ABSTRACT

LIGHTNER, CARIN ANN. A Tabu Search Approach to Multiple Sequence Alignment. (Under the direction of Dr. Shu-Cherng Fang.)

Sequence alignment methods are used to detect and quantify similarities between different DNA and protein sequences that may have evolved from a common ancestor. Effective sequence alignment methodologies also provide insight into the structure\ function of a sequence and are the first step in constructing evolutionary trees. In this dissertation, we use a tabu search approach to multiple sequence alignment. A tabu search is a heuristic approach that uses adaptive memory features to align multiple sequences. The adaptive memory feature, a tabu list, helps the search process avoid local optimal solutions and explores the solution space in an efficient manner. We develop two main tabu searches that progressively align sequences. A randomly generated bifurcating tree guides the alignment. The objective is to optimize the alignment score using either the sum of pairs or parsimony scoring function. The use of a parsimony scoring function provides insight into the homology between sequences in the alignment. We also explore iterative refinement techniques such as a hidden Markov model and an intensification heuristic to further improve the alignment. Moreover, a new approach to multiple sequence alignment is developed that provides improved alignments as compared to other methods.

A Tabu Search Approach to Multiple Sequence Alignment

by
Carin Ann Lightner

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Operations Research

Raleigh, North Carolina

2008

APPROVED BY:

_____	_____
Dr. Shu-Cherng Fang Chair of Advisory Committee	Dr. Henry L. W. Nuttle
_____	_____
Dr. Elmor Peterson	Dr. Simon M. Hsiang

Dedication

This dissertation is dedicated to my mother, Patricia Ann Lightner. Although she is only still here with us in spirit, so much of who I am has been shaped by her love, encouragement, understanding, support and selflessness. She was given the arduous task of raising three young women. With grace, poise and determination, she showed me how to be a better person and what it truly means to sacrifice everything for someone else's benefit. So, I would like to express my sincere gratitude for everything she has instilled in me and dedicate this dissertation to her loving memory.

Biography

Carin Ann Lightner was born April 25, 1977 in Cleveland, Ohio to her parents Patricia and Ardie Lightner. She is the youngest of her two siblings, Constance and Cynthia. In 1995, Carin graduated from Hathaway Brown School in Shaker Heights, Ohio.

Carin began her collegiate studies at Norfolk State University (NSU) as a Dozoretz National Institute for Mathematics and Applied Sciences (DNIMAS) scholar in the summer of 1995. After 3 ½ years in the Mathematics department at NSU, she transferred to North Carolina Central University, where she obtained a B. S. degree in Mathematics in December 1999.

Carin began her graduate studies in the Operations Research program at North Carolina State University (NCSU) as a David and Lucille Packard Fellow in the spring of 2000. She obtained an M.O.R. degree in August 2002 and continued in the graduate program as a Ph. D. student. She began working with Dr. Shu-Cherng Fang as her Ph. D. thesis advisor and pursued her research interests in the area of Bioinformatics and Soft Computing. After years of rigorous studies, she looks forward to future endeavors and pursing a career in academia.

Acknowledgments

I would like to thank my two sisters, Constance and Cynthia Lightner. As the youngest, I have often walked in their footsteps and have been guided at various junctures of my life by their words of encouragement and advice. I am so appreciative that they opened up many doors for me and helped make my path a little easier.

I would like to express my gratitude and sincere thanks to Dr. Shu-Cherng Fang. I have learned a great deal from him that extends far beyond the realms of academia. He has been extremely patient and understanding, throughout a long yet rewarding journey through NCSU. For his knowledge and wisdom, he has truly been an outstanding advisor and mentor.

A special thanks to Dr. Nuttle, Dr. Peterson and Dr. Hsiang for being on my advisory committee. I have benefited from their time, revisions and expertise.

The faculty at NCSU has been integral in my maturation as a student. I am appreciative of their instruction and superb guidance.

I am grateful to have been a part of the FANGroup. I would like to thank everyone for their help and encouragement throughout my studies.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Basics about Nucleic Acids and Genetic Sequencing

1.1.1 Nucleic Acids

The genetic information of an organism is contained in nucleic acids. Nucleic acids are composed of nucleotide chains. Each nucleotide in the chain is formed from a combination of a phosphate molecule, sugar molecule (deoxyribose or ribose) and base (adenine (A), thymine (T), cytosine (C), guanine (G) and uracil (U)). Nucleic acids composed of deoxyribose sugar molecules and bases A, T, C and G are classified as deoxyribonucleic acids (DNA), while ribonucleic acids (RNA) contain ribose sugar molecules and bases A, U, C and G. DNA and RNA contain the genetic information necessary for cell regeneration and protein synthesis. DNA is important to life because it is the long term storage facility that transfers genetic information from one generation to another [38, 18]. DNA has a double stranded helical structure, in which the bases on one strand have a binding preference to the bases on the other. Specifically, base A on one strand binds to T on the other strand and base C binds to G. This structure provides fidelity to a DNA sequence in case part of a strand is

damaged [1]. The genetic information stored by DNA is transferred to RNA for protein synthesis. Protein synthesis is important because it accomplishes most of the functions of living cells such as catalyzing chemical reactions, providing structural support and helping protect the immune system. An organism's genome contains its complete set of DNA. The arrangement of bases within the genome spells out specific instructions about an organism. These instructions spell out the way an organism looks, the organs it has, as well as all its biological components. In 2003, the human genome was sequenced with 3 billion base pairs. However, although the sequencing is known, there is still uncertainty with regard to the exact instructions spelled out for various subsequences within the human genome. To clear up some of the uncertainty about the genetic information within the human genome, researchers have used DNA sequencing. Researchers have discovered that DNA sequences of different organisms are often related. Also, similar genes are often conserved across divergent species and commonly perform similar or even identical functions. Thus, through sequence alignment of genes from various species, sequence patterns may be analyzed [27]. In general, DNA sequencing provides insight into the physical characteristics of an organism, the functions of proteins, the origins of diseases and possible cures for ailments.

1.1.2 DNA Sequencing

DNA sequencing can show the evolution of sequences that diverged from common ancestors over time. Although the most important regions of DNA are usually conserved to ensure survival, slight changes or mutations do occur as sequences evolve [38]. These mutations, shown in Figure 1-1, are any combination of insertion, deletion and/or substitution events. Methods such as sequence alignment are used to detect and quantify similarities between different DNA and protein sequences that *may* have evolved from a common ances-

Ancestral Sequence	A	T	C	T	C	G	A	G	
Insertion	A	T	C	-	C	C	G	A	G
Deletion	A	T	C	C	G	A	G		
Substitution	A	T	C	C	C	G	A	G	

Figure 1-1: Sequences evolve through a combination of mutations

tor. Sequence alignment gives insight into the structure and function of a sequence, shows a common ancestry or homology between sequences, detects mutations in DNA that lead to genetic disease and is the first step in constructing phylogenetic or evolutionary trees.

Sequence alignment (SA) is an optimal way of inserting dashes into sequences in order to minimize (or maximize) a specified scoring function [1, 38]. Generally, genetic sequencing can be classified as a pairwise sequence alignment or a multiple sequence alignment (MSA). As displayed in Figure 1-2, MSA is simply an extension of pairwise alignments that align 3 or more sequences. Both MSA and pairwise SA can further be categorized as global or local methods. Where as global methods attempt to align entire sequences, local methods only align conserved regions of similarity.

1.1.3 Dynamic Programming Approach

The most commonly used method for pairwise sequence alignment is the global dynamic programming approach proposed by Needleman and Wunsch [28]. This DP approach uses a recurrence formulation and relies on the Markovian property that future states only depend on the present state [9]. The basic idea behind the DP approach, displayed in Figure 1-3, is that any pair of sequences can be broken up into two parts in order to align them. For example, if we examine the sequence alignment CCGATT and CA- - AT, we can break this sequence up by grouping the first five base pairs together and then examining the last base

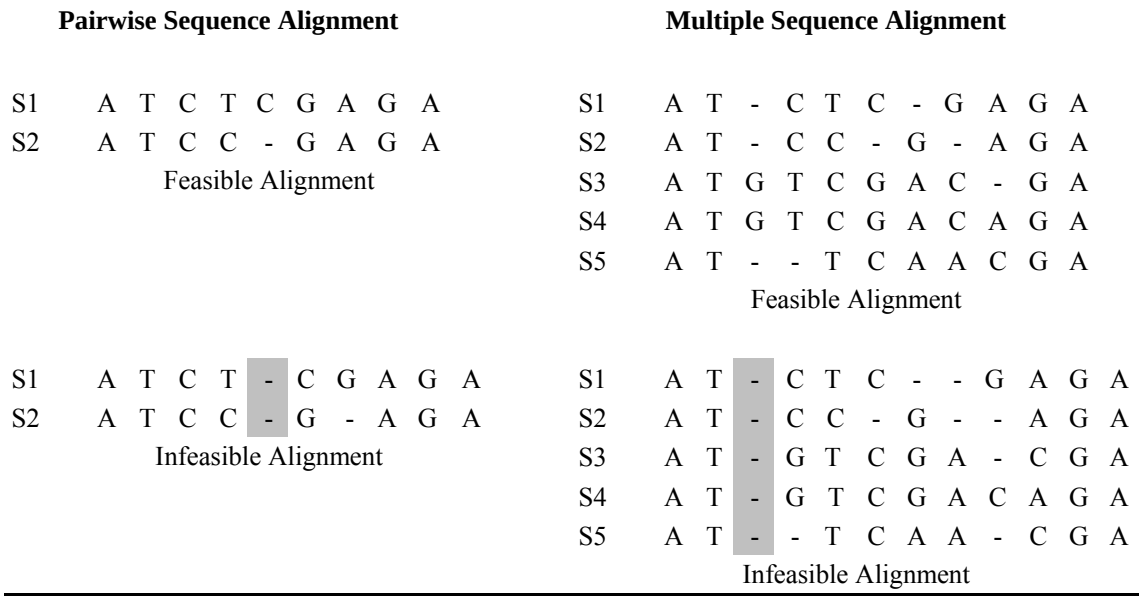


Figure 1-2: Pairwise and multiple sequence alignment

pair alone. Thus, if the scoring system is additive, the score for this alignment is the sum of the scores of the first five bases plus the score for aligning the last pair of bases.

In a pairwise sequence alignment, Figure 1-4 illustrates that for the i^{th} base of sequence one, $x_{i,}$ and the j^{th} base of sequence 2, y_j, there are only three ways x_i and y_j can be aligned; either x_i is aligned with y_j, which gives a match or mismatch, x_i is aligned with a gap or y_j is aligned with a gap [38]. In order to calculate the optimal cumulative score, $S(i, j)$, we

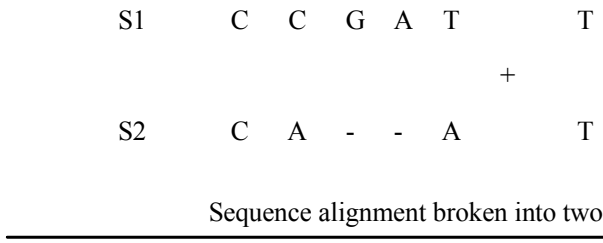


Figure 1-3: Basic idea behind DP approach

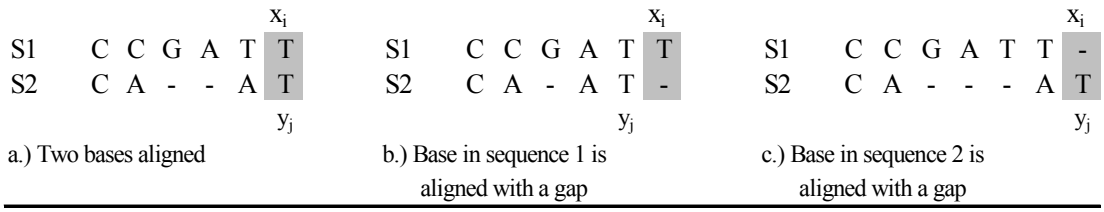


Figure 1-4: A sequence alignment can end in one of three ways

must rely on the recursive and additive nature of the DP approach. The optimal cumulative score at bases x_i and y_j is:

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

where $s(x_i, y_j)$ is the score for matching symbols x_i and y_j and d is the penalty for introducing a gap.

Calculating the optimal cumulative score for the DP approach is best illustrated using an example adapted from Needleman and Wunsch's paper. We calculate the optimal score using the array displayed in Figure 1-5a). Specifically, to find an optimal sequence alignment of sequences CCGATT and CAAT, we can create an array, displayed in Figure 1-5b), with five rows and seven columns. In this example, let the gap penalty, d, the match score and mismatch score equal 5, 1 and 0, respectively. For the first row, assign a score equal to $(-id)$ for each cell $(i, 0)$. Similarly, for the first column, assign a score equal to $(-jd)$ for each cell $(0, j)$. Then, the entire array can be filled in using the recursive equation for $S(i, j)$. The optimal cumulative score, -7, is in cell (5,7). A trace back procedure, shown with the arrows, is used to reconstruct the alignments which produce the optimal score. In this example, the optimal alignment is not unique. There are 8 optimal solutions that each have a score of -7. A multiple optimal solution is given in Figure 1-5c).

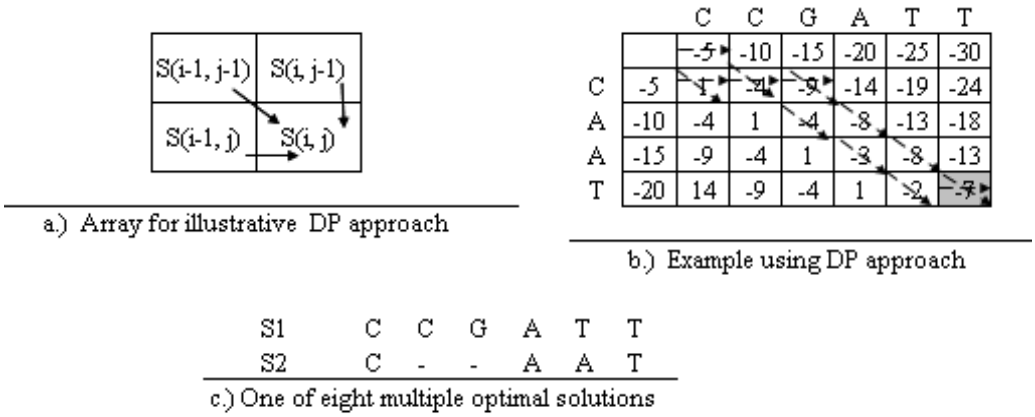


Figure 1-5: Illustration of Needleman and Wunsch's DP approach

The example above has a scoring scheme that does not take into account differing penalties for opening a gap and extending a gap. When examining the evolution of sequences, opening up a gap is much more costly than extending an existing one. Gotoh's algorithm [16] is a DP algorithm for global pairwise alignment that uses a linear gap penalty. A gap of length k is penalized $a + bk$, where a is the gap open penalty and b is the gap continuation penalty. An array representation similar to the one presented above for the Needleman and Wunsch DP approach can be used for Gotoh's algorithm. The difference is that Gotoh's algorithm has three entries per cell instead of just one. For this approach, let the first entry in a cell, $S^{AB}(A^i, B^j)$, be the penalty associated with the best alignment between sequences A^i and B^j that ends by pairing bases A_i and B_j; let the second entry, $S^{A-}(A^i, B^j)$, be the penalty associated with the best alignment between sequences A^i and B^j that ends by pairing A_i with a gap; finally, let the third cell entry, $S^{-B}(A^i, B^j)$, be the penalty associated with the best alignment between A^i and B^j that ends by pairing B_j with a gap. Also, let $s(A_i, B_j)$ be the cost/score for aligning bases A_i and B_j. Recursive formulas for $S^{AB}(A^i, B^j)$, $S^{A-}(A^i, B^j)$ and $S^{-B}(A^i, B^j)$ are as follows:

$$S^{AB}(A^i, B^j) = \min\{S^{AB}(A^{i-1}, B^{j-1}),\ S^{A-}(A^{i-1}, B^{j-1}),\ S^{-B}(A^{i-1}, B^{j-1})\} + s(A_i, B_j)$$

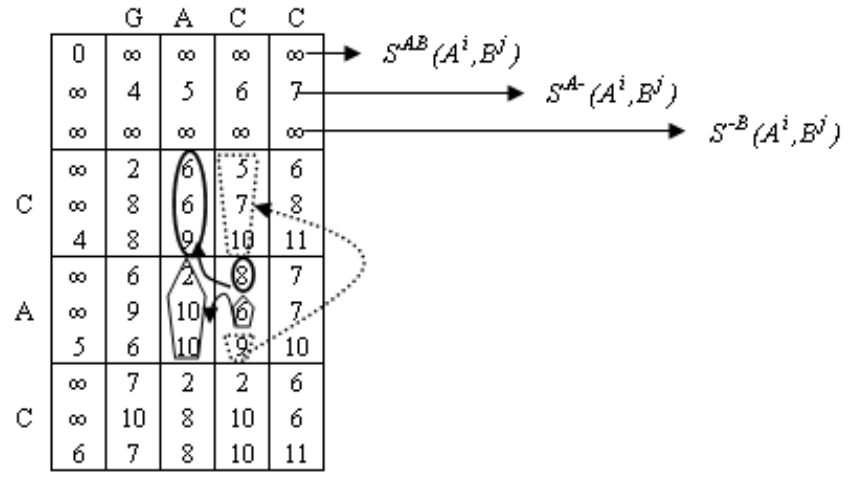
$$S^{A-}(A^i, B^j) = \min\{S^{AB}(A^{i-1}, B^j) + a,\ S^{A-}(A^{i-1}, B^j),\ S^{-B}(A^{i-1}, B^j + a)\} + b$$

$$S^{-B}(A^i, B^j) = \min\{S^{AB}(A^i, B^{j-1}) + a,\ S^{A-}(A^i, B^{j-1}) + a,\ S^{-B}(A^i, B^{j-1})\} + b$$

The score of the best alignment is the minimum of $S^{AB}(A^i, B^j)$, $S^{A-}(A^i, B^j)$ and $S^{-B}(A^i, B^j)$ or simply the minimum of the values in the cell that is in the bottom right corner. The same trace back procedure in Needleman and Wunsch's DP algorithm can be used here to recover the optimal alignment.

Sequences GACC and CAC are aligned in Figure 1-6a) using Gotoh's algorithm[43] . In this example, the match, mismatch, gap open and gap extension scores are 0, 2, 3 and 1, respectively. The three values in cell $(3, 4)$, 8, 6, and 9, are calculated using the surrounding cells in Figure 1-6a). The first entry, 8, is calculated using cell (2,3) and more specifically, 8=$\min\{6, 6, 9\} + 2$. The second entry, 6, is calculated using cell $(3, 3)$ and specifically, 6=$\min\{2 + 4, 10 + 1, 10 + 4\}$. Finally, the third entry, 9, is calculated using cell $(2, 4)$ and specifically, 9=$\min\{5 + 4, 7 + 4, 10 + 1\}$. In this alignment, the optimal score of 6 is found by taking the minimum value of the entries in cell $(4, 5)$. There are two optimal sequence alignments displayed in Figure 1-6b).

One advantage of the DP approach is that it guarantees an optimal solution. The drawback is that the time complexity for two sequences of lengths m and n is O(mn). Thus, although the DP approach can be generalized to simultaneously align 3 or more sequences, the time complexity makes this approach impractical to use for an MSA with more than 3 relatively long sequences. Therefore, instead of the DP approach, heuristics are often used to align multiple sequences.



a.) For this example, the scores are as follows:
match=0, mismatch=2, gap open=3 and gap extension=1

S1	G	A	C	C		S1	G	A	C	C
S2	C	A	-	C		S2	C	A	C	-

b.) Both of these optimal alignments have a score of 6.

Figure 1-6: Example of Gotoh's DP algorithm

1.2 Multiple Sequence Alignment

1.2.1 General Approach

Multiple sequence alignments are often classified as progressive or iterative. Typically progressive alignments involve three steps. In the first step, each pair of sequences are aligned using the DP approach. Then, the scores from the pairwise alignments in step one are used to construct a tree. Finally, the tree from step two is used to progressively align the sequences and calculate an alignment score. In progressive algorithms, once a gap is introduced in the early stages of an MSA it is always present; thus one major drawback is that an error in an initial subalignment will be propagated throughout the entire MSA [27]. To avoid these problems, iterative techniques are used and initial alignments are constantly modified. An example of a progressive MSA is shown in Figure 1-7 [19]. The array in step one has all ten pairwise distance scores. A guide tree is constructed in step two following the distance matrix. In steps 3a) and 3b), sequences one and two are aligned, then sequences three and four are aligned. Next, the two subalignments from steps 3a) and 3b) are aligned. In the last step, sequence five is aligned with the other four sequences and the final multiple sequence alignment is given.

In this example, a distance score is used to construct a tree. A distance score represents the number of changes required to change one sequence into another. In theory, the score reflects the amount of evolutionary time that has elapsed since the sequences diverged from a common ancestor; thus, a larger distance score, indicates greater evolutionary time and more sequence divergence [27]. The simplest way to calculate the distance score is to sum up the number of mismatches in an alignment and divide by the total number of matches and mismatches. In step 1 of Figure 1-7, a distance score of zero is avoided by adding a

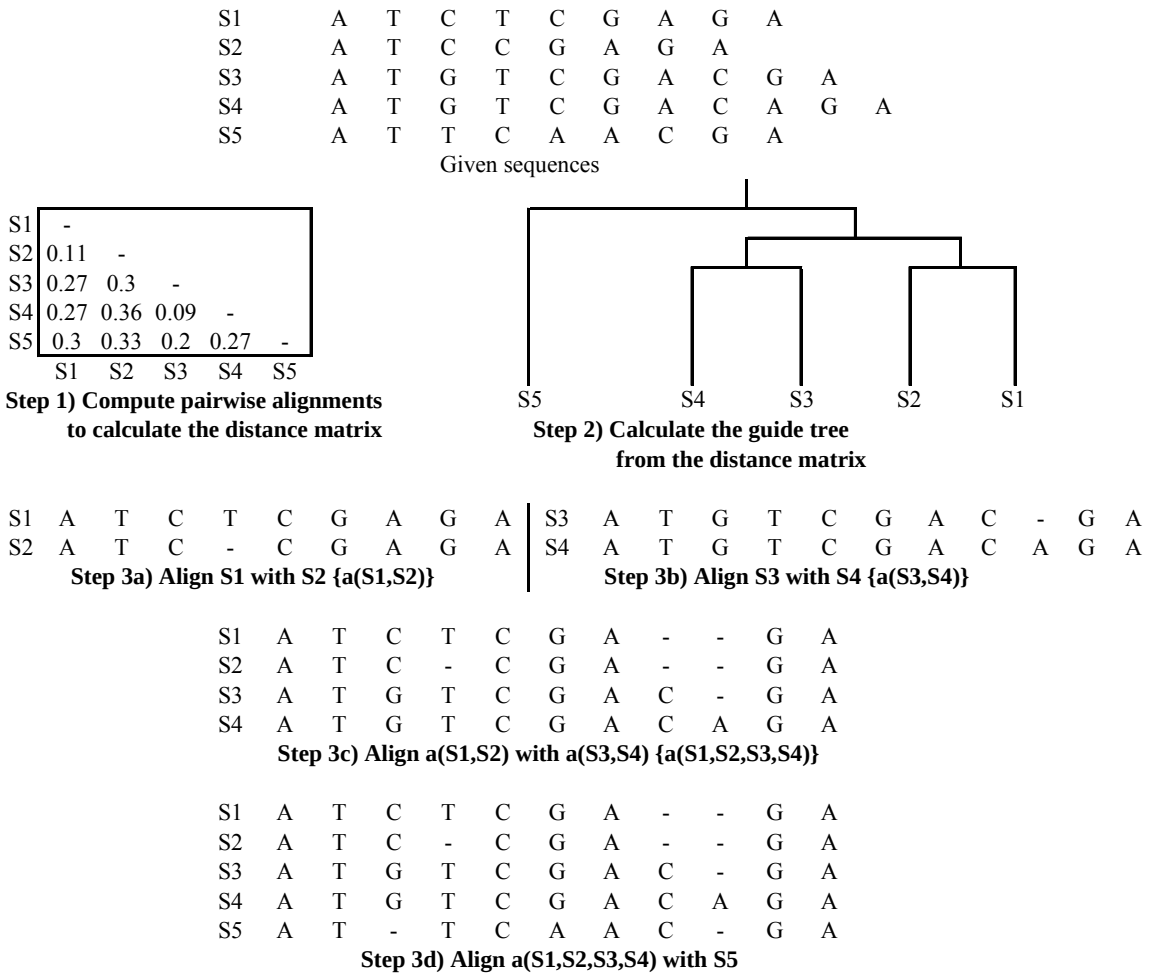


Figure 1-7: An example of multiple sequence alignment

pseudocount of 1, to the numerator and denominator. Referring to the alignment in step 3a), the distance score for sequences 1 and 2 is (0+1)/(1+8)=0.11. Conversely, a similarity score could have been used to construct a tree since typically distance and similarity scores sum to one. There is an inverse relationship between distance and similarity scores which holds even when a more complex scoring scheme involving likelihood and odds scores is used.

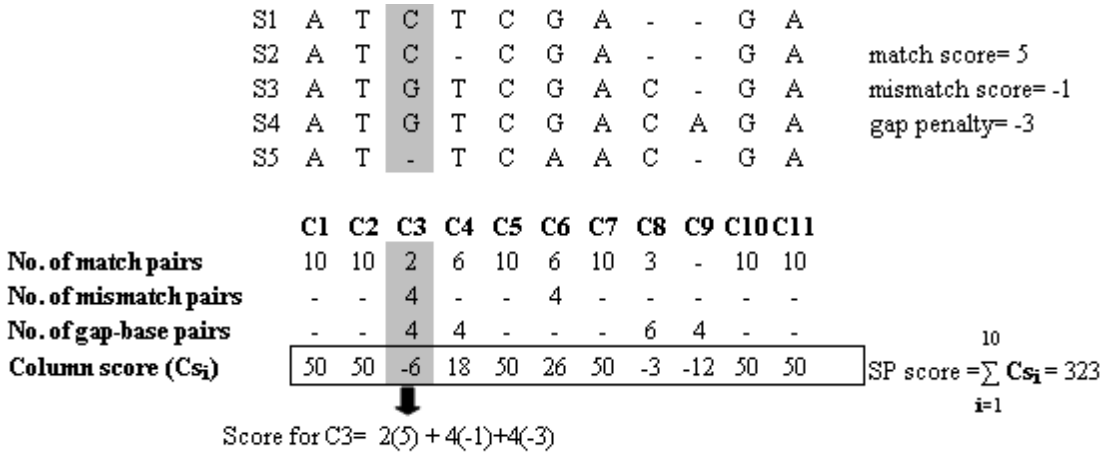


Figure 1-8: Calculation of SP score

1.2.2 Scoring an MSA

The sum of pairs (SP) method, first introduced by Carrillo and Lipman [6], is the simplest way to score an alignment. The SP score for an MSA with N sequences is defined as the sum of the scores of the N(N-1)/2 pairwise alignments For each column, this scoring scheme sums up the cost for every pair of bases as well as each gap and base pair. Then all column costs are summed to give the score for the alignment. The SP score is calculated for a multiple sequence alignment in Figure 1-8. Although the SP score is commonly used to score an MSA, there is not a biological or probabilistic justification for using this scoring scheme [9]. Also, this method tends to overestimate evolutionary events as the number of sequences increases [38, 2]. The sequences are scored as if they are not homologous and descendants of a common ancestor. Instead each sequence is scored as if it is a descendant of the others. To balance the overestimation problem, a weighted sum of pairs score is often used [9].

The weighted SP scoring function, $w(m)$, is very similar to the SP scoring function. In the weighted scoring function, $\alpha_{p,q}$ is the weight that is assigned to balance the overestimation

problem between sequences p and q in the SP score. Also, $s(m_{pj}, m_{qj})$ is the substitution scoring function that gives the score for aligning nucleotides p_j and q_j in column j. The formulation of the weighted SP score for an MSA is as follows:

$$w(MSA) = \sum_{1 \leq p < q \leq k} \left(\alpha_{p,q} * \sum_{j=1}^{L} s(m_{pj}, m_{qj}) \right)$$

There are various weighting schemes and substitution scoring functions that are used to calculate a weighted SP score. Other scoring methods include a star phylogeny, information content and graph based trace methods [27]. In general, because there is not a standard scoring scheme for multiple sequence alignments, different scoring schemes will affect and alter the final alignment. Many of the existing MSA programs use a variety of scoring functions. In Chapter 2, there is a review of commonly used MSA programs and the corresponding scoring functions.

In Chapter 3, we introduce a tabu search approach to multiple sequence alignment. Three tabu searches are implemented that attempt to maximize the SP scoring function. We start with an initial tabu search, Tabu A. The initial tabu is modified by implementing two subsequent tabu searches, Tabu B and Tabu A', to improve either the CPU time or SP score of Tabu A.

In Chapter 4, we present the computational results from using Tabu A, Tabu B and Tabu A' to align multiple sequences. We examine the MSAs, SP scores and CPU times generated from each of the tabu searches for several groups of sequences. ClustalW, PRALINE and SAGA are used to align the same groups of sequences. The results from these other MSA programs are compared with the results from the tabu searches.

In Chapter 5, we outline an improved tabu search, Tabu C. Tabu C uses components of the previous tabu searches. This tabu also incorporates a tree to help guide the alignment. Initially, we use an SP scoring function to evaluate MSAs from Tabu C. Subsequently, a

parsimony score is used to evaluate the alignments. Additional components are also added to Tabu C to help locally improve the MSAs.

In Chapter 6, we present the computational results from Tabu B, Tabu C and Tabu A' to align multiple sequence. The alignment scores and CPU times from the three tabu searches are compared with the results from ClustalW, KALIGN, PRALINE, SAGA and PRRN.

In Chapter 7, we have the conclusion and discussion of future research works.

Chapter 2

Literature Review

ClustalW is a commonly used multiple sequence alignment program [20, 42, 21]. As with any other heuristic, ClustalW does not guarantee an optimal solution. It progressively aligns sequences and exploits the fact that similar sequences are evolutionarily related. First, ClustalW aligns and scores all possible pairs of sequences to determine their distance score. Then a guide tree is constructed using the edit distances and a neighbor joining algorithm. Finally, the guide tree is used to progressively align the sequences. Although an optimal solution is not guaranteed, ClustalW usually provides a good starting point for other refinement methods such as Hidden Markov Modeling. Since ClustalW progressively aligns sequences, any regions that were misaligned early in the process can not be corrected as the program progresses and more information is introduced. Another problem with ClustalW is the choice of alignment parameters. Sequences that are not highly conserved or not very similar are extremely sensitive to the adjustments of the parameters. Thus, when aligning divergent sequences, slight parameter adjustments will drastically change the final multiple sequence alignment. In general, it is difficult to justify why one scoring matrix or parameter selection is better than another [38]. As a direct result of the uncertainty involved in select-

ing the parameters, ClustalW is most useful when sequences are known to be evolutionarily related.

Notredame and Higgins [30] have the best known genetic algorithm, Sequence Alignment by Genetic Algorithm (SAGA), for multiple sequence alignment. Similar to other genetic algorithms (GA), SAGA uses the principles of evolution to find the optimal alignment for multiple sequences. This method generates many different alignments by rearrangements that simulate gap insertion and recombination events to generate higher and higher scores for the MSA [27]. In this GA, the population consists of alignments that were formed from a complex set of twenty two different crossover and mutation operations. To determine the fitness of an alignment, SAGA uses a weighted sum of pairs approach in which each pair of sequences is aligned and scored; then the scores from all the pairwise alignments are summed to produce an alignment score. As with any heuristic approach, SAGA may not generate an optimal MSA. Although it has been shown that SAGA does produce quality alignments, the time complexity involved in the weighted sum of pairs fitness function is a major drawback to this approach.

Shyu *et al.* [38] propose two multiple sequence alignment algorithms using genetic algorithms. In the first algorithm, the GA structure is focused on developing guide trees, which determine the order of the sequences in the MSA. These guide trees are an alternative to heuristic techniques such as neighbor joining. The GA evolves the guide trees to ultimately produce quality alignments. The population consists of viable guide trees that have a coalescing binary tree structure. The fitness of these trees is measured by the natural logarithm of the alignment score. A parameterized beta distribution is used for this rank-based selection procedure. Thus, there is a strong bias for selecting more fit parents for the next generation and replacing less fit parents. The second GA focuses on evolving

an optimal consensus sequence. This consensus sequence is simply a compact formulation representing all possible alignments for virtually any given number of sequences. With this GA, the total number of iterations needed to find an optimal solution is independent of the number of sequences being aligned but dependent on the length of the sequences.

Riaz *et al.* [34] propose a tabu search algorithm for multiple sequence alignment. The algorithm implements the adaptive memory features typical of tabu searches to align multiple sequences. Both aligned and unaligned initial alignments/solutions are used as starting points for this algorithm. Aligned initial solutions are generated using Feng and Doolittle's progressive alignment algorithm [11]. Unaligned initial solutions are formed by inserting a fixed number of gaps into sequences at regular intervals. The quality of an alignment is measured by the COFFEE objective function [31]. In order to move from one solution to another, the algorithm moves gaps around within a single sequence and performs block moves. This tabu search uses a recency-based memory structure. Thus, after gaps are moved, the tabu list is updated to avoid cycling and getting trapped in a local solution. This tabu search has two parts. The first part of the algorithm implements a typical tabu search procedure, outlined in the next chapter, until the solution or an alignment is stabilized. Then the second part of the algorithm uses a intensification and diversification procedure, which focuses on poorly aligned regions. Finally, the algorithm terminates when no more poorly aligned regions are identifiable and the alignment quality cannot be improved. As with any heuristic approach, an optimal solution is not guaranteed; however generally, this tabu search produces comparable results to ClustalW and various other alignment programs.

Lukashin *et al.* [25] propose a simulated annealing (SA) approach to multiple sequence alignment. This procedure is based on the Metropolis-Monte Carlo algorithm. The efficiency of this simulated annealing method depends mainly on the behavior of the cost

function. The multiple sequence alignment problem for N sequences attempts to find the alignment that maximizes the cost function or matching score

$$M(s) = \sum_{k+1}^{L} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} Comp(S_i(k), S_j(k)) - \sum_{i=1}^{N} P(S_i)$$

where L is the new length of the aligned sequence, i and j represent the sequence number, k represents the base position, $P(S_i)$ is the gap penalty function and the function $Comp(S_i(k), S_j(k))$ compares two bases from different sequences and reflects the degree of similarity.

In general, the simulating annealing procedure starts off at a high temperature that allows many moves to alignments that have both better and worse scores. This feature, that allows moves to a worse scoring alignment, helps SA procedures avoid local optimal solutions. A cooling scheme is introduced to gradually decrease the temperature, the number of overall moves and the moves to worse scoring alignments. Eventually a global minimum is reached at the limit of zero-temperature. This algorithm was tested on E. Coli sequences and a consensus sequence of the aligned set was developed. One drawback of this algorithm is that the running time can vary tremendously depending on the selection of the cooling scheme and cost function.

Brudno *et al.* [5] propose a glocal alignment approach that combines features of both local and global alignment methods. This glocal aligner, Shuffle-LAGAN (SLAGAN), is a pairwise alignment algorithm that can be extended to multiple sequence alignment. The distinct differences between global and local alignments are merged in the SLAGAN approach. Whereas global alignments transform one sequence into the other by using a combination of insertions, deletions and substitutions (simple edits), local alignment techniques tend to focus on similarities between conserved regions; this glocal alignment combines these features and creates a map that transforms one sequence into the other while allowing for rearrange-

ment events. SLAGAN includes rearrangement events because DNA is known to mutate by simple edits, rearrangements such as translocations (a subsegment is removed and inserted in a different location but with the same orientation), inversions (a subsegment is removed from the sequence and then reinserted in the same location but with the opposite orientation) and duplications (a copy of a subsegment is inserted into the sequence and the original subsequence remains unchanged) or any combination of these simple edits. SLAGAN quickly aligns long sequences. In this technique, a penalty is incurred for the set of operations that include insertions, deletions, point mutations, inversions, translocations and duplications. This approach minimizes the sum of these penalties (edit distance). SLAGAN has three distinct stages. The first stage consists of finding local alignments using the CHAOS tool. The second stage picks the maximal scoring subset of the local alignments under certain gap penalties to form a 1-monotonic conservation map. Whereas standard global alignments are non-decreasing in both sequences, the structure of the 1-monotonic conservation map is non-decreasing in one sequence and without restrictions in the second sequence. Relaxing this assumption in the second sequence, allows the algorithm to detect rearrangements. The inclusion of rearrangement events is one of the features that makes this algorithm different than typical global alignment approaches. In the final stage of SLAGAN, the conservation map of local alignments is joined to form maximal consistent subsegments that are aligned using the LAGAN global aligner. One drawback of the SLAGAN algorithm is that it is not symmetric in the sequence order, so it frequently misses duplications in the sequences.

Ogal and Ericyes [32] propose a method of identifying local similarities between DNA sequences that can be used to determine a global alignment for a pair of sequences. This algorithm finds all locally optimal alignments. There are four distinct stages in the algorithm. The first stage represents the two sequences together by constructing a compacted

tree, called a suffix tree, that only stores the suffixes of a given sequence. The second stage finds maximum unique matches between the sequences by traversing the suffix tree. A maximum unique match cannot be a subsequence of any longer such sequence and it must occur only once in both sequences. The third stage uses a variation of the longest increasing subsequence algorithm by putting the matches in the same order with respect to both sequences. The final stage uses a dynamic program to extract all locally optimal alignments in the non-matching regions and completes the global alignment. This algorithm finds a near-optimal global alignment quickly. Also, this global pairwise method can be used in typical multiple sequence alignments to perform the initial pairwise comparisons. One drawback of this algorithm is that it only uses simple edits and does not permit rearrangements.

Lassman *et al.* [23] propose KALIGN, a method employing the Wu-Manber string-matching algorithm [46], to improve both the accuracy and speed of multiple sequence alignment. The KALIGN algorithm follows a strategy similar to many progressive methods for sequence alignment. Pairwise distances are calculated and a guide tree is constructed based upon the distance calculations. Finally, sequences/profiles are aligned in the order given by the guide tree. In contrast to previous methods, the Wu-Manber approximate string-matching algorithm is used to calculate the pairwise distances. KALIGN is a string matching algorithm that estimates sequence distances quickly and more accurately than many existing methodologies. The Wu-Manber algorithm allows string-matching with mismatches and is simply an extension of the exact Baeza-Yates-Gonnet algorithm [4]. The distance between two strings is measured using the Levenshtein edit distance [24]. Two strings A and B have an edit distance, d, if A can be changed into B by a series of d mismatches, insertions or deletions. The Wu-Manber algorithm determines pairwise distances between highly divergent sequences because it can readily identify shared mismatch patterns. The sum of

the three highest scoring diagonals quantifies the sequence similarity. This method is much faster than performing all pairwise alignments. Based upon this sequence similarity score, the UPGMA clustering method [40] is used to construct a guide tree. KALIGN follows the guide tree to progressively align the sequences using a dynamic programming algorithm.

PRRN is a global multiple sequence alignment program proposed by Gotoh [17]. PRRN is a best-first search iterative refinement strategy with tree-dependent partitioning for multiple sequence alignment. The main feature of the strategy is the iterative refinement stage. To make the alignment, phylogenetic tree and pair weights mutually consistent, a doubly-nested randomized iterative (DNR) method is used. The iterative refinement is based on a group-to-group sequence alignment (GSA) algorithm that uses an affine gap cost. In order to incorporate the piecewise linear gap cost into the GSA algorithm, the gap lengths are calculated. The true optimum may not be attained using the GSA algorithm since it is a hill-climbing heuristic. PRRN is most effective when refining an alignment that was obtained by an alternative progressive alignment method.

Schwartz *et al.* [37] propose an approach to multiple sequence alignment that is based on an algorithm for rapidly checking whether single matches are consistent with a partial multiple alignment. This sequence annealing algorithm incrementally builds multiple sequence alignments one match at a time. Homologous annealing positions are aligned first. The alignment proceeds by segment-to-segment comparisons. Unlike other alignment algorithms, segments of size 1 are allowed. The sequence annealing process uses the minimal steps that reduce the number of columns in the (partial) alignment by one. In order to incrementally reduce the number of columns, two existing columns are merged into one. The most challenging part of the sequence annealing algorithm is that not all pairs of columns can necessarily be merged in a partial multiple sequence alignment. The merge operation can

be done efficiently using the Pearch and Kelly online topological ordering algorithm (2004). Use of the Pearch and Kelly algorithm allows for rapidly constructed global multiple alignments. In general, sequence annealing has the advantage that the intermediate alignments produced during the annealing process help identify locally conserved regions of similarity in the alignment.

BLAST [3] and variations of this algorithm are some of the most commonly used heuristic local alignment programs. These programs compare DNA/ protein sequences to DNA/protein databases in order to find patches of regional similarity. Gapped BLAST attempts to optimize a similarity measure by searching the database in two phases. First, it uses a two-hit method that requires the existence of two non-overlapping segments, on the same diagonal within a specified distance. If these two word pairs each have an aligned score that is at least above a threshold parameter, T, then gap BLAST extends these subsequences. Finally, the algorithm uses dynamic programming to extend these two subsequences and only examines alignments that drop in score no more than a specified amount below the best score yet seen. Selecting the threshold parameter can be difficult because there is a trade-off between speed and sensitivity. While a higher value of T speeds up the search, it also has a lower probability of finding weak regional similarities. Generally, BLAST is most useful when there are compact highly conserved regions of similarity with sequence alignments.

ParAlign [35] is a parallel sequence alignment algorithm that quickly computes ungapped alignment scores for all diagonals in an alignment matrix. The alignment with the highest score is considered optimal. In order for other local alignment techniques such as BLAST to detect sequence similarities, it is required that these similarities be concentrated. The technique also identifies high scoring ungapped alignments first and then determines gapped alignments based on these preliminary findings. ParAlign differs from other local alignment

techniques because it detects similarities that are more spread out along one diagonal in the alignment matrix or divided among several diagonals. The ParAlign algorithm has three distinct phases. The first phase calculates the exact ungapped alignment score for each diagonal in parallel. The second phase computes an approximate gapped alignment score, by combining the scores for each diagonal in a new inter-diagonal scoring function. Finally, these approximate scores are used to select the highest scoring database sequences for a subsequent Smith-Waterman [39] parallel alignment.

Chapter 3

New Approach to Multiple Sequence Alignment

3.1 Overview

In this chapter, we outline key components for three different tabu searches that are used to align multiple sequences. In general, each tabu search improves upon the results of the previous tabu search. Initially, we implement Tabu A. The alignment score and CPU time are used to determine the quality and efficiency of a tabu search. We compare the alignment scores and CPU times from Tabu A to the corresponding results from ClustalW, PRALINE and SAGA. This analysis led us to implement Tabu B. Tabu B is a modified version of Tabu A that is implemented to speed up the algorithm and reduce the overall CPU time. Again, we compare the results from Tabu B, to the corresponding results from other MSA programs. This analysis led us to focus on improving the alignment score. We added a local improvement algorithm called a hidden markov model (HMM) at the end of Tabu B to further improve the MSA and alignment score. Due to the extensive calculations, the

HMM is only used to align 50 sequences or less. Thus, to improve the MSA score for larger groups of sequences, we implement Tabu A'. Tabu A' is also a modified version of Tabu A.

3.2 Use of Tabu Search vs. Other Sequence Alignment Algorithms

ClustalW, SAGA and most other MSA algorithms use a tree to guide the alignments. The guide tree is most commonly formed using both the distance scores and a neighbor joining algorithm. Sequences that are similar are clustered together in groups while dissimilar sequences are spread apart. The tree is used to form the MSA; then, both the tree and the MSA are used to determine phylogenetic relationships. The disadvantage to determining an MSA and predicting phylogenetic relationships based upon this type of guide tree is that tree does not attempt to minimize (or maximize) the most important measure of an MSA, the alignment score. Also, an incorrect guide tree might bias the results of an evolutionary tree used to predict phylogenetic relationships. Thus, we implement a tabu search approach based upon optimizing the sum of pairs scoring function of an MSA.

3.3 Basic Components of a Tabu Search

Tabu search is a heuristic approach that uses adaptive memory features to align multiple sequences. The adaptive memory feature, a tabu list, helps the search process avoid local optimal solutions and explores the solution space in an effective manner [34]. While the tabu list restricts the search of some neighboring solutions, there are conditions which allow exceptions to the tabu list. As displayed in Figure 3-1, a tabu procedure starts with an initial solution, generates neighbors and then moves to the best accessible neighboring

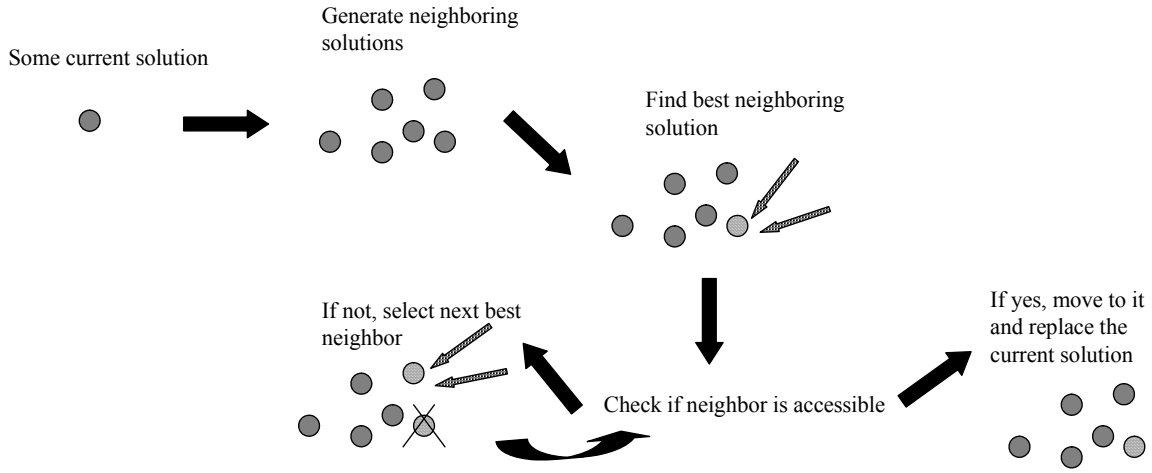


Figure 3-1: Illustration of neighbor selection in a tabu search

solution. Accessible solutions are either not on the tabu list or are on the tabu list but satisfy conditions which allow exceptions to the tabu [10]. Generally, the steps displayed in Figure 3-1 are repeated, until some termination criteria is met.

The two main features of a tabu search are its adaptive memory and responsive exploration strategies. Moving to the best available neighborhood solution, even if it is worse than the current solution, allows the algorithm to explore more of the solution space. The tabu list is updated every iteration and maintains a list of move attributes that are not allowed unless certain conditions are met. These conditions that allow for exceptions from the tabu list are called the aspiration criteria. In general, the aspiration criteria allow moves that will lead to better solutions than have previously been found. In cases where the tabu search becomes stabilized and is no longer moving toward better solutions, a diversification and intensification procedure is implemented, so that more of the solution space can be explored. The tabu search procedure used in Glover's tutorial [15] is outlined in Figure 3-2

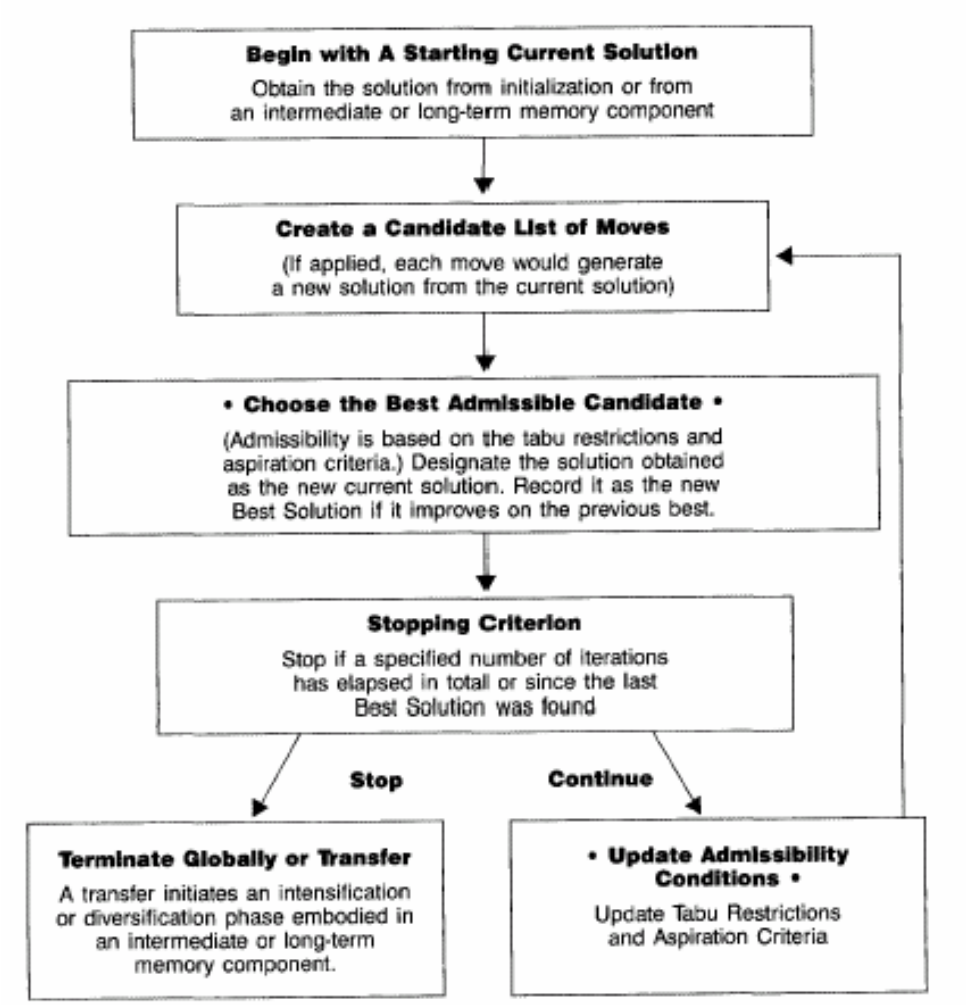


Figure 3-2: Procedure for basic tabu search

3.4 Tabu Search A and B

We initially develop and implement two tabu search algorithms, Tabu A and Tabu B. The basic components of a tabu search algorithm outlined in the previous section are used in these algorithms. A solution consists of arrays that contain the sequence order and the positions of the gaps in the corresponding MSA. The actual MSA is not stored in memory. The optimality criterion attempts to maximize the most important measure of an MSA -the alignment score. Thus, the quality of an alignment is measured using the final alignment score. Finally, the algorithm terminates after the current best solution has not improved for a specified number of iterations.

There are three types of move strategies that will generate a neighbor from a current solution. Using the solution representation from Tabu A, Figure 3-3 displays the first two types of moves. The first type of move is made by swapping pairs of sequences. In Figure 3-3a) the current solution, is composed of two arrays, s and sg, that contain the sequence order of the MSA and the gap locations, respectively. A possible neighbor, displayed in Figure 3-3b), is generated by swapping the sequences in positions 3 and 5 as well as the sequences in positions 8 and 10. For this neighbor, the arrays containing the sequence order and gap location are different from the arrays in the current solution. The second type of move is made by swapping blocks (two or more consecutive sequences) of sequences in particular positions. For example in Figure 3-3c), a possible neighbor of the current solution is generated by swapping sequences in positions 2, 3 and 4 with sequences in positions 7, 8 and 9. The third type of move is made by changing the gap positions within a sequence. So, the array containing the sequence order would remain the same and only the array with the gap positions would change.

Both tabu search algorithms use the same move strategy and stopping criteria. Although

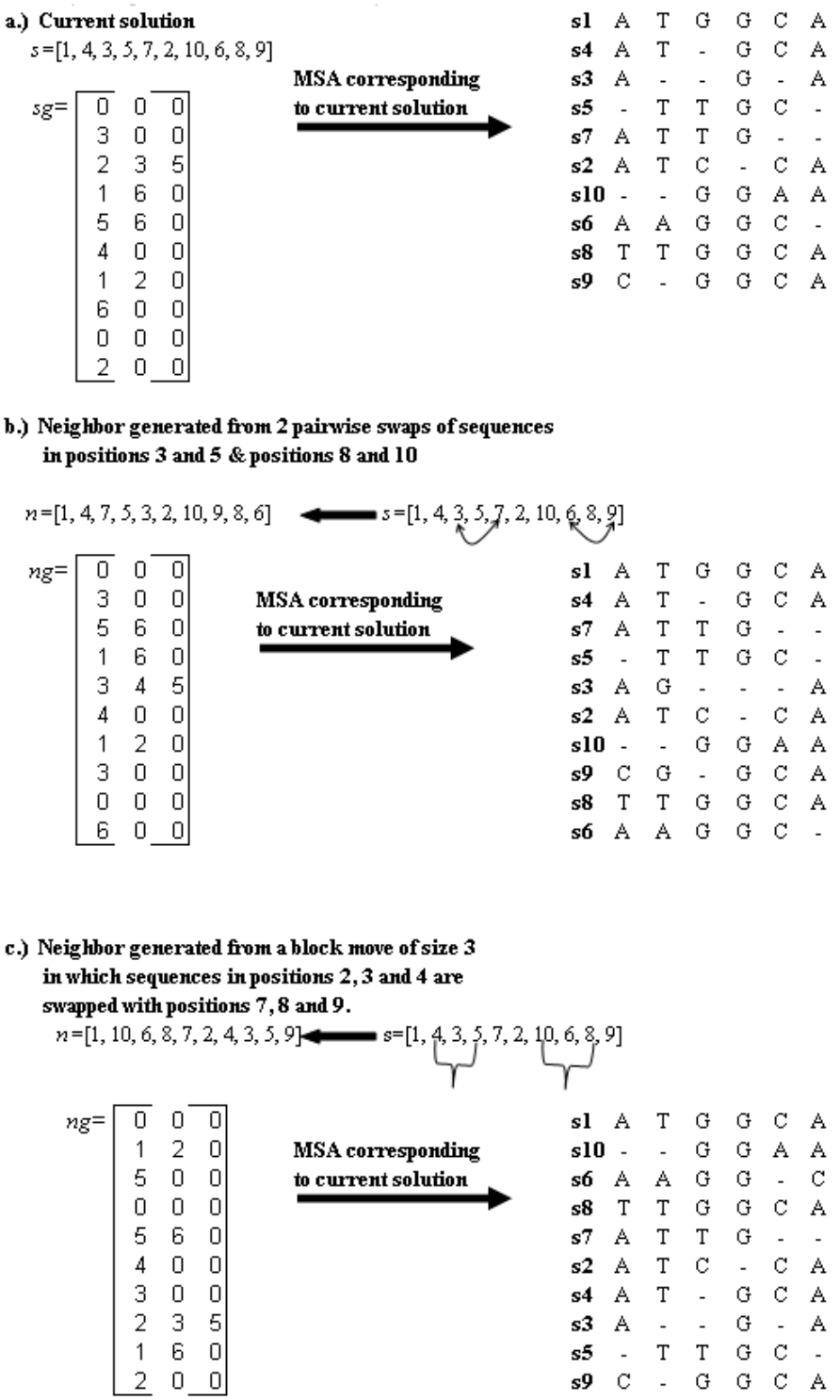


Figure 3-3: Illustration of pairwise and block moves

both algorithms have similar components, the difference lies in there basic structure. Tabu A, simply progressively aligns all N sequences together in the order specified by a solution. For the MSA in Figure 3-3a), $s1$ is aligned with $s4$ to compose the subalignment, $a(s1, s4)$. Next, $s3$ is aligned with the subalignment $a(s1, s4)$ to compose the subalignment, $a(s1, s4, s3)$. This progressive alignment continues until the entire MSA, $a(s1, s4, s3, s5, s7, s2, s10, s6, s8, s9)$, is composed. One difference between the two tabu searches is that the solution representation of Tabu A consists of two parts, while that of Tabu B consists of $g + 1$ parts, where g is the number of equal-sized ordered groups that the N sequences are divided into. In the first step of Tabu B, the array, s, containing the sequence order is divided into g equal-sized groups. Each of the ordered groups is progressively aligned (separately) and the gap locations for the g subalignments are stored in arrays $sG_1, ..., sG_g$. Then, all of the subalignments are progressively aligned to make up the MSA, $a(sG_1, sG_2, ..., sG_g)$. Figure 3-4 displays the difference in the structure of the two algorithms. After an initial solution is determined, the procedure for finding an accessible neighbor as outlined in the previous section is followed using all three move strategies. In Tabu B, block and pairwise moves can be made within a subalignment or between two or more subalignments. If moves are made that change the order of the sequences within a subalignment, then, first, all the sequences within that particular subalignment are progressively aligned and second, all g subalignments are progressively aligned to make up an MSA. In both tabu searches, this cycle of moving to new solutions is continued until the stopping criteria are met. When each of the tabu search algorithms terminate, the *best* solution is the one with the highest alignment score. After these tabu searches, a local improvement algorithm is implemented to further improve the sequence alignment score. A hidden Markov model is a local alignment algorithm that is used after Tabu B terminates.

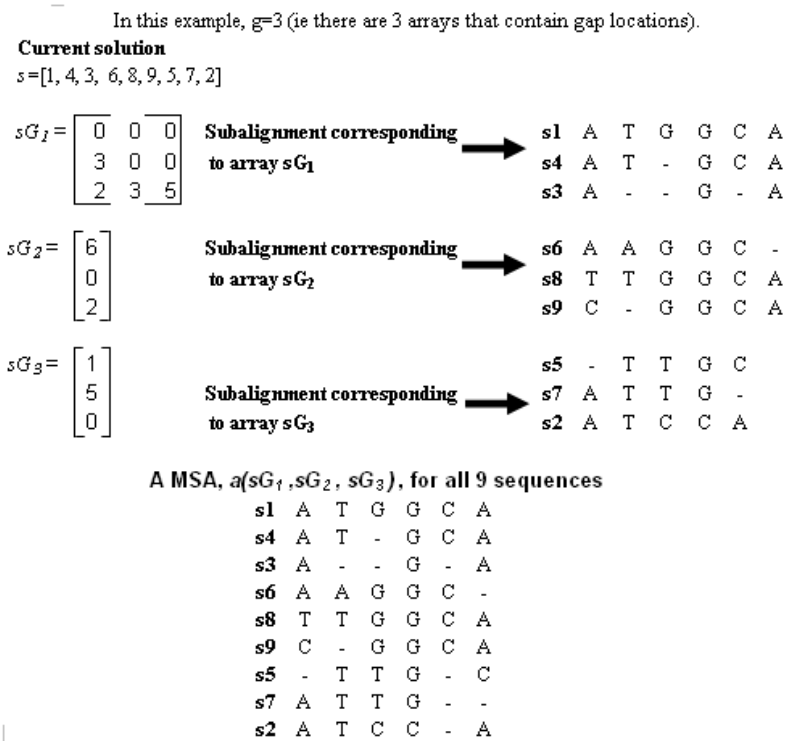


Figure 3-4: An example of a solution for Tabu A

3.5 Hidden Markov Model for Local Improvement of MSA

3.5.1 Basics Components of HMM

A hidden Markov model (HMM) is a doubly stochastic process in which the states are not directly observable (i.e. hidden). However, by using a Markov model of other states that are observable, the hidden states can be indirectly and probabilistically observed. Generally, a hidden Markov model is defined by the vector of initial state probabilities, the state transition matrix and the emission probability matrix. HMMs can be used to align multiple sequences. An advantage of using an HMM is that penalty values for opening a gap, extending a gap, scoring matches and mismatches are not used. Thus, the variability and sensitivity of alignments due to these cost parameters is not an issue with this type of model. The HMM is deeply rooted in statistics and it considers all possible combinations of matches, mismatches and gaps to generate an alignment of multiple sequences [27]. This model aligns sequences locally and is directly influenced by the initial alignment. Therefore, if this model is used in conjunction with a heuristic that initially aligns the set of sequences globally, an improved multiple sequence alignment may be determined.

Typically, HMMs have three types of hidden states that correspond to a column in an alignment -insert states, delete states and match states. The observable states are the actual DNA bases that are aligned from each position of the MSA. Figure 3-5 shows the HMM representation of a section of an MSA given in Krogh *et al.* [22]. Each column in the model represents the possibility that a match, insert or delete is in the corresponding column of an alignment [27]. A match state, represented by a square, contains DNA base probability distributions. An insert state, represented by a diamond, produces a random base insertion between aligned columns. A delete state, represented by a circle, corresponds to a gap in an

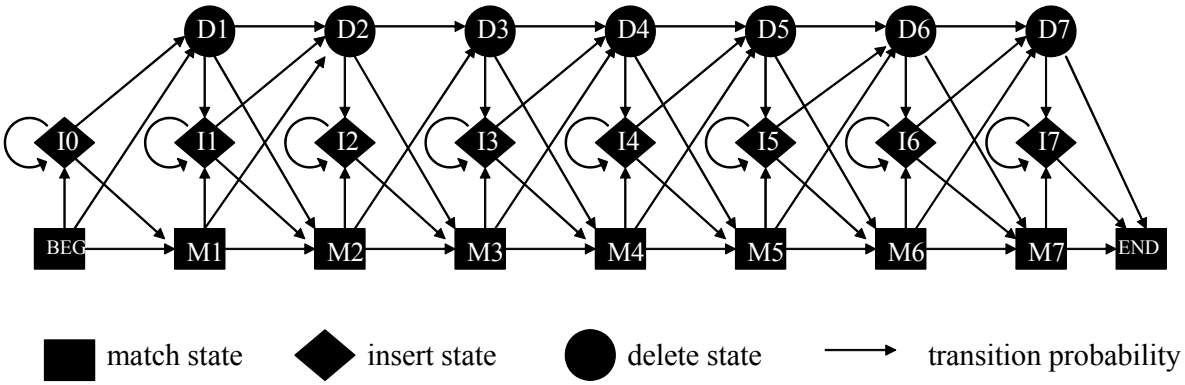


Figure 3-5: Hidden markov model for multiple sequence alignment

aligned column. Each arc represents a transition probability between hidden states. Any path through this model that starts with the BEG state and ends with the END state, has a probability associated with it. Analysis of these probabilities and the relationship between the hidden states and aligned sequences allows three basic questions to be answered. The first question is with what probability does a given HMM produce a given sequence? The next question is what path through the hidden states has the highest probability or is most probable? The final question is what are the parameters of an HMM that most probably underlie a given set of sequences. Each of these three questions can be answered using the forward-backward, Viterbi and EM algorithms which will be described in latter sections.

Given an observation sequence, O = (o_1, o_2,...,o_T), and N hidden states, an HMM is commonly defined by the following three parameters adapted from Rabiner and Juang [33]:

$\Pi = (\pi_i)$ vector of initial state probabilities

$A = (a_{ij})$ state transition matrix

$E = (e_j(o_t))$ emission (observation) probability matrix

The vector of initial state probabilities is a vector containing the probabilities of being in a match, insert, or delete state at time t=1. The state transition matrix contains the

probabilities of going from one hidden state to another and the emission probability matrix contains the probability that a base will actually be emitted, or shown, in a particular position in an alignment. The emission and state transition matrices do not change in time as the system evolves [33].

3.5.2 Forward and Backward Algorithms

The forward algorithm is a recursive algorithm that determines the probability that a given HMM emits a particular sequence. The forward algorithm sums up the probability of every possible path through the hidden states. The basic idea behind the forward algorithm is as follows:

Let

$$f_t(i) = \Pr(\text{observation} \mid \text{hidden state is } i) * \Pr(\text{all paths to state } i \text{ at time } t)$$

$$= \Pr(\text{o}_1, \text{ o}_2, ..., \text{o}_t \mid \text{hidden state is } i) * \Pr(\text{all paths to state } i \text{ at time } t)$$

Initialize at $t = 1$

$$f_1(i) = e_i(o) * \pi_i \qquad\qquad i \in [1...N]$$

Repeat recursion for hidden states $t = 2$ to $t = T - 1$

$$f_{t+1}(j) = e_j(o_{t+1}) * \left\{ \sum_{i=1}^{N} f_t(i) a_{ij} \right\} \qquad\qquad i, j \in [1..N],\ t \in [1, ..., T-1]$$

Finally, the sum of all partial probabilities gives the probability that a given observation sequence produces a given HMM or

$$\Pr(\text{O} \mid \text{HMM}) = \sum_{i=1}^{N} f_T(i)$$

An example of forward algorithm calculations for a portion of an HMM is displayed in Figure 3-6a). In this example, the observation sequence, O, is GTCG. The calculations for $f_{t+1}(j)$ are only displayed for states IO , $M1$, $D1$ and $I1$.

The backward and forward algorithms have a similar recursive structure. Whereas the

forward algorithm computes probabilities from the initial column in an observation sequence to a certain time, t, the backward algorithm computes probabilities from a certain time, t, to the final column in an observation sequence. For the backward algorithm, we have the following structure:

Let

$b_t(i) = \Pr(o_{t+1}, o_{t+2},...,o_T \text{ |hidden state is } i) * \Pr(\text{all paths to state } i \text{ at time } t)$

Initialize at $t = T$

$b_T(i) = 1 \qquad\qquad i \in [1...N]$

Repeat recursion for hidden states with $t = T - 1$ to $t = 1$

$b_t(i) = \sum_{j=1}^{N} a_{ij} * e_j(o_{t+1}) * b_{t+1}(j) \qquad\qquad i, j \in [1...N],\ t \in [1, ..., T-1]$

3.5.3 Viterbi Algorithm

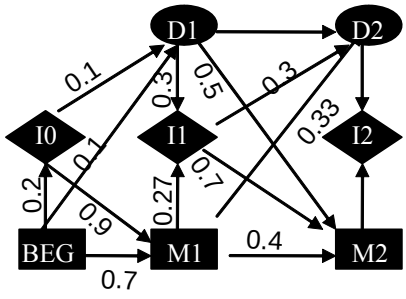
The Viterbi algorithm [33, 45] determines the best path through the hidden states. The Viterbi and forward algorithms are very similar. Whereas the Viterbi algorithm finds the partial best path into each state by taking the path with the maximum probability, the forward algorithm sums up the probability of all paths into each state. The partial best path is defined as the sequence of hidden states which achieves the maximum probability. For the Viterbi algorithm, define $\delta_t(i)$ as the maximum probability of all paths ending at state i at time t, and $\psi_t(i)$ as the previous state on the partial best path. The basic idea behind the Viterbi algorithm is as follows:

Initialize at $t = 1$

$\delta_1(i) = e_i(o_1) * \pi_i$

$\psi_1(i) = 0$

Repeat recursion for hidden states with $t > 1$



o_1=G
o_2=T
o_3=C
o_4=G

$e_{M1}(A)$=0.1
$e_{M1}(C)$=0.3
$e_{M1}(G)$=0.4
$e_{M1}(T)$=0.2

$e_{IK}(o_t)$=0.25 $\forall$ K

a.) Calculations using forward algorithm

$f_0(beg) = 1$

$f_1(IO) = e_{I0}(G) * \pi_{IO}$
$= 0.25 * 0.2$
$= \mathbf{0.05}$

$f_2(M1) = e_{M1}(T)[f_1(beg) * \alpha_{begM1} + f_1(IO) * \alpha_{IOM1}]$
$= 0.1[0 * 0.7 + 0.05 * 0.9]$
$= \mathbf{0.0045}$

$f_2(D1) = [f_1(beg) * \alpha_{begD1} + f_1(IO) * \alpha_{IOD1}]$
$= [0 * 0.1 + 0.05 * 0.1]$
$= \mathbf{0.005}$

$f_3(I1) = e_{I1}(C)[f_2(M1) * \alpha_{M1I1} + f_2(D1) * \alpha_{D1I1}]$
$= 0.25[0.0045 * 0.27 + 0.005 * 0.3]$
$= \mathbf{0.000678}$

---

b.) Calculations using Viterbi algorithm

$\delta_0(beg) = 1$

$\delta_1(IO) = e_{I0}(G) * \pi_{IO}$
$= 0.25 * 0.2$
$= \mathbf{0.05}$ $\quad \psi_1(IO) = 0$

$\delta_2(M1) = e_{M1}(T) * max\{\delta_1(beg) * \alpha_{begM1}, \delta_1(IO) * \alpha_{IOM1}\}$
$= 0.1 * max\{0 * 0.7, 0.05 * 0.9]$
$= \mathbf{0.0045}$ $\quad \psi_2(M1) = IO$

$\delta_2(D1) = max\{\delta_1(beg) * \alpha_{begD1}, \delta_1(IO) * \alpha_{IOD1}\}$
$= max\{0 * 0.1, 0.05 * 0.1\}$
$= \mathbf{0.005}$ $\quad \psi_2(D1) = IO$

$\delta_3(I1) = e_{I1}(C) * max\{\delta_2(M1) * \alpha_{M1I1}, \delta_2(D1) * \alpha_{D1I1}\}$
$= 0.25 * max\{0.0045 * 0.27, 0.005 * 0.3\}$
$= 0.25 * max\{0.001215, 0.0015\}$
$= \mathbf{0.000375}$ $\quad \psi_3(I1) = D1$

Figure 3-6: An example of calculations from the foward and Viterbi algorithms

$$\delta_t(i) = \max_{1 \le i \le N} \{\delta_{t-1}(i) a_{ij}\} * e_j(o_t)$$

$$\psi_t(i) = \arg \max_{1 \le i \le N} \{\delta_{t-1}(i) a_{ij}\}$$

Finally, the best path probability and the last state in the best path determined at $t = T$ is

$$Pr(\text{best path}) = \max_{1 \le i \le N} \{\delta_T(i)\}$$

and

$$\psi_T^{BEST} = \arg \max_{1 \le i \le N} \{\delta_T(i)\}$$

In order determine the actual sequence of hidden states in the best path, use $\psi_t(i)$ to find the state which led to ψ_T^{BEST} and continue to follow it backwards in a similar manner until $t = 1$

An example of Viterbi algorithm calculations is displayed in Figure 3-6b).

3.5.4 Expectation Maximization Method

The Expectation Maximization (EM) method is an algorithm that trains an HMM to find the best parameters that maximize the average probability of the most likely path of hidden states over all sequences. The EM method starts by initializing the estimates of transition and emission probabilities. If an initial alignment is known, this prior information can be used to calculate the probability estimates. A cycle starts in which the expected number of paths in which there is a transition from state i to state j, and the expected number of paths in which we see the emission of an observation (DNA base) from state j are calculated. These expected values are used to calculate the maximum-likelihood estimates for the transition and emission probabilities. Next, the maximum likelihood estimates are used to calculate the expected values and the cycle starts again. This process is repeated until both of the maximum likelihood estimates and the expected value parameters remain rela-

tively unchanged from one iteration to the next. Usually, this process converges within 10 iterations. Finally, after training the HMM, the Viterbi algorithm is used to find the most likely path for each sequence. These most likely paths for each sequence are then used to construct an MSA.

More specifically, the procedure for the EM method is as follows:

1.) Determine the length of the HMM directly by using an initial alignment or through averaging the length of all sequences.

2.) Initialize the transition and emission probabilities by using prior information from an initial alignment.

3.) Train the model with a subset of the data sequences.

a.) Calculate $f_t(i)$ and $b_t(i)$ for each hidden state using the forward and backward algorithms.

b.) Use the forward algorithm to calculate Pr(O | HMM).

c.) Expectation step

i.) Calculate the expected number of times an observation, O_t, is emitted from state i

$$E_i(o_t) = \frac{f_t(i) * b_t(i)}{\Pr(O|HMM)}$$

and

$$E_i = \sum_{t=1}^{T} E_i(o_t)$$

ii.) Calculate the expected number of times there is a transition from state i to state j

$$A_t(i,j) = \frac{f_t(i) * a_{ij} * e_j(o_{t+1}) * b_{t+1}(i)}{\Pr(O|HMM)}$$

and

$$A(i,j) = \sum_{t=1}^{T-1} \sum_{j} A_t(i,j)$$

d.) Maximization step

i.) Calculate maximum likelihood estimates using calculations from c.)

$$e_i(o_t = k) = \frac{\sum_{t=1}^{T} E_i(o_t=k)}{E_i}$$

ii.) Calculate maximum likelihood estimates using calculations from d.)

$$a_{ij} = \frac{\sum_{t=1}^{T-1} A_t(i,j)}{A(i,j)}$$

e.) Use the estimates from d.) to produce a new version of the HMM.

f.) Repeat steps a)-e) until convergence of the maximum likelihood estimates is reached.

4.) Use the Viterbi algorithm to align all sequences not in the training set.

5.) Build the MSA from the most likely paths of each sequence.

3.6 Modified Tabu Search: Tabu A'

Tabu A' is a modified version of Tabu A. One major difference is that Tabu A' aligns three sequences in the DP algorithm instead of using a pairwise DP algorithm. An extension of Needleman and Wunsch's pairwise DP algorithm was used to align three sequences. The search space for this DP algorithm is reduced by the Carrillo Lipman bound [41, 28, 6].

The mathematical derivation of the Carrillo-Lipman bound is as follows:

Let,

$s_1 ... s_k$- given sequences

A- any alignment,

A^o-an optimal alignment

A^h- a heuristic alignment

$c(A)$-cost for any alignment

$c(A^o)$-cost for an optimal alignment

$c(A^h)$-cost function for a heuristic alignment

$c(A^o_{|i,j})$-cost of the(i,j) projection of the optimal alignment

$c(A^h_{|i,j})$-cost of the(i,j) projection of the heuristic alignment

$d(s_i, s_j)$-cost of the pairwise optimal alignment

So, for any fixed pair of sequences, (p,q), a bound on $c(A^o_{|p,q})$, in terms of all other $d(s_i,s_j)$'s and $c(A^h_{|i,j})$'s can be found by

$$\sum_{(i,j),i<j} [c(A^h_{|i,j}) - c(A^o_{|i,j})] \geq 0$$

The equation can be rearranged to read

$$\{ \sum_{\substack{(i,j),i<j \\ (i,j)\tilde{}=(p,q)}} [c(A^h_{|i,j}) - c(A^o_{|i,j})]\} + c(A^h_{|p,q}) - c(A^o_{|p,q}) \geq 0$$

or

$$\{ \sum_{\substack{(i,j),i<j \\ (i,j)\tilde{}=(p,q)}} [c(A^h_{|i,j}) - c(A^o_{|i,j})]\} + c(A^h_{|p,q}) \geq c(A^o_{|p,q})$$

By the optimality of $d(s_i,s_j)$,

$$c(A^o_{|i,j}) \geq d(s_i,s_j), \text{ for any } i,j$$

Using this inequality,

$$\{ \sum_{\substack{(i,j),i<j \\ (i,j)\tilde{}=(p,q)}} [c(A^h_{|i,j}) - d(s_i,s_j)]\} + c(A^h_{|p,q}) \geq c(A^o_{|p,q}) \text{ for any } p,q$$

Define the upper and lower bound as

$$\text{U}= \sum_{(i,j),i<j} (c(A^h_{|i,j})$$

$$\text{L}= \sum_{(i,j),i<j} d(s_i,s_j)$$

The Carrillo-Lipman bound becomes

Finally,

$$\text{U - L} + d(s_p,s_q) \geq c(A^o_{|p,q}).$$

3.7 Summary

In this chapter, we outlined the components for each of the three tabu searches. Tabu A was the first tabu search. It was designed to determine an initial MSA using the SP score. An MSA for Tabu A was determined by initially aligning the first two sequences using DP. Then, sequences were added one by one until the entire MSA was obtained. Whenever the tabu search moved to a new solution, the *entire* MSA was determined using DP and progressively adding on each sequence. Tabu A used the DP procedure to align portions of the MSA that were and were not affected by the move to a new solution. Based upon these results, there was a need to make Tabu A run more efficiently.

Thus, Tabu B was a modified version of Tabu A that reduced the number of computations required to generate an MSA. For the initial solution, Tabu B divided the MSA up into subgroups. The sequences within a subgroup were progressively aligned. So, if we examined the alignment of a single subgroup, this would be identical to how MSAs were aligned in Tabu A. After each subgroup was aligned separately, all the subgroups were aligned together to form an MSA. Following this initial solution, there were various moves that generated neighboring solutions. If a move to a neighboring solution did not involve changing the order of the sequences or gap positions within a particular subgroup, DP was *not* used to realign the sequences within this subgroup. So, Tabu B helped reduce how much the DP procedure was used. Instead of using the DP procedure to align the entire MSA after a move (which was the case with Tabu A), DP was only used to realign sequences within subgroups that were directly affected by moving to a neighboring solution. One drawback to Tabu A and B was that sequences were progressively added to the alignment using only the pairwise DP procedure. Aligning more sequences simultaneously using DP would increase the SP score, by reducing the number of gaps in the final MSA. Thus, Tabu A' was implemented

to improve the overall SP score of Tabu A.

Tabu A' was the third tabu search. This tabu was a modified version of Tabu A that used the DP procedure to align either two or three sequences simultaneously. An MSA for Tabu A' was determined by initially aligning the first three sequences using DP. Subgroups of three sequences were each aligned separately. Then, all the subgroups with three sequences were progressively aligned together to form an MSA.

In the next chapter, the results from the three tabu searches are compared to the results from ClustalW, SAGA and PRALINE.

Chapter 4

Computational Experiments for Tabu A, B and A'

4.1 Overview

In this chapter, we generate the groups of sequences used in our MSAs. These groups vary with regard to the number of sequences and the sequence lengths. We use the SP scoring function to determine the best alignments. Also, we examine the alignment scores and CPU times for Tabu A, Tabu B, Tabu A', ClustalW, SAGA and PRALINE.

4.2 Sequence Generation and Experimental Cases

Sequences evolve over time through a series of nucleotide substitutions at various sequence positions. The simplest model for nucleic acid substitutions is the Jukes-Cantor model. The Jukes-Cantor model assumes that there is the same probability of substitution at any sequence position. Also, once a substitution or mutation has occurred, that sequence po-

sition is just as likely to change again [27]. This model assumes that each nucleotide will eventually have the same frequency in DNA sequences. The only parameter in the Jukes-Cantor model, α, represents the instantaneous rate of change for all nucleotide substitutions. More generally, this model is given by

$$P_{(ii)}(t) = \frac{1}{4} + \frac{3}{4}e^{-4\alpha t}$$

and

$$P_{(ij)}(t) = \frac{1}{4} - \frac{1}{4}e^{-4\alpha t}$$

A multiple sequence generation procedure adapted from Shyu *et al.* [38], was used to simulate DNA sequences. This procedure attempts to simulate real biological sequences, with conserved regions that often correspond to important biological functions. The procedure begins by randomly generating an ancestral DNA sequence. A trigonometric function in conjunction with the probability distribution determines whether a nucleotide substitution will occur at a given sequence position. As a result of the trigonometric function being periodic, conserved regions are simulated. This algorithm employs a Markov assumption that substitutions are independent and identically distributed at each sequence position. The procedure randomly generates a number between 0 and 1. If the number is less than a predefined probability density function for a given position, a mutation will not occur; otherwise, the Jukes-Cantor model is used to determine the most probable substitution. DNA sequences are generated so that the total number of base pairs in any sequence is divisible by 3. The specification allows the DNA sequences to easily be converted into protein sequences. Three DNA base pairs make up a codon. There are 64 possible codons that are composed by all the possible combinations of 4 DNA bases in groups of three. There are only 20 amino acids. So, some codons code for the same amino acid. These amino acids are the building blocks for proteins. MSA programs align protein sequences composed of amino acids and/or

Table 4.1: Details about the first four experimental cases

Cases	MSA Progs	Sc Fun	No of Grps	No Seq	Seq Len	No of Runs	Measures
1	TabB, Clus, PRA, SAG	SP	16	20-200	18-201 BP	10	SP score, CPU time
2	TabB+HMM, SAG	SP	8	20-50	18-201 BP	10	SP score
3	TabA, TabB	SP	16	20-200	18-201BP	10	CPU time, No of it
4	TabA, TabA'	SP	24	10-200	9-102 BP	10	SP score

nucleotide sequences composed of DNA base pairs. ClustalW generates MSAs for protein or nucleotide sequences. Tabu A, B and A' only generate MSAs for nucleotide sequences. PRALINE and SAGA only generate MSAs for protein sequences.

In this chapter, there are four different experimental cases. Initially, the Jukes-Cantor model is used to generate 16 groups of sequences. Each group contained 20-200 sequences. These sequences varied in lengths that ranged from 18-201 base pairs. All 16 groups are used in experimental cases 1 and 3. Only the 8 groups with 50 or fewer sequences are used in experimental case 2. The Jukes-Cantor model is used again to generate 24 new groups of sequences for experimental case 4. For each group of sequences, the tabu search algorithms are run 10 times each. Table 4.1 shows the MSA programs, scoring function, total number of groups, number of sequences per group, sequence lengths, number of runs and the measures of evaluation used for each of the experimental cases.

4.3 Tabu B vs. Other MSA Programs

4.3.1 Alignment Score

Experimental case 1 is used to obtain the results in this section. In Table 4.2, the highest alignment score in each group from Tabu B, was compared with the scores from SAGA,

ClustalW, and PRALINE. Also in Table 4.2, the alignment scores are ranked according to the SP alignment score. A 1 represents the best alignment and 4 represents the worst alignment. As a result of using the sum of pairs scoring function, generally, as the number of sequences or base pairs (BP) increases, the alignment score decreases. This decrease in the SP alignment score is the direct result of more gaps with negative gap penalties being introduced into larger MSAs.

MSAs from ClustalW, the most popular commercial alignment program, are used to measure the quality of the alignments from other programs. Quality is measured in terms of the alignment score. Thus, it is assumed that higher alignment scores will yield a higher quality MSA. For all 16 groups, ClustalW yielded MSAs with the highest alignment score. Conversely, in most instances, PRALINE yielded the lowest alignment score. Generally, the Tabu B scores were between the scores from ClustalW and PRALINE. The MSA scores for SAGA were higher than Tabu B for each of the 16 groups. Tabu B scores were better than PRALINE in 12 of 16 groups. There were only 2 instances in which the tabu search produced the worst alignment of all the MSA programs. In a subsequent section, we will show results in which Tabu B combined with an HMM produces comparable results to SAGA.

Tabu B is run 10 times for each of the 16 groups of sequences. Each run does not necessarily yield the best alignment score attained using Tabu B. The minimum, maximum, average, standard deviation and the percentage of times the best score was reached (for 10 SP scores per group) are displayed in Table 4.3. The variability in the alignment scores increases as the number of sequences in the group increases. The percentage of times that the best score is reached ranges between 10 and 50 percent. The lack of a diversification procedure explains the widely varying MSA scores and the low percentage of times the best score is reached. It is clear that adding a diversification procedure would help prevent Tabu

Table 4.2: SP alignment scores from ClustalW, SAGA, PRALINE and Tabu B

	No Seq, Len	**SP Score**		**No Seq, Len**	**SP Score**	
Clus	20 ,18-21	-7.150E+03	1	50,18-21	-1.067E+04	1
SAG	20 ,18-21	-7.150E+03	1	50,18-21	-1.069E+04	2
PRA	20 ,18-21	-7.512E+03	3	50,18-21	-1.074E+04	3
TabB	20 ,18-21	-7.401E+03	2	50,18-21	-1.088E+04	4
Clus	20, 39-51	-1.500E+04	1	50, 39-51	-3.193E+04	1
SAG	20, 39-51	-1.500E+04	1	50, 39-51	-3.196E+04	2
PRA	20, 39-51	-1.773E+04	3	50, 39-51	-3.298E+04	4
TabB	20, 39-51	-1.692E+04	2	50, 39-51	-3.215E+04	3
Clus	20, 75-102	-3.776E+04	1	50, 75-102	-2.040E+04	1
SAG	20, 75-102	-3.786E+04	2	50, 75-102	-2.128E+04	2
PRA	20, 75-102	-3.942E+04	3	50, 75-102	-2.279E+04	4
TabB	20, 75-102	-3.943E+04	4	50, 75-102	-2.278E+04	3
Clus	20, 150-201	-5.901E+04	1	50, 150-201	-7.028E+04	1
SAG	20, 150-201	-5.901E+04	1	50, 150-201	-7.167E+04	2
PRA	20, 150-201	-5.991E+04	3	50, 150-201	-7.391E+04	4
TabB	20, 150-201	-5.947E+04	2	50, 150-201	-7.385E+04	3
Clus	100, 18-21	-1.563E+04	1	200, 18-21	-5.479E+04	1
SAG	100,18-21	-1.564E+04	2	200, 18-21	-5.526E+04	2
PRA	100,18-21	-1.594E+04	4	200, 18-21	-5.701E+04	4
TabB	100,18-21	-1.579E+04	3	200, 18-21	-5.602E+04	3
Clus	100, 39-51	-1.693E+05	1	200, 39-51	-7.428E+05	1
SAG	100, 39-51	-1.736E+05	2	200, 39-51	-7.434E+05	2
PRA	100, 39-51	-1.785E+05	4	200, 39-51	-7.482E+05	4
TabB	100, 39-51	-1.776E+05	3	200, 39-51	-7.474E+05	3
Clus	100, 75-102	-5.919E+05	1	200, 75-102	-1.929E+06	1
SAG	100, 75-102	-5.956E+05	2	200, 75-102	-1.802E+06	2
PRA	100, 75-102	-6.192E+05	4	200, 75-102	-1.809E+06	3
TabB	100, 75-102	-6.179E+05	3	200, 75-102	-1.809E+06	3
Clus	100,150-201	-9.853E+05	1	200, 150-201	-5.240E+06	1
SAG	100,150-201	-9.869E+05	2	200, 150-201	-5.241E+06	2
PRA	100,150-201	-9.920E+05	4	200, 150-201	-5.242E+07	3
TabB	100,150-201	-9.911E+05	3	200, 150-201	-5.242E+07	3

Table 4.3: Measures for the 10 tabu search runs per group using Tabu B

		Tabu B			
No seq, Len	**Min SP**	**Max SP**	**Avg SP**	**St Dev**	**%Max**
20, 18-21	-7.52E+03	-7.40E+03	-7.46E+03	8.70E+01	50
20, 39-51	-1.80E+04	-1.69E+04	-1.75E+04	7.79E+02	40
20, 75-102	-4.14E+04	-3.94E+04	-4.04E+04	1.40E+03	40
20, 150-201	-6.14E+04	-5.95E+04	-6.05E+04	1.39E+03	30
50, 18-21	-1.96E+04	-1.09E+04	-1.52E+04	6.16E+03	30
50, 39-51	-4.12E+04	-3.22E+04	-3.67E+04	6.37E+03	30
50, 75-102	-3.01E+04	-2.28E+04	-2.65E+04	5.19E+03	20
50, 150-201	-8.19E+04	-7.38E+04	-7.79E+04	5.73E+03	20
100, 18-21	-2.31E+04	-1.58E+04	-1.94E+04	5.13E+03	30
100, 39-51	-2.83E+05	-1.78E+05	-2.30E+05	7.43E+04	30
100, 75-102	-6.97E+05	-6.18E+05	-6.58E+05	5.63E+04	40
100, 150-201	-1.44E+06	-9.91E+05	-1.22E+06	3.21E+05	10
200, 18-21	-6.00E+05	-5.60E+04	-3.28E+05	3.85E+05	20
200, 39-51	-9.02E+05	-7.47E+05	-8.25E+05	1.09E+05	20
200, 75-102	-2.75E+06	-1.81E+06	-2.28E+06	6.67E+05	30
200, 150-201	-5.61E+07	-5.24E+07	-5.43E+07	2.60E+06	20

B from cycling back into local optimal solutions.

4.3.2 CPU Time

The average CPU times (in seconds) for ClustalW, PRALINE, SAGA and Tabu B are shown in Table 4.4. Experimental case 1 is used to obtain the results in this section. Tabu A and B were coded in C++. Tabu A' was coded in Matlab. The source code for SAGA was downloaded from http://www.tcoffee.org/Projects_home_page/saga_home_page.html_saga. As for ClustalW and PRALINE, the sequences were entered directly on the webpages http://www.ebi.ac.uk/Tools/clustalw2/index.html/ and http://zeus.cs.vu.nl/programs/pralinewww/, respectively. All MSA programs were run on a 440 MHz SUN Ultra10 machine.

ClustalW has the fastest processing time for each of the 16 groups. PRALINE consistently has the next fastest processing time. With PRALINE, there is a trade-off between

quality of an MSA and CPU time. PRALINE is much faster than Tabu B; however, the MSAs produced by PRALINE are generally not as good. Also, when at least 100 sequences are aligned, SAGA uses the most CPU time but consistently produces high quality alignments (second to only ClustalW). For the 8 groups that contain 20 or 50 sequences, Tabu B takes the most CPU time of all the four alignment programs. Additionally, for the 8 groups that contain 100 or 200 sequences, SAGA takes the most CPU time of all the four alignment programs. Both ClustalW and PRALINE were expected to have a relatively small processing time since these are both commercial packages. Thus, if the actual source code for ClustalW and PRALINE were run on the same machine as the tabu searches, this would be a more adequate way to assess the speed of the programs relative to SAGA and Tabu A, B, A'.

Table 4.5 displays the minimum, maximum, average and standard deviation of the CPU times for the 10 runs of Tabu B. The CPU time is affected by increases in the sequence length and number of sequences in an alignment. Additionally, the starting solution of a tabu search has an impact on the variability of the CPU time. If the tabu search randomly starts at a good solution, then it would take less time for the algorithm to stabilize as compared with a poor starting solution.

Figures 4-3 and 4-4 display the changes in the CPU time as the number of sequences increases. Specifically, in Figure 4-3 there a four different groups of sequences that contain between 18-21 base pairs. For ClustalW, SAGA, PRALINE and Tabu B, Figure 4-3 shows how the CPU time is affected by aligning groups with 20, 50 100 or 200 total sequences that each contain between 18-21 base pairs. The CPU time for SAGA and Tabu B is extremely sensitive to changes in the total number of sequences in an MSA. For instance, the average CPU time for an MSA with 200 sequences that vary in length from 18-21 base pairs is over 22

hours for both of these MSA programs. However, for both of these algorithms, the average CPU time for an MSA with 20 sequences that vary in length from 18-21 base pairs is only around 9-13 minutes.

An increase in the sequence lengths, slightly increases the CPU time for each of the alignment programs. Figures 4-1 and 4-2 display the changes in the CPU time as the lengths of the sequences increase. For the group containing 20 sequences that vary in length from 18-21 base pairs, the Tabu B CPU time is 9.9 minutes. Where as, the Tabu B CPU time is 27.3 minutes for the group containing 20 sequences that vary in length from 150-201 base pairs. The CPU time for ClustalW and PRALINE are not significantly affected by changes in the number of sequences or the sequence lengths in an MSA. Each of the four algorithms would be efficient for aligning sequences with more than 200 base pairs. Because the CPU time for SAGA and Tabu B is extremely sensitive to an increase in the number of sequences in an MSA, these algorithms should ideally be used to align fewer that 200 sequences.

4.3.3 Tabu B with HMM Local Improvement Algorithm vs. SAGA

After Tabu B has met the termination criteria, an HMM is used to further improve the MSA. This local improvement method uses a large amount of computational time. Therefore, the HMM was only used on the 8 groups, that contained 20 or 50 sequences. Experimental case 2 is used to obtain the results in this section. Table 4.6 displays the percentage improvement in the MSA scores for Tabu B with and without the HMM. For each group, the average alignment score for Tabu B with HMM was better than the score obtained from only running Tabu B. A change in the alignment score corresponds to a change in the actual MSA. Thus, combining the tabu algorithm with an HMM has been an effective way to improve the quality

Table 4.4: Average CPU times from ClustalW, SAGA, PRALINE and Tabu B

	No Seq, Len	**CPU Time (sec)**	**No Seq, Len**	**CPU Time (sec)**
Clus	20, 18-21	1.6560E+01	100, 18-21	2.1130E+01
SAG	20, 18-21	5.9469E+02	100, 18-21	6.7138E+04
PRA	20, 18-21	5.2282E+01	100, 18-21	9.1547E+02
TabB	20, 18-21	8.2471E+02	100, 18-21	6.6921E+04
Clus	20, 39-51	1.6760E+01	100, 39-51	2.1870E+01
SAG	20, 39-51	6.1379E+02	100, 39-51	6.6908E+04
PRA	20, 39-51	1.9252E+02	100, 39-51	9.4259E+03
TabB	20, 39-51	9.3405E+02	100, 39-51	6.6485E+04
Clus	20, 75-102	1.7480E+01	100, 75-102	2.2100E+01
SAG	20, 75-102	9.2731E+02	100, 75-102	6.9336E+04
PRA	20, 75-102	2.0379E+02	100, 75-102	1.0370E+03
TabB	20, 75-102	1.1917E+03	100, 75-102	6.7170E+04
Clus	20, 150-201	1.7710E+01	100, 150-201	2.4420E+01
SAG	20, 150-201	1.0796E+03	100, 150-201	7.1002E+04
PRA	20, 150-201	3.2482E+02	100, 150-201	1.1045E+03
TabB	20, 150-201	1.6351E+03	100, 150-201	6.9893E+04
Clus	50, 18-21	2.0200E+01	200, 18-21	2.2730E+01
SAG	50, 18-21	1.9524E+03	200, 18-21	8.8215E+04
PRA	50, 18-21	6.1844E+02	200, 18-21	1.3908E+03
TabB	50, 18-21	4.4159E+04	200, 18-21	7.9964E+04
Clus	50, 39-51	2.0230E+01	200, 39-51	2.3150E+01
SAG	50, 39-51	2.0148E+03	200, 39-51	8.3958E+04
PRA	50, 39-51	6.0957E+02	200, 39-51	1.4103E+03
TabB	50, 39-51	4.5236E+04	200, 39-51	8.0453E+04
Clus	50, 75-102	2.0340E+01	200, 75-102	2.5430E+01
SAG	50, 75-102	2.1775E+03	200, 75-102	9.0125E+04
PRA	50, 75-102	7.1381E+02	200, 75-102	1.4947E+03
TabB	50, 75-102	4.9929E+04	200, 75-102	8.2833E+04
Clus	50, 150-201	2.1160E+01	200, 150-201	6.7890E+01
SAG	50, 150-201	2.3025E+03	200, 150-201	9.1833E+04
PRA	50, 150-201	7.1665E+02	200, 150-201	1.6056E+03
TabB	50, 150-201	5.4116E+04	200, 150-201	8.9526E+04

Table 4.5: CPU time measures for Tabu B

		Tabu B		
No seq, Len	**Min CPU**	**Max CPU**	**Avg CPU Time (sec)**	**St Dev**
20, 18-21	4.13E+02	9.97E+02	8.25E+02	2.51E+02
20, 39-51	5.03E+02	1.10E+03	9.34E+02	2.58E+02
20, 75-102	5.38E+02	2.96E+03	1.19E+03	1.06E+03
20, 150-201	7.48E+02	2.81E+03	1.64E+03	8.47E+02
50, 18-21	9.92E+02	4.68E+03	4.42E+04	2.39E+04
50, 39-51	1.42E+03	4.74E+04	4.52E+04	2.59E+04
50, 75-102	2.00E+03	5.72E+04	4.99E+04	3.00E+04
50, 150-201	1.94E+04	6.65E+04	5.41E+04	2.44E+04
100, 18-21	3.84E+04	8.92E+04	6.69E+04	2.52E+04
100, 39-51	3.58E+04	5.83E+04	6.65E+04	9.39E+03
100, 75-102	4.82E+04	6.91E+04	6.72E+04	1.08E+04
100, 150-201	4.58E+04	7.51E+04	6.99E+04	1.28E+04
200, 18-21	6.98E+04	1.17E+05	8.00E+04	2.03E+04
200, 39-51	7.72E+04	1.20E+05	8.05E+04	2.15E+04
200, 75-102	8.18E+04	1.46E+05	8.28E+04	2.99E+04
200, 150-201	7.80E+04	2.92E+05	8.95E+04	1.08E+05

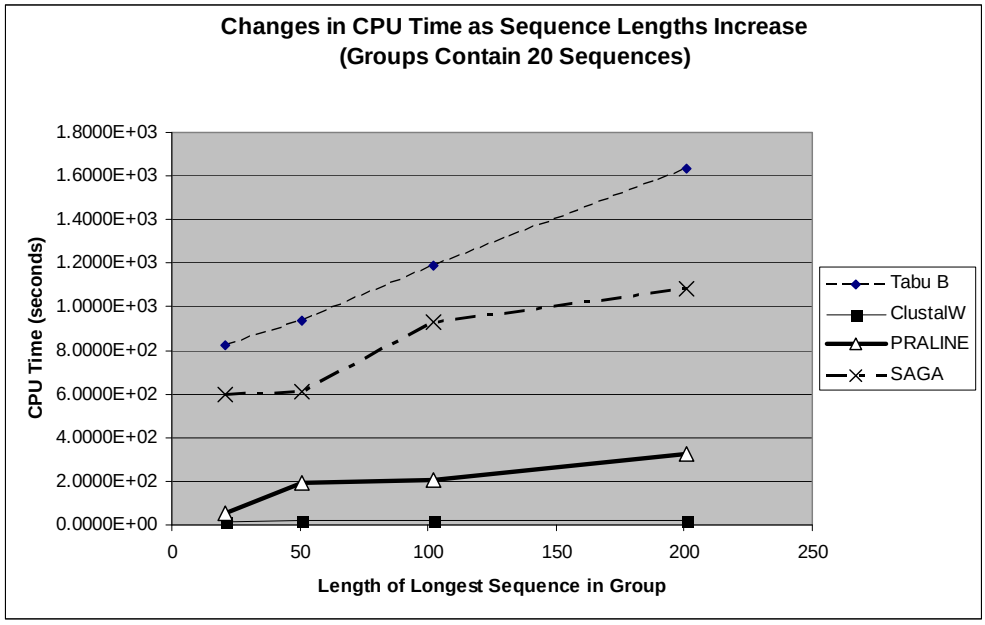


Figure 4-1: CPU time vs. Sequence length in MSAs with 20 sequences

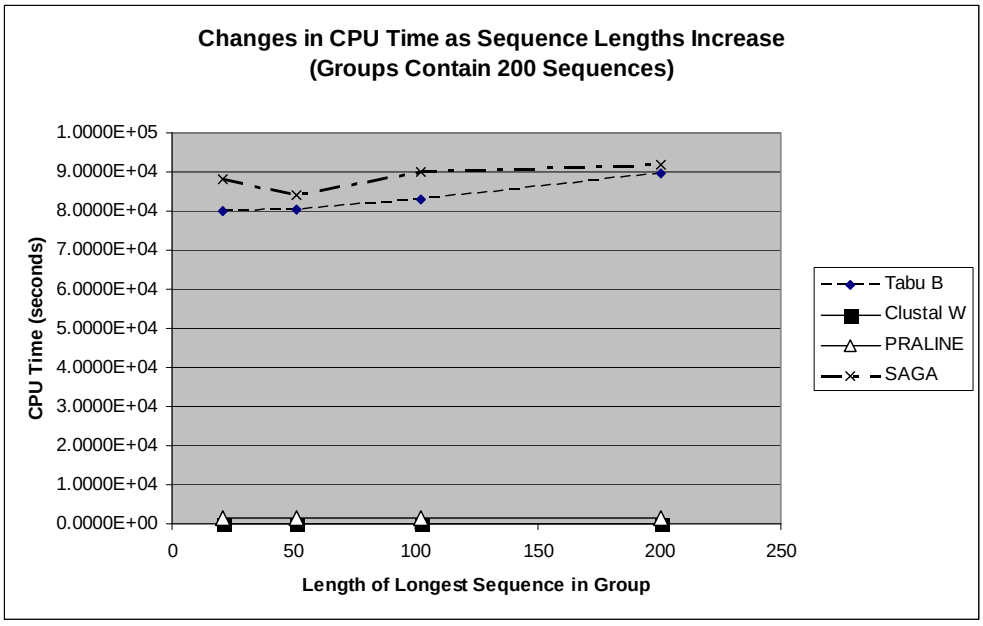


Figure 4-2: CPU time vs. Sequence length in MSAs with 200 sequences

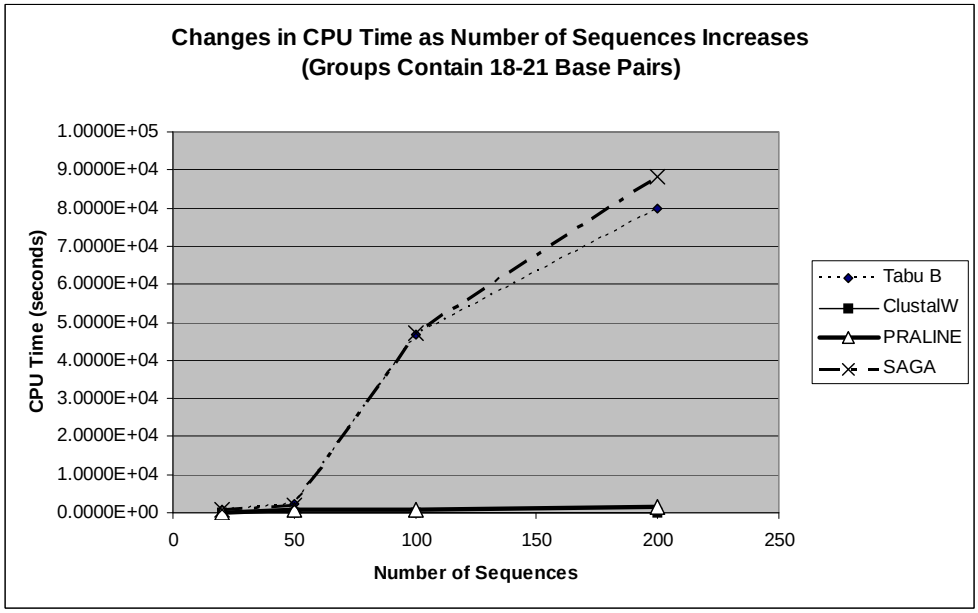


Figure 4-3: CPU time vs. Number of sequences in an MSA (with 18-21 BP)

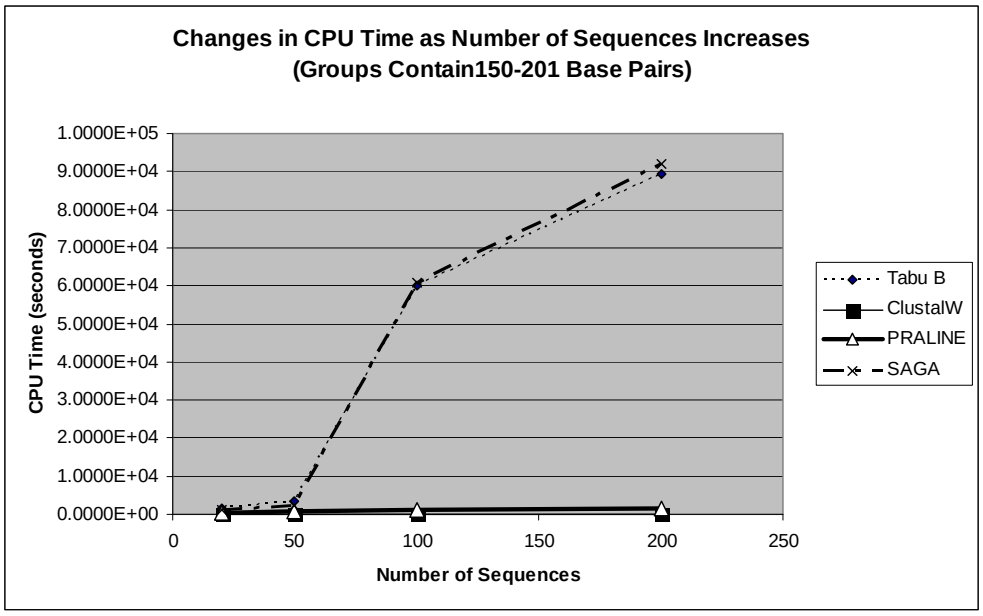


Figure 4-4: CPU time vs. Number of sequences in an MSA(with 150-201 BP)

of an MSA containing fewer than 50 sequences.

Table 4.7 displays the best MSA scores from Tabu B with HMM and SAGA. For the 8 groups with 20 or 50 sequences, the scores from Tabu B with HMM are displayed. Tabu B with HMM has higher alignment scores in 4 of the 8 groups. However, before the addition of an HMM to the tabu search, Tabu B always had lower alignment scores than SAGA. With the addition of this local improvement algorithm, Tabu B now produces better MSAs for all 4 groups with 50 sequences Also, with the group that contains 50 sequences that vary in length from 18-21 base pairs, Tabu B with HMM does as well as ClustalW. However, the SP scores for SAGA were higher than those from Tabu B with HMM for the groups that had 20 sequences These improvements in the alignment score with the addition of this HMM makes this algorithm a viable option for multiple sequence alignment. Table 4.8 displays the minimum, maximum, average, standard deviation and percentage of times the

Table 4.6: Improvement in MSA score for Tabu B with HMM

No Seq, Len	**% Improvement in SP Score**
20, 18-21	0.986
20, 39-51	5.869
20, 75-102	2.483
20, 150-201	0.3665
50, 18-21	1.961
50, 39-51	0.6532
50, 75-102	7.659
50, 150-201	3.025

Table 4.7: MSA scores for the Tabu B with HMM and SAGA

No Seq, Len	**Tabu B + HMM**	**SAGA Score**
20, 18-21	-7.3280E+03	-7.1496E+03
20, 39-51	-1.5927E+04	-1.5000E+04
20, 75-102	-3.8451E+04	-3.7858E+04
20, 150-201	-5.9252E+04	-5.9009E+04
50, 18-21	-1.0667E+04	-1.0689E+04
50, 39-51	-3.1940E+04	-3.1960E+04
50, 75-102	-2.1039E+04	-2.1281E+04
50, 150-201	-7.1612E+04	-7.1668E+04

Table 4.8: Measures for the 10 tabu search runs per group using Tabu B + HMM

	Tabu B + HMM				
No Seq, Len	**Min SP**	**Max SP**	**Avg SP**	**St Dev**	**% Max**
20, 18-21	-7.98E+03	-7.32E+03	-7.14E+03	3.29E+02	50
20, 39-51	-1.71E+04	-1.59E+04	-1.50E+04	6.39E+02	60
20, 75-102	-4.61E+04	-3.84E+04	-3.78E+04	3.86E+03	50
20, 150-201	-6.81E+04	-5.92E+04	-5.90E+04	4.45E+03	40
50, 18-21	-1.72E+04	-1.06E+04	-1.06E+04	3.41E+03	20
50, 39-51	-3.91E+04	-3.19E+04	-3.19E+04	3.72E+03	60
50, 75-102	-2.51E+04	-2.10E+04	-2.12E+04	2.23E+03	30
50, 150-201	-8.35E+04	-7.16E+04	-7.16E+04	6.18E+03	40

best SP score is reached for the 10 runs. By comparison of the percentage maximum scores, the maximum score is reached more frequently with the Tabu B + HMM than with only Tabu B. However, although the addition of the HMM did offer some improvement with the consistency of reaching the maximum score, the highest percentage maximum value is still only 60 percent. Similar to the results for Tabu B, the range of the SP scores increases with the number of sequences in the MSA.

4.4 Tabu A vs. Tabu B

Tabu A is more computationally time consuming than Tabu B. After a move is made to a new current solution, both algorithms use DP approaches to determine the MSA that corresponds to this new solution. In both algorithms, DP is used to align either two sequences or a sequence and subalignment. Additionally, in Tabu B, DP is used to align two subalignments. The time complexity of computing alignments between two sequences, a sequence and a sub-alignment or two sub-alignments is given in Figure 4-5. When aligning a group of 100 sequences (i.e. N =100) or more using Tabu A, it becomes computationally expensive to compute N-2 alignments involving a sequence and subalignment. However, by

Alignments	Time Complexity	Parameters
2 Sequences	$O(l_1 l_2)$	l_1, l_2- Lengths of sequences
A Sequence and a Subalignment	$O(4l_1 l_2 + l_2 n)$	l_1- Length of sequence l_2- Length of subalignment n- Number of sequences in subalignment
2 Subalignments	$O(16 l_1 l_2 + l_1 n_1 + l_2 n_2)$	l_1- Length of first subalignment n_1- Number of sequences in first subalignment l_2- Length of second subalignment n_2- Number of sequences in second subalignment

Figure 4-5: Time complexity for DP algorithms

simply dividing the N sequences into g_i groups and using Tabu B, the computational time is drastically reduced. Additionally, the CPU times for the Tabu B are less than the CPU times for Tabu A. Table 4.9, shows the difference in the CPU times for Tabu A and Tabu B. Experimental case 3 is used to obtain these results. The maximum alignment score for each of the 16 groups is identical for both algorithms. Since the alignment scores were the same and the CPU times were drastically different, only the results obtained from Tabu B were given in the previous sections.

For each of the 16 groups, the average number of total iterations to reach the maximum score was calculated. These values are displayed graphically in Figures 4-6 and 4-7. In Figure 4-6, the graph results are from MSAs containing 20, 50, 100 or 200 sequences. Specifically, Figure 4-6 displays the results from running the tabu searches on the 4 groups that contained exactly 20 sequences. For these 4 groups, the sequence lengths vary between 18-21 base pairs, 39-51 base pairs, 75-102 base pairs or 151-201 base pairs. In most cases, the total number of iterations slightly increases as the length of the sequences increases. Similarly, to

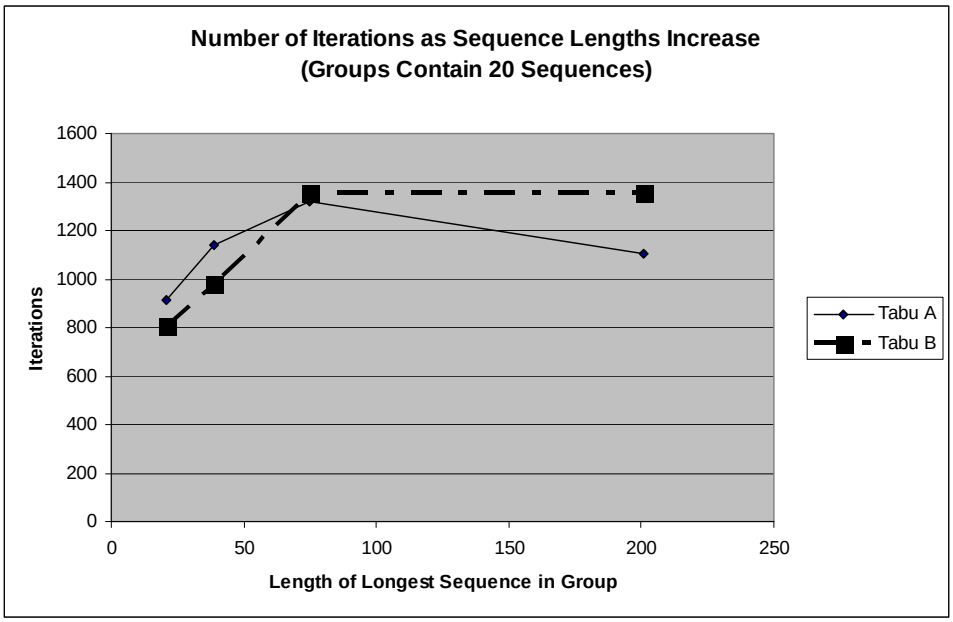


Figure 4-6: Iterations vs. Sequence lengths in MSAs with 20 sequences

the CPU times in the previous section, the total number of iterations is extremely sensitive to an increasing number of sequences in the MSA. In Figures 4-8 and 4-9, each graph displays how the total number of iterations varies with the number of sequences in the MSA.

4.5 Tabu A vs. Tabu A'

The type of DP algorithm used is one factor that differentiates Tabu A from Tabu A'. Only a pairwise DP algorithm is used in Tabu A, whereas Tabu A' generally aligns three sequences using DP. When the total number of sequences in the MSA is not divisible by 3 in Tabu A', the pairwise DP algorithm is used to align the remaining 1 or 2 sequences at the end. Whereas Tabu A progressively aligns sequences one at a time, Tabu A' progressively aligns 3 sequences at a time. Table 4.10 shows the SP scores for 24 different groups of sequences. Experimental case 4 is used to obtain these results. Tabu A produces higher

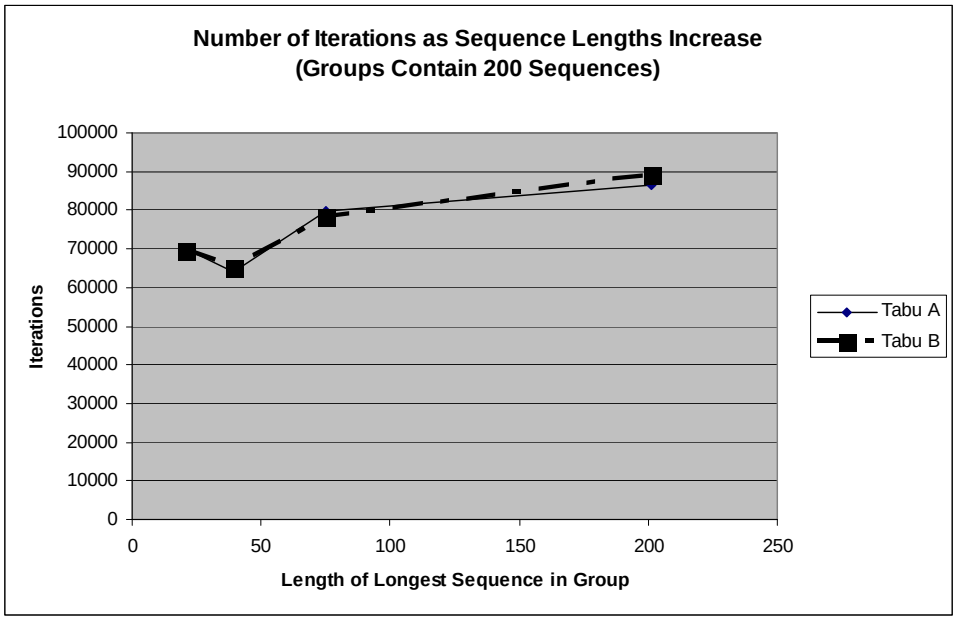


Figure 4-7: Iterations vs. Sequence lengths in MSAs with 200 sequences

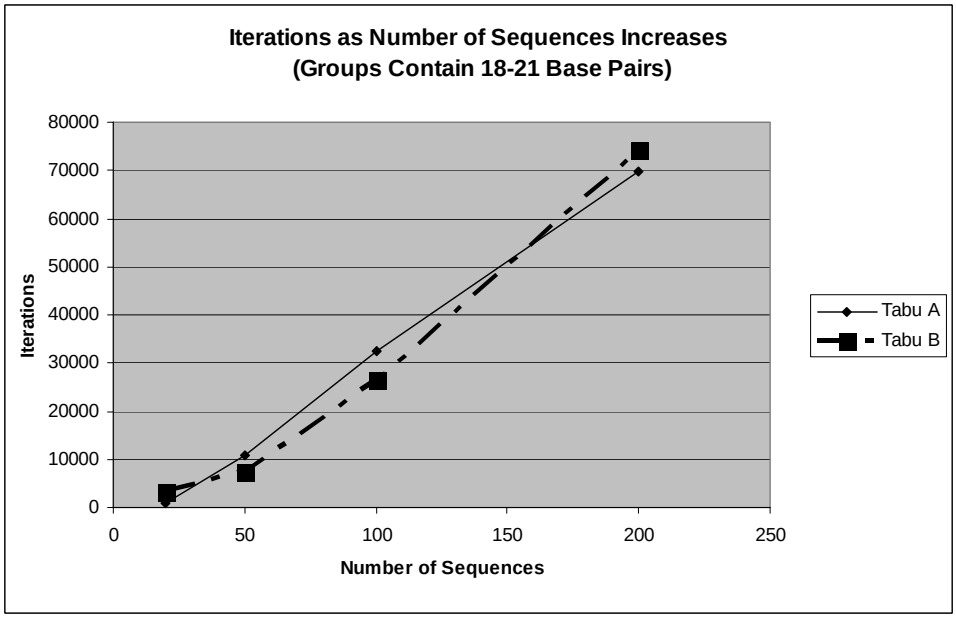


Figure 4-8: Iterations vs. Number of sequences in an MSA (with 18-21 BP)

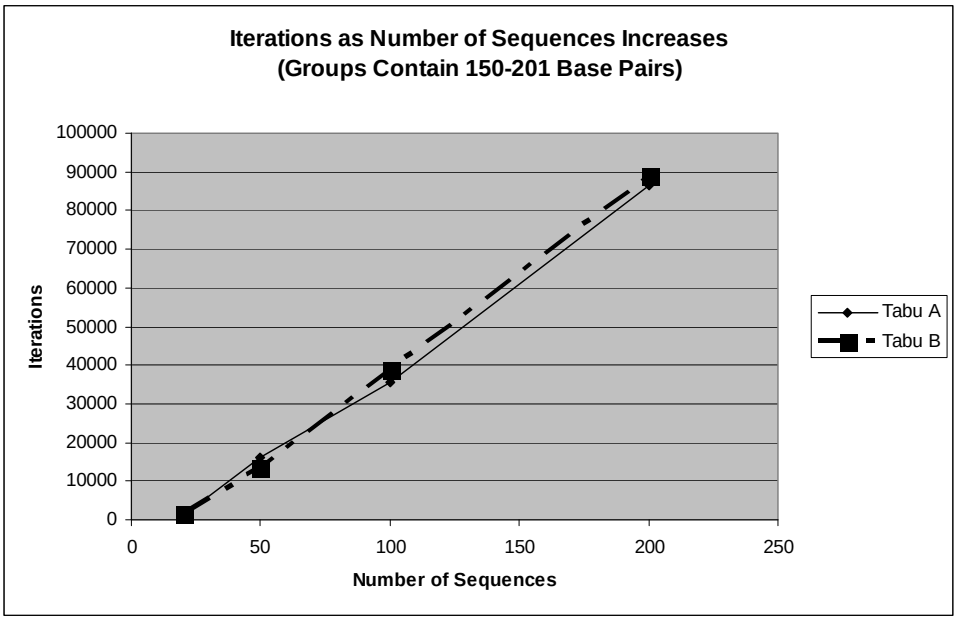


Figure 4-9: Iterations vs. Number.of sequences in an MSA (with 150-201 BP)

Table 4.9: Percentage Improvement in the CPU times between Tabu A and Tabu B

No Seq, Len	**% Improvement in CPU Time**	**No Seq, Len**	**% Improvement in CPU Time**
20, 18-21	30.14	100, 18-21	2.36
20, 39-51	28.80	100, 39-51	3.92
20, 75-102	16.77	100, 75-102	3.99
20, 150-201	5.22	100, 150-201	3.72
50, 18-21	5.89	200, 18-21	3.07
50, 39-51	11.03	200, 39-51	3.42
50, 75-102	27.90	200, 75-102	3.11
50, 150-201	32.38	200, 150-201	2.02

quality alignments for groups with 20 or fewer sequences. However, for more than 20 sequences, Tabu A' consistently has the highest alignment scores. In general, progressively aligning 3 sequences will be less productive than a DP algorithm that aligns all 3 sequences at one time. Progressively aligning 3 sequences will introduce more gaps into the alignment. Although Tabu A and Tabu A' move gaps around in the alignment, the programs do not completely remove unnecessary gaps. The only way to exclude the extra gaps from the SP score is if there is an entire column of gaps.

Table 4.11 shows the minimum, maximum, average, standard deviation and percentage of times the best MSA score was reached. Since each tabu search was run 10 times for each group of sequences, Max % =60% (row 2, column 7) in Table 4.11 means that 6 out of 10 runs ended with an SP score of -812 (for Tabu A' with 10 sequences of lengths varying from 9-15 BP). Similar to the results for Tabu B, Tabu A and A' do not always reach the best solution. Also, the standard deviation of the SP scores increases as the total number of sequences in an MSA increases.

4.6 Summary

When comparing Tabu A to Tabu B and Tabu A', in most instances, both of the modified tabu searches yielded either a higher alignment score or lower CPU time. The SP scores for Tabu B with HMM yielded results that were better than PRALINE and comparable to SAGA. However, despite improvements in the modified algorithms, ClustalW maintained the highest SP scores for each group of sequences. In order to make the tabu search approach comparable to ClustalW, an improved tabu search is developed that uses the best components of Tabu A, Tabu B and Tabu A'. In the next chapter, an improved tabu search is described that generates MSAs that are better than those from the 3 previous tabu search

Table 4.10: SP scores from Tabu A and Tabu A'

	No Seq, Len	**SP Score**	**No Seq, Len**	**SP Score**
TabA'	10, 9-15	-8.1248E+02	100, 9-15	-1.5301E+05
TabA	10, 9-15	-8.0794E+02	100, 9-15	-1.5302E+05
TabA'	10, 21-30	-9.3789E+02	100, 21-30	-2.8963E+05
TabA	10, 21-30	-9.2461E+02	100, 21-30	-2.9277E+05
TabA'	10, 39-51	-1.0326E+03	100, 39-51	-4.17220E+05
TabA	10, 39-51	-9.9172E+02	100, 39-51	-4.19431E+05
TabA'	10, 84-102	-1.2115E+03	100, 84-102	-4.92967E+05
TabA	10, 84-102	-1.1110E+03	100, 84-102	-4.99921E+05
TabA'	20, 9-15	-5.0476E+03	150, 9-15	-5.75614E+05
TabA	20, 9-15	-5.1925E+03	150, 9-15	-5.79312E+05
TabA'	20, 21-30	-5.1226E+03	150, 21-30	-5.50362E+05
TabA	20, 21-30	-5.1399E+03	150, 21-30	-5.52037E+05
TabA'	20, 39-51	-6.9755E+03	150, 39-51	-6.02545E+05
TabA	20, 39-51	-7.0347E+03	150, 39-51	-6.02793E+05
TabA'	20, 84-102	-7.5943E+03	150, 84-102	-7.57131E+05
TabA	20, 84-102	-7.5992E+03	150, 84-102	-7.57218E+05
TabA'	50, 9-15	-4.2832E+04	200, 9-15	-7.92449E+05
TabA	50, 9-15	-4.3004E+04	200, 9-15	-7.98521E+05
TabA'	50, 21-30	-4.5591E+04	200, 21-30	-8.03547E+05
TabA	50, 21-30	-4.5591E+04	200, 21-30	-8.03582E+05
TabA'	50, 39-51	-5.4783E+04	200, 39-51	-8.41492E+05
TabA	50, 39-51	-5.4927E+04	200, 39-51	-8.41502E+05
TabA'	50, 84-102	-5.9423E+04	200, 84-102	-8.35113E+05
TabA	50, 84-102	-5.9773E+04	200, 84-102	-8.35118E+05

Table 4.11: Measures for the 10 tabu search runs per group using Tabu A and Tabu A'

	No seq, Len	Min SP	Max SP	Avg SP	St Dev	% Max
TabA'	10,9-15	-1.39E+03	-8.12E+02	-1.10E+03	4.09E+02	60
TabA	10,9-15	-1.20E+03	-8.08E+02	-1.01E+03	2.80E+02	70
TabA'	10,21-30	-1.59E+03	-9.38E+02	-1.27E+03	4.64E+02	70
TabA	10,21-30	-1.83E+03	-9.25E+02	-1.38E+03	6.41E+02	70
TabA'	10,39-51	-2.01E+03	-1.03E+03	-1.52E+03	6.93E+02	60
TabA	10,39-51	-1.83E+03	-9.92E+02	-1.41E+03	5.94E+02	40
TabA'	10,84-102	-1.73E+03	-1.21E+03	-1.47E+03	3.68E+02	80
TabA	10,84-102	-2.14E+03	-1.11E+03	-1.63E+03	7.29E+02	60
TabA'	20,9-15	-6.15E+03	-5.05E+03	-5.60E+03	7.77E+02	50
TabA	20,9-15	-5.99E+03	-5.19E+03	-5.59E+03	5.66E+02	50
TabA'	20,21-30	-6.48E+03	-5.12E+03	-5.80E+03	9.61E+02	50
TabA	20,21-30	-5.81E+03	-5.14E+03	-5.48E+03	4.75E+02	30
TabA'	20,39-51	-7.87E+03	-6.98E+03	-7.42E+03	6.35E+02	20
TabA	20,39-51	-7.94E+03	-7.03E+03	-7.49E+03	6.42E+02	40
TabA'	20,84-102	-8.22E+03	-7.59E+03	-7.91E+03	4.42E+02	20
TabA	20,84-102	-8.37E+03	-7.60E+03	-7.99E+03	5.46E+02	30
TabA'	50,9-15	-4.92E+04	-4.28E+04	-4.60E+04	4.50E+03	20
TabA	50,9-15	-5.22E+04	-4.30E+04	-4.76E+04	6.49E+03	30
TabA'	50,21-30	-5.48E+04	-4.56E+04	-5.02E+04	6.53E+03	50
TabA	50,21-30	-5.34E+04	-4.56E+04	-4.95E+04	5.51E+03	40
TabA'	50,39-51	-6.48E+04	-5.48E+04	-5.98E+04	7.07E+03	50
TabA	50,39-51	-6.41E+04	-5.49E+04	-5.95E+04	6.49E+03	20
TabA'	50,84-102	-6.99E+04	-5.94E+04	-6.47E+04	7.43E+03	20
TabA	50,84-102	-6.95E+04	-5.98E+04	-6.46E+04	6.88E+03	10
TabA'	100,9-15	-1.75E+05	-1.53E+05	-1.64E+05	1.52E+04	30
TabA	100,9-15	-1.74E+05	-1.53E+05	-1.64E+05	1.51E+04	30
TabA'	100,21-30	-3.58E+05	-2.90E+05	-3.24E+05	4.81E+04	30
TabA	100,21-30	-3.57E+05	-2.93E+05	-3.25E+05	4.58E+04	40
TabA'	100,39-51	-4.47E+05	-4.17E+05	-4.32E+05	2.13E+04	20
TabA	100,39-51	-4.65E+05	-4.19E+05	-4.42E+05	3.23E+04	20
TabA'	100,84-102	-5.17E+05	-4.93E+05	-5.05E+05	1.71E+04	20
TabA	100,84-102	-5.28E+05	-5.00E+05	-5.14E+05	2.02E+04	20
TabA'	200,9-15	-8.15E+05	-7.92E+05	-8.04E+05	1.58E+04	30
TabA	200,9-15	-8.28E+05	-7.99E+05	-8.13E+05	2.11E+04	30
TabA'	200,21-30	-8.32E+05	-8.04E+05	-8.18E+05	2.05E+04	40
TabA	200,21-30	-8.61E+05	-8.04E+05	-8.32E+05	4.06E+04	20
TabA'	200,39-51	-9.29E+05	-8.41E+05	-8.85E+05	6.19E+04	30
TabA	200,39-51	-8.96E+05	-8.42E+05	-8.69E+05	3.86E+04	20
TabA'	200,84-102	-9.39E+05	-8.35E+05	-8.87E+05	7.35E+04	20
TabA	200,84-102	-9.40E+05	-8.35E+05	-8.88E+05	7.44E+04	40

algorithms.

Chapter 5

Improved Tabu Search: Tabu C

5.1 Overview

In this chapter, we outline a new and improved tabu search, Tabu C. Tabu C merges the best features from Tabu A, Tabu B and Tabu A'. Unlike the previous tabu searches, Tabu C uses a tree to guide the alignment process. Also, intensification and diversification procedures are incorporated in Tabu C to further improve the MSA. The use of the parsimony score to assess the quality of the alignment is another distinguishing feature of Tabu C.

5.2 Components of Tabu C

5.2.1 Solution Representation

The solution representation for Tabu C contains both the sequence order and tree topology. Unlike many other progressive alignment programs, this tabu search does not align every pair of sequences in order to determine the MSAs tree structure. Instead, an initial tree is randomly generated and then used to guide the order that sequences are aligned. Whereas

Tabu A and Tabu B progressively align one sequence at a time to a subalignment, Tabu C takes into account closely related sequences by aligning related clusters first. Although it does not depict evolutionary time, this algorithm does indicate which sequences are more closely related. Thus, when the program terminates more similar sequences are in the same branch of the tree and more divergent sequences are in different branches. One advantage of a randomly generated tree is that it reduces the preprocessing time involved with aligning all sequence pairs that the neighbor joining algorithm typically uses to construct a guide tree. Also, allowing a randomly generated tree structure to evolve through the tabu search helps avoid errors in the MSA that are the result of an initially incorrect guide tree.

Tabu C randomly generates the initial tree by starting with only three sequences (leaves) and three inner nodes. Only the sequence number and not the actual sequences (DNA bases) are used in this process. Sequences are added one at a time, by randomly selecting an existing node in the tree that will share the same parent as the newly added sequence. Whether a leaf or an inner node in the existing tree is randomly selected to share a parent node with the newly added sequence, the tree maintains a bifurcating structure. Once all sequences in the MSA are added, this process stops and the complete tree topology is given. The tree topology array consists of two columns. The first column lists the order of all nodes (inner and leaves) in the tree from left to right. The second column lists the parent of each node in the tree. Figure 5-1 displays how sequences are added to the tree to maintain a bifurcating structure and the corresponding solution representation [29, 13].

After the tree is generated, it is used as a guide for the MSA. The pair of sequences on the lowest level are aligned first. Then, the entire branch containing these two sequences is aligned starting from the lowest level and progressing upward to sequences on higher levels. In Figure 5-2a), sequences 5 and 6 are on the lowest level. Sequences 5 and 6 are aligned

first to form the subalignment a(5,6). Sequence 4 and then sequence 3 are added separately to form the subalignment a(3,4,5,6). Next, sequences 1 and 2 are aligned to form the subalignment a(1,2). Following the guide tree, the final step in the MSA is the alignment of a(1,2) and a(3,4,5,6). After the MSA is determined, the alignment is scored. During each iteration, the actual MSA for the current solution is not stored. Only the sequence order, topology and alignment score are maintained from iteration to iteration. The gap positions for the best solution are also stored.

5.2.2 Moving Between Solutions

Tabu C has different types of move strategies. The type of permissible moves depends on the phase of the tabu search. If the algorithm is in the phase of generating a neighborhood from a current solution, there are 3 different types of moves that change the order and/or tree topology. However, if the algorithm is in the intensification phase there are 2 types of moves that change the gaps in the MSA. When generating a neighborhood, the simplest type of move swaps the order of the sequences and maintains the same tree topology. Figure 5-2a)-b) displays an initial solution and neighbor generated by swapping two sequences. The second type of move changes the physical location of a single sequence. In Figure 5-2c), a single sequence move changes the tree topology and maintains the same sequence order. In Figure 5-3a) both the sequence order and tree topology is changed. The third type of move, displayed in Figure 5-3c), changes the physical location of an entire branch. In this case, the sequence order and tree topology are changed. After moving to a neighboring solution, the tabu list is updated to prevent the new current solution from immediately cycling back to an old solution. For example in Figure 5-2c), the tabu list is updated to prevent sequence 2 from sharing a parent with sequence 1 for a number of iterations.

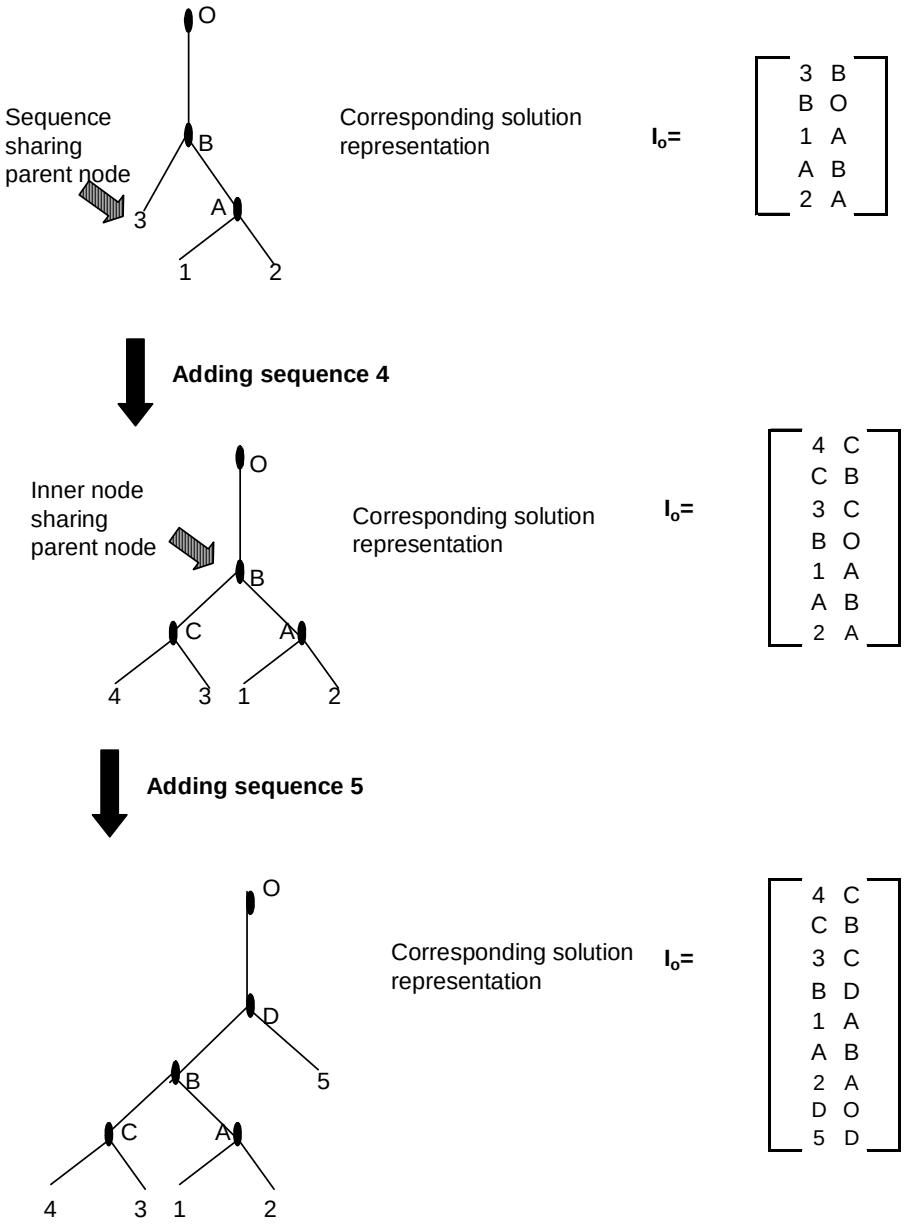
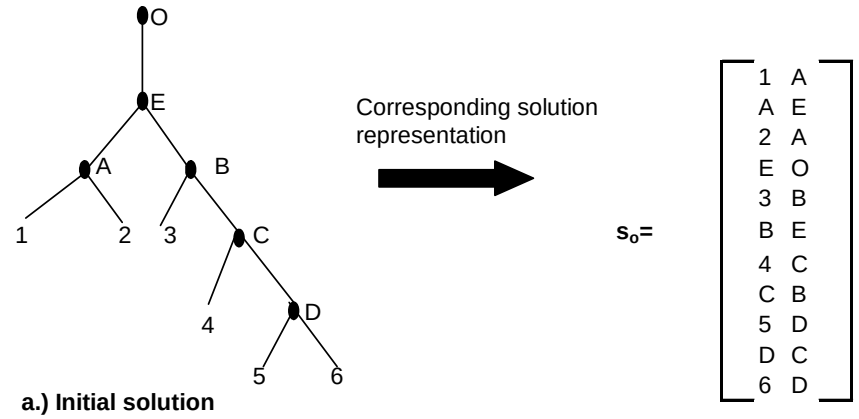
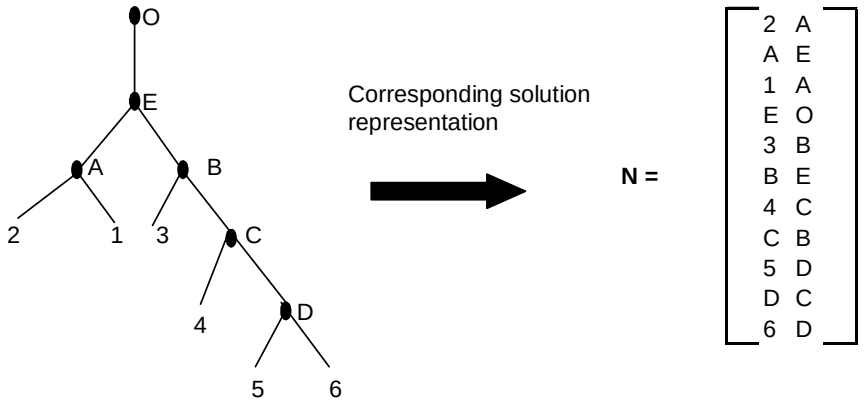


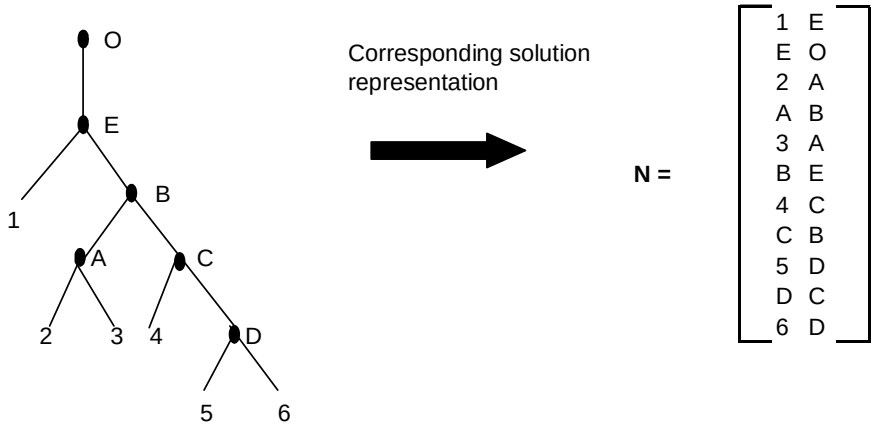
Figure 5-1: Process of generating bifurcating tree and solution representation



a.) Initial solution

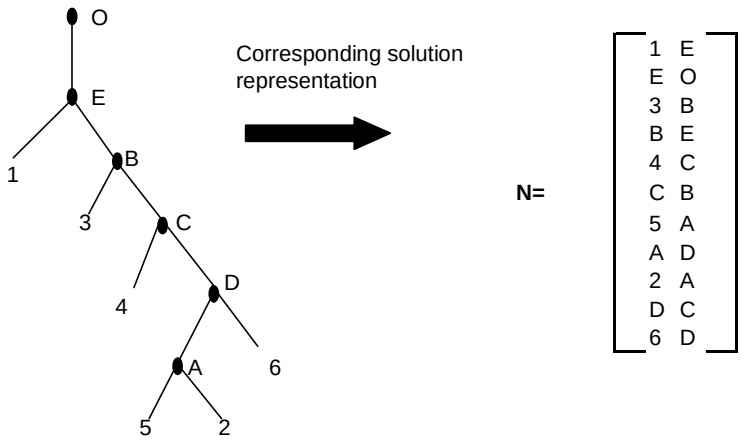


b.) Neighbor generated from initial solution by swapping sequences 1 and 2

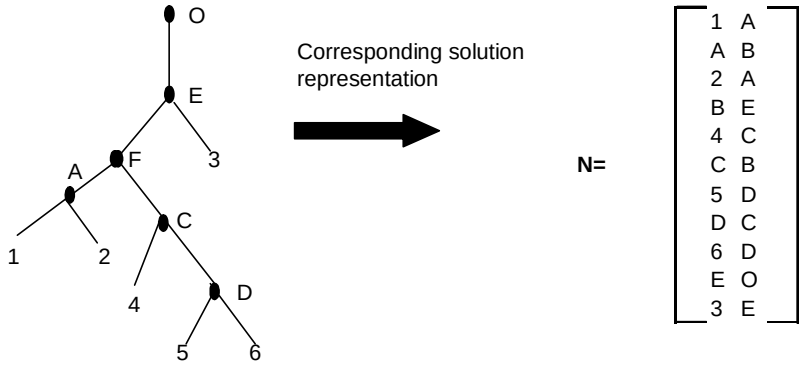


c.) Neighbor generated from initial solution by moving sequence 2 to share parent node of sequence 3

Figure 5-2: Possible moves to create a neighbor from an initial solution



a.) Neighbor generated by moving sequence 2 to share parent node of sequence 5



b.) Neighbor generated by moving branch starting with node C to share parent node of node A

Figure 5-3: Additional moves to create a neighbor from an initial solution

5.2.3 Intensification/ Diversification Components

Tabu C uses an intensification procedure. Whereas Tabu B incorporated an HMM algorithm to locally improve the best solution at the end, Tabu C goes into an intensification phase many times. Generally, an intensification procedure revisits and examines good solutions. It maintains the good portions of this solution and searches to find a better neighboring solution. Tabu C enters an intensification phase when it stabilizes. The tabu is considered stable when a single MSA continues to have the highest score for many iterations. The intensification phase starts with the best MSA. There are two types of moves that will potentially lead to a better solution during this phase. The first type of move swaps individual gap positions in the MSA. The second type of move concentrates on areas that have multiple columns with gaps in over half of the sequences. Either the gaps are removed and the sequences are adjusted or the gaps are simply randomly moved around within that area. This intensification process stops when there is no improvement for many iterations. The intensification phase iteratively refines the best MSA, so that gaps introduced early in the alignment can be removed or switched around. A solution can only enter an intensification phase one time.

A diversification procedure is also implemented in Tabu C that is not used in Tabu A or B. Generally, a diversification procedure forces the search into previously unexplored areas of the search space. It also helps prevent the search from being caught in a local optimal solution [15]. In Tabu C, the diversification phase begins after an MSA has completed the intensification phase and this solution continues to have the highest alignment score following many iterations. In the diversification phase, a new MSA is generated that is *not* in the neighborhood of the current solution. The diversification phase ends by assigning the current solution the value of the newly generated MSA. After the diversification phase, if

a new best solution is not found following a specified number of iterations, the tabu search terminates.

5.2.4 Scoring an MSA

The SP scoring function is used to evaluate MSAs because it is the simplest way of comparing pairs of sequences. This scoring function penalizes gaps within an alignment (with a negative cost) and rewards matches between base pairs (with a positive cost). Thus, the best alignments have fewer gaps and more columns with matching base pairs. While the SP scoring function does distinguish between good and poor MSAs, it does not take into account the principles of evolution. The motivation behind using the parsimony scoring function is that we wanted to use a scoring system which incorporated the minimal evolution principle. The minimal evolution principle maintains that simpler explanations of evolution are preferable to more complicated and ad hoc explanations [14]. If we apply this principle to sequence alignment, the simplest way that a sequence can evolve would be through a minimal number of changes. The parsimony score is the number of changes or mutations along the evolutionary tree. So, minimizing the parsimony score will maintain certain principles of evolution. The parsimony scoring functions is computed with regard to each column. Thus, the parsimony score does not take into account poorly aligned regions that span across several columns. In the case of poorly aligned regions that have a large number of gaps and mismatches, the SP score would adversely be affected. Since using the SP score or the parsimony score alone has drawbacks, a combination of both scores would provide a better measure for the quality of an MSA.

The sum of pairs (SP) and parsimony methods are used to score the alignments for Tabu C. The SP method used in Tabu C is similar to the one described earlier. It differs because

the SP method used in Tabu C scores gap opening and gap extension penalties differently. When calculating a column score, extending a gap is penalized less than opening a gap. Also, there are many instances in which two MSAs have the same SP score. In Tabu C, a parsimony score is used to beak a tie between two MSAs with the same SP score.

The parsimony is defined as the minimum number of mutations required for a given tree. This scoring system helps show how sequences may have evolved from a common ancestor. Whereas a gap in an SP score calculation is often overestimated, the parsimony score balances this problem. When computing an SP score for a column in an N sequence problem, the parsimony score treats a gap as a single event instead of N separate events. Fitch's algorithm is used to calculate the parsimony score [12]. The outline for Fitch's algorithm for a single column is as follows:

Input: Tree topology and MSA

Define:

N- number of sequences

S_i- set of bases for children nodes

g- gap penalty

m- mismatch penalty

C_i- parsimony score for node i

X_i= DNA base at leaf i

Initialize: C_i=0

T= $2N$-1 (sum of the inner nodes and leaves)

Iterate:

For i=1 to T

If i is a leaf

S_i=X_i

C_i=0

Otherwise

If intersection of S_j and S_k for children nodes j and k

is nonempty

S_i= intersection between S_j and S_k

C_i=0

Otherwise

S_i= union between S_j and S_k

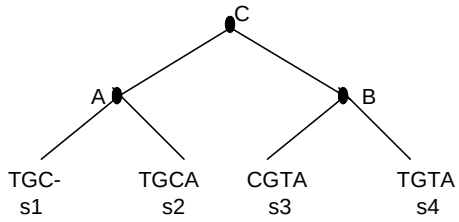
C_i=m or C_i=g

Output: Parsimony score for a column

In Figure 5-4 the SP scores for both MSAs are equal. However, if we assume the mismatch penalty (m) and gap penalty(g) are positive, the first MSA is a better alignment because its parsimony score is the smallest. In Figure 5-5, Fitch's algorithm is used to compute the parsimony scores of MSA 1 and MSA 2. Both of these MSAs have a parsimony score of 2m+3g. To determine the best alignment between MSA 1 and MSA 2, the SP score is calculated. Since most of the columns in both MSAs are identical to each other, only the SP scores for the shaded columns are computed. If we assume that g<m, then MSA 2 would be the best alignment because it has the highest SP score.

5.3 Modifications to the DP Algorithm

Two modifications to the pairwise DP algorithm used in Tabu C are made. The changes eliminate penalties for end gaps and bound the search area of the DP algorithm. The Linear Bounded Diagonal Alignment (LBD-Align) is used to bound the search area of the DP array

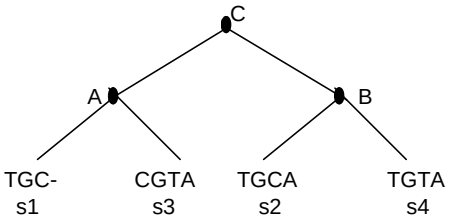


MSA 1

s1	T	G	C	-
s2	T	G	C	A
s3	C	G	T	A
s4	T	G	T	A

	Col 1	Col 2	Col 3	Col 4	**Pars score**
S_i for node A	T	G	C	{-,A)	**-**
C_i for nod A	0	0	0	g	**g**
S_i for node B	{C,T}	G	T	A	**-**
C_i for nod B	m	0	0	0	**m**
S_i for node C	T	G	(C,T)	A	**-**
C_i for nod C	0	0	m	0	**m**

Parsimony score for MSA 1=2m+g



MSA 2

s1	T	G	C	-
s3	C	G	T	A
s2	T	G	C	A
s4	T	G	T	A

	Col 1	Col 2	Col 3	Col 4	**Pars score**
S_i for node A	{T,C)	G	{C,T}	{A,-}	**-**
C_i for nod A	m	0	m	g	**2m+g**
S_i for node B	T	G	{C,T}	A	**-**
C_i for nod B	0	0	m	0	**m**
S_i for node C	T	G	{C,T}	A	**-**
C_i for nod C	0	0	m	0	**m**

Parsimony score for MSA 2=3m+g

Figure 5-4: Parsimony score calculations using Fitch's algorithm

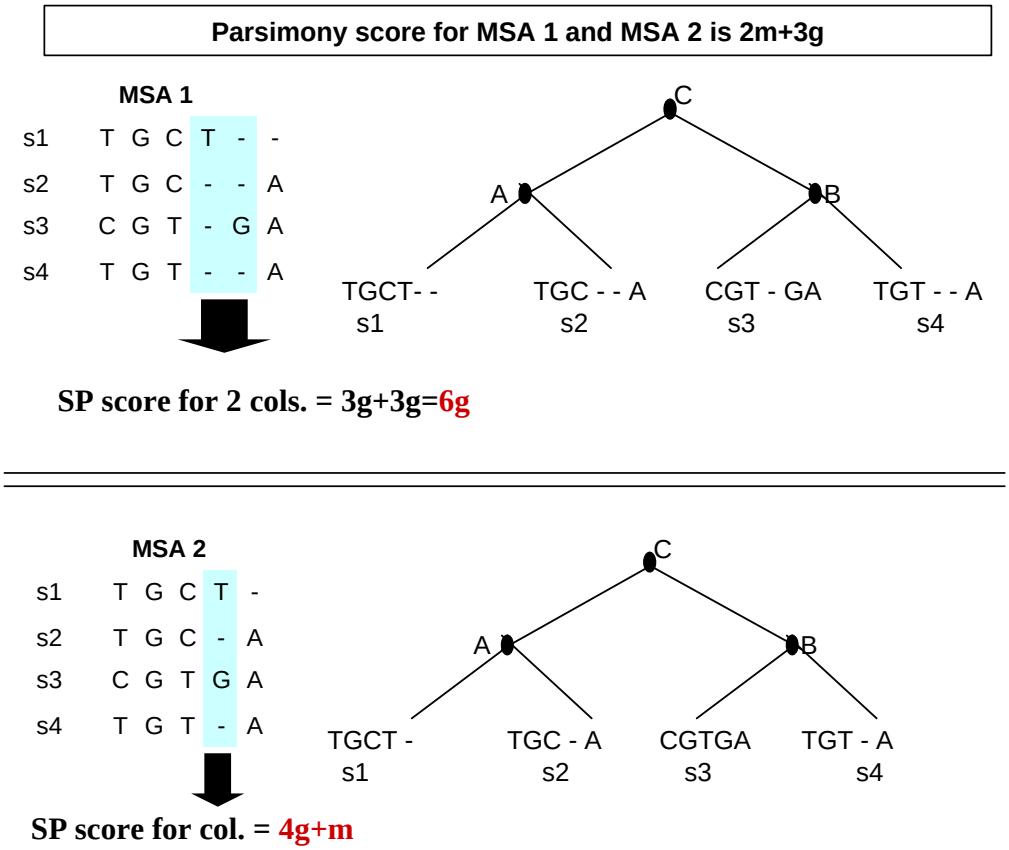


Figure 5-5: SP calculations for two MSAs with the same parsimony score.

[7]. Typically, a pairwise DP alignment for two sequences of lengths N_1 and N_2, requires calculations for each entry in an $(N_1 + 1)$ by $(\ N_2 + 1)$ array. LBD-Align prunes away portions of the search space which do not contain the optimal path by traversing through the DP array along antidiagonals. Only the first and last cells of a diagonal are tested to determine whether the pruning criterion is meet. If the first cell of a diagonal meets the pruning criterion, all cells in the same row and to the right are pruned . If the last cell of the diagonal meets the pruning criterion all cells in the same column and below are pruned. To obtain a lower bound, several small DP arrays are chained together to obtain a possible pairwise alignment. An overly optimistic upper bound is determined at any point in the array by assuming that all remaining bases are perfectly matched with gap adjustments to compensate for sequences of unequal length. If a score in the DP array plus the heuristic upper bound is less than the lower bound, the pruning criterion is met [7, 36].

5.4 Summary

In this chapter, the key components of Tabu C were outlined. Tabu C used and improved upon the best features from Tabu A, Tabu B and Tabu A'. A guide tree, intensification and diversification procedures were new components incorpoated into Tabu C. Also, the solution space in the DP procedure was bounded in Tabu C. In the next chapter, we analyze the alignment scores and CPU times of Tabu B, Tabu A' and Tabu C. The corresponding results from ClustalW, KALIGN, SAGA, PRALINE and PRRN are compared to the results from the tabu searches.

Chapter 6

Computational Experiments for Tabu B, C and A'

6.1 Overview

In this chapter, we present the computational results from Tabu B, Tabu C and Tabu A'. We compare the results from the tabu searches to the corresponding results for ClustalW, KALIGN, SAGA, PRALINE and PRRN. The SP score and CPU time are used in this comparison. In section 6.4, we compare the parsimony scores from ClustalW and Tabu C.

6.2 Sequence Generation and Experimental Cases

The Jukes-Cantor model is used to generate 20 groups of sequences. Each group contained 5-80 sequences that varied in length from 9-102 base pairs. For each group of sequences, Tabu B, Tabu C and Tabu A' are run 10 times each. The same 20 groups of sequences are used for experimental cases 5 and 6. Table 4.1 shows the MSA programs, scoring function,

Table 6.1: Details about experimental cases 5 and 6

Cases	MSA Progs	Sc Fun	No of Grps	No Seq	Seq Len	No of Runs	Measures
5	Clus, KALI, PRA, PRRN, SAG, TabB, TabC, TabA'	SP	20	5-80	9-102 BP	10	SP score, CPU time
6	TabC, Clus	Par	20	5-80	9-102 BP	10	Par score

total number of groups, number of sequences per group, sequence lengths, number of runs and the measures of evaluation used for experimental cases 5 and 6.

During experimental case 5, the SP score is used as the primary measure in determining the quality of MSAs generated from Tabu C. The parsimony score is used as a secondary measure. Thus, if the SP score is tied for two MSAs, the parsimony score is used to break the tie. During experimental case 6, the parsimony score is used as the primary measure for Tabu C. We are searching for the MSA with the lowest parsimony score. Thus, if the parsimony score is tied for two MSAs, the SP score is used as the secondary measure to break the tie.

6.3 SP Scoring Method

6.3.1 Tabu B, C, A' vs. Other MSA Programs

Experimental case 5 is used to obtain the results in this section. For Tabu B, Tabu C and Tabu A', the objective is to maximize the SP function. During experimental case 5, the SP score is used to evaluate the quality of the alignment following the termination of ClustalW, KALIGN, SAGA, PRALINE and PRRN. It is assumed that higher SP scores yield higher quality MSAs. The SP scores for the MSAs generated from ClustalW, KALIGN, PRALINE and PRRN are shown in Table 6.2. Additionally, the rank of the score among all 8 MSA

programs is listed. The SP scores for the MSAs generated from SAGA, Tabu B, Tabu C and Tabu A' are shown in Table 6.3.

Again ClustalW obtained the highest scores for all groups, while SAGA had the highest scores for groups with 20 or fewer sequences. All the alignment programs produced the same MSAs and thus the highest scores for the 4 groups that have 5 sequences. Tabu C also yielded MSAs with the highest alignment scores in the groups with 10 sequences. Tabu C finished second to ClustalW in 6 of 8 groups with 40 or 50 sequences. Figure 6-1 plots the SP scores for the 5 groups that have between 21-30 base pairs. The figure indicates that while SAGA beats Tabu C for problems with 20 or fewer sequences, Tabu C is better fit for problems with over 20 sequences.

In 16 of the 20 groups, Tabu C had higher SP scores than Tabu B and Tabu A'. There were no instances in which Tabu B or Tabu A' had a higher alignment score than Tabu C. Figure 6-2, displays the SP scores from Tabu B, Tabu C and Tabu A' for the 4 groups with 21-30 base pairs. From the plots, Tabu A' appears to perform slightly better than Tabu B. For the groups with 10, 40 and 80 sequences, Tabu A' had a higher SP score than Tabu B. It is clear from this plot that Tabu C is producing better alignments than Tabu B and Tabu A'.

Table 6.4 and 6.5 display the minimum, maximum, average, standard deviation and percentage of times that the best SP score is reached for each group. Tabu C is reaching its best solution in 70-100 percent of the runs. In contrast, Tabu B and Tabu A reach its best solution in 20-60 percent of the runs. In comparison to the other tabu searches, there is less variability in the SP score for Tabu C as the number of sequences increases. Also, the average SP score for Tabu C is greater than the average scores for the other tabu searches.

Table 6.2: SP scores from ClustalW, KALIGN, PRALINE and PRRN

	No Seq, Len	SP Score		No Seq, Len	SP Score	
Clus	5, 9-15	-3.1452E+02	1	20,39-51	-1.5206E+04	1
KALI	5, 9-15	-3.1452E+02	1	20,39-51	-1.5321E+04	3
PRA	5, 9-15	-3.1452E+02	1	20,39-51	-1.5428E+04	4
PRRN	5, 9-15	-3.1452E+02	1	20,39-51	-1.5206E+04	1
Clus	5, 21-30	-2.8874E+02	1	20, 84-102	-1.4340E+04	1
KALI	5, 21-30	-2.8874E+02	1	20, 84-102	-1.4392E+04	2
PRA	5, 21-30	-2.8874E+02	1	20, 84-102	-1.6292E+04	5
PRRN	5, 21-30	-2.8874E+02	1	20, 84-102	-1.4340E+04	1
Clus	5, 39-51	-1.7838E+02	1	40, 9-15	-3.1810E+04	1
KALI	5, 39-51	-1.7838E+02	1	40, 9-15	-3.2913E+04	3
PRA	5, 39-51	-1.7838E+02	1	40, 9-15	-3.5473E+04	6
PRRN	5, 39-51	-1.7838E+02	1	40, 9-15	-3.1810E+04	1
Clus	5, 84-102	-3.6474E+02	1	40, 21-30	-4.3534E+04	1
KALI	5, 84-102	-3.6474E+02	1	40, 21-30	-4.8011E+04	6
PRA	5, 84-102	-3.6474E+02	1	40, 21-30	-4.7997E+04	5
PRRN	5, 84-102	-3.6474E+02	1	40, 21-30	-4.3915E+04	2
Clus	10, 9-15	-1.0708E+03	1	40, 39-51	-3.9919E+04	1
KALI	10, 9-15	-1.1483E+03	4	40, 39-51	-4.1911E+04	3
PRA	10, 9-15	-1.0708E+03	1	40, 39-51	-4.2212E+04	5
PRRN	10, 9-15	-1.1360E+03	3	40, 39-51	-3.9919E+04	1
Clus	10, 21-30	-1.3086E+03	1	40, 84-102	-1.1583E+05	1
KALI	10, 21-30	-1.3086E+03	1	40, 84-102	-1.1670E+05	3
PRA	10, 21-30	-1.5134E+03	5	40, 84-102	-1.2273E+05	7
PRRN	10, 21-30	-1.4108E+03	3	40, 84-102	-1.1773E+05	4
Clus	10, 39-51	-1.1960E+03	1	80, 9-15	-1.2636E+05	1
KALI	10, 39-51	-1.1960E+03	1	80, 9-15	-1.3489E+05	7
PRA	10, 39-51	-1.3522E+03	4	80, 9-15	-1.3199E+05	6
PRRN	10, 39-51	-1.1960E+03	1	80, 9-15	-1.2780E+05	3
Clus	10, 84-102	2.4902E+02	1	80, 21-30	-1.5158E+05	1
KALI	10, 84-102	2.3010E+03	3	80, 21-30	-1.5995E+05	4
PRA	10, 84-102	2.3661E+03	2	80, 21-30	-1.6027E+05	5
PRRN	10, 84-102	2.4902E+02	1	80, 21-30	-1.5199E+05	2
Clus	20, 9-15	-4.5935E+03	1	80, 39-51	-5.4454E+05	1
KALI	20, 9-15	-4.7202E+03	4	80, 39-51	-5.4477E+05	4
PRA	20, 9-15	-4.6998E+03	3	80, 39-51	-5.4486E+05	5
PRRN	20, 9-15	-4.5935E+03	1	80, 39-51	-5.4470E+05	3
Clus	20, 21-30	-6.1211E+03	1	80, 84-102	-5.1982E+05	1
KALI	20, 21-30	-6.3074E+03	5	80, 84-102	-5.2007E+05	4
PRA	20, 21-30	-6.2474E+03	3	80, 84-102	-5.2000E+05	3
PRRN	20, 21-30	-6.1211E+03	1	80, 84-102	-5.2120E+05	5

Table 6.3: SP scores from SAGA, Tabu B, Tabu C and Tabu A'

	No Seq, Len	**SP Score**		**No Seq, Len**	**SP Score**	
SAG	5, 9-15	-3.1452E+02	1	20, 39-51	-1.5206E+04	1
TabB	5, 9-15	-3.1452E+02	1	20, 39-51	-1.5597E+04	6
TabC	5, 9-15	-3.1452E+02	1	20, 39-51	-1.5298E+04	2
TabA'	5, 9-15	-3.1452E+02	1	20, 39-51	-1.5571E+04	5
SAG	5, 21-30	-2.8874E+02	1	20, 84-102	-1.4340E+04	1
TabB	5, 21-30	-2.8874E+02	1	20, 84-102	-1.6908E+04	6
TabC	5, 21-30	-2.8874E+02	1	20, 84-102	-1.4428E+04	3
TabA'	5, 21-30	-2.8874E+02	1	20, 84-102	1.4579E+04	4
SAG	5, 39-51	-1.7838E+02	1	40, 9-15	-3.3781E+04	4
TabB	5, 39-51	-1.7838E+02	1	40, 9-15	-3.4412E+04	5
TabC	5, 39-51	-1.7838E+02	1	40, 9-15	-3.2487E+04	2
TabA'	5, 39-51	-1.7838E+02	1	40, 9-15	-3.4412E+04	5
SAG	5, 84-102	-3.6474E+02	1	40, 21-30	-4.5106E+04	4
TabB	5, 84-102	-3.6474E+02	1	40, 21-30	-4.9237E+04	7
TabC	5, 84-102	-3.6474E+02	1	40, 21-30	-4.4107E+04	3
TabA'	5, 84-102	-3.6474E+02	1	40, 21-30	-4.7997E+04	5
SAG	10, 9-15	-1.0708E+03	1	40, 39-51	-4.2212E+04	5
TabB	10, 9-15	-1.1360E+03	3	40, 39-51	-4.2212E+04	5
TabC	10, 9-15	-1.0708E+03	1	40, 39-51	-4.0912E+04	2
TabA'	10, 9-15	-1.0715E+03	2	40, 39-51	-4.2048E+04	4
SAG	10, 21-30	-1.3086E+03	1	40, 84-102	-1.1862E+05	5
TabB	10, 21-30	-1.4838E+03	4	40, 84-102	-1.1903E+05	6
TabC	10, 21-30	-1.3086E+03	1	40, 84-102	-1.1600E+05	2
TabA'	10, 21-30	-1.3910E+03	2	40, 84-102	-1.1903E+05	6
SAG	10, 39-51	-1.1960E+03	1	80, 9-15	-1.3020E+05	4
TabB	10, 39-51	-1.2443E+03	3	80, 9-15	-1.3199E+05	6
TabC	10, 39-51	-1.1960E+03	1	80, 9-15	-1.2774E+05	2
TabA'	10, 39-51	-1.2140E+03	2	80, 9-15	-1.3101E+05	5
SAG	10, 84-102	2.4902E+02	1	80, 21-30	-1.5742E+05	3
TabB	10, 84-102	2.3010E+03	3	80, 21-30	-1.6127E+05	6
TabC	10, 84-102	2.4902E+02	1	80, 21-30	-1.5242E+05	3
TabA'	10, 84-102	2.2510E+03	4	80, 21-30	-1.6127E+05	6
SAG	20, 9-15	-4.5935E+03	1	80, 39-51	-5.4473E+05	3
TabB	20, 9-15	-4.8311E+03	6	80, 39-51	-5.4486E+05	5
TabC	20, 9-15	-4.6132E+03	2	80, 39-51	-5.4461E+05	2
TabA'	20, 9-15	-4.7790E+03	5	80, 39-51	-5.4486E+05	5
SAG	20, 21-30	-6.1211E+03	1	80, 84-102	-5.2149E+05	7
TabB	20, 21-30	-6.2474E+03	3	80, 84-102	-5.2900E+05	8
TabC	20, 21-30	-6.1390E+03	2	80, 84-102	-5.1991E+05	2
TabA'	20, 21-30	-6.2530E+03	4	80, 84-102	-5.2146E+05	6

Table 6.4: Measures for the 10 tabu search runs per group using Tabu B, Tabu C and Tabu A' (Groups with 5-20 sequences)

	No seq, Len	**Min SP**	**Max SP**	**Avg SP**	**St Dev**	**% Max**
TabB	5, 9-15	-3.63E+02	-3.15E+02	-3.49E+02	2.31E+01	50
TabC	5, 9-15	-3.15E+02	-3.15E+02	-3.15E+02	0.00E+00	100
TabA'	5, 9-15	-3.89E+02	-3.15E+02	-3.66E+02	3.52E+01	60
TabB	5, 21-30	-3.91E+02	-2.89E+02	-3.62E+02	4.89E+01	60
TabC	5, 21-30	-2.89E+02	-2.89E+02	-2.89E+02	0.00E+00	100
TabA'	5, 21-30	-3.76E+02	-2.89E+02	-3.44E+02	3.95E+01	50
TabB	5, 39-51	-2.91E+02	-1.78E+02	-2.60E+02	5.47E+01	60
TabC	5, 39-51	-1.78E+02	-1.78E+02	-1.78E+02	0.00E+00	100
TabA'	5, 39-51	-2.53E+02	-1.78E+02	-2.34E+02	3.70E+01	50
TabB	5, 84-102	-4.21E+02	-3.65E+02	-3.97E+02	2.33E+01	50
TabC	5, 84-102	-3.65E+02	-3.65E+02	-3.65E+02	0.00E+00	100
TabA'	5, 84-102	-4.40E+02	-3.65E+02	-4.10E+02	3.46E+01	60
TabB	10, 9-15	-2.01E+03	-1.14E+03	-1.58E+03	4.84E+02	40
TabC	10, 9-15	-1.08E+03	-1.07E+03	-1.07E+03	5.40E+00	90
TabA'	10, 9-15	-1.97E+03	-1.07E+03	-1.50E+03	4.64E+02	40
TabB	10, 21-30	-2.16E+03	-1.48E+03	-1.85E+03	3.02E+02	40
TabC	10, 21-30	-1.34E+03	-1.31E+03	-1.32E+03	1.21E+01	80
TabA'	10, 21-30	-2.18E+03	-1.39E+03	-1.61E+03	3.84E+02	50
TabB	10, 39-51	-1.96E+03	-1.24E+03	-1.61E+03	3.44E+02	40
TabC	10, 39-51	-1.21E+03	-1.20E+03	-1.20E+03	8.53E+00	90
TabA'	10, 39-51	-2.08E+03	-1.21E+03	-1.72E+03	4.00E+02	40
TabB	10, 84-102	1.16E+03	2.30E+03	1.70E+03	5.57E+02	40
TabC	10, 84-102	2.48E+03	2.49E+03	2.49E+03	6.15E+00	90
TabA'	10, 84-102	1.74E+03	2.25E+03	2.01E+03	2.10E+02	30
TabB	20, 9-15	-5.65E+03	-4.83E+03	-5.13E+03	3.66E+02	20
TabC	20, 9-15	-4.69E+03	-4.61E+03	-4.64E+03	3.55E+01	90
TabA'	20, 9-15	-5.64E+03	-4.78E+03	-5.11E+03	3.84E+02	40
TabB	20, 21-30	-7.21E+03	-6.25E+03	-6.75E+03	4.40E+02	20
TabC	20, 21-30	-6.20E+03	-6.14E+03	-6.16E+03	2.68E+01	90
TabA'	20, 21-30	-7.15E+03	-6.25E+03	-6.82E+03	4.00E+02	30
TabB	20,39-51	-1.69E+04	-1.56E+04	-1.61E+04	5.67E+02	40
TabC	20,39-51	-1.54E+04	-1.53E+04	-1.53E+04	4.11E+01	80
TabA'	20,39-51	-1.67E+04	-1.56E+04	-1.61E+04	4.99E+02	20
TabB	20, 84-102	-1.79E+04	-1.69E+04	-1.73E+04	4.14E+02	50
TabC	20, 84-102	-1.45E+04	-1.44E+04	-1.44E+04	3.11E+01	80
TabA'	20, 84-102	-1.59E+04	-1.46E+04	-1.51E+04	5.56E+02	30

Table 6.5: Measures for the 10 tabu search runs per group using Tabu B, Tabu C and Tabu A' (Groups with 40-80 sequences)

	No seq, Len	**Min SP**	**Max SP**	**Avg SP**	**St Dev**	**% Max**
TabB	40, 9-15	-3.59E+04	-3.44E+04	-3.53E+04	7.18E+02	20
TabC	40, 9-15	-3.26E+04	-3.25E+04	-3.25E+04	3.08E+01	70
TabA'	40, 9-15	-3.62E+04	-3.44E+04	-3.53E+04	9.32E+02	30
TabB	40, 21-30	-5.07E+04	-4.92E+04	-5.01E+04	6.22E+02	40
TabC	40, 21-30	-4.42E+04	-4.41E+04	-4.41E+04	4.15E+01	90
TabA'	40, 21-30	-4.91E+04	-4.80E+04	-4.86E+04	6.12E+02	40
TabB	40, 39-51	-4.48E+04	-4.22E+04	-4.39E+04	1.22E+03	40
TabC	40, 39-51	-4.10E+04	-4.09E+04	-4.10E+04	5.02E+01	80
TabA'	40, 39-51	-4.37E+04	-4.20E+04	-4.28E+04	7.92E+02	30
TabB	40, 84-102	-1.22E+05	-1.19E+05	-1.21E+05	1.36E+03	20
TabC	40, 84-102	-1.16E+05	-1.16E+05	-1.16E+05	6.53E+01	80
TabA'	40, 84-102	-1.23E+05	-1.19E+05	-1.21E+05	1.92E+03	30
TabB	80, 9-15	-1.37E+05	-1.32E+05	-1.35E+05	2.03E+03	30
TabC	80, 9-15	-1.28E+05	-1.28E+05	-1.28E+05	6.30E+01	70
TabA'	80, 9-15	-1.37E+05	-1.31E+05	-1.35E+05	3.08E+03	20
TabB	80, 21-30	-1.68E+05	-1.61E+05	-1.65E+05	2.72E+03	20
TabC	80, 21-30	-1.53E+05	-1.52E+05	-1.53E+05	8.71E+01	80
TabA'	80, 21-30	-1.69E+05	-1.61E+05	-1.66E+05	3.52E+03	30
TabB	80, 39-51	-5.53E+05	-5.45E+05	-5.49E+05	4.69E+03	40
TabC	80, 39-51	-5.45E+05	-5.45E+05	-5.45E+05	7.30E+01	80
TabA'	80, 39-51	-5.52E+05	-5.45E+05	-5.50E+05	3.40E+03	40
TabB	80, 84-102	-5.48E+05	-5.29E+05	-5.41E+05	8.32E+03	30
TabC	80, 84-102	-5.20E+05	-5.20E+05	-5.20E+05	7.78E+01	80
TabA'	80, 84-102	-5.40E+05	-5.21E+05	-5.35E+05	8.96E+03	20

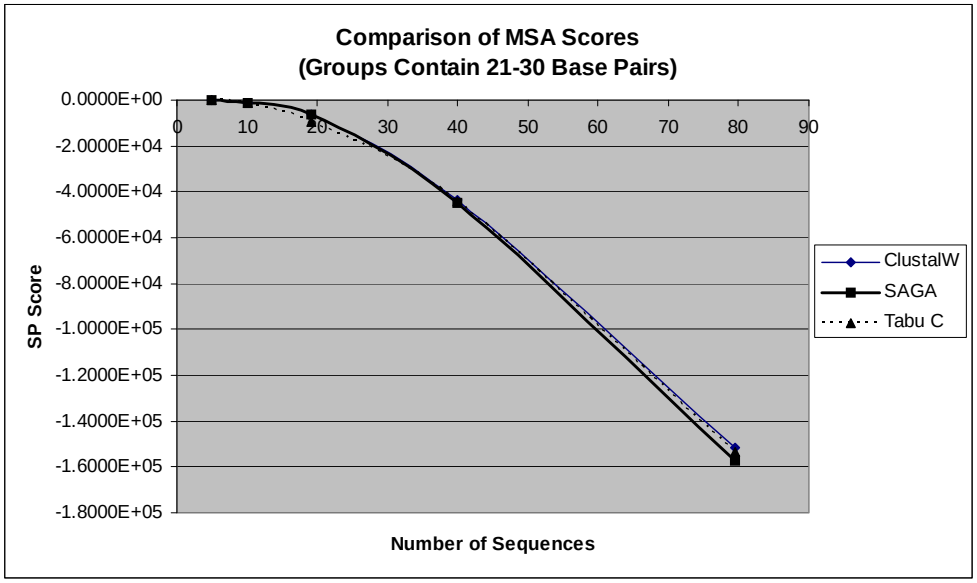


Figure 6-1: Comparison of MSA scores for ClustalW, SAGA and Tabu C

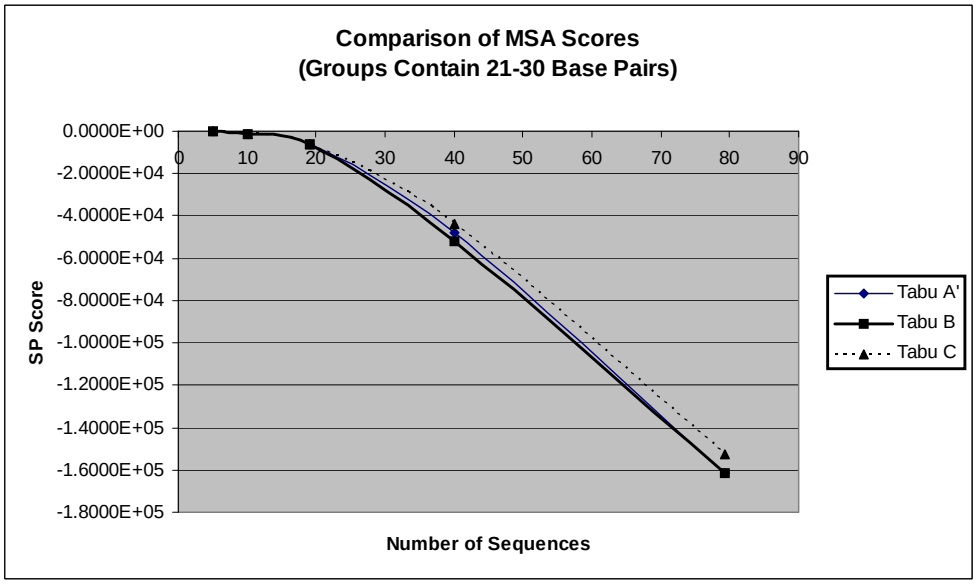


Figure 6-2: Comparison of MSA scores for Tabu B, Tabu C and Tabu A'

6.4 Parsimony Scoring Method

The parsimony scoring method is used to evaluate the quality of an alignment during experimental case 6. While running Tabu C, the objective is to minimize the parsimony score. After ClustalW terminates, the parsimony score is calculated. It is assumed that lower parsimony scores indicate a higher quality alignment. Calculation of the parsimony score requires the use of the tree topology. Tabu C and ClustalW are the only two programs that consistently generate and display the tree topology for each MSA. Thus, only the parsimony scores for the MSAs generated by ClustalW and Tabu C are shown in Table 6.6. For the 4 groups with 5 sequences, ClustalW and Tabu C yield equivalent parsimony scores. When the SP score is used to evaluate the quality of an MSA, there were no instances where Tabu C produced higher quality alignments than ClustalW. However, in 6 of 16 groups, Tabu C yields lower parsimony scores than ClustalW. This is a remarkable improvement over the results yielded with the SP scoring function.

Table 6.7 displays the minimum, maximum, average, standard deviation and percentage of times that the best parsimony score is reached for each group. For the groups with only five sequences, all 10 tabu search runs resulted in the best parsimony score. For the groups that have between 10-80 sequences, the percentage of times the best parsimony score is reached varies between 70-90 percent. The standard deviation of the parsimony scores minimally increases as the number of sequences increases.

6.5 CPU Time

The average CPU time (in seconds) for ClustalW, KALIGN, PRALINE and PRRN are shown in Table 6.8. The average CPU time (in seconds) for SAGA, Tabu B, Tabu C

Table 6.6: Parsimony scores for ClustalW and Tabu C

	No Seq, Len	**Pars score**	**No Seq, Len**	**Pars score**
TabC	5, 9-15	1.0282E+02	20, 39-51	**8.5100E+02**
Clus	5, 9-15	1.0285E+02	20, 39-51	9.2259E+02
TabC	5, 21-30	9.3470E+01	20, 84-102	9.9723E+02
Clus	5, 21-30	9.3470E+01	20, 84-102	8.4254E+02
TabC	5, 39-51	8.6220E+01	40, 9-15	1.2075E+03
Clus	5, 39-51	8.6220E+01	40, 9-15	9.5399E+02
TabC	5, 84-102	1.3481E+02	40, 21-30	**1.9640E+03**
Clus	5, 84-102	1.3481E+02	40, 21-30	2.0168E+03
TabC	10, 9-15	2.3618E+02	40, 39-51	2.2586E+03
Clus	10, 9-15	2.0153E+02	40, 39-51	2.0193E+03
TabC	10, 21-30	**2.8524E+02**	40, 84-102	2.4167E+03
Clus	10, 21-30	2.9441E+02	40, 84-102	2.2605E+03
TabC	10, 39-51	**2.1276E+02**	80, 9-15	**3.1430E+03**
Clus	10, 39-51	2.3952E+02	80, 9-15	3.2197E+03
TabC	10, 84-102	3.9650E+02	80, 21-30	3.4857E+03
Clus	10, 84-102	3.2179E+02	80, 21-30	3.1185E+03
TabC	20, 9-15	**6.4988E+02**	80, 39-51	3.1943E+03
Clus	20, 9-15	6.6252E+02	80, 39-51	2.5186E+03
TabC	20, 21-30	7.8692E+02	80, 84-102	4.8136E+03
Clus	20, 21-30	7.0345E+02	80, 84-102	4.7128E+03

Table 6.7: Measures for the 10 tabu search runs per group using Tabu C

		Tabu C			
No Seq, Len	**Min Pars**	**Max Pars**	**Avg Pars**	**St Dev**	**% Max**
5, 9-15	1.03E+02	1.03E+02	1.03E+02	0.00E+00	100
5, 21-30	9.35E+01	9.35E+01	9.35E+01	0.00E+00	100
5, 39-51	8.62E+01	8.62E+01	8.62E+01	0.00E+00	100
5, 84-102	1.35E+02	1.35E+02	1.35E+02	0.00E+00	100
10, 9-15	2.36E+02	2.51E+02	2.43E+02	6.50E+00	90
10, 21-30	2.85E+02	2.96E+02	2.89E+02	5.25E+00	80
10, 39-51	2.13E+02	2.39E+02	2.21E+02	1.20E+01	90
10, 84-102	3.97E+02	4.25E+02	4.05E+02	1.32E+01	90
20, 9-15	6.50E+02	7.01E+02	6.66E+02	2.36E+01	90
20, 21-30	7.87E+02	8.45E+02	8.05E+02	2.72E+01	70
20, 39-51	8.51E+02	9.41E+02	8.78E+02	4.25E+01	80
20, 84-102	9.97E+02	1.10E+03	1.03E+03	5.23E+01	90
40, 9-15	1.21E+03	1.28E+03	1.23E+03	3.27E+01	80
40, 21-30	1.96E+03	2.06E+03	2.01E+03	4.04E+01	90
40, 39-51	2.26E+03	2.36E+03	2.31E+03	5.24E+01	90
40, 84-102	2.42E+03	2.54E+03	2.47E+03	5.64E+01	70
80, 9-15	3.14E+03	3.27E+03	3.20E+03	5.92E+01	80
80, 21-30	3.49E+03	3.65E+03	3.58E+03	6.97E+01	80
80, 39-51	3.19E+03	3.33E+03	3.24E+03	6.27E+01	90
80, 84-102	4.81E+03	4.96E+03	4.85E+03	7.15E+01	70

and Tabu A' are shown in Table 6.9. Table 6.10 and 6.11 display the minimum, maximum, average and standard deviation of the CPU times for Tabu B, Tabu C and Tabu A'. Experimental case 5 is used in this section. Tabu B was coded in C++. Tabu A' and Tabu C were coded in Matlab. The source code for SAGA was downloaded from http://www.tcoffee.org/Projects_home_page/ saga_home_page.html_saga. As for ClustalW, KALIGN, PRALINE and PRRN, the sequences were entered directly on the webpages http://www.ebi.ac.uk/Tools/clustalw2/index.html/ and http://www.ebi.ac.uk/ kalign, http://zeus.cs.vu.nl/programs/pralinewww and /http://align.genome.jp/prrn/, respectively. All MSA programs were run on a 440 MHz SUN Ultra10 machine. ClustalW, KALIGN, PRALINE and PRRN are all commercial packages.

ClustalW had the fastest processing time for each of the 20 groups. Although these programs are not the fastest, KALIGN, PRALINE and PRRN still do have relatively small processing times. For all 20 groups, Tabu C had the longest processing time. Also, when compared to ClustalW, KALIGN, PRRN and PRALINE, SAGA had a relatively long processing time. The CPU time for SAGA, Tabu B, Tabu C and Tabu A' are significantly affected by the number of sequences in an alignment. For the group with 80 sequences that vary in length from 84-102 base pairs, the average CPU time for SAGA and Tabu C to complete the MSA is over 15 hours and 25 hours, respectively. Specifically, with Tabu C, there is a trade-off between the CPU time and the quality of the alignment. Although tabu C takes extensive CPU time, the quality of the alignments are better than most of the other programs. In Figure 6-3, CPU times for the five groups with sequences that vary from 21-30 base pairs are compared for SAGA, Tabu B, Tabu C and Tabu A'. This plot shows that SAGA and Tabu C have the shortest and longest processing times of the four algorithms, respectively. Also, the plot shows that it is unclear whether Tabu B or Tabu A' has the

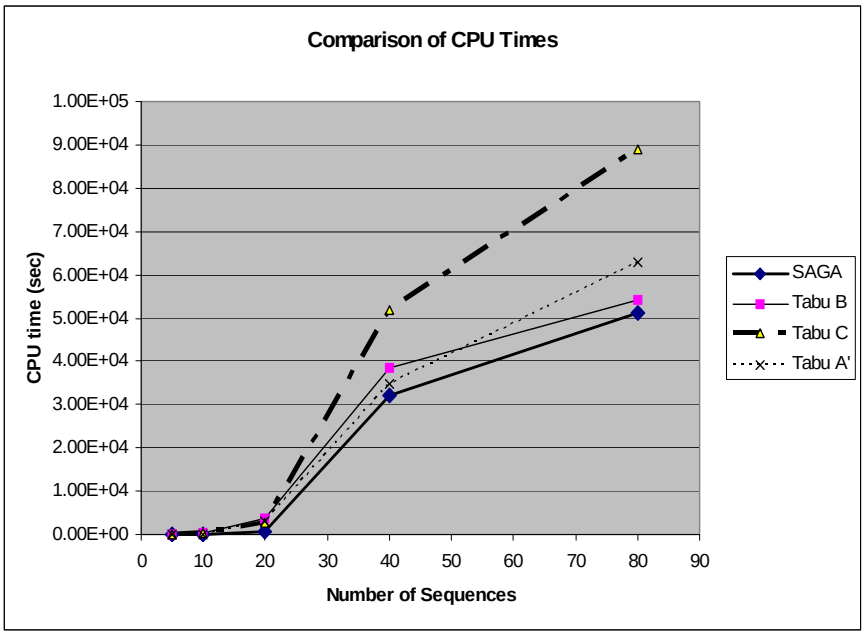


Figure 6-3: Comparison of CPU times for SAGA, Tabu B, Tabu C and Tabu A'

shortest average processing time.

6.6 Summary

In this chapter, the SP scoring function was used to compare Tabu B, Tabu A' and Tabu C to other MSA programs. The additional components added to Tabu C helped improve the alignment scores in comparison to the other tabu searches and MSA programs. Using the parsimony score as a primary measure and the SP score as the secondary measure helped Tabu C produce alignments that were comparable to ClustalW.

Table 6.8: CPU time from ClustalW, KALIGN, PRALINE and PRRN

	No Seq, Len	**CPU Time (sec)**	**No Seq, Len**	**CPU Time (sec)**
Clus	5, 9-15	<5	20, 39-51	1.9000E+01
KALI	5, 9-15	9.00E+00	20, 39-51	3.0000E+01
PRA	5, 9-15	<5	20, 39-51	2.3000E+01
PRRN	5, 9-15	1.20E+01	20, 39-51	1.4100E+02
Clus	5, 21-30	<5	20, 84-102	1.7000E+01
KALI	5, 21-30	1.10E+01	20, 84-102	3.0000E+01
PRA	5, 21-30	<5	20, 84-102	2.4000E+01
PRRN	5, 21-30	1.50E+01	20, 84-102	1.4900E+02
Clus	5, 39-51	<5	40, 9-15	2.3000E+01
KALI	5, 39-51	1.00E+01	40, 9-15	3.1000E+01
PRA	5, 39-51	<5	40, 9-15	3.1000E+01
PRRN	5, 39-51	1.40E+01	40, 9-15	1.5200E+02
Clus	5, 84-102	<5	40, 21-30	2.1000E+01
KALI	5, 84-102	1.30E+01	40, 21-30	3.3000E+01
PRA	5, 84-102	<5	40, 21-30	3.6000E+01
PRRN	5, 84-102	1.70E+01	40, 21-30	1.5500E+02
Clus	10, 9-15	<5	40, 39-51	2.7000E+01
KALI	10, 9-15	1.90E+01	40, 39-51	3.0000E+01
PRA	10, 9-15	<5	40, 39-51	3.2000E+01
PRRN	10, 9-15	1.90E+01	40, 39-51	1.5500E+02
Clus	10, 21-30	<5	40, 84-102	2.5000E+01
KALI	10, 21-30	1.90E+01	40, 84-102	3.3000E+01
PRA	10, 21-30	<5	40, 84-102	4.0000E+01
PRRN	10, 21-30	1.90E+01	40, 84-102	1.5900E+02
Clus	10, 39-51	<5	80, 9-15	2.6000E+01
KALI	10, 39-51	1.90E+01	80, 9-15	6.2000E+01
PRA	10, 39-51	<5	80, 9-15	4.5000E+01
PRRN	10, 39-51	2.20E+01	80, 9-15	1.6100E+02
Clus	10, 84-102	<5	80, 21-30	2.4000E+01
KALI	10, 84-102	1.90E+01	80, 21-30	6.4000E+01
PRA	10, 84-102	<5	80, 21-30	4.2000E+01
PRRN	10, 84-102	2.40E+01	80, 21-30	1.6300E+02
Clus	20, 9-15	1.30E+01	80, 39-51	2.9000E+01
KALI	20, 9-15	3.20E+01	80, 39-51	6.9000E+01
PRA	20, 9-15	2.10E+01	80, 39-51	4.6000E+01
PRRN	20, 9-15	2.60E+01	80, 39-51	1.7200E+02
Clus	20, 21-30	1.60E+01	80, 84-102	3.6000E+01
KALI	20, 21-30	3.00E+01	80, 84-102	6.7000E+01
PRA	20, 21-30	2.40E+01	80, 84-102	4.8000E+01
PRRN	20, 21-30	2.50E+01	80, 84-102	1.5000E+02

Table 6.9: CPU time from SAGA, Tabu B, Tabu C and Tabu A'

	No Seq, Len	**CPU Time (sec)**	**No Seq, Len**	**CPU Time (sec)**
SAG	5, 9-15	3.70E+01	20, 39-51	6.1200E+02
TabB	5, 9-15	6.30E+01	20, 39-51	9.9200E+02
TabC	5, 9-15	4.90E+01	20, 39-51	1.3960E+03
TabA'	5, 9-15	4.30E+01	20, 39-51	1.2630E+03
SAG	5, 21-30	4.20E+01	20, 84-102	7.3200E+02
TabB	5, 21-30	5.80E+01	20, 84-102	1.2870E+03
TabC	5, 21-30	7.40E+01	20, 84-102	1.5220E+03
TabA'	5, 21-30	5.10E+01	20, 84-102	1.3190E+03
SAG	5, 39-51	4.50E+01	40, 9-15	2.4860E+03
TabB	5, 39-51	6.20E+01	40, 9-15	3.8164E+04
TabC	5, 39-51	9.30E+01	40, 9-15	4.9557E+04
TabA'	5, 39-51	5.70E+01	40, 9-15	3.4127E+04
SAG	5, 84-102	4.70E+01	40, 21-30	3.1630E+03
TabB	5, 84-102	7.20E+01	40, 21-30	3.8503E+04
TabC	5, 84-102	1.01E+02	40, 21-30	5.1679E+04
TabA'	5, 84-102	6.30E+01	40, 21-30	3.4902E+04
SAG	10, 9-15	9.20E+01	40, 39-51	2.9883E+04
TabB	10, 9-15	1.51E+02	40, 39-51	3.9142E+04
TabC	10, 9-15	2.41E+02	40, 39-51	5.4955E+04
TabA'	10, 9-15	1.29E+02	40, 39-51	3.8594E+04
SAG	10, 21-30	9.60E+01	40, 84-102	3.0694E+04
TabB	10, 21-30	1.70E+02	40, 84-102	3.9105E+04
TabC	10, 21-30	2.47E+02	40, 84-102	5.5137E+04
TabA'	10, 21-30	1.58E+02	40, 84-102	4.1977E+04
SAG	10, 39-51	1.27E+02	80, 9-15	4.9035E+04
TabB	10, 39-51	1.93E+02	80, 9-15	5.4039E+04
TabC	10, 39-51	2.69E+02	80, 9-15	8.8712E+04
TabA'	10, 39-51	1.87E+02	80, 9-15	5.5318E+04
SAG	10, 84-102	1.56E+02	80, 21-30	5.1099E+04
TabB	10, 84-102	2.11E+02	80, 21-30	5.4217E+04
TabC	10, 84-102	2.91E+02	80, 21-30	8.9102E+04
TabA'	10, 84-102	2.37E+02	80, 21-30	6.2799E+04
SAG	20, 9-15	4.79E+02	80, 39-51	5.2748E+04
TabB	20, 9-15	8.12E+02	80, 39-51	5.4133E+04
TabC	20, 9-15	9.83E+02	80, 39-51	8.9117E+04
TabA'	20, 9-15	8.04E+02	80, 39-51	5.4132E+04
SAGA	20, 21-30	5.20E+02	80, 84-102	5.1006E+04
TabB	20, 21-30	8.31E+02	80, 84-102	5.4213E+04
TabC	20, 21-30	9.94E+02	80, 84-102	9.1196E+04
TabA'	20, 21-30	8.48E+02	80, 84-102	5.4133E+04

Table 6.10: CPU time measures for Tabu B, Tabu C and Tabu A' (Groups contain 10-40 sequences)

	No Seq, Len	**Min CPU**	**Max CPU**	**Avg CPU Time (sec)**	**St Dev**
TabB	10, 9-15	3.80E+01	3.02E+02	1.51E+02	1.32E+02
TabC	10, 9-15	9.50E+01	5.95E+02	2.41E+02	2.57E+02
TabA'	10, 9-15	4.10E+01	2.78E+02	1.29E+02	1.60E+02
TabB	10, 21-30	3.80E+01	3.11E+02	1.70E+02	5.57E+02
TabC	10, 21-30	1.01E+02	5.99E+02	2.47E+02	2.56E+02
TabA'	10, 21-30	5.30E+01	2.63E+02	1.58E+02	2.95E+02
TabB	10, 39-51	7.20E+01	4.42E+02	1.93E+02	3.89E+02
TabC	10, 39-51	9.90E+01	5.83E+02	2.69E+02	2.46E+02
TabA'	10, 39-51	6.30E+01	4.59E+02	1.87E+02	2.03E+02
TabB	10, 84-102	8.90E+01	5.04E+02	2.11E+02	2.13E+02
TabC	10, 84-102	1.15E+02	6.39E+02	2.91E+02	4.67E+02
TabA'	10, 84-102	7.20E+01	6.13E+02	2.37E+02	2.77E+02
TabB	20, 9-15	4.32E+02	1.36E+03	8.12E+02	4.68E+02
TabC	20, 9-15	6.84E+02	2.39E+03	9.83E+02	9.13E+02
TabA'	20, 9-15	5.01E+02	1.19E+03	8.04E+02	3.47E+02
TabB	20, 21-30	3.29E+02	2.50E+03	8.31E+02	1.73E+03
TabC	20, 21-30	6.52E+02	3.01E+03	9.94E+02	1.28E+03
TabA'	20, 21-30	2.11E+02	1.97E+03	8.48E+02	1.09E+03
TabB	20,39-51	4.53E+02	2.78E+03	9.92E+02	1.22E+03
TabC	20,39-51	6.91E+02	3.50E+03	1.40E+03	1.46E+03
TabA'	20,39-51	2.52E+02	2.99E+03	1.26E+03	2.38E+03
TabB	20, 84-102	5.98E+02	3.52E+03	1.29E+03	2.53E+03
TabC	20, 84-102	8.20E+02	3.65E+03	1.52E+03	1.97E+03
TabA'	20, 84-102	6.94E+02	2.96E+03	1.32E+03	2.17E+03
TabB	40, 9-15	1.63E+04	7.49E+04	3.82E+04	2.97E+04
TabC	40, 9-15	3.27E+04	1.59E+05	4.96E+04	6.86E+04
TabA'	40, 9-15	1.50E+04	6.98E+04	3.41E+04	2.78E+04
TabB	40, 21-30	1.10E+04	7.73E+04	3.85E+04	3.33E+04
TabC	40, 21-30	4.05E+04	1.72E+05	5.17E+04	7.89E+04
TabA'	40, 21-30	1.63E+04	1.58E+05	3.49E+04	7.70E+04
TabB	40, 39-51	1.88E+04	1.18E+05	3.91E+04	5.43E+04
TabC	40, 39-51	3.19E+04	1.67E+05	5.50E+04	7.20E+04
TabA'	40, 39-51	1.28E+04	1.49E+05	3.86E+04	7.22E+04
TabB	40, 84-102	1.69E+04	1.57E+05	3.91E+04	7.65E+04
TabC	40, 84-102	2.22E+04	1.63E+05	5.51E+04	7.36E+04
TabA'	40, 84-102	1.69E+04	1.49E+04	4.20E+04	1.51E+04

Table 6.11: CPU time measures for Tabu B, Tabu C and Tabu A' (Groups contain 80 sequences)

	No Seq, Len	**Min CPU**	**Max CPU**	**Avg CPU Time (sec)**	**St Dev**
TabB	80, 9-15	3.08E+04	1.08E+05	5.40E+04	3.99E+04
TabC	80, 9-15	6.80E+04	1.64E+05	8.87E+04	5.05E+04
TabA'	80, 9-15	2.39E+04	1.13E+05	5.53E+04	4.53E+04
TabB	80, 21-30	3.10E+04	1.37E+04	5.42E+04	2.63E+04
TabC	80, 21-30	7.14E+04	1.94E+05	8.91E+04	6.62E+04
TabA'	80, 21-30	2.50E+04	9.98E+04	6.28E+04	3.74E+04
TabB	80, 39-51	3.90E+04	1.58E+05	5.41E+04	6.59E+04
TabC	80, 39-51	7.78E+04	1.98E+05	8.91E+04	6.93E+04
TabA'	80, 39-51	2.80E+04	1.82E+05	5.41E+04	8.24E+04
TabB	80, 84-102	3.39E+04	1.39E+05	5.42E+04	5.25E+04
TabC	80, 84-102	8.68E+04	1.85E+05	9.12E+04	5.54E+04
TabA'	80, 84-102	4.50E+04	1.84E+05	5.41E+04	7.97E+04

Chapter 7

Conclusion and Future Works

Tabu search is an effective way to align multiple sequences. The proposed Tabu A, B and A', do not use a tree to guide the alignment process. One advantage of not using a guide tree is that the tabu search avoids performing pairwise alignments for each pair of sequences in an MSA. When aligning large groups of sequences, making all of the pairwise alignments could become computationally expensive. Another advantage of not using a guide tree, produced from a neighbor joining algorithm, is that we can avoid predicting incorrect evolutionary trees. With the addition of the HMM to Tabu B, the tabu search produces better alignments than PRALINE and SAGA for all groups containing 50 sequences. Although ClustalW performs the best in terms of the speed and quality of the alignments when the SP scoring function is used, Tabu B with the local improvement algorithm does yield promising results for groups of 50 sequences or less.

Similar to Tabu A, B and A', Tabu C also eliminates the preprocessing step of performing pairwise alignments for each pair of sequences in an MSA. While not using a tree has some benefits, Tabu C was modified to incorporate a tree that would take advantage of similarities between sequences and suggest how sequences may have evolved. Tabu C is different from

many other alignment programs that use a guide tree because it does not perform pairwise alignments for each pair of sequences or use a neighbor joining algorithm. Instead, Tabu C randomly generates a tree that guides the MSA. Tabu C avoids many of the typical pitfalls made by progressive alignment programs that use guide trees. Gaps introduced early in a subalignment are not necessarily fixed. Because moves are allowed which essentially change the gap locations and tree topology of an MSA, an error made earlier in the alignment could easily be corrected.

The modifications made in Tabu C help improve the quality of the MSAs produced. The results from Tabu C are comparable to the results from ClustalW, SAGA, PRALINE, KALIGN and PRRN. When using the SP scoring scheme, Tabu C performs well with MSAs containing only 5 sequences or more than 20 sequences. In most groups with more than 20 sequences, the SP scores for Tabu C are second only to ClustalW. The implementation of a heuristic intensification procedure and a diversification procedure helps stop the algorithm from getting trapped in a local optimal solution.

The SP score is a simple measure that evaluates the quality of an alignment. Initially, each of the 4 tabu searches simply attempt to find the MSA which has the best SP alignment score. However, using the SP scoring function, Tabu C was unable to obtain better alignments than those from ClustalW. Replacing the SP score with the parsimony score in Tabu C, finally produces some alignments that are better than ClustalW. In this case, the objective of Tabu C is to minimize the parsimony score. Having lower parsimony scores than ClustalW in 6 of 16 groups is a substantial improvement.

The only drawback to the tabu searches is the tremendous amount of CPU time used. While bounds limit the DP search space, a more efficient method for searching solutions in the tabu search space should be explored. One option may be to consider doing pairwise

alignments in a preprocessing step. Although doing the pairwise alignments in a preprocessing step may be a viable option, continuing to eliminate the neighbor joining algorithm would still be advantageous. Thus, when a tree is randomly generated, more similar sequences would be in the same branch. Then the tabu may not spend so much time with randomly generated tree topologies that initially have dissimilar sequences sharing the same parent node. The time spent doing a preprocessing may be less than the time spent exploring portions of the solution spaces that have little chance of producing a high quality alignment.

The tabu search objective is to either maximize the SP score or minimize the parsimony score. In the future, attempting to optimize both of these measures simultaneously, may further improve the MSA. The Ant Colony Optimization algorithm (ACO), will attempt to use bicriterion optimization to find the MSA with the best SP and parsimony score, simultaneously [44, 8]. The algorithm uses the natural behavior of ants in a colony to find the best paths to 2 different food sources. An extension of the ACO algorithm to sequence alignment will yield the first colony of ants to find the best path (MSA) to the lowest parsimony score, while the second colony finds the best path (MSA) to the highest SP score. The overall objective when the algorithm terminates is to find the best MSA that optimizes both of these measures. Although many components of the tabu search are advantageous to finding the best MSA, it is worth exploring more complex measures, such as the ACO algorithm, to measure the quality of an alignment.

Bibliography

[1] Abbas, A. and S. Holmes, "Bioinformatics and management science: some common tools and techniques", Operations Research, vol. 52 (2), pp. 165-190, 2004.

[2] Altschul, S.F., R.J. Carroll and D. Lipman, "Weights for data related by a tree", Journal of Molecular Biology, vol. 207, pp. 647-653, 1989.

[3] Altuschul, S.F.,T.L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller and D. Lipman, "Gapped BLAST and PSI BLAST: a new generation of protein database search programs" , Nucleic Acids Research, vol. 25(17), pp. 3389-3402, 1997.

[4] Baeza-Yates, R. A. and G. H. Gonnet, "A new approach to text searching", Proceedings of the 12th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp.168-175, 1989.

[5] Brudno, M., S. Malde, A. Poliakov, C. Do, O. Couronne, I. Dubbchak and S. Batzoglou, "Glocal alignment: finding rearrangements during alignment", Bioinformatics, vol. 19, pp. 154-162, 2003.

[6] Carrillo, H. and D. Lipman, "The multiple sequence alignment problem in biology", SIAM Journal of Applied Mathematics, vol. 48, pp. 1073-1082, 1988.

[7] Davidson, A., "A fast pruning algorithm for optimal sequence alignment", Proceedings of the IEEE 2nd International Symposium on Bioinformatics and Bioengineering Conference, vol 4(6), pp. 49-56, 2001.

[8] Dorigo, M. and T. Stutzle.Ant Colony Optimization, MIT Press, Cambridge, MA, 2004.

[9] Durbin, S., S. Eddy, A. Krogh and G. Mitchison, Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids, Cambridge University Press, Cambridge, UK, 1998.

[10] http://www.ie.ncsu.edu/fangroup/ps.dir/tabusearch.pdf

[11] Feng, D.F. and R. F. Doolittle, "Progressive sequence alignment as a prerequisite to correct phylogenetic trees", Journal of Molecular Evolution, vol.24(4), pp. 351-360, 1987.

[12] Fitch, W. and J. Farris, "Evolutionary trees with minimum nucleotide replacements from amino acid sequences", Journal of Molecular Evolution, vol. 3, pp. 263-278, 1974.

[13] Gabrani, N. and P. Shankar, "A note on the reconstruction of a binary tree from its traversals", Information Processing Letters, vol. 42(2), pp. 117 -119, 1992.

[14] Gascuel, O, D. Bryant and F. Denis, "Strengths and limitations of the minimum-evolution principle", Systematic Biology, vol. 50(5), pp. 621-627, 2001.

[15] Glover, F., "Tabu search: a tutorial", Interfaces, vol. 20(4), pp. 74-94, 1990.

[16] Gotoh, O., "An improved algorithm for matching biological sequences", Journal of Molecular Biology, vol. 162, pp. 705-708, 1982.

[17] Gotoh, O., "Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments." , Journal of Molecular Biology, vol. 264, pp. 823-838, 1996.

[18] Graur, D. and W.H. Li, Fundamentals of Molecular Evolution (2nd ed.), Sinauer Associates, Sunderland, MA, 2002.

[19] Higgins, D.G. and W. Taylor, Bioinformatics: Sequence Structure and Databanks, Oxford University Press, New York, NY, 2000.

[20] Higgins, D.G, and P.M. Sharp, "CLUSTAL: A package for performing multiple sequence alignment on a microcomputer", Gene, vol. 73 ,pp. 237-244, 1988.

[21] Higgins, D.G, J.D. Thompson and T.J. Gibson, "Using CLUSTAL for multiple sequence aligments", Methods Enzymol, vol. 266, pp. 383-402, 1996.

[22] Krogh, A., M. Brown, I.S. Mian, K. Sjolander and D. Haussler, "Hidden markov models in computational biology: applications to protein modeling", Journal of Molecular Biology, vol. 235, pp. 1501-1531, 1994.

[23] Lassmann, T. and E. Sonnhammer, "Kalign: an accurate and fast multiple sequence alignment algorithm", BMC Bioinformatics, vol 6, pp. 298-307 2005

[24] Levenshtein, V., "Binary codes capable of correcting deletions, insertions, and reversals", Soviet Physics Doklady, vol.10(8), pp. 707-710, 1966.

[25] Lukashin, A., J. Engelbrecht and S. Brunak, "Multiple alignment using simulated annealing: branch point definition in human mRNA splicing", Nucleic Acids Research, vol. 20(10), pp. 2511-2516, 1992.

[26] Zhu, M., G. Hu, Q. Zeng and H. Peng, "Multiple sequence alignment using minimum spanning tree", Proceedings of 2005 International Conference on Machine Learning and Cybernetics, vol. 6, pp.,3352-3356, 2005.

[27] Mount, D. W., Bioinformatics: Sequence and Genome Analysis, Cold Spring Harbor Laboratory, Cold Spring Harbor, NY, 2001.

[28] Needleman, S. B. and C. D. Wunsch, "A general method applicable to the search for similarities in amino acid sequences of two proteins", Journal of Molecular Biology, vol. 48, pp. 443-453, 1970.

[29] Nei, M. and S. Kumar, Molecular Evolution and Phylogenetics , Oxford University Press, New York, NY, 2000.

[30] Notredame, C. and D.G. Higgins, "SAGA: sequence alignment by genetic algorithm", Nucleic Acids Research, vol. 24, pp. 1515-1524, 1996.

[31] Notredame, C., L. Holmes and D.G. Higgins, "COFFEE: an objective function for multiple sequence alignments", Bioinformatics, vol. 14(5), pp. 407-422, 1998.

[32] Ogul, H. and K. Ericyes, "Identifying all local and global alignments between two DNA sequences", International Computing Institute, Ege University, Izmir, Turkey

[33] Rabiner, L.R and B.H. Juang, "An introduction to hidden markov models", IEEE ASSP Magazine, vol. 3(1), pp. 4-16, 1986.

[34] Riaz, T., Y. Wang and K. Li, "Multiple sequence alignment using tabu search", Asia-Pacific Bioinformatics Conference (APBC2004), vol. 29, pp.1-10, 2004.

[35] Rognes, T., "ParAlign: a parallel sequence alignment algorithm for rapid and sensitive database searches", Nucleic Acids Research, vol. 29(7), pp. 1647-1652, 2001.

[36] Schroedl, S., "An improved search algorithm for optimal multiple sequence alignment", Journal of Aritificial Intelligence Research, vol. 23, pp. 587-623, 2005.

[37] Shwartz, A., and L. Pachter, "Multiple alignment by sequence annealing", Bioinformatics, vol. 23(2), pp. 24-29, 2007.

[38] Shyu, C., L. Sheneman and J. Foster, "Multiple sequence alignment with evolutionary computation", Genetic Programming and Evolvable Machines, vol. 5, pp. 121-144, 2004.

[39] Smith, T. F. and M. S. Waterman, "Identification of Common Molecular Subsequences", Journal of Molecular Biology, vol. 147, pp. 195-197, 1981.

[40] Sneath, P. H. A. and R. R. Sokal, Numerical Taxonomy Freeman, San Francisco, CA, 1973.

[41] http://www.techfak.uni-bielefeld.de/bcd/Curric/MulAli/node3.html

[42] Thompson, J.D., D.G. Higgins and T.J. Gibson, "CLUSTALW: improving the sensitvity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice, Nucleic Acids Research, vol. 22, pp. 4673-4680, 1994.

[43] http://statgen.ncsu.edu/thorne/bioinf2.html

[44] T'kindt, V., N. Monmarche, F. Tercinet and D. Laugt, "An ant colony optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem", European Journal of Operational Research, vol. 142(2), pp. 250-257, 2002.

[45] http://www.comp.leeds.ac.uk/roger/HiddenMarkovModels/html_dev/viterbi_algorithm/s1_pg1.html

[46] Wu, S., U. Manber, E. W. Myers and W. Miller, "An O(NP) sequence comparison algorithm," Information Processing Letters, vol. 35, pp. 317-323, 1990.