

IMPROVING THE QUALITY OF MULTIPLE SEQUENCE ALIGNMENT

A Dissertation

by

YUE LU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2008

Major Subject: Biochemistry

IMPROVING THE QUALITY OF MULTIPLE SEQUENCE ALIGNMENT

A Dissertation

by

YUE LU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Sing-Hoi Sze
Committee Members,	John Mullet
	J. Martin Scholtz
	Tiffani Williams
Head of Department,	Gregory Reinhart

December 2008

Major Subject: Biochemistry

ABSTRACT

Improving the Quality of Multiple Sequence Alignment. (December 2008)

Yue Lu, B.S., Peking University

Chair of Advisory Committee: Dr. Sing-Hoi Sze

Multiple sequence alignment is an important bioinformatics problem, with applications in diverse types of biological analysis, such as structure prediction, phylogenetic analysis and critical sites identification. In recent years, the quality of multiple sequence alignment was improved a lot by newly developed methods, although it remains a difficult task for constructing accurate alignments, especially for divergent sequences.

In this dissertation, we propose three new methods (PSAlign, ISPAAlign, and NRAAlign) for further improving the quality of multiple sequences alignment.

In PSAlign, we propose an alternative formulation of multiple sequence alignment based on the idea of finding a multiple alignment which preserves all the pairwise alignments specified by edges of a given tree. In contrast with traditional NP-hard formulations, our preserving alignment formulation can be solved in polynomial time without using a heuristic, while still retaining very good performance when compared to traditional heuristics.

In ISPAlign, by using additional hits from database search of the input sequences, a few strategies have been proposed to significantly improve alignment accuracy, including the construction of profiles from the hits while performing profile alignment, the inclusion of high scoring hits into the input sequences, the use of intermediate sequence search to link distant homologs, and the use of secondary structure information.

In NRAlign, we observe that it is possible to further improve alignment accuracy by taking into account alignment of neighboring residues when aligning two residues, thus making better use of horizontal information. By modifying existing multiple alignment algorithms to make use of horizontal information, we show that this strategy is able to consistently improve over existing algorithms on all the benchmarks that are commonly used to measure alignment accuracy.

To my parents

ACKNOWLEDGEMENTS

I would never have been able to finish this dissertation without the guidance of my committee members, help from friends, and support from my family.

I would like to express my deepest gratitude to my advisor, Dr. Sing-Hoi Sze, for his tremendous guidance, support, patience, and providing me with an excellent atmosphere for doing research. I have encountered many difficulties during my graduate years, Dr. Sze has had the wisdom to point me in the right direction, yet allow me to find my own answers.

I am grateful to all the members of my committee, Dr. David Giedroc, Dr. Thomas Ioerger, Dr. John Mullet, Dr. J. Martin Scholtz and Dr. Tiffani Williams, for their input and interest in my research.

I would like to thank my labmates, Qingwu Yang and Xiaoyan Zhao, for valuable discussion and input. My first project was joint work with Qingwu. I would also like to thank all my friends at Texas A&M University. I greatly value their friendship and deeply appreciate their help during these years.

Finally, and most importantly, none of this would have been possible without the love and patience of my family. My very special thanks to my fiancé, Jie Meng, who was

always there cheering me up and stood by me through the good times and bad, and my parents, who were always there supporting me and encouraging me with their best wishes throughout all these years.

TABLE OF CONTENTS

	Page
ABSTRACT	iii
DEDICATION	v
ACKNOWLEDGEMENTS	vi
TABLE OF CONTENTS	viii
LIST OF FIGURES	x
LIST OF TABLES	xi
 CHAPTER	
I INTRODUCTION	1
A. Overview of Multiple Sequence Alignment.....	1
B. Our Contribution.....	8
II PSAlign: A POLYNOMIAL TIME SOLVABLE FORMULATION OF MULTIPLE SEQUENCE ALIGNMENT	13
A. Introduction	13
B. Problem Formulation	17
C. Exact Algorithm	20
D. Performance.....	27
E. Discussion.....	34
III ISPAlign: MULTIPLE SEQUENCE ALIGNMENT BASED ON PROFILE ALIGNMENT OF INTERMEDIATE SEQUENCES	37
A. Introduction	37
B. Finding Intermediate Sequences.....	41
C. Choosing Intermediate Sequences.....	42
D. Constructing Sequence Profiles.....	44
E. Alignment via Modified Pair-HMM.....	45
F. Detailed Algorithm	47
G. Performance.....	50

CHAPTER		Page
	H. Discussion	63
IV	NRAAlign: MULTIPLE SEQUENCE ALIGNMENT BASED ON ALIGNMENT OF NEIGHBORING RESIDUES	65
	A. Introduction	65
	B. Methods	67
	C. Benchmark Alignments	72
	D. Performance.....	75
	E. Discussion.....	85
	F. Conclusion	91
V	SUMMARY AND FUTURE WORK.....	92
	REFERENCES.....	96
	VITA	107

LIST OF FIGURES

FIGURE	Page
1.1 Representation of progressive alignment algorithm.....	3
1.2 Consistency-based strategy called “library extension” from TCoffee	7
1.3 The flow of MUSCLE.....	8
1.4 Pair-HMM models used by ProbCons and MUMMALS.....	9
2.1 Illustration of the exact algorithm.	22
3.1 Greedy algorithm to choose a small subset of intermediate sequences to add to the input sequences.....	43
3.2 The original and the modified pair-HMM models	46
3.3 Running time of SPEM and ISPAlign on PREFAB 4.0	58
4.1 Illustration of the window on two sequences s and s' with $\omega = 2$	68

LIST OF TABLES

TABLE	Page
2.1 Average SPS and CS scores (in %) of PSAlign on BALiBASE 2.01	30
2.2 Average Q scores (in %) of PSAlign on PREFAB 4.0	31
2.3 Average Q scores (in %) of PSAlign on SABmark 1.65	32
2.4 Performance of ProbCons ($ir = 0$) and PSAlign[ProbCons] when consistency transformation in ProbCons is disabled.....	35
3.1 Average SPS scores (in %) of ISPAAlign on the full length sequence set in BALiBASE 3.0	52
3.2 Average CS scores (in %) of ISPAAlign on the full length sequence set in BALiBASE 3.0	53
3.3 Average SPS and CS scores (in %) of ISPAAlign on HOMSTRAD	54
3.4 Average Q scores (in %) of ISPAAlign on PREFAB 4.0	56
3.5 Average f_D and f_M scores (in %) of ISPAAlign on the Twilight and Superfamily subsets of SABmark 1.65	57
3.6 Average CS scores (in %) of ISPAAlign on HOMSTRAD using a few methods that are of increasing levels of complexity.....	59
3.7 Average Q scores (in %) of ISPAAlign on PREFAB 4.0 using a few methods that are of increasing levels of complexity	60
3.8 Average CS scores (in %) of ISPAAlign on HOMSTRAD by varying the parameter k that specifies the maximum number of intermediate sequences that are added to the input sequences.....	61
3.9 Average CS scores (in %) of ISPAAlign on HOMSTRAD by varying the parameters α and β that modify the pair-HMM probabilities.....	62
4.1 Parameter settings for the modified version of each algorithm that uses horizontal information	69

TABLE	Page
4.2 Average SPS scores (in %) of NRAlign on full length sequences in BAliBASE 3.0.....	76
4.3 Average CS (in %) scores of NRAlign on full length sequences in BAliBASE 3.0.....	78
4.4 Average SPS scores (in %) of NRAlign on HOMSTRAD	79
4.5 Average CS scores (in %) of NRAlign on HOMSTRAD	80
4.6 Average Q(2) scores (in %) of NRAlign on PREFAB 4.0.....	81
4.7 Average Q(50) scores (in %) of NRAlign on PREFAB 4.0.....	82
4.8 Average f_D and f_M scores and average normalized Dali Z-score, GDT_TS score, and ContactA and ContactB scores (in %) of NRAlign on the Twilight subset of SABmark 1.65	83
4.9 Average f_D and f_M scores and average normalized Dali Z-score, GDT_TS score, and ContactA and ContactB scores (in %) of NRAlign on the Superfamily subset of SABmark 1.65	84
4.10 Average f_D and f_M scores and average normalized Dali Z-score, GDT_TS score, and ContactA and ContactB scores (in %) of NRAlign on the Twilight subset of SABmark 1.65 when pairwise alignments are performed over all sequence pairs instead of obtaining a single multiple alignment.....	86
4.11 Average f_D and f_M scores and average normalized Dali Z-score, GDT_TS score, and ContactA and ContactB scores (in %) of NRAlign on the Superfamily subset of SABmark 1.65 when pairwise alignments are performed over all sequence pairs instead of obtaining a single multiple alignment.....	87
4.12 Average SPS scores (in %) of NRAlign on HOMSTRAD. Each subset includes all alignments with number of sequences within the specified range.....	88
4.13 Average CS scores (in %) of NRAlign on HOMSTRAD. Each subset includes all alignments with number of sequences within the specified range.....	89

TABLE	Page
-------	------

4.14 Average SPS and CS scores (in %) of NRAlign on HOMSTRAD by varying the parameter ω that specifies the maximum number of horizontal positions that are included to the left and to the right, and the parameter β that specifies the weight of the neighboring scores	90
--	----

CHAPTER I

INTRODUCTION

A. Overview of Multiple Sequence Alignment

Introduction

The goal of multiple sequence alignment (MSA) is to bring functionally or structurally similar regions across different biological sequences as close as possible, with applications in diverse types of biological analysis, such as structure prediction, phylogenetic analysis and critical sites identification (Taylor, 1987; Carillo and Lipman, 1988; Thompson *et al.*, 1994; Gotoh, 1996; Morgenstern *et al.*, 1996; Stoye, 1998; Notredame *et al.*, 2000; Lee *et al.*, 2002; Edgar, 2004; Van Walle *et al.*, 2004; Do *et al.*, 2005). In recent years, a lot of multiple sequence alignment methods have been developed and many of them successfully improved the quality of multiple alignment, however, these purely automatic methods are still not good enough when comparing to manually refined alignments, especially for divergent sequences.

Computational Approaches for Multiple Sequence Alignment

The traditional formulation of multiple sequence alignment is to identify the alignment that has the maximal sum-of-pairs (SP) score. The sum-of-pairs score of an alignment is

This dissertation follows the style of *Journal of Computational Biology*.

typically defined as the sum of substitution matrix scores for each aligned pair of residues minus gap penalties. Usually, this mathematically optimal alignment can be obtained in seconds for a pair of sequences by dynamic programming techniques, however, the computational time and memory required by dynamic programming is too expensive for multiple sequences. The best known exact algorithm employs dynamic programming techniques with time complexity $O(n^k)$ (Carillo and Lipman, 1988), where n is the maximum sequence length and k is the number of sequences, and thus is useful only when k is very small. Stoye (1998) proposed a divide-and-conquer heuristic to limit the search space by subdividing the input sequences into shorter segments, but it is still not efficient enough for large-scale alignments.

The inherent difficulty of the multiple alignment problem leads naturally to the development of heuristic approaches. The most popular approach to construct multiple sequence alignments is by employing a progressive alignment algorithm (Figure 1.1), in which each sequence is treated initially as an alignment and the next two most similar alignments are repeatedly combined until a single multiple alignment is obtained (Feng and Doolittle, 1987; Thompson *et al.*, 1994; Notredame *et al.*, 2000; Edgar 2004; Do *et al.*, 2005; Pei and Grishin, 2006; Roshan and Livesay, 2006). Alternatively, other non-progressive approaches assemble a final multiple alignment from short alignments of local similarities (Morgenstern *et al.*, 1996; Van Walle *et al.*, 2004).

Since progressive alignment approach cannot correct errors that may happen during each

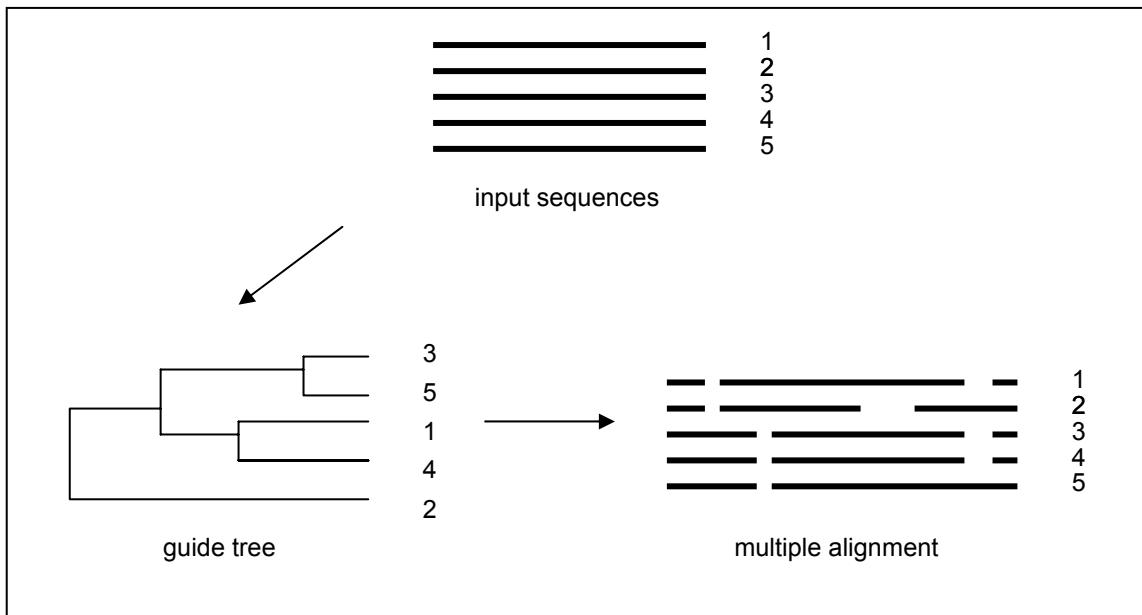


Figure 1.1: Representation of progressive alignment algorithm. First a guide tree is constructed according to the similarity scores of the input sequences, then following the guide tree, each sequence is treated initially as an alignment and the next two most similar alignments are repeatedly combined until a single multiple alignment is obtained. In this case, there are five input sequences. Following the guide tree, sequence 3 and sequence 5 are aligned first, and then sequence 1 and sequence 4 are aligned. Then the alignment of sequence 3 and 5 is aligned to the alignment of sequence 1 and 4. Finally the alignment of sequence 3, 5, 1 and 4 are aligned to sequence 2 to obtain the final multiple alignment.

progressive alignment step, two techniques are often used to minimize the errors and further improve the alignment: performing iterative refinements after the initial

alignment is constructed (Gotoh, 1996; Edgar, 2004; Do *et al.*, 2005; Roshan and Livesay, 2006; Yamada *et al.*, 2006) and using consistency-based pairwise alignments during progressive approach (Notredame *et al.*, 2000; Do *et al.*, 2005; Pei and Grishin, 2006; Roshan and Livesay, 2006). There are a variety of subtly different methods for iterative refinements. Generally, iterative refinements are carried out by repeatedly dividing the alignment into two sub-alignments and realigning these two sub-alignments until the score of the alignment can't be improved anymore. The goal of consistency-based approach is to improve the quality of the initial pairwise alignments by aligning through other sequences to increase their agreement with the final multiple alignment. This is often achieved by adjusting the residue pair scores of two sequences by their alignments to other sequences. For example, given three sequences x , y and z , instead of using the original residue pair scores directly, consistency-based approach adjusts the matching score for a residue pair $x_i - y_j$ according to support from some residue z_k that aligns to both x_i and y_j in the respective $x - z$ and $y - z$ pairwise comparisons.

Other recent efforts that lead to significant improvement of alignment accuracy also include using maximal expected accuracy alignment instead of the less accurate Viterbi alignment (Do *et al.*, 2005; Pei and Grishin, 2006; Roshan and Livesay, 2006), incorporating secondary structure predictions (Zhou and Zhou, 2005; Pei and Grishin, 2007), incorporating local structural information (O'Sullivan *et al.*, 2004; Van Walle *et al.*, 2004; Pei *et al.*, 2008), incorporating additional sequences from database search (Marti-Renom *et al.*, 2004; Simossis *et al.*, 2005; Zhou and Zhou, 2005; Pei and Grishin,

2007) and combining alignments from existing methods to obtain an improved alignment (Bucka-Lassen *et al.*, 1999; Wallace *et al.*, 2006).

Benchmarks and Evaluation of Alignment Quality

Two types of quality scores are often used to evaluate multiple sequence alignments: reference-dependent scores and reference-independent scores. Reference-dependent scores are computed by comparing a test alignment to its corresponding reference alignment, which is considered as a gold standard. Reference alignments are generated from protein structural alignments, which are constructed without any sequence information. To increase the quality of reference alignments, they are often manually refined or only the regions that are agreed by several different structural alignment methods are considered. Commonly used reference alignment benchmarks include BALiBASE (Thompson *et al.*, 2005), which contains manually refined structural alignments that are subdivided into five categories, HOMSTRAD (Mizuguchi *et al.*, 1998), which contains a collection of manually edited structure-based alignments, PREFAB (Edgar, 2004), which contains structural alignments of two sequences and automatically generated alignments that are obtained from adding high scoring hits of the two sequences from database search, and SABmark (Van Walle *et al.*, 2004), which contains alignments that are derived from the SCOP classification (Murzin *et al.*, 1995).

Although reference alignments are often in very high quality, it is still possible that they contain errors, especially for divergent sequences. Reference-independent scores can

avoid this problem. When structures are available, reference-independent scores are computed as structural similarity scores by comparing two structures using the aligned residues in the test alignment (Pei and Grishin, 2006; Pei and Grishin, 2007; Pei *et al.*, 2008). Reference-independent scores can be classified as intra-molecular based scores and inter-molecular based scores. Inter-molecular based scores require superimposition of the two structures. Previous studies show that reference-independent scores are consistent with reference-dependent scores when using large alignment benchmarks (Pei and Grishin, 2006; Pei and Grishin, 2007; Pei *et al.*, 2008).

Multiple Alignment Methods

ClustalW (Thompson *et al.*, 1994) is probably the most widely used multiple alignment method. It employs a classic progressive alignment approach, and to our best knowledge, it has not been improved significantly for a long time. Current popular and accurate methods include TCoffee (Notredame *et al.*, 2000), MUSCLE (Edgar, 2004), ProbCons (Do *et al.*, 2005) and MUMMALS (Pei and Grishin, 2006). TCoffee is the first multiple alignment tool that utilizes consistency-based pairwise alignments (Figure 1.2). MUSCLE is one of the most efficient multiple alignment methods that also have high accuracy (Figure 1.3). ProbCons is the first multiple alignment tool that utilizes the maximal expected accuracy alignment based on a pair-HMM model (Figure 1.4). MUMMALS uses secondary structure information during pair-HMM training to further improve alignment accuracy (Figure 1.4). An alignment method can outperform other methods statistically based on large benchmarks. However, there is no guarantee that

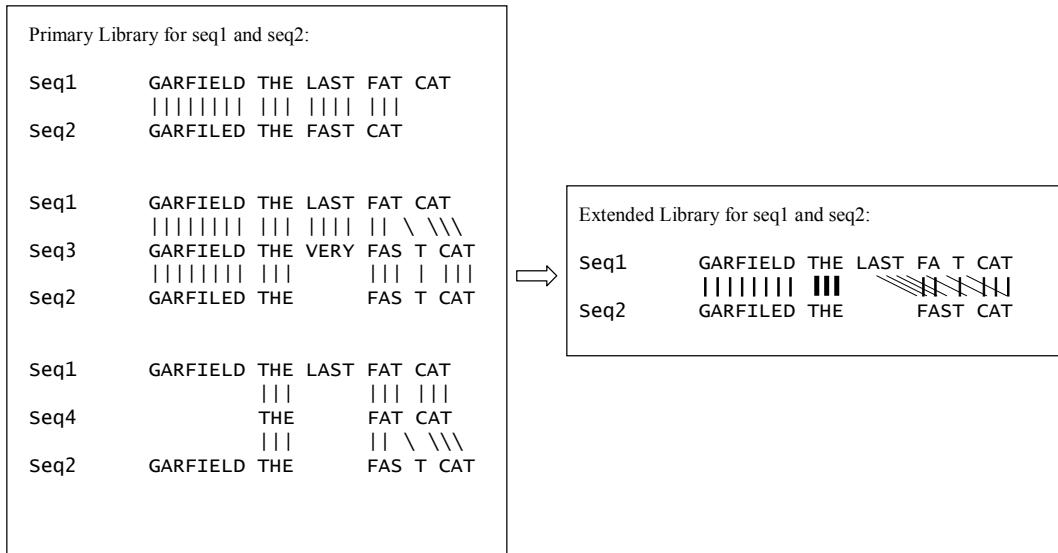


Figure 1.2: Consistency-based strategy called “library extension” from TCoffee (Notredame *et al.*, 2000). The primary library is constructed from global and local alignment methods such as ClustalW (Thompson *et al.*, 1994) and Lalign (Huang and Miller, 1991). The library is extended by using a triplet approach, which modifies the score of two residues from two sequences according to their alignments to residues from a third sequence. The extended library is then used in progressive alignment to obtain the multiple sequence alignment.

one method is always better than the others on all individual cases. It is highly recommended to try several methods together for a specific alignment, and the regions that are agreed by different methods are more likely to be reliable.

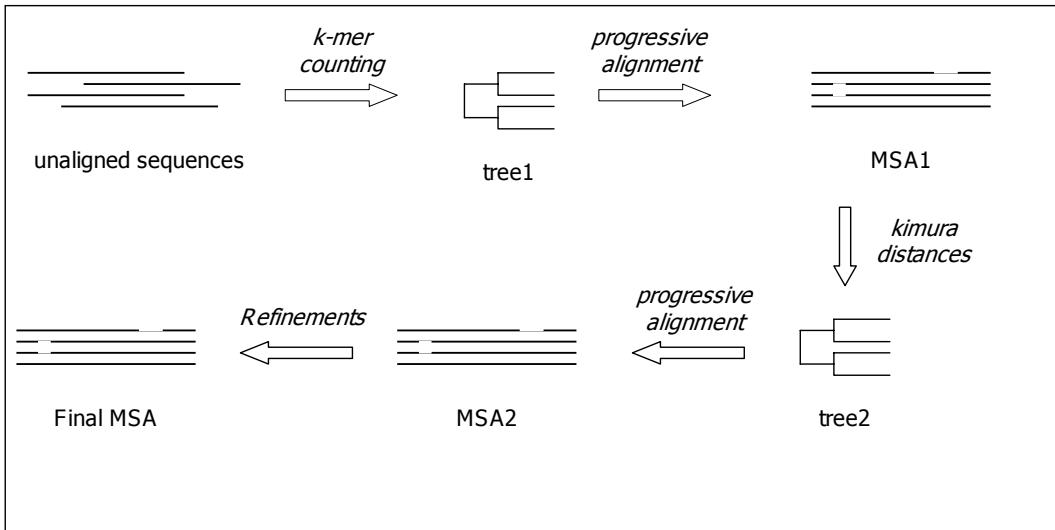


Figure 1.3: The flow of MUSCLE (Edgar, 2004). A draft multiple alignment is first constructed by progressive alignment from a tree estimated by *k*-mer distances of the input sequences. Then a second progressive alignment is performed using a re-estimated tree by kimura distances from the first multiple alignment. Iterative refinements are performed to the second multiple alignment to obtain the final alignment. By avoiding the time-consuming pairwise comparisons of all the input sequences, MUSCLE is one of the most efficient methods for large-scale multiple alignments.

B. Our Contribution

In despite of the noticeable improvement in alignment accuracy, current available multiple alignment methods are far from perfect, especially for divergent sequences and

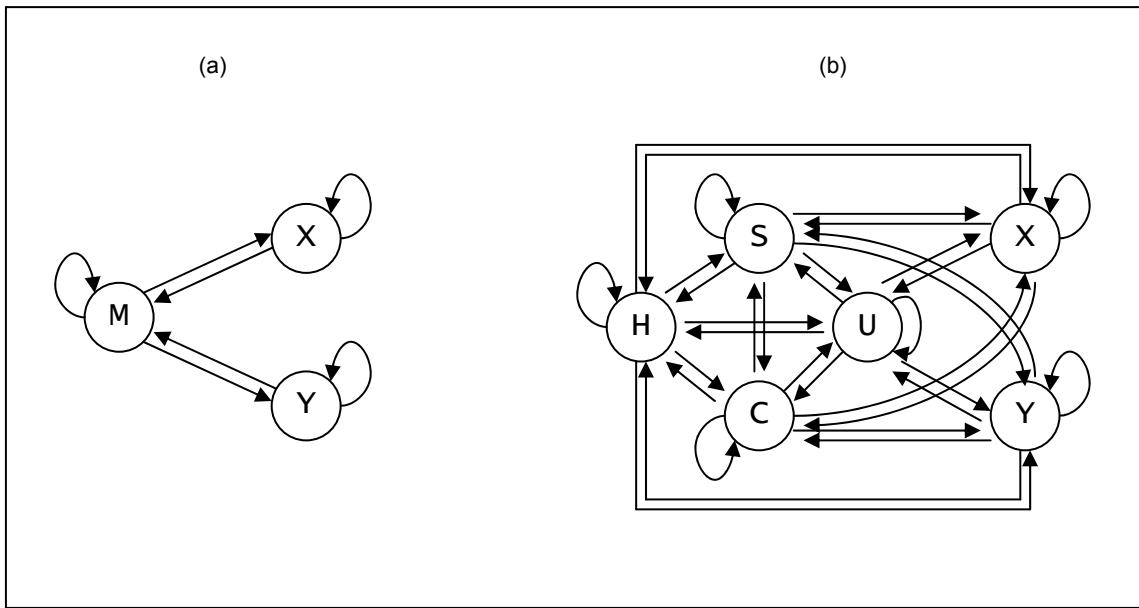


Figure 1.4: Pair-HMM models used by ProbCons (Do *et al.*, 2005) and MUMMALS (Pei and Grishin, 2006). (a) Basic pair-HMM model used by ProbCons. Basic HMM contains three different states. State M emits an aligned pair of residues, states X and Y emit a residue in the first and the second sequences respectively that is aligned to a gap. (b) The HMM_1_3_1 model used by MUMMALS. The state M in the basic model is further divided into 4 states (state H , state S , state C and state U) in this model. State H , state S and state C represent helix, strand and coil respectively according to the secondary structure types of the first sequence. State U represents unaligned regions.

large sequence datasets. In Chapter II – Chapter IV, we propose three new multiple alignment methods, which improve multiple sequence alignment using three different approaches.

Since traditional multiple alignment formulations are NP-hard, heuristics are commonly employed to find acceptable alignments with no guaranteed performance bound. This causes a substantial difficulty in understanding what the resulting alignment means and in assessing the quality of these alignments. In Chapter II, we propose an alternative formulation of multiple alignment based on the idea of finding a multiple alignment of k sequences which preserves $k - 1$ pairwise alignments as specified by edges of a given tree. Although it is well known that such a preserving alignment always exists, it did not become a mainstream method for multiple alignment since it seems that a lot of information is lost from ignoring pairwise similarities outside the tree. In contrast, by using pairwise alignments that incorporate consistency information from other sequences, we show that it is possible to obtain very good accuracy with the preserving alignment formulation. We show that a reasonable objective function to use is to find the shortest preserving alignment, and, by a reduction to a graph-theoretic problem, that the problem of finding the shortest preserving multiple alignment can be solved in polynomial time. We demonstrate the success of this approach on three sets of benchmark multiple alignments by using consistency-based pairwise alignments from the first stage of two of the best performing progressive alignment algorithms TCoffee (Notredame *et al.*, 2000) and ProbCons (Do *et al.*, 2005) and replace the second heuristic progressive step of these algorithms by the exact preserving alignment step. We apply this strategy to TCoffee and show that our approach outperforms TCoffee on two of the three test sets. We apply the strategy to a variant of ProbCons with no iterative refinements and show that our

approach achieves similar or better accuracy except on one test set. We also compare our performance to ProbCons with iterative refinements and show that our approach achieves similar or better accuracy on many subcategories even without further refinements. The most important advantage of the preserving alignment formulation is that we are certain that we can solve the problem in polynomial time without using a heuristic. A software tool implementing this approach (PSAlign) is available at <http://faculty.cs.tamu.edu/shsze/psalign>.

In Chapter III, by using additional hits from database search of the input sequences, a few strategies have been proposed to significantly improve alignment accuracy, including the construction of profiles from the hits while performing profile alignment, the inclusion of high scoring hits into the input sequences, the use of intermediate sequence search to link distant homologs, and the use of secondary structure information. We develop an algorithm that integrates these strategies to further improve alignment accuracy by modifying the pair-HMM approach in ProbCons to incorporate profiles of intermediate sequences from database search and utilize secondary structure predictions as in SPEM (Zhou and Zhou, 2005). We test our algorithm on a few sets of benchmark multiple alignments, including BAliBASE, HOMSTRAD, PREFAB and SABmark, and show that it significantly outperforms MAFFT (Katoh *et al.*, 2005) and ProbCons, which are among the best multiple alignment algorithms that do not utilize additional information, and SPEM, which is among the best multiple alignment algorithms that utilize additional hits from database search. The improvement in accuracy over SPEM

can be as much as 5 to 10% when aligning divergent sequences. A software tool that implements this approach (ISPAAlign) is at <http://faculty.cs.tamu.edu/shsze/ispalign>.

While most of the recent improvements in multiple sequence alignment accuracy are due to better use of vertical information, which include the incorporation of consistency-based pairwise alignments and the use of profile alignments, we observe that it is possible to further improve accuracy by taking into account alignment of neighboring residues when aligning two residues, thus making better use of horizontal information. By modifying existing multiple alignment algorithms to make use of horizontal information, In Chapter IV, we show that this strategy is able to consistently improve over existing algorithms on four sets of benchmark alignments that are commonly used to measure alignment accuracy, and the improvements in accuracy can be as much as a few percent. Unlike previous algorithms, consistent improvements can be obtained across all identity levels. A software tool that implements this approach (NRAAlign) is available at <http://faculty.cs.tamu.edu/shsze/nralign>.

CHAPTER II

PSAlign: A POLYNOMIAL TIME SOLVABLE FORMULATION OF MULTIPLE SEQUENCE ALIGNMENT^{*}

A. Introduction

The goal of the multiple alignment problem is to bring similar regions from different sequences as closely together as possible, with applications in diverse types of biosequence analysis (Taylor, 1987; Carillo and Lipman, 1988; Thompson *et al.*, 1994; Gotoh, 1996; Morgenstern *et al.*, 1996; Stoye 1998; Notredame *et al.*, 2000; Lee *et al.*, 2002; Edgar, 2004; Van Walle *et al.*, 2004; Do *et al.*, 2005). Since traditional multiple alignment formulations are NP-hard (Just, 2001), it is unreasonable to expect that one will ever be able to find an efficient approach that always returns an optimal alignment. The best known exact algorithm employs dynamic programming techniques with time complexity $O(n^k)$ (Carillo and Lipman, 1988), where n is the maximum sequence length and k is the number of sequences, and thus is useful only when k is small. Stoye (1998) proposed a divide-and-conquer heuristic to limit the search space by subdividing the input sequences into shorter segments, but it is not efficient enough for large-scale

^{*}Part of the data reported in this chapter is reprinted with permission from “A Polynomial Time Solvable Formulation of Multiple Sequence Alignment” by Sze, S.-H., Lu, Y. and Yang, Q., 2005, *RECOMB'2005, Lecture Notes in Computer Science*, 3500, 204-216. Copyright 2005 with kind permission of Springer Science+Business Media.

applications.

The inherent difficulty of the multiple alignment problem leads naturally to the development of heuristic approaches. Among the most successful of these are progressive approaches, which combine the given sequences in some order to obtain a multiple alignment (Feng and Doolittle, 1987; Thompson *et al.*, 1994; Notredame *et al.*, 2000; Edgar, 2004; Do *et al.*, 2005). These heuristics are often coupled with iterative refinement of the initial multiple alignment to obtain improved performance (Gotoh, 1996; Edgar, 2004; Do *et al.*, 2005). Alternatively, other non-progressive approaches assemble a final multiple alignment from short alignments of local similarities (Morgenstern *et al.*, 1996; Van Walle *et al.*, 2004). Although in most cases, a scoring scheme and an accompanying objective function can be defined for these heuristics, it is often unclear how close the final alignment is to the optimal. Some efforts have been spent to develop approximation algorithms for multiple alignment with guaranteed performance bound, but the theoretical bound is usually too weak to reflect the actual performance (Gusfield, 1993).

We propose an alternative formulation of multiple alignment that is solvable in polynomial time. Such a formulation is very important as it makes it possible to know what the alignment means and also ensures that the optimal solution can be found. Instead of employing objective functions that are very difficult to optimize, the formulation is based on the idea of finding a multiple alignment of k sequences which

preserves $k - 1$ pairwise alignments as specified by edges of a given tree. In particular, one can use the optimum spanning tree that includes the best $k - 1$ pairwise alignments covering all the sequences. Although it is well known that such a preserving alignment always exists (Feng and Doolittle, 1987; Gusfield, 1993; Pevzner, 2000), it did not become a mainstream method for multiple alignment since it seems that a lot of information is lost from ignoring pairwise similarities outside the tree.

The preserving alignment approach can be seen as a restricted version of a broader class of consistency based approaches, which aim to maximize the consistency between the resulting multiple alignment and a given set of pairwise alignments on aligned residue pairs (Kececioglu, 1993; Notredame *et al.*, 1998). A distinct advantage of these consistency-based approaches is that once the pairwise alignments are fixed, the multiple alignment follows logically without the need to define a multiple alignment score (Notredame *et al.*, 1998). Since the pairwise alignments are not restricted to be on a tree in this more general formulation, they can be conflicting, and thus the objective function is likely to be more accurate, but it is also likely to be intractable to optimize. In another direction, by incorporating consistency information from other sequences when computing individual pairwise alignments, these consistency-based pairwise alignments have been successfully used in the pairwise stage of progressive approaches to give some of the best performing multiple alignment approaches to date (Notredame *et al.*, 2000; Edgar 2004; Do *et al.*, 2005). By employing these consistency-based pairwise alignments, we will show that it is possible to obtain very good accuracy even with the

more restrictive preserving alignment formulation. Other studies that made use of the notion of consistency include Gotoh (1990) and Vingron and Argo (1991).

A complication with the preserving alignment formulation is that there may be many multiple alignments which preserve the given $k - 1$ pairwise alignments and previous studies did not suggest how to choose among them. Ideally, one would like to maximize the similarity level over all columns. However, many of the objective functions that attempt to exploit these similarities are likely to be intractable to optimize. One natural way that allows us to develop a tractable approach is to find the shortest preserving multiple alignment with the smallest number of columns, corresponding to adding as few gaps as possible while preserving the pairwise alignments along the tree edges. Without being able to control the similarity level in each individual column, this formulation discourages gaps (similarly to other traditional formulations) while making sure that the resulting multiple alignment resembles the given pairwise alignments. One important advantage of the preserving alignment formulation is that once the tree and the pairwise alignments are fixed, no additional parameters or a scoring scheme for multiple alignment are needed. This makes it possible to use any tree or pairwise alignments directly from other approaches, including ones that have made use of structural information. Similar ideas of utilizing structural pairwise alignments have been proposed by a few studies (O'Sullivan *et al.*, 2004; Van Walle *et al.*, 2004).

We will show, by a reduction to a graph-theoretic problem, that the problem of finding the shortest preserving multiple alignment can be solved in polynomial time, and, by using consistency-based pairwise alignments, that the accuracy of this exact approach is comparable to the best heuristic multiple alignment approaches on three sets of benchmark multiple alignments from Thompson *et al.* (1999), Edgar (2004), and Van Walle *et al.* (2004), thus justifying the use of the proposed polynomial time formulation over other NP-hard formulations. In particular, we reduce the multiple alignment problem to finding a topological partial ordering in a directed acyclic graph where each vertex represents a partially aligned column and unordered vertices are allowed to share the same label. The label assigned to each vertex from the ordering specifies its position in the multiple alignment, and the ordering itself represents a preserving multiple alignment.

B. Problem Formulation

Let $S = \{s_1, \dots, s_k\}$ be a given set of sequences. Assume that we are given a tree T with k vertices where each vertex of T is labeled by a distinct sequence and each edge (i, j) of T represents a pair of sequences s_i and s_j , and we are also given a pairwise alignment P_{ij} between sequences s_i and s_j for each edge (i, j) of T . A multiple alignment M of S is said to preserve all the $k - 1$ pairwise alignments on T if for each edge (i, j) of T , the induced pairwise alignment on sequences s_i and s_j is the same as P_{ij} when the columns containing

only gap characters are removed. Since the tree T specifies pairwise alignments that can be simultaneously preserved, it is obvious that such a preserving multiple alignment M always exists (Feng and Doolittle, 1987; Gusfield, 1993; Pevzner, 2000). Likewise, a multiple alignment M of S is said to preserve all matches and mismatches (or just the matches) in the $k - 1$ pairwise alignments on T if for each edge (i, j) of T , each column containing a match or a mismatch (or just a match) has to stay in the same column in M . We formulate the multiple alignment problem as follows: given a tree T and a pairwise alignment P_{ij} for each edge (i, j) of T , find a preserving multiple alignment M with the smallest number of columns.

The simplest way to obtain pairwise alignments is by applying standard techniques, including global (Needleman and Wunsch, 1970) and local (Smith and Waterman, 1981) approaches, or a combination of these approaches. However, we found that it is often better to use other kinds of pairwise alignments such as those that have incorporated consistency information from other sequences. These consistency-based pairwise alignments can be obtained from the pairwise stage of a few progressive approaches (Notredame *et al.*, 2000; Do *et al.*, 2005). From these pairwise alignments, one reasonable tree T to use is an optimum spanning tree on the complete graph C_k where each vertex is labeled by a distinct sequence and each edge (i, j) is labeled by the score of P_{ij} , which can either be the pairwise alignment score of P_{ij} or other scores given by the approach generating the pairwise alignments. To compute the optimum spanning tree from C_k , Prim's algorithm can be used, which has time complexity $O(k^2 \log k)$ (Cormen

et al., 2001). Alternatively, one can use a star tree with a central vertex and $k-1$ leaves (Gusfield, 1993; Pevzner, 2000). Although computational results show that using a star tree works well when all the given sequences are closely related, it does not give very good performance when none of the given sequences can act as the center, such as when there is more than one cluster of closely related sequences. In contrast, by using a general tree, it is possible to utilize the best non-conflicting pairwise alignments.

Note that the tree used here represents each sequence as a vertex, which is different from the phylogenetic tree (Morrison, 1996) or the guide tree (Thompson *et al.*, 1999) used in traditional progressive approaches in which sequences are represented only at the leaves. As a result, only alignments between sequences are needed in the preserving alignment formulation, which is similar to the comparisons made between sequences during the greedy extension step of the multiple alignment algorithm of Taylor (1987, 1988), while progressive approaches need to consider alignments between alignments (Altschul and Lipman, 1989). However, it is possible to make use of a phylogenetic tree when one is given. Instead of using pairwise alignment scores, we can use the distances between sequences s_i and s_j on the phylogenetic tree to compute an optimum spanning tree. In this case, we only need to compute $k - 1$ pairwise alignments along the spanning tree.

C. Exact Algorithm

For simplicity of analysis, assume that each of the input sequences is of the same length n . Given a tree T and a pairwise alignment P_{ij} for each edge (i, j) of T , we give an algorithm to solve the shortest preserving multiple alignment problem in linear time by two successive graph reductions. Gusfield (1993) gave an algorithm to solve the problem in the important special case when the given tree is a star. We first consider preserving only the matches and mismatches instead of entire pairwise alignments. We will show that this strategy also preserves the indel columns under normal situations. Let s_{ij} be the letter at the j th position of sequence s_i . The first reduction constructs an undirected graph $G = (V, E)$ as follows (see Figure 2.1 a–d):

Let $V = \{v_{ij}\}$, where v_{ij} represents the j th position of sequence s_i , and $E = \{\{v_{ip}, v_{jq}\} \mid (i, j) \in T \text{ and } (s_{ip}, s_{jq}) \text{ is a match or a mismatch column in } P_{ij}\}$.

Intuitively, E contains all the match and mismatch columns within the pairwise alignments along the edges of T and thus specifies exactly all the preservation constraints. The observation below follows directly from T being a tree and P_{ij} being pairwise alignments.

Proposition 2.1. *Each connected component C in G is a tree and contains at most one vertex from each sequence s_i .*

To obtain a preserving multiple alignment, letters within each connected component in G must be put into the same column. On the other hand, we are free to put two different connected components in the same column as long as they do not contain vertices from the same sequence. Also, when assigning components to different columns to obtain a multiple alignment, the order of the letters within each sequence must be maintained. To represent these constraints precisely, the second reduction constructs a directed graph $G' = (V', E')$ from G as follows (see Figure 2.1 e for a transitively reduced version of G'):

Let V' be the set of all connected components C in G and let $s(C)$ be the set of sequences in which the vertices in C reside. Connect a component C_1 to another component C_2 by a directed edge in E' if $s(C_1) \cap s(C_2) \neq \emptyset$ and for every sequence $s_i \in s(C_1) \cap s(C_2)$ shared by C_1 and C_2 , the vertex v_{ip} in C_1 appears before the vertex v_{iq} in C_2 (i.e., $p < q$). (2.1)

Note that two connected components that contain vertices from the same sequence are strictly ordered and thus will be connected by an edge (in one of the directions), since if there are two vertices v_{ip} and v_{jq} in C_1 and two vertices v_{ir} and v_{js} in C_2 with $p < r$, then we must have $q < s$. The reasoning is as follows. Let $v_{ip} = u_1, \dots, u_t = v_{jq}$ be the unique path between v_{ip} and v_{jq} in C_1 and $v_{ir} = w_1, \dots, w_t = v_{js}$ be the unique path between v_{ir} and v_{js} in C_2 . For $1 \leq l < t$, since u_l and w_l are on the same sequence, the adjacent pairs (u_l, u_{l+1}) and (w_l, w_{l+1}) represent two match or mismatch columns within one single

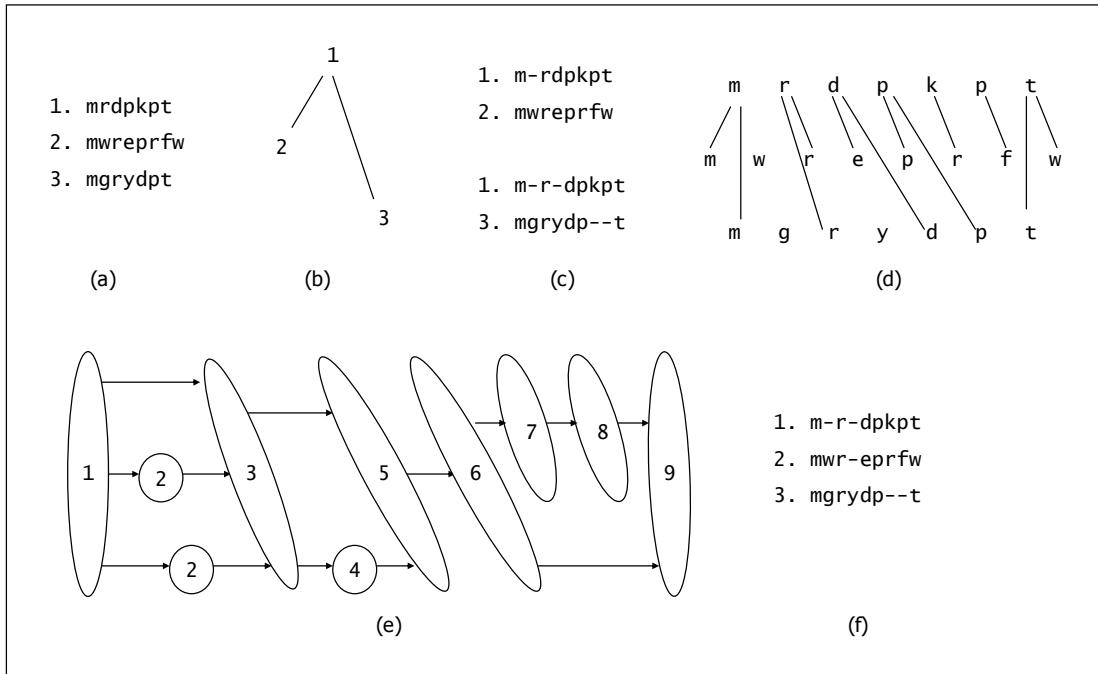


Figure 2.1: Illustration of the exact algorithm. (a) Set of sequences S . (b) Tree T . (c) Pairwise alignments P_{12} and P_{13} . (d) Undirected graph G constructed from S , T , P_{12} , and P_{13} . (e) Directed graph G' (transitively reduced) constructed from G by taking connected components in G as vertices. Labels of vertices in G' are assigned by the topological partial ordering algorithm. (f) Shortest preserving alignment by interpreting labels as columns.

pairwise alignment. The fact that $p < r$ and $q \geq s$ contradicts these being columns in the pairwise alignments. A more elaborate argument gives the following.

Proposition 2.2. G' is a directed acyclic graph.

Proof. Let C_1, \dots, C_t be a cycle in G' . Then there exist sequences s_{i1}, \dots, s_{it} such that v_{i1p1} is in C_1 and v_{i1q1} is in C_2 with $p_1 < q_1$, v_{i2p2} is in C_2 and v_{i2q2} is in C_3 with $p_2 < q_2$, and so on, until finally, v_{ip_t} is in C_t and v_{iq_t} is in C_1 with $p_t < q_t$. Between these vertices, there is a unique path $v_{i1q1}, \dots, v_{i2p2}$ on C_2 , $v_{i2q2}, \dots, v_{i3p3}$ on C_3 , and so on, until finally, $v_{iq_t}, \dots, v_{i1p1}$ on C_1 . Thus the cycle can be represented by the path $v_{i1p1} \xrightarrow{j} v_{i1q1} \xrightarrow{w} v_{i2p2} \xrightarrow{j} v_{i2q2} \xrightarrow{w} \dots \xrightarrow{w} v_{ip_t} \xrightarrow{j} v_{iq_t} \xrightarrow{w} v_{i1p1}$, where \xrightarrow{j} denotes a jump to a later connected component on the same sequence, and \xrightarrow{w} denotes walking along the tree edges in T within a connected component which visits each sequence at most once. On this path, whenever a sequence s is visited again, the position of visit on s must increase, since a jump increases the position of visit on the same sequence from p_l to q_l , at least one jump has to occur before s is visited again, and whenever a walk moves from s to another sequence t along a tree edge in T , the only way to return to s is through t along the same edge in T . Walking along T this way with no increase in the position of visit on s when s is revisited contradicts the given pairwise alignments. In particular, this is true for sequence s_{i1} , a contradiction to the above cycle which keeps the position of visit on s_{i1} at p_1 .

Since the primary purpose of G' is to specify ordering constraints, a transitively reduced version of G' suffices (see Figure 2.1 e). Instead of performing the transitive reduction step, such a graph G' can be obtained directly from G by requiring further that there

exists a sequence s_i such that $p + 1 = q$ in (2.1). This reduces the number of edges in G' substantially, and we will be using this definition in what follows. The above results suggest that a multiple alignment can be obtained by finding a topological partial ordering in G' .

Definition 2.3. *A topological partial ordering of a directed acyclic graph $G' = (V', E')$ is an assignment of an integer label $l(v)$ to each vertex $v \in V'$ such that for each edge $(u, v) \in E'$, $l(u) < l(v)$.*

Since it is possible that two vertices are assigned the same label, the result is not necessarily a total order (a total order corresponds to a topological sorting [Knuth, 1997]). Without loss of generality, assume that the labels are consecutive integers from 1 to m . From a given graph G' , there are many ways to realize such an ordering. For each fixed ordering, a multiple alignment can be obtained by putting each letter within a connected component C in G (C is a vertex in G') in column $l(C)$ and filling other unassigned spaces by gap characters (see Figure 2.1 f). The set of all possible ways to do this represents the solution space of all preserving alignments.

Proposition 2.4. *Each topological partial ordering of G' specifies a multiple alignment preserving all matches and mismatches in the $k - 1$ pairwise alignments on T .*

By making additional assumptions, it is possible to ensure that the resulting multiple alignment preserves the given pairwise alignments entirely, which includes the indel columns in addition to the match and the mismatch columns, without requiring an algorithm change. The observation below follows directly from the constraints imposed on the placement of gap characters between two match or mismatch columns that must be preserved and are separated only by indel columns.

Proposition 2.5. *If for each pairwise alignment P_{ij} on T , there do not exist two adjacent indel columns (without match or mismatch columns in between) such that the gap character is on sequence s_i in column l and on sequence s_j in column $l-1$ or $l+1$, then each topological partial ordering G' specifies a multiple alignment preserving the $k - 1$ pairwise alignments on T .*

It is very rare to be given pairwise alignments that violate the condition given in Proposition 2.5, and thus in most cases the resulting multiple alignment also preserves the given pairwise alignments entirely. In the other direction, one can relax the constraints by requiring only the match columns (or the columns with a positive score with respect to a given substitution matrix) to be preserved, which can be achieved by allowing only edges representing these columns to be added to G . Computational results show that simply finding the shortest preserving multiple alignment in this case does not give very good performance since the flexibility in the placements of the resulting

smaller connected components in G becomes excessive. The following observation completes the reduction.

Proposition 2.6. *A topological partial ordering of G' that uses the smallest number of labels specifies a shortest preserving multiple alignment.*

Since edges in G correspond to match or mismatch columns in the given $k - 1$ pairwise alignments along the given tree T and each pairwise alignment is of length $O(n)$, there are $O(kn)$ vertices and edges in G . Thus, there are $O(kn)$ connected components in G which make the vertices in G' . These connected components can be obtained in $O(kn)$ time by a depth-first search on G . In the simplest case, each connected component in G is of size one (which represents one position on a single sequence), and edges in G' are constructed between components that represent adjacent positions within the same sequence, resulting in a total of $O(kn)$ edges in G' . The graph G' in any other case with larger connected components can be obtained from this simplest case by merging the corresponding vertices and collapsing each resulting multiedge into a single edge, and thus the number of edges in G' is $O(kn)$ in all cases. To find a topological partial ordering that uses the smallest number of labels in G' , an algorithm very similar to the standard topological sorting algorithm (Knuth, 1997) can be used: initially, all vertices are unmarked. Repeatedly find an unmarked vertex v with all its incoming vertices marked. Set the label of v to be one plus the maximum label among all its incoming vertices and mark v (see Figure 2.1 f). If a count is kept in each vertex representing the

number of remaining unmarked incoming vertices, the algorithm can be implemented in time linear in the number of edges in G' . Thus, with appropriate data structures, the overall time complexity of the entire procedure is $O(kn)$, which is linear in the input size. If it is not important to obtain a totally ordered multiple alignment, it is possible to return the graph G' directly as a partially ordered multiple alignment, which is similar in concept but different in structure to the notion of partial order multiple alignment proposed by Lee *et al.* (2002). In this case, there is no need to define any objective function.

D. Performance

We evaluate the accuracy of the preserving alignment algorithm (PSAlign) on three sets of benchmark multiple alignments: BAliBASE from Thompson *et al.* (1999), PREFAB from Edgar (2004), and SABmark from Van Walle *et al.* (2004). We compare our performance to TCoffee (Notredame *et al.*, 2000) and ProbCons (Do *et al.*, 2005), which are currently considered to be among the best multiple alignment algorithms. To make fair comparisons, we compare our performance (PSAlign[TCoffee]) to TCoffee when pairwise alignments from TCoffee are used, and we compare our performance (PSAlign[ProbCons]) to ProbCons when pairwise alignments from ProbCons are used. For TCoffee, we use pairwise alignments computed from the extended library that have incorporated consistency information from other sequences (Notredame *et al.*, 2000).

For ProbCons, we use pairwise alignments computed after consistency transformation (Do *et al.*, 2005). All these pairwise alignments incorporate consistency information from other sequences, and it has been shown that this significantly improves the quality of the pairwise alignments with respect to the overall consistency. We then compute an optimum spanning tree from these consistency-based pairwise alignments using pairwise scores given by the two algorithms (normalized by the length of each pairwise alignment) and apply the preserving alignment step. Since our main goal is to show that the heuristic progressive step of these approaches can be replaced by the exact preserving alignment step, we compare to a variant of ProbCons with no iterative refinements. We also compare our performance (without performing further refinements) to ProbCons with iterative refinements.

Following Thompson *et al.* (1999), two score measures are used to evaluate the accuracy of each algorithm in finding the core blocks in BAliBASE (which are annotated regions that can be reliably aligned): the sum-of-pairs score (SPS) measures how well an algorithm can align pairs of residues within the same column correctly, while the column score (CS) measures how well an algorithm can align entire columns correctly. For PREFAB, we follow Edgar (2004) and use the Q score, which has the same meaning as SPS used in BAliBASE. For SABmark, we define the Q score for each test case as the average Q score over all pairs of reference sequences. For both PREFAB and SABmark, the reference alignments are based on pairwise comparisons, and thus the CS score is not applicable. For each test set, we compare average accuracy over meaningful subsets and

use the Wilcoxon matched-pairs signed-ranks test (Wilcoxon, 1947) to check whether there are significant performance differences with $p = 0.05$ as cutoff. Note that in the preserving alignment computation, the shortest solution is not necessarily unique and we simply report an arbitrary one.

Tables 2.1, 2.2 and 2.3 show performance comparisons of the various algorithms on BALiBASE, PREFAB, and SABmark, respectively. A general trend was that ProbCons tends to perform better than TCoffee, whether PSAlign is used or not. When we consider the ability of PSAlign[TCoffee] to replace TCoffee and the ability of PSAlign[ProbCons] to replace ProbCons, PSAlign was a much better replacement when used in conjunction with TCoffee than with ProbCons and the only case where PSAlign[TCoffee] is worse than TCoffee was in reference 4 of BALiBASE. Also, PSAlign was a better replacement when used on SABmark than on BALiBASE, but PSAlign[ProbCons] was not an adequate replacement when used on PREFAB.

On BALiBASE, when compared to TCoffee and ProbCons ($ir = 0$) that do not perform iterative refinements, PSAlign had better accuracy on references 1V1 and 5, but this was offset by worse accuracy on references 3 and 4. Although there were no noticeable differences in the overall accuracy, the Wilcoxon matched-pairs test revealed that ProbCons ($ir = 0$) performed better than PSAlign[ProbCons] in the SPS score with $p = 0.02$. The differences in all the other cases on the overall accuracy were insignificant with TCoffee or in the CS score. We did not apply the Wilcoxon test to any of the

Table 2.1: Average SPS and CS scores (in %) of PSAlign on BAliBASE 2.01. Reference 1 is further subdivided into three subsets: V1 (< 25% identity), V2 (20%–40% identity) and V3 (> 35% identity). Comparisons are made between TCoffee 1.37 and PSAlign utilizing TCoffee pairwise alignments (PSAlign[TCoffee]) and between ProbCons 1.10 and PSAlign utilizing ProbCons pairwise alignments (PSAlign[ProbCons]). No iterative refinements are performed for ProbCons ($ir = 0$). Default parameters are used otherwise, with ProbCons ($ir = 100$) performing 100 rounds of iterative refinements.

<i>SPS</i>	<i>IV1</i>	<i>IV2</i>	<i>IV3</i>	<i>I(Overall)</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>Overall</i>
TCoffee	64.1	95.2	98.5	87.6	93.5	78.6	93.9	96.0	89.1
PSAlign[TCoffee]	67.3	95.1	98.4	88.4	93.6	78.5	89.1	97.3	89.2
ProbCons($ir = 0$)	69.1	96.6	98.5	89.5	94.2	84.0	90.8	97.4	90.6
ProbCons($ir = 100$)	74.5	96.8	98.5	91.1	94.2	84.0	93.7	97.4	91.8
PSAlign[ProbCons]	71.5	96.6	98.4	90.1	94.0	80.9	90.1	98.0	90.7
<i>CS</i>	<i>IV1</i>	<i>IV2</i>	<i>IV3</i>	<i>I(Overall)</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>Overall</i>
TCoffee	41.5	90.9	96.8	79.1	58.4	50.9	80.4	90.3	74.4
PSAlign[TCoffee]	47.3	90.6	96.5	80.5	58.3	54.8	68.4	90.0	74.5
ProbCons($ir = 0$)	49.6	93.4	96.9	82.3	61.6	63.5	72.1	89.3	77.1
ProbCons($ir = 100$)	59.6	94.0	96.9	85.3	61.6	63.5	81.1	89.3	79.6
PSAlign[ProbCons]	55.8	93.5	96.6	84.0	61.7	52.2	69.7	93.6	77.2

Table 2.2: Average Q scores (in %) of PSAlign on PREFAB 4.0. Each subset includes all structure pairs with identity within the specified range.

	0–20%	20–40%	40–70%	70–100%	Overall
TCoffee	49.2	82.9	93.8	95.1	66.5
PSAlign[TCoffee]	51.0	84.6	95.2	97.9	68.3
ProbCons($ir = 0$)	55.6	87.2	95.4	97.3	71.7
ProbCons($ir = 100$)	55.6	87.2	95.4	97.3	71.7
PSAlign[ProbCons]	52.1	85.2	93.0	94.2	68.8

subsets of BALiBASE due to their small sizes. When compared to ProbCons ($ir = 100$) that performs iterative refinements, PSAlign[ProbCons] still maintained better accuracy on reference 5, but it no longer had better accuracy on reference 1V1. The Wilcoxon test revealed that ProbCons ($ir = 100$) had significantly better overall accuracy than PSAlign[ProbCons] in both the SPS and the CS scores with $p < 0.001$, which was mainly due to large accuracy improvements on references 1V1 and 4 from iterative refinements (no noticeable improvements were observed on the other references).

On PREFAB, PSAlign[TCoffee] had better accuracy than TCoffee in all five categories and these improvements were significant with $p < 0.001$ for the subsets with 0% to 20% identity, with 20% to 40% identity, with 70% to 100% identity, and for the entire set. On

Table 2.3: Average Q scores (in %) of PSAlign on SABmark 1.65. The FP variant of the two subsets includes false positive sequences.

	<i>Superfamily</i>	<i>Superfamily-FP</i>	<i>Twilight</i>	<i>Twilight-FP</i>	<i>Overall</i>
TCoffee	52.9	45.6	23.7	17.0	39.8
PSAlign[TCoffee]	54.8	54.1	25.8	25.4	45.0
ProbCons($ir = 0$)	56.7	52.5	28.6	23.0	45.2
ProbCons($ir = 100$)	57.1	53.2	29.3	24.0	45.8
PSAlign[ProbCons]	56.1	53.6	28.1	25.1	45.6

the other hand, ProbCons ($ir = 0$) performed significantly better than PSAlign[ProbCons] in all five categories with $p < 0.001$, while no noticeable improvements were observed with ProbCons ($ir = 100$) over ProbCons ($ir = 0$).

On SABmark, PSAlign[TCoffee] showed highly significant improvements over TCoffee with $p < 0.001$ in all five categories. However, on the Superfamily and Twilight subsets, ProbCons ($ir = 0$) performed significantly better than PSAlign[ProbCons] (with $p < 0.001$ for the Superfamily subset and $p = 0.03$ for the Twilight subset). The situation was reversed on the Superfamily-FP and Twilight-FP subsets when PSAlign[ProbCons] performed significantly better than ProbCons ($ir = 0$) (with $p = 0.01$ for Superfamily-FP subset and $p < 0.001$ for Twilight-FP subset), while the difference between ProbCons ($ir = 0$) and PSAlign[ProbCons] on the overall accuracy was insignificant. Although

ProbCons ($ir = 100$) further increased the performance differences from PSAlign[ProbCons] on the Superfamily and Twilight subsets to $p < 0.001$ in both cases, PSAlign[ProbCons] was able to maintain significantly better accuracy than ProbCons ($ir = 100$) on the Twilight-FP subset with $p < 0.001$ while having an insignificant difference in accuracy on the Superfamily-FP subset. Although ProbCons ($ir = 100$) performed significantly better than PSAlign[ProbCons] on the overall accuracy with $p = 0.003$, one important advantage of PSAlign is that it had a much smaller accuracy decrease when either the Superfamily or Twilight subset is replaced by its FP variant with false positive sequences.

Overall, PSAlign[TCoffee] performed at least as well as TCoffee on BALiBASE and was much better than TCoffee on PREFAB and SABmark. When compared to ProbCons ($ir = 0$), PSAlign[ProbCons] achieved similar or better accuracy on BALiBASE and SABmark, but did not perform as well on PREFAB. When compared to ProbCons ($ir = 100$), PSAlign[ProbCons] achieved similar or better accuracy on many subcategories even without further refinements, but had worse overall accuracy. These results did not lead to a conclusive statement that shows that using PSAlign has a definite advantage or disadvantage, and thus it is hard to predict whether there will be significant accuracy differences if we replace the heuristic progressive step of some given multiple alignment algorithm by the exact preserving alignment step. Nevertheless, the most important advantage of the preserving alignment formulation is that we are certain that we can solve the problem in polynomial time without using a heuristic. Since the time

complexity is dominated by the computations of the pairwise alignments, the preserving alignment step does not add much to the running time. What we actually observe was at most a two times slowdown when PSAlign was used to replace TCoffee or ProbCons, due to the need to compute all consistency-based pairwise alignments to obtain the optimum spanning tree.

E. Discussion

The proposed multiple alignment formulation divides the multiple alignment problem into two sub-problems. The first sub-problem requires the computation of pairwise alignments and a tree, which can be defined systematically so that optimal solutions can be computed in polynomial time. For example, one can use a technique similar to that used by Notredame *et al.* (2000) to compute consistency-based pairwise alignments based on comparisons of three sequences so that each of them can be computed in $O(kn^2)$ time. It is especially important to obtain high quality pairwise alignments in this stage, since we found that good accuracy cannot be obtained when simple non-consistency-based pairwise alignments are used. This was confirmed by a much bigger decrease in accuracy and a much worse performance of PSAlign[ProbCons] in most cases as compared to ProbCons($ir = 0$) when the consistency transformation in ProbCons was disabled (Table 2.4). The second stage computes a shortest preserving multiple

Table 2.4: Performance of ProbCons ($ir = 0$) and PSAlign[ProbCons] when consistency transformation in ProbCons is disabled. Both algorithms use ordinary pairwise alignments instead of consistency-based pairwise alignments. (a) Average SPS and CS scores (in %) on BALiBASE 2.01. (b) Average Q scores (in %) on PREFAB 4.0. (c) Average Q scores (in %) on SABmark 1.65.

(a) BALiBASE										
<i>SPS</i>	<i>IV1</i>	<i>IV2</i>	<i>IV3</i>	<i>I(Overall)</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>Overall</i>	
ProbCons($ir = 0$)	63.2	95.5	98.3	87.4	93.3	82.4	88.6	94.7	88.7	
PSAlign[ProbCons]	65.2	93.3	97.1	86.7	90.8	76.6	85.2	93.0	86.9	
<i>CS</i>										
	<i>IV1</i>	<i>IV2</i>	<i>IV3</i>	<i>I(Overall)</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>Overall</i>	
ProbCons($ir = 0$)	42.7	91.1	96.5	79.4	56.4	50.9	64.7	81.2	72.1	
PSAlign[ProbCons]	44.6	86.7	94.1	77.4	47.7	44.5	57.1	80.6	68.3	
(b) PREFAB										
	<i>0–20%</i>	<i>20–40%</i>	<i>40–70%</i>	<i>70–100%</i>	<i>Overall</i>					
ProbCons($ir = 0$)	52.6	85.1	95.5	97.6	69.4					
PSAlign[ProbCons]	41.5	80.0	92.2	96.0	61.4					
(c) SABmark										
	<i>Superfamily</i>	<i>Superfamily-FP</i>	<i>Twilight</i>	<i>Twilight-FP</i>	<i>Overall</i>					
ProbCons($ir = 0$)	54.8	50.5	26.3	20.3	43.1					
PSAlign[ProbCons]	51.5	50.8	24.3	23.0	42.2					

alignment from this information, which can be used to replace the progressive step of any approach in which unified pairwise alignments are available before the progressive step, or as a second step to construct multiple alignments for algorithms that only produce a set of pairwise alignments from the given sequences (Heger *et al.*, 2003; Van Walle *et al.*, 2004), without requiring additional parameters.

The graph-theoretic technique employed allows further extensions to consider more general models of pairwise similarity. In its full generality, all we need from each pairwise comparison is an ordered list of non-intersecting connections (representing matches or mismatches) that reflect significant pairwise similarities. With these inputs, the preserving alignment approach naturally returns either local or incomplete multiple alignments. To further improve accuracy, it is possible to consider formulations other than finding the shortest solution, although many of these objective functions may be intractable to optimize. One possible strategy is to employ various heuristics to find a preserving alignment from G' that tries to assign related connected components in G to the same column as much as possible. Other directions include improving the quality of the pairwise alignments and devising strategies to perform iterative refinements (Gotoh, 1996; Edgar, 2004; Do *et al.*, 2005) after the preserving multiple alignment is obtained.

CHAPTER III

ISPAAlign: MULTIPLE SEQUENCE ALIGNMENT BASED ON PROFILE ALIGNMENT OF INTERMEDIATE SEQUENCES^{*}

A. Introduction

Although many algorithms have been proposed for multiple sequence alignment (Thompson *et al.*, 1994; Morgenstern *et al.*, 1996; Stoye, 1998; Notredame *et al.*, 2000; Lee *et al.*, 2002; Edgar, 2004; Van Walle *et al.*, 2004; Do *et al.*, 2005; Katoh *et al.*, 2005; Lassmann and Sonnhammer, 2005; Pei and Grishin, 2006; Roshan and Livesay, 2006; Yamada *et al.*, 2006), it remains difficult to obtain accurate alignments. Common techniques to improve alignment accuracy include performing iterative refinements after the initial alignment is constructed (Gotoh, 1996; Edgar, 2004; Do *et al.*, 2005; Roshan and Livesay, 2006; Yamada *et al.*, 2006), using consistency-based pairwise alignments in progressive approaches (Notredame *et al.*, 2000; Do *et al.*, 2005; Pei and Grishin, 2006; Roshan and Livesay, 2006), and incorporating structural alignments (O’Sullivan *et al.*, 2004; Van Walle *et al.*, 2004). A few other strategies combine alignments from existing algorithms to obtain an improved alignment (Bucka-Lassen *et al.*, 1999; Wallace *et al.*, 2006).

^{*}Part of the data reported in this chapter is reprinted with permission from “Multiple sequence alignment based on profile alignment of intermediate sequences” by Lu, Y. and Sze, S.-H., 2007, *RECOMB'2007, Lecture Notes in Computer Science*, 4453, 283-295. Copyright 2007 with kind permission of Springer Science+Business Media.

With the rapidly increasing number of sequences in biological databases, it has been observed that the use of additional sequences from database search can significantly improve alignment accuracy. Among the most successful approaches that use this strategy are profile alignment algorithms that use database search to find related sequences for each input sequence, construct a profile from the hits, and then align the profiles instead of the sequences, including algorithms that start from two sequences (Marti-Renom *et al.*, 2004), and algorithms that start from multiple sequences (Simossis *et al.*, 2005; Zhou and Zhou, 2005). Alternatively, Heger *et al.* (2004) identified clusters of residues to form columns of a multiple alignment by linking distant homologs through the hits.

We observe that instead of constructing a profile for each input sequence from the hits, which only compares each hit to the input sequence that generates it, it may be more accurate to perform a more extensive multiple alignment of the hits together with the input sequences, which allows comparisons among all the sequences involved. The usefulness of such a strategy has been demonstrated during the construction of the PREFAB database (Edgar, 2004), in which the incorporation of additional hits from database search into the input sequences significantly improves accuracy as opposed to aligning the input sequences alone. One drawback of this approach is that the inclusion of hits that are not intermediate between the input sequences can introduce noise, since these hits do not contribute to defining a better alignment between them. We will show that a careful definition of intermediate sequences from database search in addition to

the computation of profiles for these sequences will significantly improve alignment accuracy.

By defining an intermediate sequence as a common hit from database search that links two input sequences, an intermediate sequence search technique has been used successfully to establish distant homologs (Park *et al.*, 1997; Gerstein, 1998). The strategy was later generalized to multiple intermediate sequence search (Salamov *et al.*, 1999; Li *et al.*, 2000), in which chains of intermediate sequences found through iterative database search are used to link very distant homologs. Bolten *et al.* (2001) used such transitive homologies to cluster protein sequences for structure predictions. Heger *et al.* (2004) used a graph-theoretic approach to link intermediate sequences through transitive homologies to detect short active site motifs, while Margelevičius and Venclovas (2005) used the intermediate sequence search strategy to distinguish between reliable and unreliable regions in alignments. Instead of defining intermediate sequences as common hits, we will develop a more relaxed definition to maximize the amount of information that can be extracted from the hits.

Since the number of hits that are also intermediate sequences can be very large, it is not practical to simply add them to the input sequences and perform a multiple alignment on the combined sequence set. Motivated by the fact that similar sequences are likely to contain redundant information, our algorithm uses a greedy strategy to choose a small subset of intermediate sequences that are far away from each other, which, together with

the original sequences form a combined set of input sequences. Instead of aligning these sequences directly, we construct a profile for each sequence in the combined set by incorporating information from other intermediate sequences and aligning the profiles by modifying the pair-Hidden Markov Model (HMM) approach (Durbin *et al.*, 1998) in ProbCons (Do *et al.*, 2005). This is in contrast with the strategy used in Simossis *et al.* (2005) and Zhou and Zhou (2005), which constructs a profile from the hits of an input sequence. We will show that our strategy of constructing profiles from intermediate sequences instead of from the hits helps to prevent the introduction of excessive noise when aligning closely related sequences. To further improve alignment accuracy, we obtain a secondary structure prediction for each sequence in the combined set and incorporate these predictions into the pair-HMM alignment. While this strategy of using secondary structure predictions is similar to the one employed in Zhou and Zhou (2005), it is different from the technique used in Pei and Grishin (2006) which employs secondary structure information during HMM training without explicitly using secondary structure predictions in alignments.

We compare the performance of our algorithm to MAFFT (Katoh *et al.*, 2005) and ProbCons (Do *et al.*, 2005), which are among the best multiple alignment algorithms that do not utilize additional information, and SPEM (Zhou and Zhou, 2005), which is among the best multiple alignment algorithms that utilize additional hits from database search, on benchmark multiple alignments from BAliBASE (Thompson *et al.*, 2005), HOMSTRAD (Mizuguchi *et al.*, 1998), PREFAB (Edgar, 2004), and SABmark (Van

Walle *et al.*, 2004). We will show that our algorithm outperforms MAFFT, ProbCons, and SPEM in almost all situations, with very significant improvements when aligning divergent sequences. Before presenting the algorithm in detail, we first describe the general strategies employed in each stage in the next few sections.

B. Finding Intermediate Sequences

Although most intermediate sequence search strategies define an intermediate sequence either as a common hit from database search that links two input sequences (Park *et al.*, 1997; Gerstein, 1998), or as hits that form a chain linking two input sequences (Salamov *et al.*, 1999; Li *et al.*, 2000), such a requirement is very stringent since it may not be possible to link very divergent sequences together even if the database search is performed iteratively. We consider the following relaxed definition of an intermediate sequence which only requires that it is intermediate between the two input sequences.

Definition 3.1. Given two sequences s_1 and s_2 , and a distance score $d(s_1, s_2)$ between them, a sequence r is intermediate between s_1 and s_2 if $d(r, s_1) < d(s_1, s_2)$ and $d(r, s_2) < d(s_1, s_2)$.

The problem of finding intermediate sequences between multiple input sequences is defined as follows.

Definition 3.2. Given n input sequences s_1, \dots, s_n , and m hits r_1, \dots, r_m from database search of these sequences, find all hits r_k that are intermediate between some pair of input sequences s_i and s_j .

Similar to previous approaches, our goal is to find an appropriate subset of sequences that contain useful information between the input sequences s_1, \dots, s_n . We do not require that these intermediate sequences have a phylogenetic interpretation or have an appropriate evolutionary relationship to the input sequences. Also, since any hit that is intermediate between some pair of input sequences is potentially useful, it is included in the definition. Note that there is no need to compute pairwise distances between the potentially very large number of hits. The number of pairwise distance score computations that are needed to identify the intermediate sequences from among the hits is $O(mn+n^2)$, while the number of score comparisons is $O(mn^2)$.

C. Choosing Intermediate Sequences

The next problem of choosing a small subset of intermediate sequences to add to the input sequences is defined as follows. Our goal is to identify a combined set of sequences that are as divergent as possible.

Input: n input sequences s_1, \dots, s_n , m intermediate sequences r_1, \dots, r_m ,

distance score $d(r, s)$ between two sequences r and s .

Output: k intermediate sequences s_{n+1}, \dots, s_{n+k} added to s_1, \dots, s_n .

```

 $R \leftarrow \{ r_1, \dots, r_m \};$ 
for each  $r_i$  in  $R$  do  $\{ d_i \leftarrow \min_{1 \leq j \leq n} d(r_i, s_j); \}$ 
for  $j \leftarrow 1$  to  $k$  do {
     $s_{n+j} \leftarrow r_i$  with the maximum  $d_i$ ; remove  $r_i$  from  $R$ ;
    for each  $r_i$  in  $R$  do  $\{ d_i \leftarrow \min(d_i, d(r_i, s_{n+j})); \} \}$ 
}

```

Figure 3.1: Greedy algorithm to choose a small subset of intermediate sequences to add to the input sequences.

Definition 3.3. Given n input sequences s_1, \dots, s_n , m intermediate sequences r_1, \dots, r_m , add k intermediate sequences from among r_1, \dots, r_m , denoted by s_{n+1}, \dots, s_{n+k} , so that the minimum distance between sequences in the combined set s_1, \dots, s_{n+k} is the largest possible when distances between the input sequences s_1, \dots, s_n are ignored.

Figure 3.1 shows a greedy algorithm that iteratively adds an intermediate sequence s_{n+j} that is farthest away from the current sequence set s_1, \dots, s_{n+j-1} , in which the minimum

distance between s_{n+j} and s_1, \dots, s_{n+j-1} is the largest possible. Although the greedy strategy does not guarantee optimum divergence of the sequences s_1, \dots, s_{n+k} , they should be reasonably far away from each other. The total number of pairwise distance score computations needed is $O(m(n + k))$, and there is no need to compute distances between all pairs of the potentially very large number of intermediate sequences.

D. Constructing Sequence Profiles

Instead of aligning the sequences s_1, \dots, s_{n+k} directly, a profile is constructed for each of these sequences as follows: for each intermediate sequence r_i from among r_1, \dots, r_m , assign it to the s_j from among s_1, \dots, s_{n+k} that is most similar to r_i . For each sequence s_j with assigned sequences r_{i1}, \dots, r_{iu} , we combine all the pairwise alignments between s_j and each r_{ip} into a star alignment with s_j as the center (Gusfield, 1993). For each column in the star alignment that contains a residue of s_j , the relative frequency of each residue within the column is then used to construct a profile as a probability distribution of residues (gap characters are ignored). Here the choice of scoring functions for the profile is not very important since Edgar and Sjölander (2004) showed that most scoring functions do not have significant performance differences. One caution is that we need to make sure that the number of very closely related sequences assigned to each s_j is not excessively large to avoid over-contribution of these sequences to the profile. This can be achieved by removing sequences from the original set of intermediate sequences so

that none of the remaining sequences are very similar to each other before choosing the subset of intermediate sequences. Different from the approach in Simossis *et al.* (2005) and Zhou and Zhou (2005), hits that are not intermediate sequences are not used to avoid noise from these hits.

E. Alignment via Modified Pair-HMM

We modify the pair-HMM approach in Durbin *et al.* (1998) to incorporate profiles and secondary structure predictions. The original model consists of three states: M emits an aligned pair of residues (x, y) with probability $e(x, y)$, X emits a residue x in the first sequence that is aligned to a gap with probability $e(x)$, while Y emits a residue y in the second sequence that is aligned to a gap with probability $e(y)$ (Figure 3.2). In addition to the original residue, each position is now associated with a probability distribution of residues which corresponds to the profile at that position. Let $p_1(x, i)$ be the probability of finding the residue x at position i in the first sequence and let $p_2(y, j)$ be the probability of finding the residue y at position j in the second sequence. We modify the model to incorporate profiles as follows: define the emission probability of state M as $e'(i, j) = \sum_x \sum_y p_1(x, i)p_2(y, j)e(x, y)$ if the emission is at position i in the first sequence and at position j in the second sequence, the emission probability of state X as $e'(i) = \sum_x p_1(x, i)e(x)$ if the emission is at position i in the first sequence, and the emission probability of state Y as $e'(j) = \sum_y p_2(y, j)e(y)$ if the emission is at position j in the

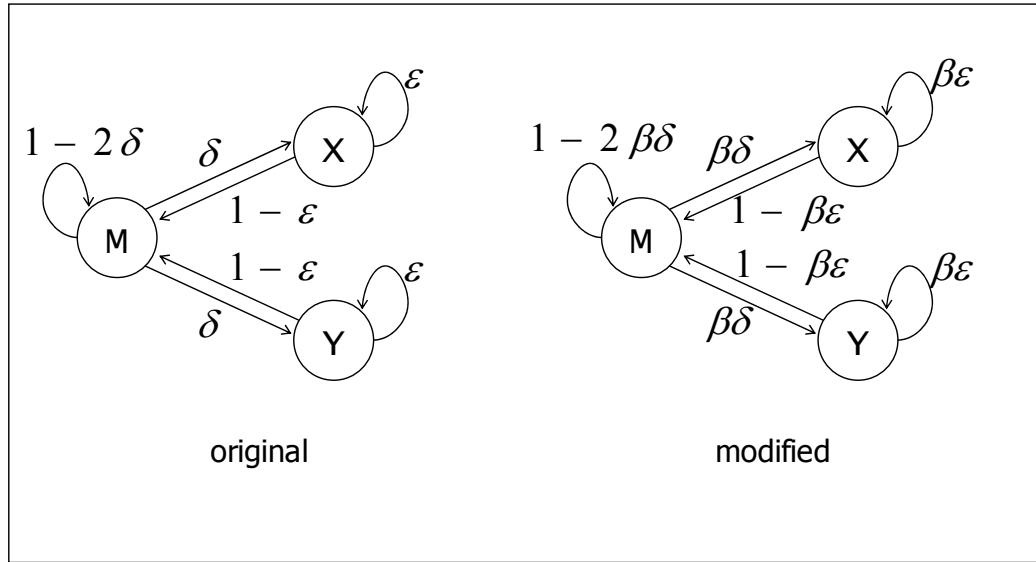


Figure 3.2: The original and the modified pair-HMM models. In the original model, state M emits an aligned pair of residues, states X and Y emit a residue in the first and the second sequences respectively that is aligned to a gap, δ is the gap opening probability, and ε is the gap extension probability (Durbin *et al.*, 1998). In the modified model, the state $M(\alpha)$ is obtained from M with emission probability $e(x, y)$ by defining the emission probability to be $\alpha e(x, y)$ if the paired residues (x, y) have the same secondary structure type and $(1-\alpha)e(x, y)$ otherwise. The factor β is applied to δ and ε to compensate for the change. To incorporate profiles, the residue emission probabilities are replaced by the average emission probabilities over a distribution of residues.

second sequence. These changes replace the original emission probabilities of the single residues by the average emission probabilities over a distribution of residues so that in

the degenerate case when the profiles represent simple sequences, the effect is the same as before.

We incorporate secondary structure predictions into the pair-HMM model as follows: in state M , we introduce an additional parameter α and subdivide the emission probability $e'(i, j)$ into two cases to obtain a modified state $M(\alpha)$ with emission probability $\alpha e'(i, j)$ if the original paired residues (x, y) at position i in the first sequence and at position j in the second sequence have the same secondary structure type, and with emission probability $(1-\alpha)e'(i, j)$ otherwise. Since this decrease in emission probability will tend to allow more gaps than before in the ideal case in which every aligned residue pair has the same secondary structure type, we apply the factor β to the gap opening and extension probabilities to compensate for it while keeping the ratio between the two probabilities unchanged to preserve the affine gap model (Figure 3.2). This modified pair-HMM can then be utilized within a progressive alignment strategy to obtain a multiple alignment (Do *et al.*, 2005).

F. Detailed Algorithm

We now describe a procedure and the associated parameters that give very good results for our algorithm. Note that this is only among one of the many possible ways to implement the algorithm.

Following SPEM (Zhou and Zhou, 2005), for each input sequence, we use PSI-BLAST (Altschul *et al.*, 1997) to perform database search on a filtered version of the non-redundant protein database (NR) that excludes low complexity regions, transmembrane regions and likely coiled-coil regions (Jones, 1999), and retain hits that have less than 98% identity to the input sequence and have *e*-value less than 0.001. One advantage of using PSI-BLAST is that it performs iterative database search automatically to look for distant homologs. Instead of keeping the entire sequence of a hit, only the regions within a PSI-BLAST local alignment are retained to avoid the introduction of noise from unrelated regions. Note that if there are more than one PSI-BLAST local alignment that satisfy the above condition within a hit, they are considered to be separate hits.

We then extract intermediate sequences from among these hits according to Definitions 3.1 and 3.2. To obtain an accurate distance score $d(s_1, s_2)$ between two sequence s_1 and s_2 , we use SSEARCH (Smith and Waterman, 1981) to obtain an optimal alignment between s_1 and s_2 and define $d(s_1, s_2)$ as the *e*-value of the alignment. Note that the use of *e*-values here does not pose any problems since no addition operations are performed.

To avoid over-contribution of very similar intermediate sequences in the later profile construction step, we use CD-HIT (Li *et al.*, 2002) to remove some of the closely related sequences so that the identity between the remaining intermediate sequences is less than 85%. We then use Definition 3.3 and the algorithm in Figure 3.1 with $k = 5$ to add at

most five intermediate sequences to the input sequences to obtain s_1, \dots, s_{n+k} . We choose $k = 5$ so that the final multiple alignment step will not become much slower than simply aligning the original input sequences. The identity of a pairwise alignment from SSEARCH is used to obtain an accurate distance score $d(s_1, s_2)$ between two sequences s_1 and s_2 by defining $d(s_1, s_2)$ as $1 - \text{identity}$ (note that this distance is different from what we use above). Note that CD-HIT cannot be used for this purpose since it initially uses counts of short tuples to estimate pairwise similarity, which is inaccurate when the identity level between the sequences s_1, \dots, s_{n+k} is low.

We then construct profiles according to the algorithm in Section D in which an intermediate sequence r_i is assigned to the sequence from among s_1, \dots, s_{n+k} that has the best SSEARCH alignment to r_i . To obtain a secondary structure prediction for each of the sequences s_1, \dots, s_{n+k} , we follow SPEM (Zhou and Zhou, 2005) and use PSIPRED (Jones, 1999) to assign one of the three possible types (helix, strand, or coil) to each residue.

With the profiles and secondary structure predictions, we modify ProbCons (Do *et al.*, 2005) by changing its pair-HMM model according to Section E. The parameters in Figure 3.2 are as follows: the original residue emission probabilities and the transition probabilities δ and ε are from ProbCons. The parameter α that modifies the emission probabilities is 0.65, while the parameter β that modifies the transition probabilities is 0.75. These two parameters are determined by testing a few combinations and choosing

one that gives satisfactory performance in PREFAB (Edgar, 2004). We use the default setting in ProbCons that utilizes two sets of gap states with the same modifying parameter β for both sets. There is no change in the later progressive alignment or the iterative refinement steps and the alignment on the original input sequences is returned.

G. Performance

We test our algorithm (ISPAAlign) on benchmark multiple alignments from BALiBASE 3.0 (Thompson *et al.*, 2005), HOMSTRAD (Mizuguchi *et al.*, 1998), PREFAB 4.0 (Edgar, 2004), and SABmark 1.65 (Van Walle *et al.*, 2004). We compare our performance to MAFFT 5.8 (using the most accurate linsi strategy, Katoh *et al.*, 2005), ProbCons 1.10 (Do *et al.*, 2005) and SPEM (Zhou and Zhou, 2005).

For BALiBASE and HOMSTRAD, two score measures are used to perform accuracy assessment of each multiple alignment on the original input sequences: the sum-of-pairs score (SPS) evaluates the percentage of residue pairs that an algorithm can align correctly in the reference alignment, while the column score (CS) evaluates the percentage of entire columns that an algorithm can align correctly (Thompson *et al.*, 1999). For PREFAB, evaluations are made on the original pairs of input sequences using the Q score defined in Edgar (2004), which has the same meaning as the SPS score. For BALiBASE and PREFAB, evaluations are made only on the core regions that are

assigned to the reference alignments. While we test MAFFT and ProbCons both on the original pairs in PREFAB and on the full set of sequences that includes random hits from database search, we test SPEM and ISPAAlign only on the original pairs since these algorithms utilize hits from database search automatically. For SABmark, reference sequences are specified in pairs and evaluations are based on the f_D and the f_M scores in Van Walle *et al.* (2004), in which f_D has the same meaning as SPS and f_M evaluates the percentage of correctly aligned residue pairs in the test alignment. We define the f_D score and the f_M score for each alignment as the average f_D score and the average f_M score respectively over all these pairs. For each test set, we use the Wilcoxon matched-pairs signed-ranks test (Wilcoxon, 1947) over large enough subsets with 0.05 as the p -value cutoff for significance.

Table 3.1 and Table 3.2 shows performance comparisons on the full length sequence set in BAliBASE 3.0. For both reference 1 and the entire set, ISPAAlign improved over MAFFT, ProbCons and SPEM very significantly, with the biggest improvements in the 1V1 subset when identity is very low (improvement in the CS score was over 5%). SPEM improved over MAFFT and ProbCons very significantly for the SPS score. For the CS score, SPEM significantly improved over MAFFT and ProbCons for reference 1, but the overall improvement was not significant for the entire set.

Table 3.1: Average SPS scores (in %) of ISPALign on the full length sequence set in BALiBASE 3.0. Reference 1 is further subdivided into two subsets: 1V1 (< 25% identity), and 1V2 (20–40% identity). The number in braces denotes the number of alignments in each subset. Within each subset, the best accuracy value is in bold. The values in parentheses denote the *p*-values, with — indicating insignificant differences. Since most of the subsets are very small, *p*-values are computed only for reference 1 and the entire set. Twenty-two cases are omitted due to unavailability of results from SPEM.

	<i>SPS</i>			
	<i>MAFFT</i>	<i>ProbCons</i>	<i>SPEM</i>	<i>ISPALign</i>
1V1{38}	64.8	64.5	73.1	76.0
1V2{42}	92.8	93.4	92.1	93.5
1 (V1-V2){80}	79.5	79.7	83.1	85.2
(vs MAFFT)			(4e-5)	(5e-8)
(vs ProbCons)			(7e-4)	(2e-6)
(vs SPEM)				(0.002)
2{37}	91.8	89.7	88.0	91.9
3{29}	81.4	78.8	82.8	83.5
4{36}	89.2	86.8	87.5	90.3
5{14}	88.2	87.5	87.0	90.3
All (1–5){196}	84.5	83.3	85.0	87.5
(vs MAFFT)			(0.005)	(2e-11)
(vs ProbCons)			(5e-4)	(2e-13)
(vs SPEM)				(3e-7)

Table 3.2: Average CS scores (in %) of ISPAAlign on the full length sequence set in BALiBASE 3.0. Twenty-two cases are omitted due to unavailability of results from SPEM.

	CS			
	<i>MAFFT</i>	<i>ProbCons</i>	<i>SPEM</i>	<i>ISPAAlign</i>
1V1{38}	44.6	40.4	51.6	56.9
1V2{42}	83.9	85.6	82.6	85.8
1 (V1-V2){80}	65.2	64.2	67.9	72.1
(vs MAFFT)			(0.01)	(2e-7)
(vs ProbCons)			(0.01)	(2e-5)
(vs SPEM)				(9e-5)
2{37}	46.0	40.8	47.1	53.8
3{29}	56.8	54.3	51.4	59.9
4{36}	67.9	60.9	55.4	63.3
5{14}	57.6	59.4	55.9	63.9
All (1-5){196}	60.3	57.3	58.3	64.6
(vs MAFFT)			(—)	(2e-10)
(vs ProbCons)			(—)	(4e-10)
(vs SPEM)				(5e-11)

Table 3.3: Average SPS and CS scores (in %) of ISPALign on HOMSTRAD. Each subset includes all alignments with average pairwise identity within the specified range, with * indicating worse performance in *p*-value. Since ProbCons consistently performs better than MAFFT, comparisons are made only between ProbCons, SPEM and ISPALign.

	SPS			CS		
	<i>ProbCons</i>	<i>SPEM</i>	<i>ISPALign</i>	<i>ProbCons</i>	<i>SPEM</i>	<i>ISPALign</i>
0-20%{156}	49.7	67.2	68.5	43.1	61.0	62.7
(vs ProbCons)		(2e-23)	(3e-24)		(4e-23)	(5e-24)
(vs SPEM)			(0.0002)			(4e-5)
20-40%{459}	80.5	85.6	86.8	74.7	80.4	81.9
(vs ProbCons)		(5e-33)	(2e-55)		(2e-29)	(2e-53)
(vs SPEM)			(2e-06)			(7e-7)
40-70%{348}	94.8	94.9	95.5	92.2	92.3	93.2
(vs ProbCons)		(0.03)	(1e-09)		(0.03)	(2e-9)
(vs SPEM)			(0.003)			(0.003)
70-100%{69}	99.1	98.5	99.0	99.1	98.4	98.9
(vs ProbCons)		(0.008*)	(—)		(0.007*)	(—)
(vs SPEM)			(—)			(—)
All{1032}	81.9	86.8	87.8	77.4	82.7	84.0
(vs ProbCons)		(4e-51)	(1e-89)		(2e-46)	(8e-87)
(vs SPEM)			(6e-12)			(1e-12)

Table 3.3 shows performance comparisons on HOMSTRAD. Except for 70–100% identity, all the *p*-values of ISPALign over SPEM, ISPALign over ProbCons, and SPEM over ProbCons were highly significant. For 70–100% identity, SPEM performed significantly worse than ProbCons, while the differences between ISPALign and ProbCons or SPEM were not significant. In general, as identity increases, less improvements were observed for both SPEM and ISPALign.

Table 3.4 shows performance comparisons on PREFAB 4.0 using two versions of MAFFT and ProbCons: MAFFT² and ProbCons² use the original input pair, while MAFFT⁵⁰ and ProbCons⁵⁰ use the full sequence set that includes random hits from database search and has at most 50 sequences. For 0–20% identity and 20–40% identity, the improvements of SPEM or ISPALign over MAFFT⁵⁰ were highly significant, while the improvements of ISPALign over SPEM were significant but not as much. For 40–70% identity, SPEM performed significantly worse than MAFFT⁵⁰, while the differences between ISPALign and MAFFT⁵⁰ or SPEM were not significant. For 70–100% identity, ISPALign performed significantly better than SPEM but did not improve over MAFFT⁵⁰, while SPEM performed significantly worse than MAFFT⁵⁰. For the entire set, all the *p*-values of ISPALign over MAFFT⁵⁰, ISPALign over MAFFT⁵⁰ and SPEM over MAFFT⁵⁰ were highly significant.

Table 3.5 shows performance comparisons on the Twilight and Superfamily subsets of SABmark 1.65. While the improvements of SPEM or ISPALign over ProbCons for both

Table 3.4: Average Q scores (in %) of ISPAlign on PREFAB 4.0. Each subset includes all structure pairs with identity within the specified range. Comparisons are made between MAFFT and ProbCons using two sequences (MAFFT², ProbCons²) and using all (at most 50) sequences (MAFFT⁵⁰, ProbCons⁵⁰), SP² (which is a specialized version of SPEM for two sequences), and ISPAlign² (ISPAlign starting from two sequences). Since MAFFT⁵⁰ has the best accuracy among MAFFT and ProbCons, *p*-value comparisons are made only against MAFFT⁵⁰.

	<i>MAFFT</i> ²	<i>ProbCon</i> ²	<i>MAFFT</i> ⁵⁰	<i>ProbCons</i> ⁵⁰	<i>SP</i> ²	<i>ISPAlign</i> ²
0–20%{887}	36.2	38.9	56.7	55.6	64.6	64.8
(vs MAFFT ⁵⁰)					(3e-36)	(5e-46)
(vs SP ²)						(0.03)
20–40%{588}	81.0	82.8	87.1	87.2	89.7	90.1
(vs MAFFT ⁵⁰)					(2e-16)	(6e-28)
(vs SP ²)						(0.01)
40–70%{112}	96.2	96.4	96.0	95.4	95.3	97.6
(vs MAFFT ⁵⁰)					(0.02*)	(—)
(vs SP ²)						(—)
70–100%{95}	97.9	97.8	98.0	97.3	97.2	98.0
(vs MAFFT ⁵⁰)					(6e-4*)	(—)
(vs SP ²)						(0.005)
All{1682}	59.4	61.4	72.3	71.7	77.3	77.7
(vs MAFFT ⁵⁰)					(1e-46)	(7e-69)
(vs SP ²)						(2e-4)

Table 3.5: Average f_D and f_M scores (in %) of ISPAlign on the Twilight and Superfamily subsets of SABmark 1.65. Four cases are omitted in the Twilight subset and three cases are omitted in the Superfamily subset since no reference alignments of sufficiently good quality are available. None of these subsets include false positive sequences. Since ProbCons consistently performs better than MAFFT, comparisons are made only between ProbCons, SPEM and ISPAlign.

	f_D			f_M		
	<i>ProbCons</i>	<i>SPEM</i>	<i>ISPAlign</i>	<i>ProbCons</i>	<i>SPEM</i>	<i>ISPAlign</i>
Twilight{205}	29.3	44.2	46.1	21.0	30.8	32.0
(vs ProbCons)		(2e-26)	(6e-29)		(1e-27)	(3e-29)
(vs SPEM)			(0.01)			(0.005)
Superfamily{422}	57.1	68.3	69.0	43.6	50.9	51.6
(vs ProbCons)		(4e-49)	(1e-51)		(1e-48)	(1e-51)
(vs SPEM)			(0.02)			(7e-4)

subsets were highly significant, the improvements of ISPAlign over SPEM were significant but not as much.

In all the subsets that we have assessed, ISPAlign always performs at least as well as ProbCons and SPEM and is much better in many cases, especially when the input sequences are divergent in which the improvements are always significant and in many

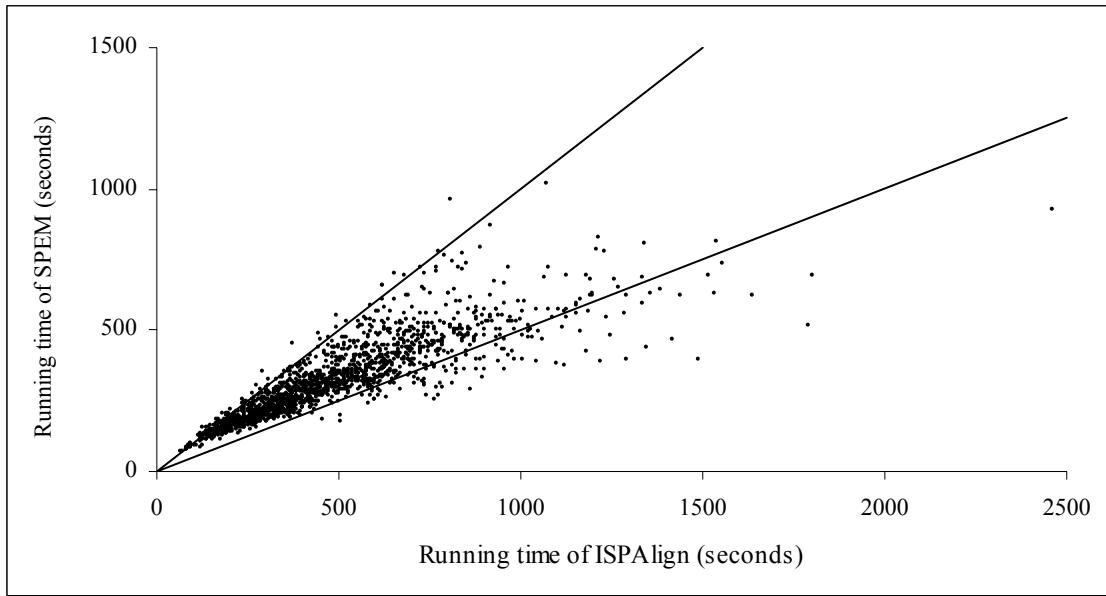


Figure 3.3: Running time of SPEM and ISPAlign on PREFAB 4.0. The upper line denotes the region in which SPEM and ISPAlign have the same speed. The lower line denotes the region in which ISPAlign is two times slower than SPEM.

cases highly significant. Also, the improvements in the CS scores are sometimes more significant than the improvements in the SPS scores. In general, the contribution from utilizing additional sequences from database search decreases as the input sequences become more closely related. When the input sequences become very similar, while SPEM has significant accuracy decreases in many cases, ISPAlign still always performs at least as well. Since not many intermediate sequences are added to the input sequences before performing the profile alignment step, ISPAlign is efficient enough to perform an

Table 3.6: Average CS scores (in %) of ISPAAlign on HOMSTRAD using a few methods that are of increasing levels of complexity. Method 1 constructs a profile from the hits of each input sequence and performs profile alignment using the modified HMM model that incorporates profiles but not secondary structure predictions. Method 2 removes the hits that are not intermediate sequences before performing profile alignment. Method 3 adds intermediate sequences to the input sequences, constructs profiles based on the intermediate sequences and performs profile alignment on the combined sequence set. Method 4 is the full ISPAAlign algorithm that also utilizes secondary structure predictions. The *p*-value comparisons are made against the previous method to the left, with * indicating worse performance.

	<i>HOMSTRAD CS</i>				
	<i>ProbCons</i>	<i>Method1</i>	<i>Method2</i>	<i>Method3</i>	<i>Method4</i>
0-20%	43.1	59.1 (3e-22)	59.2 (—)	59.4 (0.04)	62.7 (6e-8)
(vs previous)					
20-40%	74.7	79.1 (2e-24)	79.6 (0.003)	81.4 (7e-14)	81.9 (0.005)
(vs previous)					
40-70%	92.2	92.1 (—)	92.5 (8e-4)	93.1 (0.001)	93.2 (—)
(vs previous)					
70-100%	99.1	98.2 (6e-4*)	99.1 (1e-4)	99.2 (—)	98.9 (0.003*)
(vs previous)					
All	77.4	81.7 (5e-38)	82.2 (1e-6)	83.2 (1e-14)	84.0 (1e-6)
(vs previous)					

Table 3.7: Average Q scores (in %) of ISPAAlign on PREFAB 4.0 using a few methods that are of increasing levels of complexity (described in Table 3.6). For PREFAB, ProbCons uses the original input pair while all the methods start from this input pair. The *p*-value comparisons are made against the previous method to the left, with * indicating worse performance.

	<i>PREFAB Q</i>				
	<i>ProbCons</i>	<i>Method1</i>	<i>Method2</i>	<i>Method3</i>	<i>Method4</i>
0-20%	38.9	58.2	58.6	61.3	64.8
(vs previous)		(2e-103)	(—)	(6e-12)	(7e-29)
20-40%	82.8	88.7	89.0	89.7	90.1
(vs previous)		(9e-45)	(—)	(2e-4)	(0.004)
40-70%	96.4	94.4	96.6	97.8	97.6
(vs previous)		(—)	(0.002)	(—)	(0.008*)
70-100%	97.8	97.0	96.9	98.1	98.0
(vs previous)		(0.04*)	(0.02)	(—)	(—)
All	61.4	73.5	73.9	75.7	77.7
(vs previous)		(7e-146)	(—)	(2e-15)	(4e-28)

individual multiple alignment of moderate size in a reasonable time. Figure 3.3 shows that ISPAAlign was at most two times slower than SPEM in most cases.

Table 3.8: Average CS scores (in %) of ISPAlign on HOMSTRAD by varying the parameter k that specifies the maximum number of intermediate sequences that are added to the input sequences. (α, β) is fixed to $(0.65, 0.75)$. The p -value comparisons are made against the previous entry to the left, with * indicating worse performance.

	k				
	5	10	15	20	25
0-20%	62.7	63.5	63.2	63.4	62.9
(vs previous)		(0.02)	(—)	(—)	(0.02*)
20-40%	81.9	81.9	81.9	81.9	81.9
(vs previous)		(—)	(—)	(—)	(—)
40-70%	93.2	93.0	92.9	93.0	93.1
(vs previous)		(—)	(—)	(—)	(—)
70-100%	98.9	98.9	98.9	98.9	98.9
(vs previous)		(—)	(—)	(—)	(—)
All	84.0	84.0	83.9	84.0	84.0
(vs previous)		(—)	(—)	(—)	(—)

To evaluate the contributions from various components of the algorithm to the alignment accuracy under different identity levels, we compare the performance of a few methods that are of increasing levels of complexity on HOMSTRAD and PREFAB 4.0 (Table 3.6 and Table 3.7). When the identity is low, the biggest improvements were from the use of

Table 3.9: Average CS scores (in %) of ISPAlign on HOMSTRAD by varying the parameters α and β that modify the pair-HMM probabilities. k is fixed to 5 and a few (α, β) pairs that give the best performance are shown.

	(α, β)					
	(0.55,0.55)	(0.65,0.55)	(0.65,0.65)	(0.55,0.65)	(0.65,0.75)	(0.55,0.75)
0-20%	61.3	62.7	63.3	62.8	62.7	62.8
(vs previous)		(0.004)	(0.002)	(—)	(—)	(—)
20-40%	81.4	81.5	81.8	81.8	81.9	82.0
(vs previous)		(—)	(2e-4)	(—)	(—)	(—)
40-70%	93.0	93.0	93.1	93.1	93.2	93.2
(vs previous)		(—)	(0.04)	(0.03)	(—)	(—)
70-100%	99.0	98.9	98.9	99.0	98.9	99.1
(vs previous)		(—)	(—)	(—)	(—)	(—)
All	83.4	83.7	84.0	83.9	84.0	84.0
(vs previous)		(—)	(5e-7)	(—)	(—)	(0.02)

profiles, while significant improvements were obtained from the addition of intermediate sequences to the input sequences and from the use of secondary structure predictions. When the identity is high, improvements were mainly from the removal of hits that are not intermediate sequences.

To evaluate the effect of different parameter settings on the algorithm, we vary the parameter k that specifies the maximum number of intermediate sequences that are added to the input sequences and the parameters α and β that modify the pair-HMM probabilities. While Table 3.8 and Table 3.9 shows that the parameters $k = 5$ and $(\alpha, \beta) = (0.65, 0.75)$ that were determined from testing on PREFAB gave very good performance on HOMSTRAD, they did not always produce the best possible performance on HOMSTRAD. For 0–20% identity, it was better to use larger values of k such as $k = 10$, but the performance started to decrease when $k = 25$ and the overall performance across all identity levels was not significantly different for these values of k . The algorithm had the best performance when α is 0.55–0.65 and when β is 0.55–0.75, showing that α and β should be close in magnitude. For 0–20% identity, the best performance was obtained when $(\alpha, \beta) = (0.65, 0.65)$, although it was not significantly different from the performance obtained from a few other settings. Overall, the best performance on HOMSTRAD was obtained when $(\alpha, \beta) = (0.55, 0.75)$, with $(\alpha, \beta) = (0.65, 0.75)$ performing slightly worse.

H. Discussion

While we have described a procedure for ISPAAlign that gives very good performance, there are still many opportunities to further improve its accuracy. Instead of adding a fixed number of intermediate sequences to the input sequences, it may be better to add

more sequences as the number of input sequences increases. Alternatively, intermediate sequences can be added until all the minimum distances between each of the remaining intermediate sequences and the current set of sequences fall below a threshold. Also, instead of modifying the parameters used by ProbCons by applying the factors α and β , it may be better to re-train the pair-HMM using a set of confirmed secondary structures. This can be done in a framework suggested by Do *et al.* (2006). It is also possible to use other multiple alignment algorithms to perform the profile alignment step as long as profiles and secondary structure predictions can be incorporated, which can lead to further improvements as better multiple alignment algorithms become available. It may also be beneficial to utilize three-dimensional structures when they are available.

CHAPTER IV

NRAAlign: MULTIPLE SEQUENCE ALIGNMENT BASED ON ALIGNMENT OF NEIGHBORING RESIDUES

A. Introduction

The construction of multiple sequence alignments is among the most important techniques to perform biological sequence analysis, with important applications to many areas of computational biology. The most popular strategy to construct multiple sequence alignments is by employing a progressive alignment algorithm, in which each sequence is treated initially as an alignment and the next two most similar alignments are repeatedly combined until a single multiple alignment is obtained (Thompson *et al.*, 1994; Notredame *et al.*, 2000; Edgar, 2004; Do *et al.*, 2005; Pei and Grishin, 2006; Roshan and Livesay, 2006). This is often followed by iterative refinements that improve the accuracy of the final alignment (Gotoh, 1996; Edgar, 2004; Do *et al.*, 2005; Roshan and Livesay, 2006).

There are many recent efforts that lead to significant improvement of alignment accuracy, including the incorporation of consistency-based pairwise alignments that improve the quality of the initial pairwise alignments by aligning through other sequences to increase their agreement with the final multiple alignment (Notredame *et*

al., 2000; Do *et al.*, 2005; Pei and Grishin, 2006; Roshan and Livesay, 2006), the use of maximal expected accuracy alignment instead of the less accurate Viterbi alignment (Do *et al.*, 2005; Pei and Grishin, 2006; Roshan and Livesay, 2006), the incorporation of secondary structure predictions (Zhou and Zhou, 2005; Pei and Grishin, 2007), the use of local structural information (O’Sullivan *et al.*, 2004; Van Walle *et al.*, 2004; Pei *et al.*, 2008), and the incorporation of additional sequences from database search (Martí-Renom *et al.*, 2004; Simossis *et al.*, 2005; Zhou and Zhou, 2005; Pei and Grishin, 2007).

While most of these algorithms are able to significantly improve alignment accuracy by making better use of vertical information, either by incorporating consistency-based pairwise alignments or by using profiles in which each column of an alignment is modeled independently, we observe that most of these algorithms do not make use of horizontal information when constructing alignments, and it may be useful to take into account alignment of neighboring residues when aligning two residues.

There are a few previous approaches that use neighboring information to obtain significant performance improvements in other applications. Spang *et al.* (2002) obtained a jumping alignment that is suitable for remote homology detection between a given sequence and a multiple alignment by aligning each position in the given sequence to one of the sequences in the multiple alignment while penalizing each vertical jump between horizontal moves. Panchenko *et al.* (2004) used average conservation scores across spatial neighboring sites in the local structural environment to improve functional

site prediction, while Capra and Singh (2007) used conservation scores from neighboring residues to improve the prediction of functionally important residues in aligned sequences.

To incorporate horizontal information in alignments, we develop a window-based method that adjusts the pairwise score of a residue pair between two sequences (or a column pair between two sub-alignments) by incorporating the scores of neighboring residue pairs (or column pairs). This method can be applied to any multiple alignment algorithm that uses pairwise scores during the construction of a multiple alignment. We test this strategy by modifying existing multiple alignment algorithms to make use of horizontal information and show that consistent improvements can be obtained for these algorithms on all sets of benchmark alignments that we have tested. By using a statistical test that pairs the alignments before and after algorithm modification, we show that highly statistically significant improvements are obtained not just in relative performance but also in paired performance.

B. Methods

Incorporating Horizontal Information into Pairwise Scores

Given a residue (or column) at position x in the first sequence (or sub-alignment) $s = s_1 \dots s_m$, a residue (or column) at position y in the second sequence (or sub-alignment) s'

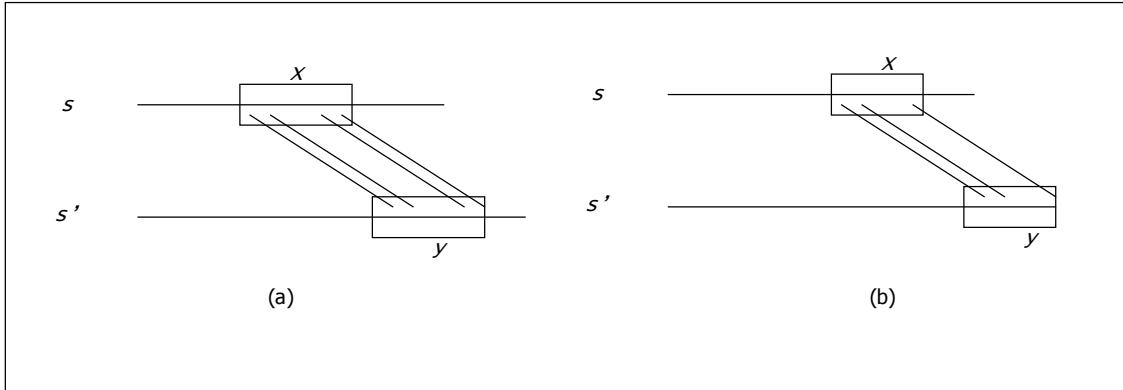


Figure 4.1: Illustration of the window on two sequences s and s' with $\omega = 2$. (a) The offsets in $N_\omega(x, y) = \{-2, -1, 1, 2\}$ are included. (b) Since $y + 1$ is the last position in s' , only one position is used to the right of (x, y) and the offsets in $N_\omega(x, y) = \{-2, -1, 1\}$ are included.

$= s'_1 \dots s'_n$, and a parameter ω , define the window that includes at most ω positions to the left and to the right of (x, y) by the following set of valid offsets in the neighborhood of (x, y) (see Figure 4.1):

$$N_\omega(x, y) = \{i \mid 0 < |i| \leq \omega, 1 \leq x + i \leq m, 1 \leq y + i \leq n\}.$$

We use the following equation to incorporate the scores of the neighboring pairs at position $(x + i, y + i)$ over all offsets i in $N_\omega(x, y)$ into the score of the given pair (x, y) :

$$S_{new}(x, y) = \frac{S_{old}(x, y) + \beta \sum_{i \in N_\omega(x, y)} S_{old}(x + i, y + i)}{1 + \beta |N_\omega(x, y)|}, \quad (4.1)$$

where S_{old} is the original score, S_{new} is the adjusted score, and β is a parameter that specifies the weight of the neighboring scores during the adjustment.

Table 4.1: Parameter settings for the modified version of each algorithm that uses horizontal information.

	TCoffee	MUSCLE	ProbCons	MUMMALS
ω	3	2	5	1
β	0.7	1.0	1.0	0.8

We apply this strategy to TCoffee 5.31 (Notredame *et al.*, 2000) without using structural information, which is among the first multiple alignment algorithms that utilize consistency-based pairwise alignments, to MUSCLE 3.6 (Edgar 2004), which is among the most efficient multiple alignment algorithms that also have high accuracy, to ProbCons 1.10 (Do *et al.*, 2005), which is among the first multiple alignment algorithms that utilize the maximal expected accuracy alignment based on a pair-HMM model, and to MUMMALS 1.01 (Pei and Grishin, 2006), which uses secondary structure information during pair-HMM training to further improve alignment accuracy. In each case, we evaluate the performance of each of the modified algorithms (called NRAAlign) against each of the original algorithms while using the same parameter setting across different benchmark alignments for each modified algorithm (Table 4.1). Horizontal information is incorporated into each of the algorithms either during the computation of consistency-based pairwise alignments or during the progressive alignment step.

Modification of TCoffee

The TCoffee algorithm consists of the following steps: construct a library of pairwise alignments from the input sequences by using global alignments from ClustalW (Thompson *et al.*, 1994) and local alignments from Lalign (Huang and Miller, 1991), assign a weight to each pair of aligned residues in the library according to sequence identity, apply library extension to all the weights in the library to obtain an extended library that utilizes consistency-based information by using a triplet approach, and perform progressive alignment according to a guide tree by aligning two groups of pre-aligned sequences using the average scores between column pairs in the extended library. In NRAAlign, we apply equation (4.1) to adjust the average extended library scores between column pairs before each progressive alignment step.

Modification of MUSCLE

The MUSCLE algorithm consists of the following steps: compute the k -mer distance for each pair of input sequences to produce an initial tree and perform progressive alignment according to the tree by utilizing log-expectation scores between two aligned columns to obtain an initial multiple alignment, re-estimate the tree using Kimura distances (Kimura, 1983) computed from the multiple alignment and perform progressive alignment according to the new tree, then perform iterative refinements to obtain the final alignment. In NRAAlign, we apply equation (4.1) to adjust the log-expectation scores before each progressive alignment step.

Modification of ProbCons

The ProbCons algorithm consists of the following steps: compute the posterior probability matrix for each pair of input sequences according to the pair-HMM model, compute maximal expected accuracy alignment for each sequence pair by dynamic programming, re-estimate the match quality score matrix for each sequence pair by performing probabilistic consistency transformation, construct a guide tree according to the maximal expected accuracy alignments, perform progressive alignment according to the guide tree by using the transformed scores, and perform iterative refinements to obtain the final alignment. In NRAlign, we apply equation (4.1) to adjust the match quality scores for each sequence pair before consistency transformation is performed.

Modification of MUMMALS

The MUMMALS algorithm consists of the following steps: compute the k -mer distance for each pair of input sequences to produce an initial tree and perform progressive alignment according to the tree to obtain an initial multiple alignment, re-estimate the tree using sequence identities computed from the multiple alignment, perform a two-stage progressive alignment in which highly similar sequences are first aligned by using weighted sum-of-pairs BLOSUM62 scores (Henikoff and Henikoff 1992), and a representative is chosen from each pre-aligned group to perform progressive consistency-based multiple alignment based on transformed pairwise maximal expected accuracy alignments that are obtained from a pair-HMM model that also includes secondary structure states, then merge the pre-aligned groups according to the alignment

of the representatives to obtain the final alignment. In NRAAlign, we apply equation (4.1) to adjust the scores between column pairs before each progressive alignment step.

C. Benchmark Alignments

We evaluate the performance of NRAAlign on benchmark multiple alignments from BALiBASE 3.0 (Thompson *et al.*, 2005), which contains manually refined structural alignments that are subdivided into five categories, from HOMSTRAD (Mizuguchi *et al.*, 1998), which contains a collection of manually edited structure-based alignments, from PREFAB 4.0 (Edgar 2004), which contains structural alignments of two sequences and automatically generated alignments that are obtained from adding high scoring hits of the two sequences from database search, and from SABmark 1.65 (Van Walle *et al.*, 2004), which contains alignments that are derived from the SCOP classification (Murzin *et al.*, 1995).

Two reference-dependent scores are used to evaluate the accuracy of each algorithm, including the sum-of-pairs score (SPS), which measures the percentage of residue pairs that are aligned correctly in the reference alignment, and the column score (CS), which measures the percentage of entire columns that are aligned correctly (Thompson *et al.*, 1999). For BALiBASE and PREFAB, evaluations are made only on the core regions that are specified in the reference alignments. For PREFAB and SABmark, the reference

alignments are based on sequence pairs and the CS score is not used. For PREFAB, the Q score in Edgar (2004) is computed on the original input sequence pair, which has the same meaning as the SPS score. For SABmark, reference alignments are specified for each sequence pair, and the f_D score, which is a sensitivity score that has the same meaning as the SPS score, and an additional f_M score, which is a specificity score that measures the percentage of residue pairs that are aligned correctly in the test alignment, are computed by averaging the scores over all sequence pairs for each multiple alignment (Van Walle *et al.*, 2004).

In addition to reference-dependent scores, four reference-independent scores are used in the presence of known three-dimensional structures to evaluate the structural agreement between aligned sequence pairs, including the Dali Z-score (Holm and Sander, 1998), which computes a structural similarity score as a weighted sum of similarities of intra-molecular distances between residues in aligned columns normalized according to alignments of random structure pairs (see equations (2) to (4) in Holm and Sander (1998)), the GDT_TS score (Zemla *et al.*, 1999; Venclovas *et al.*, 2003), which computes the average of the maximum number of aligned residue pairs that can be superimposed within four different distance thresholds of 1, 2, 4 and 8 Å (see the equation in Venclovas *et al.* (2003)), and two LiveBench contact scores ContactA and ContactB (Rychlewski *et al.*, 2003; Wallner and Elofsson, 2003), which compute an overlap score that is the lower of two contact scores, one for each structure, computed based on intra-molecular distances between residues in aligned columns that are

separated by at least five residues (see equation (2) in Wallner and Elofsson (2003)), with ContactA normalized for each residue and ContactB normalized over the entire contact map.

To compute the GDT_TS score, multiple superpositions of aligned residue pairs are needed that optimize the individual score components, and the software from Zhang and Skolnick (2004) is used with the set of aligned residue pairs as input while omitting the final normalization step. Following the procedure in Pei and Grishin (2006), each score is further weighted and normalized against the reverse alignment that represents a random model. For SABmark, three-dimensional coordinates are extracted from the given PDB files (Berman *et al.*, 2000), and the scores for each multiple alignment are computed by averaging the scores over all sequence pairs.

To evaluate whether the use of NRAAlign leads to significant improvements, we use the Wilcoxon matched-pairs signed-ranks test (Wilcoxon 1947) over subsets that are large enough with $p = 0.05$ as significance cutoff, in which the alignments before and after algorithm modification are paired to evaluate whether the improvements are consistent not just in relative performance but also in paired performance.

D. Performance

Table 4.2 and Table 4.3 show performance comparisons on full length sequences in BALiBASE 3.0. Among all the subsets that are large enough, NRAlign always performed as least as well as the original algorithm. Except for MUSCLE, the improvements for NRAlign were more significant in the CS score when compared to the SPS score, and this is especially evident on TCoffee. The improvements in the CS score were more than 2% in references 1V2, 2, 3 and 4 over MUSCLE and in reference 1V2 over MUMMALS, more than 4% in reference 5 over MUMMALS, and more than 1% in the entire set over MUSCLE and MUMMALS.

Table 4.4 and Table 4.5 show performance comparisons on HOMSTRAD. Except for 70 to 100% identity when the improvements for NRAlign were significant only over MUMMALS, all improvements at other identity levels were significant (except for 0 to 20% over MUMMALS). The improvements were especially significant when the identity is moderately low (20 to 40%), while the overall improvements were highly significant over all algorithms.

Table 4.6 and Table 4.7 show performance comparisons on PREFAB 4.0. When only the original input sequence pair are aligned, the performance improvement characteristics of NRAlign were similar to those of HOMSTRAD, except that the improvements for NRAlign were significant also over ProbCons for 70 to 100% identity. The

Table 4.2: Average SPS scores (in %) of NRAlign on full length sequences in BALiBASE 3.0. Reference 1 contains alignments of sequences that are subdivided into two subsets 1V1 (< 20% identity) and 1V2 (20–40% identity). Reference 2 contains alignments that include orphan sequences. Reference 3 contains alignments of clusters of sequences from different families. Reference 4 contains alignments of sequences with large terminal extensions, while reference 5 contains alignments of sequences with internal insertions. The number in braces denotes the number of alignments in each subset. For each algorithm, the first number shows the performance of the original algorithm that does not use horizontal information. The second number shows the performance of the modified algorithm NRAlign that makes use of horizontal information, with the higher accuracy value in bold. The third number shows the *p*-value, with — indicating insignificant differences. Since many of the subsets are small, *p*-values are computed only for reference 1 and for the entire set.

<i>SPS</i>	<i>TCoffee</i>		<i>SPS</i>	<i>MUSCLE</i>	
1V1{38}	53.81	54.21	1V1{38}	56.21	56.98
1V2{44}	91.55	91.98	1V2{44}	90.62	91.50
1{82}	74.06	74.48	1{82}	74.67	75.50 0.003
2{41}	89.04	88.82	2{41}	88.08	88.24
3{30}	71.09	71.19	3{30}	75.01	76.27
4{49}	82.21	82.37	4{49}	84.83	85.64
5{16}	81.94	80.98	5{16}	82.69	82.83
All{218}	78.88	78.97	0.04	All{218}	80.11 80.82 0.006

Table 4.2: Continued.

<i>SPS</i>	<i>ProbCons</i>		<i>SPS</i>	<i>MUMMALS</i>	
1V1{38}	64.46	64.48	1V1{38}	64.41	64.23
1V2{44}	93.50	93.65	1V2{44}	93.53	94.00
1{82}	80.05	80.13	—	1{82}	80.03
2{41}	89.93	89.94	2{41}	89.18	89.39
3{30}	78.62	78.30	3{30}	80.76	80.79
4{49}	87.43	87.25	4{49}	83.69	83.97
5{16}	87.69	87.87	5{16}	86.33	87.40
All{218}	83.93	83.89	—	All{218}	83.14
					83.39
					—

improvements were more significant in this case than in the case when the full set of at most 50 sequences are aligned, although using the full set of sequences gives better performance for each of the original and modified algorithms on divergent sequences (0 to 40% identity).

Table 4.8 and Table 4.9 show performance comparisons on the Twilight and Superfamily subsets of SABmark 1.65. Unlike previous algorithms that have improvements mostly on divergent sequences, the improvements for NRAAlign were more significant on the Superfamily subset than on the more divergent Twilight subset. Similar to the results in Pei and Grishin (2006), there are strong correlations between the

Table 4.3: Average CS (in %) scores of NRAlign on full length sequences in BALiBASE 3.0.

<i>CS</i>	<i>TCoffee</i>			<i>CS</i>	<i>MUSCLE</i>		
1V1{38}	31.34	32.21		1V1{38}	35.63	33.95	
1V2{44}	81.64	82.68		1V2{44}	80.75	82.93	
1{82}	58.33	59.29	1e-4	1{82}	59.84	60.23	0.01
2{41}	37.85	38.88		2{41}	35.27	37.61	
3{30}	36.00	36.83		3{30}	40.57	42.73	
4{49}	48.20	48.78		4{49}	47.37	49.67	
5{16}	50.63	49.31		5{16}	47.94	44.94	
All{218}	48.56	49.27	7e-9	All{218}	48.89	50.07	0.002
<i>CS</i>	<i>ProbCons</i>			<i>CS</i>	<i>MUMMALS</i>		
1V1{38}	40.45	41.00		1V1{38}	41.61	41.39	
1V2{44}	85.52	85.77		1V2{44}	83.98	86.41	
1{82}	64.63	65.02	0.02	1{82}	64.34	65.55	—
2{41}	40.63	40.49		2{41}	42.83	43.46	
3{30}	54.37	54.80		3{30}	49.40	49.57	
4{49}	53.67	53.14		4{49}	48.55	49.76	
5{16}	57.38	57.31		5{16}	52.88	57.00	
All{218}	55.71	55.77	0.04	All{218}	53.85	55.02	0.001

Table 4.4: Average SPS scores (in %) of NRAlign on HOMSTRAD. Each subset includes all alignments with average pairwise identity within the specified range.

<i>SPS</i>	<i>TCoffee</i>			<i>SPS</i>	<i>MUSCLE</i>		
0–20%{156}	46.68	47.21	0.005	0–20%{156}	48.08	50.18	4e–4
2–40%{459}	79.19	79.71	4e–13	20–40%{459}	78.86	80.11	1e–10
40–70%{348}	94.48	94.80	2e–11	40–70%{348}	94.45	94.77	1e–4
70–100%{69}	99.10	99.16	—	70–100%{69}	99.02	99.07	—
All{1032}	80.76	81.19	2e–22	All{1032}	80.82	81.80	6e–16
<i>SPS</i>	<i>ProbCons</i>			<i>SPS</i>	<i>MUMMALS</i>		
0–20%{156}	49.67	50.67	6e–8	0–20%{156}	54.39	54.44	—
2–40%{459}	80.55	81.44	3e–22	2–40%{459}	82.67	82.71	4e–4
40–70%{348}	94.75	95.19	7e–12	40–70%{348}	95.04	95.14	9e–4
70–100%{69}	99.10	99.08	—	70–100%{69}	98.94	99.14	0.005
All{1032}	81.91	82.60	6e–38	All{1032}	83.65	83.72	5e–8

reference-dependent and reference-independent results, which indicate that the improvements are not only at the sequence level but also at the structural level.

When comparisons were made on the improvements among the different algorithms, we found that MUMMALS was the hardest to improve on HOMSTRAD and on PREFAB when using the original input sequence pair for moderate to low identity. ProbCons was

Table 4.5: Average CS scores (in %) of NRAlign on HOMSTRAD.

<i>CS</i>	<i>TCoffee</i>			<i>CS</i>	<i>MUSCLE</i>		
0–20%{156}	39.97	40.64	2e–4	0–20%{156}	41.77	43.70	0.003
2–40%{459}	72.97	73.76	9e–17	20–40%{459}	73.01	74.61	2e–11
40–70%{348}	91.79	92.33	2e–13	40–70%{348}	91.90	92.28	8e–5
70–100%{69}	99.03	99.10	—	70–100%{69}	98.98	99.03	—
All{1032}	76.07	76.71	1e–30	All{1032}	76.39	77.53	8e–16
<i>CS</i>	<i>ProbCons</i>			<i>CS</i>	<i>MUMMALS</i>		
0–20%{156}	43.12	44.15	3e–7	0–20%{156}	47.94	48.00	—
2–40%{459}	74.67	75.80	5e–24	2–40%{459}	77.31	77.43	0.001
40–70%{348}	92.20	92.84	6e–13	40–70%{348}	92.61	92.77	3e–5
70–100%{69}	99.06	99.02	—	70–100%{69}	98.87	99.08	0.007
All{1032}	77.44	78.32	6e–40	All{1032}	79.47	79.60	4e–8

the hardest to improve on BAliBASE, the easiest to improve on HOMSTRAD except for 70 to 100% identity and on PREFAB when using the original input sequence pair, while the improvements on TCoffee and MUSCLE varied across different benchmarks. This is in contrast with the better performance of ProbCons and MUMMALS over TCoffee and MUSCLE for moderate to low identity. The improvement characteristics were especially

Table 4.6: Average Q(2) scores (in %) of NRAlign on PREFAB 4.0. Each subset includes all structure pairs with identity within the specified range, with * indicating worse performance in p -value. The Q(2) scores are obtained from aligning only the original input sequence pair.

$Q(2)$	<i>TCoffee</i>			$Q(2)$	<i>MUSCLE</i>		
0–20%{887}	37.92	38.27	1e–5	0–20%{887}	38.22	39.69	8e–8
2–40%{588}	82.60	82.92	4e–8	2–40%{588}	81.75	83.87	1e–29
40–70%{112}	96.37	96.51	0.005	40–70%{112}	96.24	96.58	0.01
70–100%{95}	97.94	98.04	—	70–100%{95}	97.97	97.91	—
All{1682}	60.82	61.13	1e–12	All{1682}	60.68	62.21	7e–29
$Q(2)$	<i>ProbCons</i>			$Q(2)$	<i>MUMMALS</i>		
0–20%{887}	38.95	40.17	7e–31	0–20%{887}	43.59	43.62	0.005
2–40%{588}	82.84	84.30	4e–39	2–40%{588}	85.39	85.45	2e–4
40–70%{112}	96.41	96.83	5e–6	40–70%{112}	96.59	96.75	5e–4
70–100%{95}	97.76	98.05	3e–4	70–100%{95}	97.75	97.93	0.03
All{1682}	61.44	62.64	3e–71	All{1682}	64.79	64.85	5e–8

different on PREFAB depending on whether the original input sequence pair or the full set of sequences are aligned, when it was easier to improve on MUMMALS than on MUSCLE in the latter case.

Table 4.7: Average Q(50) scores (in %) of NRAlign on PREFAB 4.0. Each subset includes all structure pairs with identity within the specified range, with * indicating worse performance in p -value. The Q(50) scores are obtained from aligning the full set of sequences (at most 50) that also includes random hits from database search and evaluations are made on the original input sequence pair.

$Q(50)$	<i>TCoffee</i>			$Q(50)$	<i>MUSCLE</i>		
0–20%{887}	49.67	50.00	6e–6	0–20%{887}	50.71	50.95	—
2–40%{588}	83.94	84.20	8e–7	2–40%{588}	85.09	85.13	—
40–70%{112}	95.99	95.55	0.02*	40–70%{112}	94.72	96.46	—
70–100%{95}	97.97	98.04	—	70–100%{95}	97.50	97.69	—
All{1682}	67.46	67.70	2e–9	All{1682}	68.30	68.57	—
$Q(50)$	<i>ProbCons</i>			$Q(50)$	<i>MUMMALS</i>		
0–20%{887}	55.63	55.72	0.02	0–20%{887}	57.68	57.91	0.02
2–40%{588}	87.24	87.38	3e–7	2–40%{588}	87.24	87.30	0.02
40–70%{112}	95.39	95.48	0.004	40–70%{112}	95.34	95.41	—
70–100%{95}	97.26	97.40	0.001	70–100%{95}	96.68	97.04	0.005
All{1682}	71.68	71.79	1e–7	All{1682}	72.73	72.89	5e–4

In all the subsets that we have assessed, NRAlign always performed at least as well as the original algorithm (except for one case). The overall improvements were highly significant in most cases, even when the improvements in accuracy can sometimes be

Table 4.8: Average f_D and f_M scores and average normalized Dali Z-score, GDT_TS score, and ContactA and ContactB scores (in %) of NRAlign on the Twilight subset of SABmark 1.65. The Twilight subset contains alignments that represent a SCOP fold ($\leq 25\%$ identity). Four cases are omitted in the Twilight subset since no high quality reference alignments are available.

	<i>Twilight{205}</i>			<i>TCoffee</i>			<i>Twilight{205}</i>			<i>MUSCLE</i>		
f_D	24.07	23.99	—	f_D	24.07	25.29	0.008					
f_M	18.08	18.08	—	f_M	16.47	16.84	—					
Dali Z-score	11.10	11.19	0.02	Dali Z-score	13.14	13.68	0.02					
GDT_TS	10.67	10.78	0.007	GDT_TS	12.45	12.91	0.03					
ContactA	6.72	6.76	—	ContactA	7.62	7.95	0.03					
ContactB	8.98	9.03	—	ContactB	10.06	10.47	—					
	<i>Twilight{205}</i>			<i>ProbCons</i>			<i>Twilight{205}</i>			<i>MUMMALS</i>		
f_D	29.26	29.72	0.01	f_D	31.57	31.63	0.04					
f_M	21.00	21.02	—	f_M	22.87	22.97	0.009					
Dali Z-score	13.88	14.32	3e-5	Dali Z-score	15.32	15.38	0.03					
GDT_TS	13.38	13.68	5e-4	GDT_TS	14.52	14.54	—					
ContactA	8.67	8.87	0.01	ContactA	9.41	9.45	—					
ContactB	12.10	12.37	0.01	ContactB	12.59	12.61	—					

Table 4.9: Average f_D and f_M scores and average normalized Dali Z-score, GDT_TS score, and ContactA and ContactB scores (in %) of NRAAlign on the Superfamily subset of SABmark 1.65. The Superfamily subset contains alignments that represent a SCOP superfamily ($\leq 50\%$ identity). Three cases are omitted in the Superfamily subset since no high quality reference alignments are available.

<i>Superfamily{422}</i>	<i>TCoffee</i>			<i>Superfamily{422}</i>	<i>MUSCLE</i>		
f_D	52.91	53.30	2e-5	f_D	53.12	53.91	0.008
f_M	41.30	41.52	5e-4	f_M	39.87	40.26	0.04
Dali Z-score	33.09	33.25	0.04	Dali Z-score	35.34	35.85	0.002
GDT_TS	31.07	31.23	0.01	GDT_TS	32.98	33.47	5e-4
ContactA	23.07	23.14	—	ContactA	24.23	24.54	0.001
ContactB	28.91	28.94	—	ContactB	30.30	30.59	0.007
<i>Superfamily{422}</i>	<i>ProbCons</i>			<i>Superfamily{422}</i>	<i>MUMMALS</i>		
f_D	57.06	57.30	8e-8	f_D	59.50	59.65	0.004
f_M	43.57	43.61	0.03	f_M	45.15	45.25	0.01
Dali Z-score	35.84	36.26	9e-21	Dali Z-score	37.79	37.87	0.001
GDT_TS	33.67	33.92	1e-17	GDT_TS	35.05	35.11	0.01
ContactA	25.29	25.45	2e-9	ContactA	26.41	26.45	—
ContactB	32.10	32.22	1e-4	ContactB	33.11	33.17	0.04

small. Unlike previous algorithms that have improvements mostly on divergent sequences, consistent improvements can be obtained across all identity levels, and it is not always the case that the most improvements were obtained on highly divergent sequences.

E. Discussion

Pairwise Alignment versus Multiple Alignment

The above results on PREFAB show that the improvements for NRAlign were more significant on pairwise alignments. Since the reference alignments for SABmark are based on sequence pairs, we investigate this further by performing pairwise alignments over all sequence pairs instead of obtaining a single multiple alignment, and computing the scores for each multiple alignment by averaging the scores over all sequence pairs. When compared to Table 4.8 and Table 4.9, Table 4.10 and Table 4.11 shows that the improvements in SABmark were more significant when pairwise alignments are performed, and this is especially evident on the Superfamily subset, although obtaining a single multiple alignment gives better performance on both the Twilight and Superfamily subsets for each of the original and modified algorithms of ProbCons and MUMMALS, and on the Superfamily subset for each of the original and modified algorithms of MUSCLE.

Table 4.10: Average f_D and f_M scores and average normalized Dali Z-score, GDT_TS score, and ContactA and ContactB scores (in %) of NRAAlign on the Twilight subset of SABmark 1.65 when pairwise alignments are performed over all sequence pairs instead of obtaining a single multiple alignment.

<i>Twilight{205}</i>	<i>TCoffee</i>			<i>Twilight{205}</i>	<i>MUSCLE</i>		
f_D	24.88	25.06	0.005	f_D	25.30	26.50	4e-5
f_M	16.78	16.85	—	f_M	17.05	17.68	3e-4
Dali Z-score	13.41	13.60	1e-4	Dali Z-score	13.83	14.40	3e-5
GDT_TS	12.74	12.89	7e-8	GDT_TS	13.16	13.69	2e-7
ContactA	7.70	7.79	0.002	ContactA	8.01	8.34	4e-4
ContactB	10.17	10.29	0.01	ContactB	10.75	10.98	—
<i>Twilight{205}</i>	<i>ProbCons</i>			<i>Twilight{205}</i>	<i>MUMMALS</i>		
f_D	26.23	26.49	4e-4	f_D	29.13	29.17	0.02
f_M	17.92	17.96	0.04	f_M	19.64	19.65	—
Dali Z-score	13.46	13.74	4e-10	Dali Z-score	15.06	15.10	0.03
GDT_TS	12.88	13.10	5e-11	GDT_TS	14.24	14.26	—
ContactA	8.09	8.15	5e-4	ContactA	8.93	8.94	—
ContactB	10.99	10.94	—	ContactB	11.90	11.92	—

Table 4.11: Average f_D and f_M scores and average normalized Dali Z-score, GDT_TS score, and ContactA and ContactB scores (in %) of NRAAlign on the Superfamily subset of SABmark 1.65 when pairwise alignments are performed over all sequence pairs instead of obtaining a single multiple alignment.

<i>Superfamily{422}</i>	<i>TCoffee</i>			<i>Superfamily{422}</i>	<i>MUSCLE</i>		
f_D	50.73	51.01	1e-13	f_D	50.79	51.79	3e-16
f_M	38.09	38.24	5e-9	f_M	38.16	38.85	3e-11
Dali Z-score	33.82	33.98	3e-11	Dali Z-score	33.80	34.60	2e-19
GDT_TS	31.81	31.95	2e-13	GDT_TS	31.84	32.52	2e-22
ContactA	23.11	23.19	2e-6	ContactA	23.21	23.74	9e-20
ContactB	28.85	28.91	0.003	ContactB	29.10	29.51	3e-9
<i>Superfamily{422}</i>	<i>ProbCons</i>			<i>Superfamily{422}</i>	<i>MUMMALS</i>		
f_D	51.60	52.27	1e-28	f_D	54.79	54.83	3e-6
f_M	39.10	39.45	7e-19	f_M	41.06	41.08	5e-5
Dali Z-score	33.56	34.23	7e-45	Dali Z-score	35.67	35.64	1e-5
GDT_TS	31.72	32.18	3e-39	GDT_TS	33.34	33.36	1e-5
ContactA	23.29	23.63	3e-25	ContactA	24.65	24.64	0.01
ContactB	29.28	29.53	4e-9	ContactB	30.74	30.73	—

Table 4.12: Average SPS scores (in %) of NRAlign on HOMSTRAD. Each subset includes all alignments with number of sequences within the specified range.

<i>SPS</i>	<i>TCoffee</i>			<i>SPS</i>	<i>MUSCLE</i>		
2 seqs{630}	80.88	81.17	1e-6	2 seqs{630}	80.40	81.55	1e-11
3 seqs{169}	80.52	81.29	2e-9	3 seqs{169}	81.26	82.52	1e-5
4-5 seqs{122}	79.78	80.45	1e-9	4-5 seqs{122}	80.97	81.23	—
≥ 6 seqs{111}	81.55	81.94	4e-4	≥ 6 seqs{111}	82.34	82.72	0.04
<i>SPS</i>	<i>ProbCons</i>			<i>SPS</i>	<i>MUMMALS</i>		
2 seqs{630}	81.65	82.42	1e-20	2 seqs{630}	83.50	83.56	6e-6
3 seqs{169}	81.50	82.20	2e-8	3 seqs{169}	83.33	83.43	0.002
4-5 seqs{122}	82.26	82.89	2e-9	4-5 seqs{122}	83.53	83.59	0.04
≥ 6 seqs{111}	83.64	83.95	1e-7	≥ 6 seqs{111}	85.15	85.27	—

To further investigate the effect of the number of sequences on the performance of NRAlign, we group the results on HOMSTRAD according to the number of sequences in each alignment. Table 4.12 and Table 4.13 shows that except for TCoffee, the improvements on HOMSTRAD were more significant when the number of sequences is small, and the differences are especially evident when comparing pairwise alignments to multiple alignments.

Table 4.13: Average CS scores (in %) of NRAlign on HOMSTRAD. Each subset includes all alignments with number of sequences within the specified range.

<i>SPS</i>	<i>TCoffee</i>			<i>SPS</i>	<i>MUSCLE</i>		
2 seqs{630}	80.88	81.17	1e-6	2 seqs{630}	80.40	81.55	1e-11
3 seqs{169}	74.51	75.50	1e-9	3 seqs{169}	75.41	77.14	6e-6
4-5 seqs{122}	68.38	69.48	1e-10	4-5 seqs{122}	70.17	70.69	—
≥ 6 seqs{111}	59.59	61.23	8e-10	≥ 6 seqs{111}	62.03	62.80	0.04
<i>SPS</i>	<i>ProbCons</i>			<i>SPS</i>	<i>MUMMALS</i>		
2 seqs{630}	81.65	82.42	1e-20	2 seqs{630}	83.50	83.56	6e-6
3 seqs{169}	75.54	76.43	1e-6	3 seqs{169}	77.92	78.06	0.007
4-5 seqs{122}	71.69	72.94	2e-10	4-5 seqs{122}	73.58	73.74	0.02
≥ 6 seqs{111}	62.77	63.79	3e-8	≥ 6 seqs{111}	65.47	65.93	0.04

Effect of Parameters ω and β

While the same parameters ω and β are used for each modified algorithm across different benchmarks, we found that not only different algorithms have different preferences of ω and β , different benchmarks also have different preferences of ω and β , even when the same algorithm is used. Table 4.14 shows that the effect of varying ω that specifies the maximum number of horizontal positions that are included to the left and to the right was much more pronounced than varying β that specifies the weight of

Table 4.14: Average SPS and CS scores (in %) of NRAlign on HOMSTRAD by varying the parameter ω that specifies the maximum number of horizontal positions that are included to the left and to the right, and the parameter β that specifies the weight of the neighboring scores. For each modified algorithm and each score measure, the highest accuracy value and the values of ω and β that correspond to our chosen parameter setting that is the same across different benchmarks are in bold.

SPS	ProbCons					SPS	MUMMALS				
	$\omega=1$	$\omega=3$	$\omega=5$	$\omega=7$	$\omega=9$		$\omega=1$	$\omega=3$	$\omega=5$	$\omega=7$	$\omega=9$
$\beta=0.2$	81.97	82.27	82.50	82.61	82.54	$\beta=0.2$	83.72	83.72	83.64	83.49	83.18
$\beta=0.4$	82.02	82.36	82.56	82.62	82.48	$\beta=0.4$	83.70	83.70	83.60	83.34	82.94
$\beta=0.6$	82.06	82.38	82.59	82.62	82.45	$\beta=0.6$	83.72	83.72	83.54	83.27	82.84
$\beta=0.8$	82.07	82.40	82.60	82.61	82.44	$\beta=0.8$	83.72	83.72	83.52	83.24	82.78
$\beta=1.0$	82.08	82.40	82.60	82.59	82.44	$\beta=1.0$	83.73	83.70	83.50	83.21	82.75
CS	ProbCons					CS	MUMMALS				
	$\omega=1$	$\omega=3$	$\omega=5$	$\omega=7$	$\omega=9$		$\omega=1$	$\omega=3$	$\omega=5$	$\omega=7$	$\omega=9$
$\beta=0.2$	77.51	77.89	78.17	78.33	78.28	$\beta=0.2$	79.60	79.61	79.56	79.39	79.03
$\beta=0.4$	77.58	78.01	78.27	78.34	78.24	$\beta=0.4$	79.57	79.61	79.52	79.22	78.74
$\beta=0.6$	77.63	78.04	78.30	78.35	78.21	$\beta=0.6$	79.59	79.64	79.44	79.13	78.62
$\beta=0.8$	77.64	78.06	78.30	78.34	78.20	$\beta=0.8$	79.60	79.63	79.41	79.08	78.55
$\beta=1.0$	77.66	78.07	78.32	78.32	78.20	$\beta=1.0$	79.60	79.61	79.39	79.04	78.51

the neighboring scores on HOMSTRAD, and our chosen parameter setting was not the one that gives the best performance. It is possible to further improve performance significantly if another parameter setting is chosen that is different across benchmarks, even when no significant differences in performance were obtained with our chosen parameter setting.

F. Conclusion

We have developed a strategy that incorporates horizontal information in alignments that proves to be useful in all situations. Unlike previous algorithms, consistent improvements can be obtained that are mostly not dependent on the identity level, even for very high identity. To further improve performance, it may be useful to utilize different weights for neighboring scores that are at different distances from the given pair (x, y) . In addition to using horizontal information from neighboring scores in sequences, it is also possible to utilize spatial neighboring information in the local structural environment when such information is available and combine the scores from both types of neighbors.

CHAPTER V

SUMMARY AND FUTURE WORK

In this dissertation, we develop three different methods to further improve the quality of multiple sequence alignment.

In Chapter II, we propose an alternative formulation of multiple sequence alignment based on the idea of finding a multiple alignment of k sequences which preserves $k-1$ pairwise alignments as specified by edges of a given tree. Although it seems that a lot of information is lost from ignoring pairwise similarities outside of the tree, by using pairwise alignments that incorporate pairwise consistency information from other sequences, we show that it is possible to obtain very good accuracy with the preserving alignment formulation. We show that a reasonable objective function to use is to find the shortest preserving alignment, and, by a reduction to a graph-theoretic problem, that the problem of finding the shortest preserving multiple alignment can be solved in polynomial time. Such a formulation is very important as it makes it possible to know what the alignment means and also ensures that the optimal solution can be found. A software program implementing this approach (PSAlign) is available at <http://faculty.cs.tamu.edu/shsze/psalign>. To further improve accuracy from current version of PSAlign, it is possible to consider formulations other than finding the shortest solution, although many of these objective functions may be intractable to optimize. Other

directions include improving the quality of the pairwise alignments and devising strategies to perform iterative refinements after the preserving multiple alignment is obtained. A minor application of PSAlign is to identify critical sites of a multiple sequence alignment, by choosing highly scored connected components in G (corresponding to reliable columns in the final multiple alignment).

In Chapter III, we propose a few strategies for using additional hits from database search of the input sequences to significantly improve alignment accuracy. These strategies include the construction of profiles from the hits while performing profile alignment, the inclusion of high scoring hits into the input sequences, the use of intermediate sequence search to link distant homologs and the use of secondary structure information. We develop an algorithm that integrates these strategies to further improve alignment accuracy by modifying the pair-HMM model to incorporate profiles of intermediate sequences from database search and utilize secondary structure predictions. We show that our algorithm significantly outperforms current best multiple alignment algorithms with and without using additional information from database search. A software program that implements this approach (ISPAAlign) is at <http://faculty.cs.tamu.edu/shsze/ispalign>. There are still many opportunities to further improve the accuracy of ISPAAlign. Instead of adding a fixed number of intermediate sequences to the input sequences, it may be better to add more sequences as the number of input sequences increases. Alternatively, intermediate sequences can be added until all the minimum distances between each of the remaining intermediate sequences and the current set of sequences fall below a

threshold. Also, instead of modifying the parameters used by ProbCons, it may be better to re-train the pair-HMM using a set of confirmed secondary structures. It is also possible to use other multiple alignment algorithms to perform the profile alignment step as long as profiles and secondary structure predictions can be incorporated, which can lead to further improvements as better multiple alignment algorithms become available. It may also be beneficial to utilize three-dimensional structures when they are available. Besides ISPAAlign, some of the strategies proposed here may also be helpful in other applications. For example, the definition of intermediate sequence may help some approaches which need database search for remote homolog detection, and the idea of using profiles may be utilized in multiple sequence alignment approaches without additional information, by summarizing the horizontal information that is along a single sequence instead of vertical information that is across different sequences.

In Chapter IV, instead of making better use of vertical information, which include the incorporation of consistency-based pairwise alignments and the use of profile alignments, we propose a strategy to further improve accuracy of multiple alignment by taking into account alignment of neighboring residues when aligning two residues, thus making better use of horizontal information. By modifying existing multiple alignment algorithms to make use of horizontal information, we show that this strategy is able to consistently improve over existing algorithms. A software program that implements this approach (NRAAlign) is available at <http://faculty.cs.tamu.edu/shsze/nralign>. To further improve NRAAlign, it may be useful to utilize different weights for neighboring scores

that are at different distances from the given pair. In addition to using horizontal information from neighboring scores in sequences, it is also possible to utilize spatial neighboring information (or predictions) in the local structural environment when such information is available and combine the scores from both types of neighbors.

It is also possible to combine the three methods we propose to further improve multiple sequence alignment. For example, we can combine the strategy of using horizontal information from NRAlign with the strategies of using vertical information from ISPAlign. Although it is possibly useful to use the predicted contact maps of input sequences when employing NRAlign, contact maps prediction is very time-consuming and not suitable for large number of sequences. By applying NRAlign strategy to PSAlign, it is possible to only predict one contact map instead of predicting all the contact maps, thus making it possible for large scale alignments.

REFERENCES

- Altschul, S.F., and Lipman, D.J. 1989. Trees, stars, and multiple biological sequence alignment. *SIAM J. Appl. Math.* 49, 197–209.
- Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D.J. 1997. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Res.* 25, 3389–3402.
- Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., and Bourne, P.E. 2000. The protein data bank. *Nucleic Acids Res.* 28, 235–242.
- Bolten, E., Schliep, A., Schneckener, S., Schomburg, D., and Schrader, R. 2001. Clustering protein sequences - structure prediction by transitive homology. *Bioinformatics* 17, 935–941.
- Bucka-Lassen, K., Caprani, O. and Hein, J. 1999. Combining many multiple alignments in one improved alignment. *Bioinformatics* 15, 122–130.
- Capra, J.A., and Singh, M. 2007. Predicting functionally important residues from sequence conservation. *Bioinformatics* 23, 1875–1882.

- Carillo, H., and Lipman, D. 1988. The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.* 48, 1073–1082.
- Cormen, T.H., Leiserson, C.E., Rivest, R.L., and Stein, C. 2001. *Introduction to Algorithms*, 2nd ed., MIT Press, Cambridge, MA.
- Do, C.B., Gross, S.S., and Batzoglou, S. 2006. CONTRAlign: discriminative training for protein sequence alignment. *Lect. Notes Bioinform.* 3909, 160–174.
- Do, C.B., Mahabhashyam, M.S., Brudno, M., and Batzoglou, S. 2005. ProbCons: Probabilistic consistency-based multiple sequence alignment. *Genome Res.* 15, 330–340.
- Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. 1998. *Biological Sequence Analysis*. Cambridge University Press, Cambridge, UK.
- Edgar, R.C. 2004. MUSCLE: Multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.* 32, 1792–1797.
- Edgar, R.C., and Sjölander, K. 2004. A comparison of scoring functions for protein sequence profile alignment. *Bioinformatics* 20, 1301–1308.

- Feng, D., and Doolittle, R. 1987. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.* 25, 351–360.
- Gerstein, M. 1998. Measurement of the effectiveness of transitive sequence comparison, through a third “intermediate” sequence. *Bioinformatics* 14, 707–714.
- Gotoh, O. 1990. Consistency of optimal sequence alignments. *Bull. Math. Biol.* 52, 509–525.
- Gotoh, O. 1996. Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. *J. Mol. Biol.* 264, 823–838.
- Gusfield, D. 1993. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bull. Math. Biol.* 55, 141–154.
- Heger, A., Lappe, M., and Holm, L. 2003. Accurate detection of very sparse sequence motifs. *RECOMB '03*, 139–147.
- Heger, A., Lappe, M., and Holm, L. 2004. Accurate detection of very sparse sequence motifs. *J. Comput. Biol.* 11, 843–857.

- Henikoff, S. and Henikoff, J.G. 1992. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA* 89, 10915–10919.
- Holm, L. and Sander, C. 1998. Dictionary of recurrent domains in protein structures. *Proteins* 33, 88–96.
- Huang, X. and Miller, W. 1991. A time-efficient linear-space local similarity algorithm. *Adv. Appl. Math.* 12, 337–357.
- Jones, D.T. 1999. Protein secondary structure prediction based on position-specific scoring matrices. *J. Mol. Biol.* 292, 195–202.
- Just, W. 2001. Computational complexity of multiple sequence alignment with SP-score. *J. Comp. Biol.* 8, 615–623.
- Katoh, K., Kuma, K., Toh, H., and Miyata, T. 2005. MAFFT version 5: Improvement in accuracy of multiple sequence alignment. *Nucleic Acids Res.* 33, 511–518.
- Kececioglu, J.D. 1993. The maximum weight trace problem in multiple sequence alignment. *Lect. Notes Comp. Sci.* 684, 106–119.
- Kimura, M. 1983. *The Neutral Theory of Molecular Evolution*. Cambridge University

Press, Cambridge, UK.

Knuth, D.E. 1997. *The Art of Computer Programming, Vol. 1: Fundamental Algorithms*, 3rd ed., Addison-Wesley, Reading, MA.

Lassmann, T., and Sonnhammer, E.L.L. 2005. Kalign—an accurate and fast multiple sequence alignment algorithm. *BMC Bioinformatics* 6, 298–306.

Lee, C., Grasso, C., and Sharlow, M.F. 2002. Multiple sequence alignment using partial order graphs. *Bioinformatics* 18, 452–464.

Li, W., Jaroszewski, L., and Godzik, A. 2002. Tolerating some redundancy significantly speeds up clustering of large protein databases. *Bioinformatics* 18, 77–82.

Li, W., Pio, F., Pawłowski, K., and Godzik, A. 2000. Saturated BLAST: An automated multiple intermediate sequence search used to detect distant homology. *Bioinformatics* 16, 1105–1110.

Margelevičius, M., and Venclovas, Č. 2005. PSI-BLAST-ISS: An intermediate sequence search tool for estimation of the position-specific alignment reliability. *BMC Bioinformatics* 6, 185–194.

- Marti-Renom, M.A., Madhusudhan, M.S., and Sali, A. 2004. Alignment of protein sequences by their profiles. *Protein Sci.* 13, 1071–1087.
- Mizuguchi, K., Deane, C.M., Blundell, T.L., and Overington, J.P. 1998. HOMSTRAD: A database of protein structure alignments for homologous families. *Protein Sci.* 7, 2469–2471.
- Morgenstern, B., Dress, A., and Werner, T. 1996. Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proc. Natl. Acad. Sci. USA* 93, 12098–12103.
- Morrison, D.A. 1996. Phylogenetic tree-building. *Int. J. Parasitology* 26, 589–617.
- Murzin, A.G., Brenner, S.E., Hubbard, T., and Chothia, C. 1995. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.* 247, 536–540.
- Needleman, S.B., and Wunsch, C.D. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.* 48, 443–453.
- Notredame, C., Higgins, D.G., and Heringa, J. 2000. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.* 302, 205–217.

- Notredame, C., Holm, L., and Higgins, D.G. 1998. COFFEE: An objective function for multiple sequence alignments. *Bioinformatics* 14, 407–422.
- O'Sullivan, O., Suhre, K., Abergel, C., Higgins, D.G., and Notredame, C. 2004. 3DCoffee: Combining protein sequences and structures within multiple sequence alignments. *J. Mol. Biol.* 340, 385–395.
- Panchenko, A.R., Kondrashov, F., and Bryant, S. 2004. Prediction of functional sites by analysis of sequence and structure conservation. *Protein Sci.* 13, 884–892.
- Park, J., Teichmann, S.A., Hubbard, T., and Chothia, C. 1997. Intermediate sequences increase the detection of homology between sequences. *J. Mol. Biol.* 273, 349–354.
- Pei, J. and Grishin, N.V. 2006. MUMMALS: Multiple sequence alignment improved by using hidden Markov models with local structural information. *Nucleic Acids Res.* 34, 4364–4374.
- Pei, J. and Grishin, N.V. 2007. PROMALS: Towards accurate multiple sequence alignments of distantly related proteins. *Bioinformatics* 23, 802–808.
- Pei, J., Kim, B.-H., and Grishin, N.V. 2008. PROMALS3D: A tool for multiple protein

sequence and structure alignments. *Nucleic Acids Res.* 36, 2295–2300.

Pevzner, P.A. 2000. *Computational Molecular Biology: An Algorithmic Approach*, MIT Press, Cambridge, MA.

Roshan, U. and Livesay, D.R. 2006. Probalign: Multiple sequence alignment using partition function posterior probabilities. *Bioinformatics* 22, 2715–2721.

Rychlewski, L., Fischer, D., and Elofsson, A. 2003. LiveBench-6: Large-scale automated evaluation of protein structure prediction servers. *Proteins* 53, 542–547.

Salamov, A.A., Suwa, M., Orengo, C.A., and Swindells, M.B. 1999. Combining sensitive database searches with multiple intermediates to detect distant homologues. *Protein Eng.* 12, 95–100.

Simossis, V.A., Kleinjung, J., and Heringa, J. 2005. Homology-extended sequence alignment. *Nucleic Acids Res.* 33, 816–824.

Smith, T.F., and Waterman, M.S. 1981. Identification of common molecular subsequences. *J. Mol. Biol.* 147, 195–197.

Spang, R., Rehmsmeier, M., and Stoye, J. 2002. A novel approach to remote homology

detection: jumping alignments. *J. Comput. Biol.* 9, 747–760.

Stoye, J. 1998. Multiple sequence alignment with the divide-and-conquer method. *Gene* 211, GC45–56.

Taylor, W.R. 1987. Multiple sequence alignment by a pairwise algorithm. *Comp. Appl. Biosci.* 3, 81–87.

Taylor, W.R. 1988. A flexible method to align large numbers of biological sequences. *J. Mol. Evol.* 28, 161–169.

Thompson, J.D., Higgins, D.G., and Gibson, T.J. 1994. CLUSTALW: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. *Nucleic Acids Res.* 22, 4673–4680.

Thompson, J.D., Koehl, P., Ripp, R., and Poch, O. 2005. BAliBASE 3.0: Latest developments of the multiple sequence alignment benchmark. *Proteins* 61, 127–136.

Thompson, J.D., Plewniak, F., and Poch, O. 1999. A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Res.* 27, 2682–2690.

Van Walle, I., Lasters, I., and Wyns, L. 2004. Align-m—a new algorithm for multiple alignment of highly divergent sequences. *Bioinformatics* 20, 1428–1435.

Venclovas, Č., Zemla, A., Fidelis, K., and Moult, J. 2003. Assessment of progress over the CASP experiments. *Proteins* 53, 585–595.

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

Wallace, I.M., O’Sullivan, O., Higgins, D.G. and Notredame, C. 2006. M-Coffee: Combining multiple sequence alignment methods with T-Coffee. *Nucleic Acids Res.* 34, 1692–1699.

Wallner, B. and Elofsson, A. 2003. Can correct protein models be identified? *Protein Sci.* 12, 1073–1086.

Wilcoxon, F. 1947. Probability tables for individual comparisons by ranking methods. *Biometrics* 3, 119–122.

Yamada, S., Gotoh, O., Yamana, H. 2006. Improvement in accuracy of multiple sequence alignment using novel group-to-group sequence alignment algorithm with piecewise linear gap cost. *BMC Bioinformatics* 7, 524–540.

- Zemla, A., Venclovas, Č., Moult, J., and Fidelis, K. 1999. Processing and analysis of CASP3 protein structure predictions. *Proteins Suppl.* 3, 22–29.
- Zhang, Y. and Skolnick, J. 2004. Scoring function for automated assessment of protein structure template quality. *Proteins* 57, 702–710.
- Zhou, H. and Zhou, Y. 2005. SPEM: Improving multiple sequence alignment with sequence profiles and predicted secondary structures. *Bioinformatics* 21, 3615–3621.

VITA

Name: Yue Lu

Address: Department of Biochemistry & Biophysics
Texas A&M University
2128 TAMU
College Station, TX 77843

Email Address: luyue68@hotmail.com

Education: B.S., Cell Biology and Genetics, Peking University, 2002
Ph.D., Biochemistry, Texas A&M University, 2008