UNIVERSITY OF CALIFORNIA, SAN DIEGO

# Discovery and analysis of mosaic arrangements in biological sequences and structures

A dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in

Bioinformatics

by

Degui Zhi

Committee in charge:

Professor Pavel Pevzner, Chair
Professor Vineet Bafna
Professor Charles Elkan
Professor Adam Godzik
Professor Trey Ideker
Professor William Loomis

2006

The dissertation of Degui Zhi is approved, and it is acceptable in quality and form for publication on microfilm:

_____

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2006

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

# ACKNOWLEDGEMENTS

While my doctoral study is part of my personal pursuit of scientific career, I realize it is definitely impossible without all the wonderful people I met.

First, I thank my advisor, Pavel Pevzner, for his vision, guidance, and support throughout my doctoral study. Among other things, he teaches me the importance of a problem-oriented attitude as a bioinformatician. I also thank Bill Loomis, my co-advisor, for constantly reminding me how bioinformatics is deeply rooted in biology. I enjoy every conversation with him since my first year in PhD. I am grateful to Adam Godzik, who introduces me into the field of structural bioinformatics, and gives me support in need.

I am fortunate to be in a research group with talented people. I cannot forget Haixu Tang, Ben Raphael, Alkes Price, Uri Keich, Steffen Heber, Shaojie Zhang, Mark Chaisson, Max Alekseyev, Nuno Bandeira, Neil Jones, Vikas Bansal and other members of Pevzner group, who share vivid discussions on topics from scientific research to everything in life. I spend a significant amount of time at the Burnham Institute during my final year of PhD. I enjoy meeting Yuzhen Ye, Haibo Cao, SS Krishna, and other members of Godzik group. Also, I thank my classmates at the Bioinformatics program at UCSD, who share happy times with me during study and play, especially Eugene, who also shares an apartment with me.

I thank Neil Jones, Haixu Tang, Yuzhen Ye, Ben Raphael, Shaojie Zhang, Eugene Ke, and Lu Tang for critical reading of the dissertation.

Finally, I would like to thank my family and friends for their support through my doctoral study. Especially, I thank my wife Lu, for her constant trust and support. Last but not least, I am indebt to my parents and brothers who helped me to become the person I am today.

The material in Chapter 3 is a reprint of material appearing in Genome Research. The material in Chapter 4 is a reprint of material to appear in Genome Biology. The material in Chapter 6 is a reprint of material submitted for publication. I wish to thank my co-authors Ben Raphael, Haixu Tang, Alkes Price, S.S. Krishna, Haibo Cao, Adam Godzik, and Pavel Pevzner.

# VITA

| | |
|---|---|
| 1997 | B.S., Peking University, Beijing, China |
| 1997 – 1999 | Graduate Student Researcher, School Of Computing, National University Of Singapore |
| 1999 | M.S., National University Of Singapore |
| 1999 – 2000 | Research Assistant, Department Of Computational Science, National University Of Singapore |
| 2001 – 2005 | Research Assistant, Department Of Computer Science And Engineering, University Of California, San Diego |
| 2006 | Ph.D., University Of California, San Diego |

# PUBLICATIONS


D. Zhi, S.S. Krishna, H. Cao, P. Pevzner, and A. Godzik. "Representing and comparing protein structures as curves in three-dimensional space", *submitted to Bioinformatics*, 2005.

D. Zhi, B. Raphael, A. Price, H. Tang and P. Pevzner. "Identifying repeat domains in large genomes", *Genome Biology, accepted*, 2005.

D. Zhi, U. Keich, P. Pevzner, S. Heber, and H. Tang. "Checking for base-calling errors in repeats", *IEEE/ACM Transactions in Computational Biology and Bioinformatics, accepted*, 2005.

S. Roepcke, D. Zhi, M. Vingron, and P. Arndt. "Identification of localized sequence motifs in the proximal promoters of human ribosomal protein genes", *Gene, accepted*, 2005.

B. Raphael, D. Zhi, H. Tang, and P. Pevzner. “A Novel method for multiple sequence alignment of sequences with repeated and shuffled elements”, *Genome Research*, 14: 2336-46, 2004.

X. Chen, Z.L. Ji, D.G. Zhi, Y.Z. Chen. “CLiBE: a database of computed ligand binding energy for ligand/receptor complexes”, *Computers & Chemistry*. 26(6):661-6, 2002.

Y.Z. Chen, and D.G. Zhi. “Ligand-protein inverse docking and its potential use in computer search of putative protein targets of a drug”, *Proteins*, 43(2): 217-26, 2001.

ABSTRACT OF THE DISSERTATION

Discovery and analysis of mosaic arrangements in biological sequences and structures

By

Degui Zhi

Doctor of Philosophy in Bioinformatics

University of California, San Diego, 2006

Professor Pavel Pevzner, Chair

Biological molecules are composed of discrete units, called *domains*. The study of the identity and organization of these domains can reveal the correspondence between individual units in different molecules, and the history of domains themselves, which may guide our understanding of the evolutionary history of individual molecules. Currently, the study of domain organization in protein sequences is a mature field; however, the studies of domain organization in other types of biological sequences and protein structures are still in their infancy. There is currently no general framework and specific tools for the identification of domains or for the discovery of the domain organization.

Existing tools do not explicitly define what a domain is. In some cases, existing tools (e.g., multiple sequence alignment tools) ignore domain organizations entirely, or represent only a limited subset of domain organization. As a result, the mosaic structures of biological data are left undetected, and we demonstrate that the prevalence of mosaic arrangements is under-appreciated.

This dissertation considers shortcomings of current technologies and develops a generic framework for the discovery and analysis of domain organizations in any types of sequential data. We apply this framework in several biological contexts. First we develop the A-Bruijn Aligner (ABA), which represents a multiple sequence alignment (MSA) as a graph that automatically reveals the domain structures. Second, we develop a repeat domain graph approach that decomposes a repeat family library into repeat domains, which is the first method for the comprehensive identification of repeat domains in large genomes. Third, we extend the A-Bruijn graph approach to an exploration of the mosaic arrangements in protein structures. Finally, we propose a new method for structure comparison based on a simplified representation of protein structures using the local curvatures along their generalized backbones.

# 1 Introduction: mosaic arrangements and open biological problems

One of the important results of the analysis of the human genome [1] is the discovery of a large number of low-copy repeats, also called segmental duplications [2, 3]. Segmental duplications have drawn considerable attention due to their implications in mammalian evolution [3-7] and genomic diseases [8-10]. Segmental duplications exhibit a complex mosaic structure [11-13], which is hypothesized to be a result of a series of duplication events: initially a set of independent ancestral donor loci (duplicons) are copied to the so-called acceptor region, which tend to lie in pericentromeric regions, thus forming a mosaic of segments from different genomic locations. Subsequently, the now-mosaic acceptor region is duplicated to various genomic loci. Although Eichler and colleagues [14, 15] had systematically identified segmental duplication regions in the human genome, the problems of identification of duplicons and the elucidation of segmental duplication events remained open, due to the lack of algorithmic tools for disentangling the complex mosaic structures. The challenge is that it is non-trivial to derive duplicons from a set of more than 25,000 pairwise alignments [14-16].

Only recently, Jiang et al [17] presented an elegant solution to this problem by constructing a segmental duplication graph such that the edges in the graph correspond to duplicons and the topology of the graph reveals the evolutionary history of segmental duplications. While this study demonstrated how computational methods can be crucial for solving the key problems in evolution of segmental duplications, similar approaches for revealing mosaic structures in other biological sequences

remain poorly developed, and are the focus of this work. Specifically, the goal of present work is to develop a general framework for the discovery and analysis of mosaic arrangements in other types of biological data, which include nucleotide and amino acid sequences, as well as protein structures (as sequences of structural units).

It is well established that proteins consist of protein domains. A protein domain is generally defined as a conserved sequence, which folds into a specific 3-D conformation, and has a specific function. Protein domains are the units of protein structure and function. Therefore, domain duplication, shuffling, and other domain rearrangements are among the most important events in protein evolution. With the availability of protein domains databases and automatic tools [18-25], large scale studies of organization of protein domains become possible [26, 27], and the tools for automatic domain identification [28] are being developed. While protein domains and their organization have been the focus of active research for decades, the understanding of other biological sequences' mosaic structures has just begun. As we will demonstrate later in this dissertation, interspersed transposons/retrotransposons also often exhibit a mosaic structure.

Transposon repeats are classified into repeat families according to their global sequence similarity. However, different repeat families typically share subparts of their sequences with other families or with other parts of their own sequences. For example, the ubiquitous human *Alu* family is dimeric [29]. Different retroviral elements in human genome exhibit a complicated scenario of recombinations. An extreme example of such recombination is the Harlequin family, which is a mosaic of 12 fragments from 7 distinct other retroviral families [30]. The understanding of this

mosaic structure yields important insights into the evolution of repeat families. However, even with the extensive manual curation of repeat families in the Repbase [30], the sharing of sequences among repeat families in the human genome and their mosaic structures is still poorly understood. It is even more difficult to study the sharing of sequences among repeat families in newly sequenced genomes as the repeat family sequences are typically generated by *de novo* repeat identification programs. An even more challenging task is to compare repeat families across different organisms. When comparing repeats in the newly sequenced *C. briggsae* genome with that in the *C. elegans* genome, Stein et al [31] noted that no simple one-to-one mapping exists between *C. briggsae* repeat families and *C. elegans* repeat families. Obviously, without first delineating the underlying mosaic structure of repeat families, it is not possible to answer questions about how repeats are shared across species, nor about which repeat domains are shared among species. In fact, it is not clear that we can say anything at all about the evolution of repeat domains without first discovering what they are.

The prevalence of mosaic arrangements in biological sequences calls for the development of a general framework for the analysis of mosaic arrangements in biological sequences. In particular, the following questions need to be addressed: given a set of sequences, (i) how can we identify all *domains* (subsequences) shared in the set? (ii) how can we identify domain *organizations* among the sequences in the set? (iii) what evolutionary events can we infer from the domain organizations? and finally (iv) how can we compare domains shared between two different sets of sequences?

Close to the core of the problems of identifying domains, lies the problem of multiple sequence alignment. Multiple sequence alignment has been an active research area since the mid-1970's. Traditionally a multiple alignment of several sequences is represented in a row-column format. This representation is sufficient if all input sequences share the same domain organization. However, when the input contains sequences with different domain organizations, traditional multiple alignment programs can only align a subset of domains that appear in the same order in all the sequences. Domains that do not appear in the same order are left unaligned and padded with long gaps. Thus, the alignment of sequences with multiple domains and varied domain organizations requires a representation more flexible than the rigid row-column format.

The A-Bruin graph, a generalization of the classical de Bruijn graph, provides a framework for the analysis of a set of sequences with a set of arbitrarily defined similarities. Thus the A-Bruijn graph approach is particularly well-suited for the modeling of mosaic arrangements. The most important contribution of this work is that it provides the first framework capable of discovering and analyzing mosaic arrangements in biological sequences and structures. We further demonstrate the utility of this approach in several important problems in bioinformatics. The remaining chapters of the dissertation are organized as following.

Chapter 2 introduces the general concept of the A-Bruijn graph and algorithms for the construction and simplification of A-Bruijn graphs, which serves as the background and a starting point for this work.

Chapter 3 presents A-Bruijn Alignment (ABA), a new method for the problem of multiple sequence alignment. The major difference between ABA and existing multiple alignment methods is that ABA represents an alignment as a directed graph, possibly containing cycles. This representation provides more flexibility than a traditional alignment matrix or the recently introduced Partial Order Alignment (POA) [32] graph by allowing for a larger class of evolutionary relationships between the aligned sequences. While this is a collaboration work, I developed the ABA software package and the case studies; also I participated the entire writing process.

Chapter 4 develops new methods for the analysis of repeat family libraries via the A-Bruijn graph approach. We build a repeat domain graph that decomposes a repeat library into repeat domains, defined as short subsequences shared by multiple repeat families, and reveals the mosaic structure of repeat families. Our method recovers documented mosaic repeat structures and suggests additional putative ones. Our method is useful for elucidating the evolutionary history of repeats and annotating de novo generated repeat libraries.

While A-Bruijn graph is developed for modeling biological sequences, its application is not limited to sequences. Chapter 5 explores the possibility of extending the A-Bruijn graph approach to the analysis of repeated and shuffled protein domains in structures. This exploration inspired the development of a new method for structure alignments.

In chapter 6 we propose a new structure comparison approach based on a simplified representation of proteins that describes its three-dimensional path by local curvature along the generalized backbone of the polypeptide. We implement a

dynamic programming procedure that aligns curvatures of the two proteins optimizing a defined sum turning angle deviation measure.

Chapter 7 concludes this dissertation and points out several directions for future research.

# 2 A-Bruijn graph

The notion of A-Bruijn graph is the key concept in the present work. This chapter reviews the definitions and algorithms for A-Bruijn graph. The A-Bruijn graph is a generalization of the classical de Bruijn graph. In Section 2.1 the concept of de Bruijn graph and its applications in computational biology are discussed. The limitations of de Bruijn graph lead to the development of A-Bruijn graph. Section 2.2 gives a formal definition of A-Bruijn graph. The A-Bruijn graphs built from biological sequences are typically very complicated and the algorithms for the simplification of A-Bruijn graph are essential to their successful application in computational biology. Section 2.3 reviews the key algorithms for the simplification of A-Bruijn graph, and discusses alternative problem formulations. This chapter concludes with a brief discussion of the applications of A-Bruijn graph outsides the scope of the remaining chapters in the dissertation.

## 2.1 de Bruijn graph and its applications in computational biology

The de Bruijn graph [33] is a well-known concept in mathematics and computer science. Given a set of sequences of length $l$, the de Bruijn graph over these sequences can be constructed as following: represent each sequence as a vertex, and connect an directed edge from vertex $a$ to vertex $b$ if the ($l$-1)-suffix of the sequence $a$ is identical to the ($l$-1)-prefix of the sequence $b$. The de Bruijn graph of a single long sequence can be constructed by first fragmenting the entire sequence into overlapping $l$-mers, and then applying the above procedure over the set of fragments. The entire sequence corresponds to an Eulerian path of the de Bruijn graph.

In bioinformatics, de Bruijn graph was first applied to the problem of Sequencing by Hybridization [34]. Later, its application has been extended to many other bioinformatics problems including fragment assembly [35, 36], resequencing using DNA arrays [37, 38], EST analysis [39], and computational mass spectrometry [40].

Effectively, in constructing the de Bruijn graph of a sequence, all occurrences of identical *l*-mers in the sequence are "glued" together into the same node. If the input sequence is a genomic sequence, the de Bruijn graph can glue all identical repeat copies into a single edge with a high multiplicity. This provides a scheme for de novo repeat identification.

However, the *l*-mer gluing in the original de Bruijn graph definition is limited to *identical* repeat copies, which is too rigid for the analysis of biological sequences that are subject to mutations during evolution. Indeed, minor variations among different repeat copies can greatly complicate the topology of resulting de Bruijn graph.

Pevzner, Tang, and Tesler [41] proposed the A-Bruijn graph approach, which extended the classical de Bruijn graph concept, in an attempt to allow for a broader definition of sequence similarity. In principle, given a set of sequences as well as a set of pairwise local alignments between these sequences, A-Bruijn graph is built as following: represent each sequence as a linear chain of vertex, and glue two vertices together if they are aligned in one of the input pairwise alignments. In practice, as the input pairwise alignments may not be consistent, the straightforward result of A-Bruijn graph construction procedure may contain many short cycles in two classes, bulges and whirls (see below). Several heuristics must be applied to further simplify

the graph to make it coherent and interpretable. In the remaining of this chapter, I first give a formal introduction to the A-Bruijn graph, and then describe algorithms for the simplification of A-Bruijn graphs.

## 2.2 A-Bruijn graph

Our presentation of A Bruijn graph includes an introduction of the concept of A-Bruijn graph and discussions on alternative definitions and problem formulations that may provide room for future improvements.

Let $S$ be a sequence of $n$ letters and $\mathscr{A}$ be a set of pairwise alignments between subsequences of $S$. Let $A = \{a_{ij}\}$ be a binary $n$ x $n$ "similarity matrix" that is based on $\mathscr{A}$. Assuming transitivity of the similarity, matrix $A$ is defined as $a_{ij} = 1$ if the positions $i$ and $j$ are aligned in any pairwise alignment and $a_{ij} = 0$ otherwise. Notice that matrix $A$ naturally defines the "adjacency matrix" of a graph on $n$ vertices 1,..., $n$ (vertices $i$ and $j$ are connected iff $a_{ij} = 1$). This graph is called the A-graph. Each connected component of the A-graph represents a set of positions that are (transitively) aligned. In short, A-Bruijn graph is built by “gluing” these transitively aligned positions into a single node[1]. Formally, let $V$ be the set of connected components of A-graph and let $v_i \in V$ be the connected component containing vertex $i$ ($1 \le i \le n$). The $A$-Bruijn graph is defined as the multi-graph on the vertex set $V$ with ($n$-1) directed edges ($v_i$, $v_{i+1}$) for $1 \le i < n$. The A-Bruijn graph can be viewed as the Eulerian path obtained from the path (1, ..., $n$) after contracting each connected

[1] We limit our use of terminology: *vertices* in the A-graph and *nodes* in the A-Bruijn graph.

component in the A-graph into a single node. $v_1$ and $v_n$ are called the source and sink. See Figure 2.1 for an example of A-Bruijn graph construction.

Based on this constructive definition, there is a one-to-one correspondence between the positions in the input sequences and the vertices in the A-graph, and there is a many-to-one mapping from the vertices in the A-graph to the nodes in the A-Bruijn graph. In other words, for each position in an input sequence, there is a corresponding node in the A-Bruijn graph. Thus, the A-Bruijn graph is accurate to the single letter level, i.e., it is possible to "thread" the input sequence onto the A-Bruijn graph letter-by-letter.

It is straightforward to define the A-Bruijn graph over $n$ ($n > 1$) sequences based on a set of pairwise alignments among subsequences of them. One can build the A-Bruin graph for $n$ sequences by first concatenating them into a single one and constructing the A-Bruijn graph for it, then removing the concatenation edges. The resulting A-Bruijn graph for $n$ sequences can be viewed as the amalgamation of $n$ paths with $n$ sources and $n$ sinks[2].

The definition of A-Bruijn graph allows an arbitrary definition of the matrix $A$. When the matrix $A$ corresponds to all perfect $l$-mer matches in the sequence S, the A-Bruijn graph corresponds to the classical de Bruijn graph (with minor technical modifications). In the context of biological sequences, $\mathscr{A}$ is often a set of significant pairwise local alignments produced by a pairwise sequence alignment program.

[2] To avoid the situation when two sources or two sinks are glued into a single vertex (i.e. when the ends of two different sequences are aligned in one alignment in $\mathscr{A}$), we add virtual vertices at the ends of a sequence with zero length edges.

Therefore, the concept of A-Bruijn graph generalizes the concept of de Bruijn graph, and allows biologically meaningful sequence similarities to be represented in a graph.

Although the construction of A-Bruijn graph is relatively straightforward, the direct application of the above procedure to real biological sequences often results in a very noisy graph, i.e., a graph with a large number of short cycles. Short cycles can be defined as cycles longer than a threshold, called *girth*. Two types of short cycles are identified: whirls and bulges (Figure 2.2). Whirls are short cycles in which all edges are oriented in the same direction, while bulges are short cycles that contain edges with different directions. Whirls and bulges are formed by different causes. Bulges are often caused by gaps in alignments, while whirls are usually the result of inconsistencies among input pairwise alignments. When there is a large number of similar sequences with inconsistent alignments, bulges and whirls can form a complicated network and consequently the resulting A-Bruijn graph might no longer be intuitive and informative. To better interpret A-Bruijn graph, one has to distinguish the graph topology that corresponds to true sequence similarities from the graph structures caused by technical inconsistencies among input alignments. Thus, algorithms for simplifying A-Bruijn graph are essential to the application of A-Bruijn graph to modeling biological sequences.

## 2.3 Algorithms for the simplification of A-Bruijn graph

The major challenge of simplifying A-Bruijn graph is the removal of whirls, which result from inconsistencies among input pairwise alignments. Reconciling inconsistencies among a set of pairwise alignments is a difficult task [42]. The A-

Bruijn graph representation transforms this problem into a graph theoretic problem. Whirls correspond to nodes in the A-Bruijn graph (connected components in the A-graph) that contain two positions on a sequence that are less than *girth* apart. The goal of whirl removal is to split the positions (by removing minimal number of edges in the A-graph) in such nodes so that no two positions in a node are less than *girth* apart on a sequence. Although it was not stated directly in [41], this is a hard combinatorial problem, and in [41] a greedy iterative node splitting procedure is introduced.

After whirl removal, the resulting A-Bruijn graph represents a set of consistent alignments among the input sequences. Since the whirl removal procedure is essentially performed on the A-graph, and thus preserves the mapping between the vertices in the A-graph and the nodes in the A-Bruijn graph, therefore, the letter-by-letter correspondence between the A-Bruijn graph and the input sequences is preserved.

However, because the whirl-free A-Bruijn graph may still contain bulges and other complications that obscure the overall graph topology, further simplification of the A-Bruijn graph is desired. In [41] a bulge removal procedure is described.

Comparing the whirl removal, the problem of bulge removal is relative simple. The A-Bruijn graphs can be viewed as weighted graphs with the weight (multiplicity) of an edge between two vertices equal to the number of edges connecting these vertices. The problem of bulge removal can be formulated as the removal of edges with minimal total weights such that the remaining graph does not have a cycle longer than girth. Although theoretically this is the Maximum Subgraph with Large Girth (MSLG) Problem, which is a hard problem, in [41] a heuristic approach that peels a

bulge network into its maximum spanning tree (MST) provides a practical solution. After the removal of a bulge network, the remaining graph can be further simplified through the erosion and zigzag path straightening steps. A warning with the bulge removal procedure is that it performed on the A-Bruijn graph, thus the mapping between the A-graph and the A-Bruijn graph may be destroyed, and the resulting A-Bruijn graph may not be an exact letter-by-letter threading of the input sequences.

While the above solution to MSLG problem formulation provides a practical procedure for bulge removal, here I provide an alternative approach to the bulge-removal problem, which may be of some theoretical interests. First, an observation is that network of bulges represents a section of multiple alignment with partial order. Since the goal of bulge removal is to reveal such sections and give them a simplified representation, one can consider the alternative procedure which first identify a network of bulge, and then optimizes the alignment within the network via traditional multiple alignment procedures or partial order alignment procedures [32].

In A-Bruijn graph there are simple chains of nodes linked by parallel edges. These chains represent aligned regions among several sequences, and the A-Bruijn graph can be simplified by collapsing such simple chains into single edges with the length being the number of nodes in the path and the multiplicity being the multiplicity of the edges (Figure 2.3).

After this collapsing, there still may be a number of short collapsed edges, due to ambiguities at the boundaries of aligned regions. [41] provides one heuristic for the reconciliation of the alignment boundaries. First define *important edges* as edges of high multiplicity and edges with length greater than some threshold. Then apply a 2

step rethreading procedure: (i) remove the unimportant edges; (ii) thread the input sequences through the remaining important edges. For an aligned region, this procedure essentially takes the minimum among all pairwise alignments. The problem of boundary reconciliation is not well formulated. One possibility is to minimize some entropy measure around the boundaries.

When the bulge removal procedure is performed on the A-Bruijn graph, the original mapping between the A-graph and the A-Bruijn graph may be lost, and the resulting A-Bruijn graph may not be the exact letter-by-letter threading of the input sequences. However, the A-Bruijn graph is still accurate at the boundaries of the simple chains, which define boundaries of aligned regions.

## 2.4 Some applications of A-Bruijn graph

In [41], A-Bruijn graph approach is developed for the tasks of *de novo* repeat identification and fragment assembly. For the problem of *de novo* repeat identification, the A-Bruijn graph framework provides an explicit representation to the repeat boundary problem, and reveals more details of the similarity/dissimilarities between repeat families/subfamilies. The algorithms for graph construction and simplification are implemented as the `RepeatGluer` package. For the problem of fragment assembly, the A-Bruijn graph approach is implemented as the `EULER+` assembler, which is an improvement over the `EULER` assembler [35] in handling low-quality regions of reads, where *l*-mer matches for de Bruijn graph are problematic.

As we mentioned in the introduction, the A-Bruijn graph approach has been applied to the analysis of segmental duplications [17]. Segmental duplications

represent an important feature of mammalian evolution. Utilizing the A-Bruijn graph approach, segmental duplications in human genome can be decomposed to a complex mosaic of independent duplication units and the ancestors of these units can be derived.

## 2.5 Figures

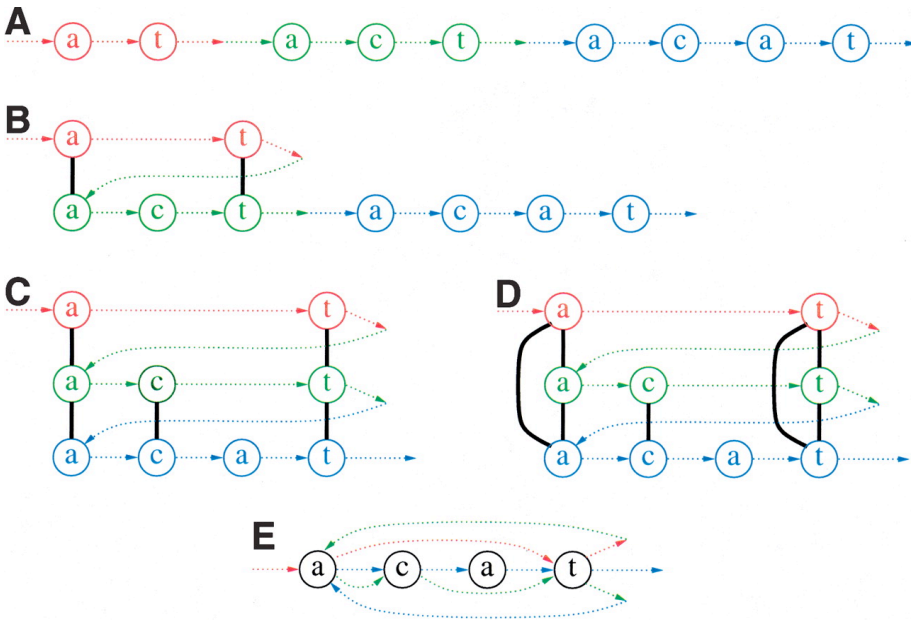



**Figure 2.1: (*A*) Construction of the *A*-graph from the sequence...at...act...acat by applying three pairwise alignments (*B*) a-t versus act, (C) act versus acat, and (*D*) a-t versus acat. (*D*) The *A*-graph consists of the eight nodes plus the seven thick, black edges created from the alignments; the colored edges are shown to indicate the relation of the nodes to the sequence, but they are not part of the *A*-graph. (*E*) Each of these alignments serves as "gluing instructions" that transform the sequence into the *A*-Bruijn graph on four vertices; the colored edges are in the *A*-Bruijn graph, although the coloring itself is not. (Source: Pevzner, P.A., H. Tang, and G. Tesler, Genome Res, 2004. 14(9) [41])**

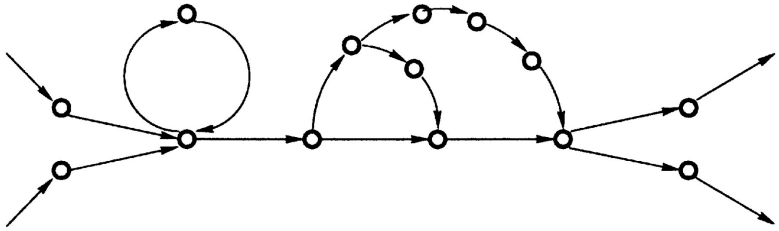


**Figure 2.2: A repeat region in an *A*-Bruijn graph in which alignment inconsistencies have caused a whirl and a network of bulges. (Source: Pevzner, P.A., H. Tang, and G. Tesler, Genome Res, 2004. 14(9) [41])**

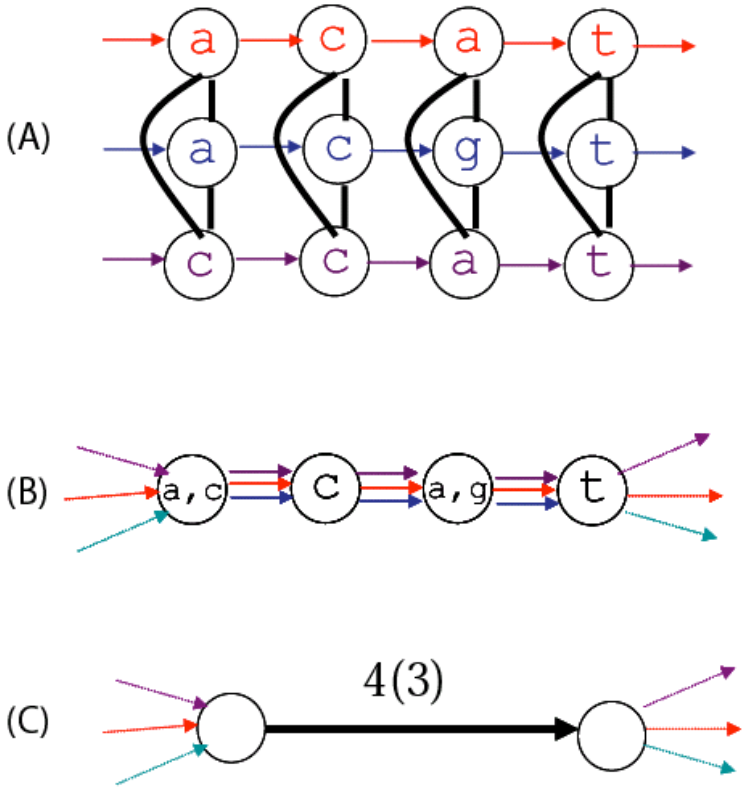



**Figure 2.3: Merging simple chain into a single edge. (A) A-graph; (B) A-Bruijn graph; (C) collapsed simple chain, lebeled with *l*(*m*), where *l* is the number of nodes in the chain and the *m* is the number of vertices in a node.**

# 3 Multiple alignment of sequences with shuffled and repeated domains

## 3.1 Introduction

Multiple sequence alignment (MSA) is arguably among the most studied [42-54] and difficult [55] problems in computational biology. An optimal MSA of $t$ sequences, each of length $n$, can be computed in $\Theta((2n)t)$ time by dynamic programming [43, 44]. However, such an approach is not practical for more than a few sequences. Consequently, a large body of research exists for the design of efficient heuristics for MSA; see [56] for a recent review. Currently, popular programs include CLUSTALW [49], T-COFFEE [57], DIALIGN [42], MultiPipMaker [53], and MACAW [51]. However, these programs (and the majority of alignment algorithms) consider the sequences to be aligned as having resulted from an evolutionary process that includes only point mutations and (small) insertions/deletions. Accordingly, a multiple sequence alignment of $t$ sequences is often represented in row-column format: the sequences are listed in $t$ rows with "space characters" (dashes) inserted in positions of indels, and columns indicating aligned positions.

This representation of the alignment as a linear sequence of alignment columns implicitly assumes that all regions of all sequences are similar over their entire length. However, for many biological sequences this assumption does not hold. For example, multi-domain protein families evolve not only through mutation of individual amino acids, but also through operations such as domain duplications and domain

recombinations [58]. Experienced users of multiple alignment programs often manually clip their sequences into similar blocks and compute a global alignment of each block [59]. In a sense, this manual procedure attempts to overcome the limitations of the alignment program by trimming the sequences to the parts that are related by point mutations and small indels. A more attractive alternative is to change the alignment program to include a larger set of operations that more accurately reflect the changes that occur in biological sequences.

Recently, in a pioneering paper, Lee, Grasso, and Sharlow [32] asked the question "Should multiple sequence alignments be linear?" In answer to this question, they proposed Partial Order Alignment (POA), an algorithm that replaces the row-column representation of a multiple alignment by a directed acyclic graph (DAG). Figure 1 illustrates the intuition behind the POA approach. An alignment is a mapping from a set of sequences to a graph. In row-column alignment the graph is always a single directed path, while the POA approach expands the allowable graph structures to include directed acyclic graphs. The POA approach opens a new perspective on the multiple alignment problem by removing the rigid structure of the linear row-column representation that has been the basis for multiple alignment research over the three decades. POA permits domain recombinations making it a useful tool for the alignment of multi-domain proteins and ESTs.

However, even the directed acyclic graph representation employed in POA is not flexible enough to capture the full complexity of the similarities between biological sequences. For example, related protein sequences frequently share common domains, but the order of the domains may be different in different proteins

(shuffled domains), or a single domain may be repeated in a single protein (repeated domains). To represent shuffled or repeated domains, the alignment representation must permit directed cycles (Figure 3.2). Hence, neither the row-column representation nor the DAG representation provides an accurate representation of shuffled or repeated domains. We take the approach of Lee, Grasso, and Sharlow [32] a step further and ask "Should multiple alignments be represented by acyclic graphs?" In the case of proteins with repeated or shuffled domains, the answer is no.

We emphasize that the real multiple alignment problem is more difficult that the schematic representation in Figure 3.2. For example, when aligning multi-domain protein sequences, the delineation of the sequences into domains is not known in advance, and needs to be derived from raw protein sequences. Neuwald et al. [60] recognized this problem and developed a program that can automatically identify local blocks of significant multiple alignment. However, their program restricts the blocks to be in same order and converges onto a single strongest domain. Different domains may have different lengths in different proteins and pairwise alignments between them are often inconsistent. Resolving these inconsistencies is a major challenge in multiple alignment. In proteins with preserved domain order, local similarities should not "cross" (Figure 3.3). However, alignments of proteins with shuffled domains often contain many such crossing similarities. Since a row-column or partial order alignment does not permit crossing similarities, in building such an alignment one must decide which of the crossing similarities to represent in the alignment. Once we allow cycles in our alignment representation, crossing local similarities are permissible, and distinguishing the crossing similarities that indicate domains from

"spurious" similarities becomes much more difficult. Therefore, the problem of dealing with crossing local similarities calls for development of a new MSA approach that adequately reflects the varieties of domain architectures in different proteins.

In this chapter, we describe a new representation for a multiple alignment as a weighted directed graph (possibly containing cycles) called the A-Bruijn graph. The A-Bruijn graph was recently introduced and applied to fragment assembly and *de novo* repeat identification [41]. Our work is the first application of the A-Bruijn graph to multiple sequence alignment. The A-Bruijn graph is an extension of the classical de Bruijn graph that has been successfully applied to many bioinformatics problems [34-40, 61]. Zhang and Waterman [62] pioneered the use of the de Bruijn graph approach for global multiple alignment of DNA sequences. However, the question of how to generalize their approach for highly diverged DNA or protein sequences remained open. In this chapter, we show how the notion of A-Bruijn graph addresses this problem. We describe A-Bruijn Aligner (ABA), a program to produce an alignment representation from the A-Bruijn graph. We apply ABA to multi-domain protein sequences and genomic sequences with repeated and shuffled elements. The alignment representation produced by ABA is similar to the threaded blocksets recently introduced by Blanchette et al. [63] to represent the complex multiple alignments of large genomic sequences. We demonstrate that ABA provides a solution to the open problem posed by Blanchette et al. [63] of how to automatically generate threaded blocksets. The ABA software is available at http://nbcr.sdsc.edu/euler/.

## 3.2 Methods

The MSA problem involves two tasks: finding a graph that represents the domain structure and finding a mapping of each sequence to this graph. Our approach constructs this graph based on a predetermined set of local similarities (e.g. pairwise alignments) between the input sequences. We describe our methods in the framework of A-Bruijn graph introduced in previous chapter.

While Figure 3.2f is illustrative of the alignment representation that we wish to obtain, it is not immediately clear how to obtain such a representation. The major challenge is the determination of the regions of similarity that should be "glued together" in the graph to represent the protein domains (boxes in Figure 3.2). One cannot use a stringent criterion for similarity, such as exact *l*-tuple matches, because relatively few, if any, exact matches are present in distantly related sequences. Therefore, the traditional de Bruijn graph approach that is based on perfect *l*-tuple matches does not work for this application. With a less stringent criteria (e.g. local alignments), local similarities will frequently be inconsistent, and one must decide which local similarities to respect in the multiple alignment, a nontrivial task. Morgenstern, Dress, and Werner [42] give a mathematical condition for the consistency of a set of local similarities among sequences. A number of heuristics have been proposed for selecting sets of consistent local similarities and building alignments from these sets [48, 64]. The problem is compounded by our desire to permit directed cycles that result from crossing alignments that indicate domain structures. In the ABA approach, we distinguish crossing alignments from local

inconsistencies by using graph heuristics that remove the short cycles resulting from local inconsistencies while retaining longer cycles that result from multi-domain organization.

Following the A-Bruijn graph approach, ABA represents an alignment of $t$ sequences, $S^1$, $S^2$, . . . , $S^t$, as a directed graph (possibly containing cycles) with $t$ source and $t$ sink vertices. Each sequence $S^i$ corresponds to a directed path in the graph from the $i$th source to the $i$th sink. Aligned regions from different sequences or repeated regions in a single sequence correspond to high multiplicity edges in the ABA graph. This latter feature - aligning regions in the same sequence - is not found in existing approaches to multiple alignment, and is similar to the use of the A-Bruijn graph in repeat analysis [41]. Thus, our representation reveals repeated and shuffled regions in the input sequences, features that are not apparent in a row-column or partial order alignment. The input to ABA is a set of $t$ sequences and their pairwise alignments. We first construct the A-Bruijn graph of the alignments by "gluing" together the aligned positions in the sequences $S^1$, $S^2$, . . . , $S^t$, and then apply the bulge and whirl removal procedure as described in [41]. After removing bulges and whirls, the resulting graph may still contain many short edges, due to ambiguities at the ends of aligned regions. These edges add unnecessary complexity to the A-Bruijn graph. To reveal aligned regions, we are interested in *important edges*: edges of high multiplicity and edges with length greater than some threshold. Therefore, we apply a 2 step rethreading procedure: (i) remove the unimportant edges; (ii) thread each sequence $S^i$ through the remaining important edges.

Finally, for visual display purposes, we apply a short edge removal heuristic that simply collapses any connected component of short edges in the graph into a single *super-vertex*, represented by boxes in the figures. We use the Graphviz package [65] to draw the resulting ABA graph. As an illustration, the construction of the ABA graph in Figure 8 is shown in Appendix Figure A.4.

For short sequences (e.g. protein sequences) the running time of ABA is negligible compared to the time taken in computing all local pairwise alignments that form the input to ABA. For longer sequences (e.g. megabase-sized genomic sequence), the major constraint is memory. The human-mouse-rat sequences considered below required two hours of processing time and three gigabytes of memory on an Alpha ES40 workstation. Improvements in memory usage will be necessary for scaling ABA to larger genomic regions. We are currently implementing a version of ABA with reduced memory requirements.

## 3.3 Result I: ABA for protein sequences

### 3.3.1 Case Study: Proteins with SH2, SH3, and Pkinase Domains

The ABA graph can represent alignments of proteins with shuffled domains. As an illustration, we first examine the pairwise alignment of two proteins: SHK1 protein in Dictyostelium (SwissProt id: Q9BI25) and the ABL1 protein in human (SwissProt id: ABL1 HUMAN). Both proteins function as kinases in signal transduction pathways and contain a protein Kinase domain and SH2 domain, but in different order (Figure 3.4a).

The ABA graph reveals the shared domains as edges of multiplicity two (Figure 3.4b). Furthermore, the ABA graph reflects the shuffled domain structure as a cycle consisting of two edges of multiplicity two – corresponding to the shared domains – and two edges of multiplicity one – corresponding to the unique interdomain regions in each sequence. This cyclic structure cannot be represented as a row-column alignment or as a POA graph.

As a second example, we present an alignment of four human proteins: MATK, ABL1, GRB2, and CRKL. Lee, Grasso, and Sharlow [32] use this example to illustrate the ability of the POA graph to reveal domain structures and to demonstrate the advantage of the partial order representation over a row-column representation. In their representation (Figure 3.5a), the alignment of the SH2 domains present in all four sequences is shown as an edge in the center of the graph. However, POA does not align the five SH3 domains present in these sequences. In fact, the acyclic property of the POA graph prohibits an alignment with the five SH3 domains aligned and the four SH2 domains aligned. The alignment of the four SH2 domains by POA forces the five SH3 domains into two alignments: one preceding the aligned SH2 domains, and one succeeding the aligned SH2 domains. This rigidity is not present in the ABA graph (Figure 3.5b,c). The SH3 domains on both sides of the SH2 domains align in a single unit. As a result, the edges corresponding to the SH2 domain and SH3 domain form a cycle in the ABA graph.

To obtain the ABA graph, we identify pairwise local alignments between the four protein sequences using the BLAST program with BLOSUM80 matrix. Hits with minimal length of 40 and at least 40% conserved (as defined by BLAST) are input to

the ABA algorithm. The resulting ABA graph (Figure 3.5c) clearly shows the domain structures as edges with high multiplicity. In the ABA graph, the edge (*2 –> 1*) of multiplicity four corresponds to the SH2 domain shared by all four sequences. The edge (*1 –> 2*) of multiplicity five corresponds to the five SH3 domains in four sequences. Notably, the two SH3 domains in GRB2 are glued together on this edge. As a result, the path through the graph corresponding to the GRB2 sequence contains a cycle signifying duplication of the SH3 domain. Also note that there is a second SH3 domain at the C-terminal end of CRKL that is not glued by ABA to the other SH3 domains. The reason for the isolation of this SH3 domain is that it is sufficiently diverged from the other SH3 domains so that there are no significant pairwise local alignments (satisfying our criteria above) between this SH3 domain and the other sequences detected by BLAST.

### 3.3.2 Case Study: Proteins with GAF, Response Reg, GGDEF and EAL Domains

Since ABA has the ability to align proteins with shuffled domains, we wanted to explore the prevalence of domain shuffling in proteins from SwissProt [66], based on the SwissPfam domains annotation [18]. The *domain shuffling network* (Figure 3.6) summarizes our findings. Vertices in the domain shuffling network are Pfam domains, and a pair of vertices are joined by an edge if they appear in different orders in some proteins in the SwissProt database, i.e. they are shuffled. The domain shuffling graph is similar to the *domain network* [26] or the *domain graph* [27], in which domains

(vertices) are linked if they appear in the same protein. Clearly, the domain shuffling network is a subgraph of the domain graph.

To construct the domain shuffling network, we select a subset of Pfam domains (7316 domains, as at Feb 18, 2004) according to the following criteria: (i) longer than 50 aa; (ii) more than 21% conserved; and (iii) contained in at least 500 proteins. A total of 119 domains satisfy these criteria, and 47 of these appear in different orders in the Pfam annotation of some SwissProt proteins. There are a total of 56 edges representing shuffles between these 47 domains. The network has 10 connected components. The largest connected component of the domain shuffling network (Figure 3.6) prominently displays the protein kinase domain (Pkinase) as the highest degree vertex. This reflects the fact that domain shuffling is a common feature in the kinase family. However, the domain shuffling network reveals that shuffling of domains is not restricted to kinases. We now describe an example of domain shuffling and duplication outside the protein kinase family.

We analyze four proteins Q82U13, ETR1 ARATH, PHY2 SYNY3, and Q7MD98 from SwissProt, each containing some but not all of the Pfam domains GAF, Response reg, GGDEF and EAL (Figure 3.7). The BLAST program with the BLOSUM80 matrix gives eight significant pairwise local alignments between these sequences satisfying the constraints that alignment length is longer than 40 aa and is more than 40% conserved. We input these alignments into the ABA program, and obtain the graph shown in Figure 3.7.

Edges of high multiplicity (or a chain of high multiplicity edges) in the ABA graph corresponds to domains shared by the sequences. Table 2.1 shows four edges

(chain of edges), each representing a significant local multiple alignment. We emphasize that the correspondence between these edges and known protein domains is approximate, since the edges result directly from significant alignments from BLAST. We can extract the subsequences corresponding to high multiplicity edges, and refine the multiple alignment using an existing tool like CLUSTALW. We remark that ABA eliminates a time-consuming, manual clipping procedure.

Domain shuffling creates directed cycles in the ABA graph. In this example there are two domain shuffles: Response reg *vs.* GAF, and EAL *vs.* GAF/GGDEF represented by two directed cycles ( *2* –> *0* –> *1* –> *2* and *2* –> *3* –> *4* –> *5* –> *7* –> *2*) in Figure 3.7. The different domain orders in individual sequences are reflected by the different paths traversing the ABA graph that visit edges in different orders.

When we align these four sequences using POA [32], we observe that the DAG representation used by POA cannot adequately represent the shuffled domain structure (Appendix Figure A.1a,b). Among the four significant local alignments listed in Table 1, POA correctly identifies the first one: an alignment between the GGDEF-EAL domains in the three sequences. However, depending on the order that the sequences were input into the iterative alignment procedure, POA detects either alignment No. 2 or No. 3 in Table 1, but not both. Alignment No. 4 (self-alignment) is always missing. We emphasize that the ABA graph (Appendix Figure A.1c), in contrast to the POA graph, is independent of the order in which the sequences are considered.

### 3.3.3 Case Study: Proteins with Condensation, AMP-binding and PP-binding Domains

We use ABA to align 16 protein sequences from SwissProt containing the condensation domain. Appendix Table A.1 gives the SwissProt ID for each of the 16 proteins. The condensation domain is 249 aa long and is found in multi-domain enzymes that synthesize peptide antibiotics. Many of these proteins also contain an AMP-binding domain, a 330 aa long domain that covalently binds AMP to their substrates in an ATP-dependent manner, and a PP-binding domain, a short domain (65 aa) that serves as a "swinging arm" for the attachment of activated fatty acid and amino-acid groups.

We obtain the ABA graph shown in Figure 3.8 using the same BLAST parameters as the previous case studies. Most of the long edges in the graph correspond to the long domains: condensation and AMP-binding. These domains are typically not well conserved over their full length, and ABA reveals the well conserved parts as high multiplicity edges (e.g. A –> B and C –> D) and splits the less conserved parts into multiple edges, e.g. B –> C, E –> F, and G –> A.

## 3.4 Result II: ABA for genomic sequences

ABA is also applicable to the alignment of genomic sequences, and the ABA graph directly reveals duplications and inversions that are often found in alignments of long mammalian genomic sequences. The input to ABA is a set of $t$ DNA sequences (with the $t$ reverse complements), and the pairwise local alignments between the $2t$ sequences. The resulting ABA graph is a collection of $2t$ paths – corresponding to the $t$

input sequences and their *t* reverse complements – that are 15 glued together according to the local alignments. A duplication in a single sequence corresponds to a directed cycle in the path corresponding to this sequence, while an inversion corresponds to a gluing of the direct strand of one sequence to the reverse strand of another sequence.

We apply ABA to a pair of plant chloroplast genomes, *Arabidopsis thaliana* and *Oenothera elata*, and produce the graph in Figure 3.9a. We compare our results to the alignment obtained by the Threaded Blockset Aligner (TBA) of Blanchette et al. [63] (Figure 3.9b). TBA represents a multiple alignment as a set of alignment blocks (a blockset) that is ordered according to one of the input sequences; i.e. one "threads" one of the input sequences through the set of blocks. Blocksets in TBA are analogous to long edges in the ABA graph. We observe a striking correspondence between long edges in the ABA graph of the chloroplast genomes (Figure 3.9a) and the blocks obtained by Blanchette et al. [63] (Figure 3.9b). A single block (block 3) is missing from the ABA graph, which probably could be rescued by a more sensitive parameter setting when computing the pairwise BLASTZ alignments that are input to ABA. Thus, in this example ABA automatically generates threaded blocks as long edges of high multiplicity in the ABA graph.

We note that the current implementation of TBA produces a limited type of threaded blockset, namely TBA "does not accommodate inversions[3] and duplications, and it is restricted to finding matches that occur in the same order and orientation in the given sequences". ABA has no such restrictions. Indeed, in the ABA graph of the

[3] TBA now can handle reverse-strand matches and inversions (W. Miller, *pers. comm.*).

chloroplast genomes, block 7 appears twice along the path corresponding to the *Arabidopsis* genome, once in the direct strand and once in the reverse strand.

To further our comparison of the blocks extracted from the long edges of the ABA graph with the blocks produced by TBA, we examined the human, mouse, and rat sequences from the NISC target region T1. The complete set of sequences from 12 species was first analyzed in Thomas et al. [67]. The ABA graph (Appendix Figure A.3) contains 13536 edges (for both strands of the three genomic sequences), while TBA generates 4445 blocks. When projected on the direct strand of human genome, the ABA graph (Appendix Figure A.2) has 3726 multiple edges (i.e. edges of multiplicity larger than one) and TBA has 1624 multiple blocks (i.e. blocks containing more than one sequence). We display the ABA and TBA blocks in the UCSC genome browser [68] as custom tracks[4] for a visual comparison. A region surrounding the CAV2 gene along human genome is shown in Figure 3.10.

ABA and TBA use different algorithmic approaches to blockset generation (discussed below), yet most of the blocks produced by TBA and ABA have significant overlaps. However, we observe three differences. First, ABA generates blocks of multiplicity higher than 3 (dark grey blocks in Figure 3.10), demonstrating the ability of ABA to handle duplications and inversions. Second, TBA detects a few blocks that are missed by ABA: these blocks represent short 3-way alignments. ABA misses these short alignments because it uses only pairwise alignments while TBA implements a progressive multiple alignment engine (MULTIZ). Third, ABA generally produces longer blocks (or concatenations of blocks).

[4] http://www.cse.ucsd.edu/groups/Bioinformatics/browser-tba-aba-human.bed

The above results demonstrate that: (i) ABA is able to automatically generate threaded blocksets for genomic sequence alignment, and (ii) ABA handles duplications and matches of sequences that are in different orders in different genomes. A more detailed comparison of the two approaches and the possibility of synergistic combinations of both approaches are important questions for future study.

## 3.5 Discussion and Future Directions

The important feature of ABA is the ability to produce multiple alignments of sequences that include shuffled and repeated regions, a feature lacking in other alignment methods. We now compare ABA with other approaches to multiple sequence alignment, and describe further applications and extensions of ABA.

### 3.5.1 Alignment Representation

ABA represents a multiple alignment as a directed graph, possibly containing cycles. This is in contrast to most existing alignment programs that use a linear, row-column representation. Recently, Lee, Grasso, and Sharlow [32] introduced Partial Order Alignment (POA) that uses a variation of *network alignment* (first presented in [69] and analyzed in [70]) to align a sequence to a directed acyclic graph representation of an alignment. The method is order dependent, as each sequence is aligned to the graph in turn. In a later paper, Grasso and Lee [32] generalize the method to include alignment of two partial order graphs and thus implement a progressive alignment. However, their partial order graph is not able to represent shuffled or repeated domains.

Lee, Grasso, and Sharlow [32] comment that their partial order representation "expresses a more complex set of relationships than can easily be discovered by phylogenetic tree", and accordingly introduce a new edit operator: domain recombination. In a similar fashion, ABA implements two other operations: *domain rearrangement* (change in order of two domains in a single sequence) and *domain duplication* (repetition of a domain in a single sequence). Both domain rearrangement and domain duplication are common in protein sequences. Domain rearrangement is similar to "string edit distance with moves" [71] or block edit distance [72] studied in string matching. However, to our knowledge, ABA is the first multiple alignment program that implements the domain rearrangement operation.

Zhang and Waterman [62] were the first to propose a multiple alignment method based on the de Bruijn graph approach for DNA sequences. Following the Eulerian method for fragment assembly in DNA sequencing [35, 36], their EulerAlign algorithm starts with the de Bruijn graph of *k*-mers contained in the set of sequences to be aligned. Their algorithm transforms the de Bruijn graph into a DAG, and then aligns all sequences to a consensus represented by a high weight path through the DAG. Thus, their method aligns all sequences to a single consensus, and removes all cycles present in the de Bruijn graph.

The Zhang and Waterman [62] algorithm presents a powerful new technique for alignment of similar DNA sequences that eliminates the time-consuming task of performing pairwise alignments. Thus, their method is suitable for aligning a large number of DNA sequences. However, the question of how to generalize their method to align highly diverged DNA sequences (e.g. DNA sequences that are less than 70-

80% similar) remains open. Furthermore, while the similarity between DNA sequences can often be captured by shared *k*-mers, protein sequences typically share very few *k*-mers and the similarity between protein sequences often requires non-trivial scoring matrices. Furthermore, shared *k*-mers are very sensitive to indels. Our A-Bruijn graph approach bypasses these limitations by abandoning the *k*-mer analysis.

Recently, Blanchette et al. [63] introduced the Threaded Blockset Aligner (TBA) for multiple alignment of megabase-sized regions of genomic sequences. The development of TBA and ABA share a common philosophy: overcoming the limitations of the row-column representation of a multiple alignment. The blocksets employed by TBA are analogous to the high multiplicity edges in the ABA graph, and the threading procedure in TBA to create "ref-blocksets" is analogous to following the path in the ABA graph from source *i* to sink *i*. However, the current implementation of TBA – similar to Partial Order Alignment – handles only alignments of blocks that occur in the same order and orientation in the sequences. They leave open the problem of "automatically, accurately, and reliably" identifying blocksets in genomic sequences that resulted from inversions, duplications, and other complex rearrangements. We demonstrate that ABA solves this problem for protein and genomic sequences, and it is possible to use ABA to automatically generate blocksets for TBA.

The algorithmic approach of TBA is very different from ABA. TBA progressively aligns input sequences along a phylogenetic tree from leaves to the root. The blocks in the blockset at a parent node result from intersections or exclusive-ORs of the blocks at its children. The blocks are only split into smaller blocks during the

progressive steps of TBA. There is no mechanism to merge blocks – TBA follows the maxim: “once a block boundary, always a block boundary”. In contrast, ABA permits the merging and simplification of very small blocks.

Every blockset represented by TBA is a somewhat simplified linear view of a multiple alignment. In reality, some alignments within a blockset may extend over several blocks while other alignments may be significantly shorter. In a sense, the individual blocks in a blockset have the same limitations as the row-column alignment, in comparison to a DAG alignment that was discussed in the introduction. ABA has a more flexible approach to defining the block boundaries that are expressed as “tangles” in the ABA graph.

### 3.5.2 Applications and Extensions

ABA integrates well with existing multiple alignment tools, and can serve as a preprocessor for these multiple alignment programs. For example, given a set of sequences with complex domain structure, we can first run ABA to uncover this structure, and then apply an existing multiple alignment program like CLUSTALW to refine the alignments given by high multiplicity edges in the ABA graph. In this scenario, ABA automates the time-consuming clipping of sequences frequently recommended for multiple alignment tools and performs this clipping in a rigorous way.

The ability of the ABA graph to succinctly represent proteins with shuffled and repeated domains makes it useful for *de novo* domain finding and studies of domain structure. Galperin and Koonin [73] highlighted how the multi-domain organization of

proteins can trigger mistakes in functional annotation, and thus ABA graphs may be useful in this context. Since some protein domains cannot be determined solely from pairwise similarity, alternative similarity measures will be necessary for these applications. ABA can utilize different measures of similarity in the construction of the *A*-Bruijn graph. In this chapter, we focused on similarities given by pairwise sequence alignments, but we can also use *k*-way similarities, or similarity measures given by profiles (e.g. PSI-BLAST), structural comparisons, Hidden Markov Models, etc. In particular, we can use reverse position specific BLAST (rpsBLAST) with profiles found in domain libraries such as the Conserved Domain Database [21]. Use of rpsBLAST will reveal alignments corresponding to known domains. Other high multiplicity edges in the ABA graph might suggest novel domains.

Further refinements in the ABA algorithm will be required to extend its application. One possible improvement to ABA is the implementation of an iterative refinement procedure: after we construct the initial ABA graph using pairwise similarities, we identify the important edges and refine the alignment at each important edge using more accurate alignment procedures. Finally, we rethread individual sequences through the important edges; if the topology of the graph changes, the procedure is repeated.

The text in this chapter, in part or in full, is a reprint of material as it appears in Genome Research. The dissertation author was the secondary author of the paper. I thank my co-authors Ben Raphael, Haixu Tang, and Pavel Pevzner.

## 3.6 Tables and figures

**Table 3.1: Four high multiplicity edges in the ABA graph. [a]Average pairwise percent of conserved amino acids. [b]Number of occurrences of domain or domain combinations in the proteins. [c]The alignment corresponding to this domain extends into block 2 in the graph (data not shown), and thus the alignment is longer than the length of the edge *7 –>2* in Figure 3.7b.**

| No. | Edge | Length | Conservation[a] | Domain(s) | Domain occurrence[b] | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | Q7 | P | E | Q8 |
| 1 | *2 –> 3 –> 4 –> 5 –> 6* | 420 | 53% | GGDEF-EAL | 1 | 1 | | 1 |
| 2 | *0 –> 1* | 58 | 51% | Response_reg | | | 1 | 1 |
| 3 | *7 –> 2* | 49[c] | 60% | GAF | | 1 | 1 | |
| 4 | *2 –> 3* | 157 | 43% | GGDEF | | 2 | | |

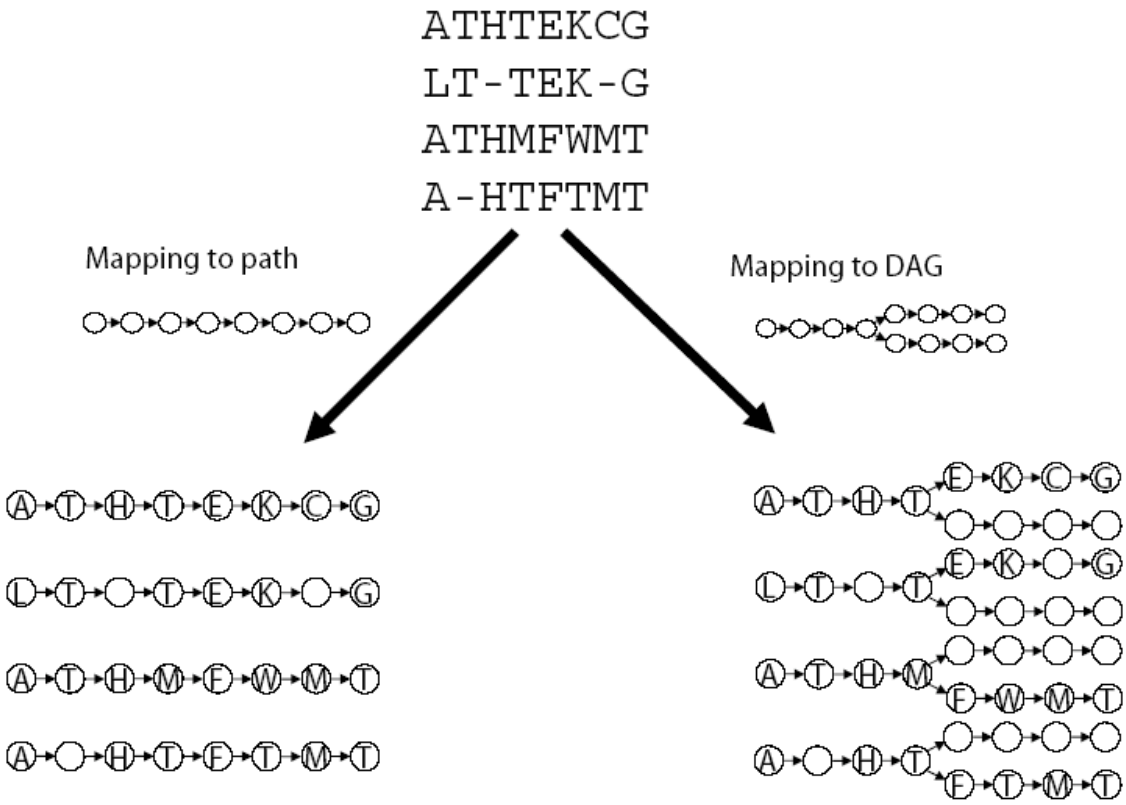



**Figure 3.1: An alignment is a mapping from a set of input sequences to a directed graph. Positions that map to the same vertex are aligned. Standard MSA programs map each sequence to a single path. Each vertex on the path contains either a letter or a gap character from each sequence. POA maps each sequence to a directed acyclic graph (DAG). The structure of the DAG permits alignments where a subset of the sequences is aligned at a position.**

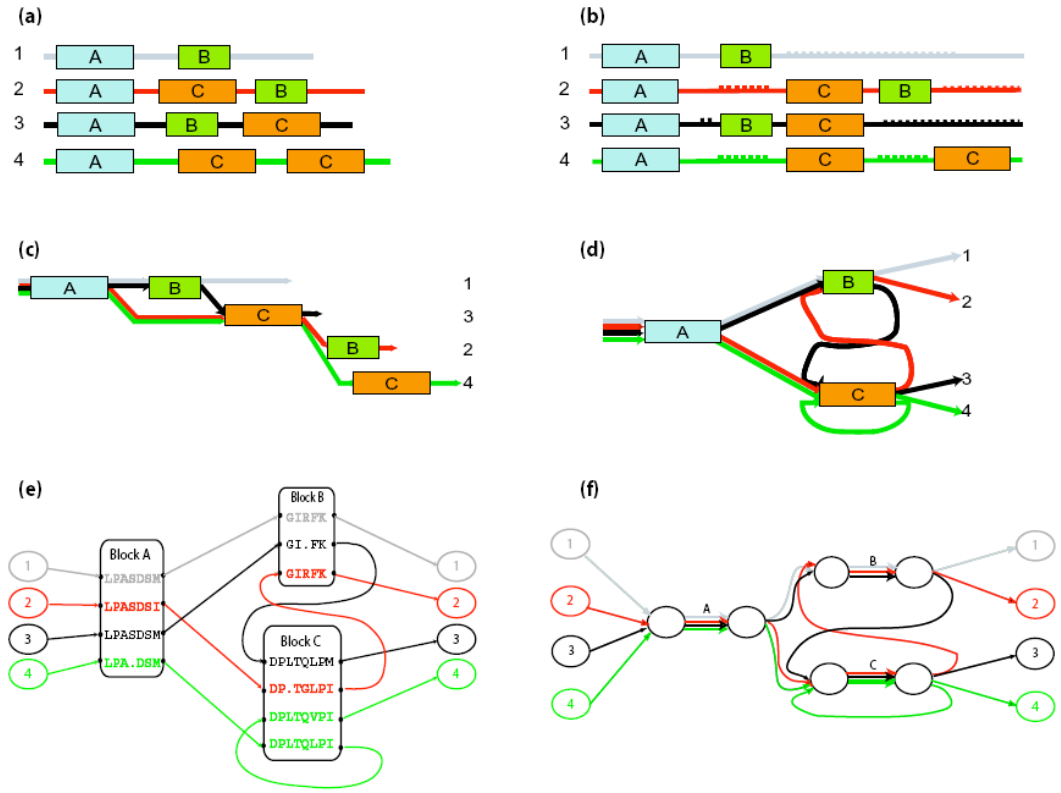



**Figure 3.2: (a) Four "protein" sequences containing three "domains" A, B, C (shown as boxes) and unique regions (shown as lines). (b) A row-column multiple alignment introduces gaps (dotted lines) to align domains A and C, but cannot represent the alignment of all three domains. (c) The Partial Order Alignment (POA) graph improves the alignment in (b) by reducing the number of gaps, but also does not align all copies of the domains. (d) A representation of the domain structure as a graph with cycles. (e) We obtain a representation of the multiple alignment of the four sequences by "gluing" together similar regions in the sequences. However, the sequences do not align over their entire length, and the shuffled domains create cycles in the resulting graph. (f) A simplified representation of the ABA graph shows the domains as edges of high multiplicity, and the unaligned regions as edges of multiplicity one.**

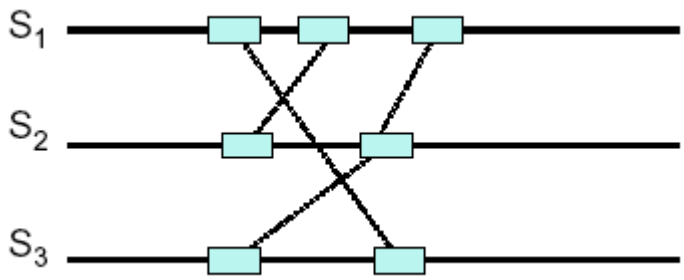



**Figure 3.3: With a row-column or partial order representation, any local similarities that "cross" are inconsistent. In our representation, these local similarities are permissible and lead to cycles in the alignment.**

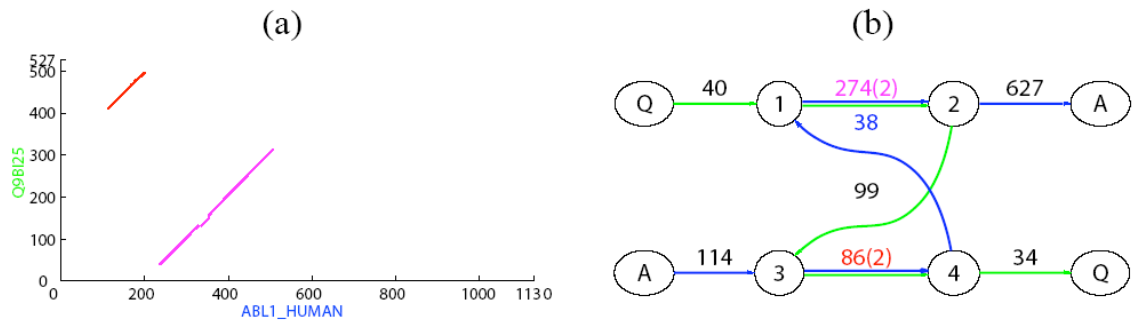



**Figure 3.4: (a) Dot matrix representation of similarities between Q9BI25 and ABL1 HUMAN protein sequences as revealed by BLAST [74]. The two diagonals of length 274 and 86 represent two domains: Pkinase (gray) and SH2 (black). (b) The corresponding ABA graph. Each multiple edge has a label of the form *l*(*m*) where *l* is the length of the sequences represented by that edge, and *m* is the multiplicity of the edge. Each single edge is labeled simply as *l* (length) for brevity. Source/sink vertices are labeled A and Q for protein sequences ABL1 HUMAN and Q9BI25, respectively. Other vertices are numbered. The gray path through the graph corresponds to Q9BI25 and the black path through the graph corresponds to ABL1 HUMAN. The Pkinase domain corresponds to the edge (1–>2) of length 274, and the SH2 domain corresponds the edge (3–> 4) of length 86.**

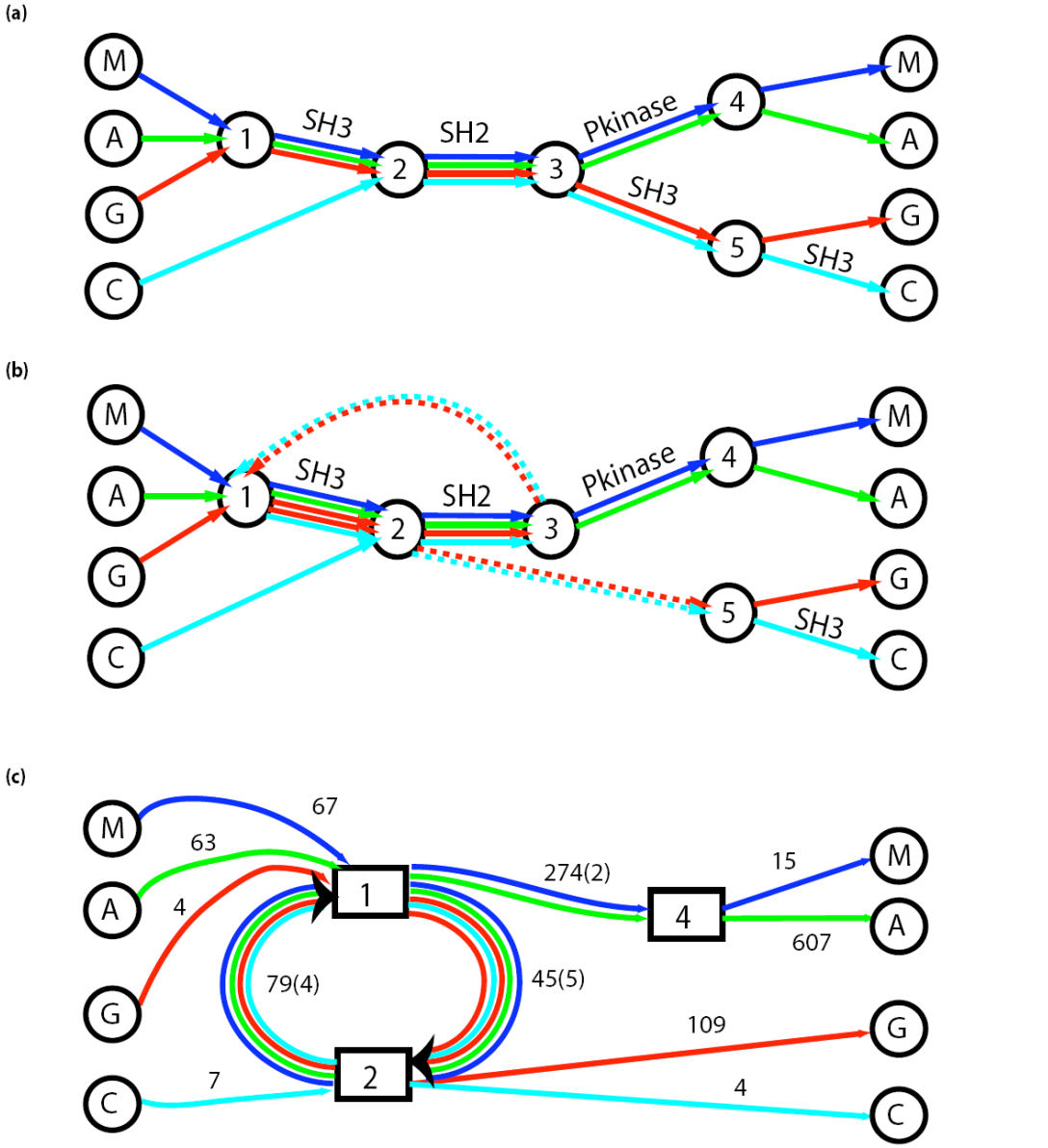



**Figure 3.5: Comparison of POA and ABA representations of the domain structure of four human SH2 domain containing proteins: MATK (M), ABL1 (A), GRB2 (G), and CRKL (C). (a) A simplified representation of the POA graph, as obtained in Lee, Grasso, and Sharlow [32]. Each input sequence forms a path through the graph. Edges with a high multiplicity are labeled with protein domains. (b) A simplified representation of the ABA graph. Dotted edges have length zero and connect nodes that are glued together in the ABA graph. (c) The ABA graph with collapsed multiple edges. Boxed vertices represent small subgraphs that have been contracted (cf. Methods). In this graph, high multiplicity edges correspond to protein domains: SH2, SH3, and Pkinase domains with estimated lengths of 79, 45, and 274 nucleotides, respectively.**

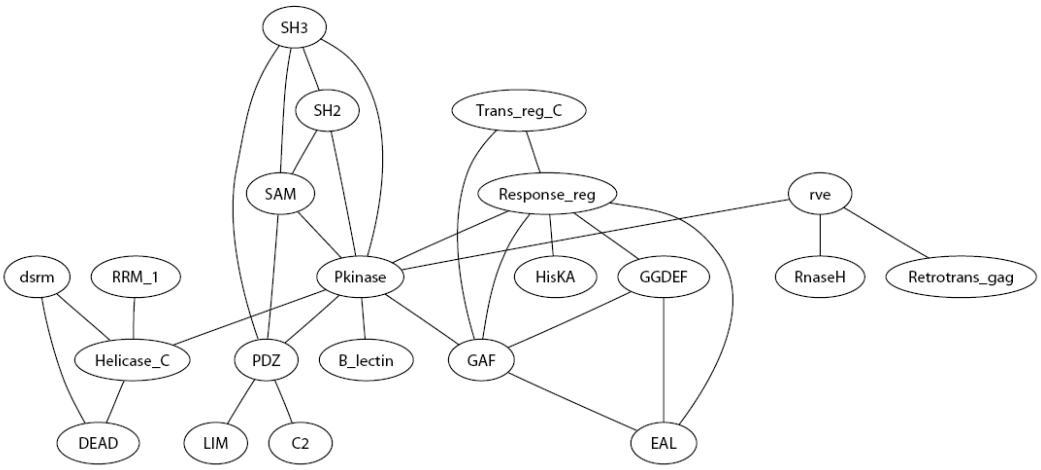



**Figure 3.6: The largest connected component of the Domain Shuffling Network of Pfam domains. Only long, conserved, and common domains are shown. Pfam domains that appear in different orders in proteins from SwissProt are connected by an edge. We omit loops in the network that indicate repeated domains.**

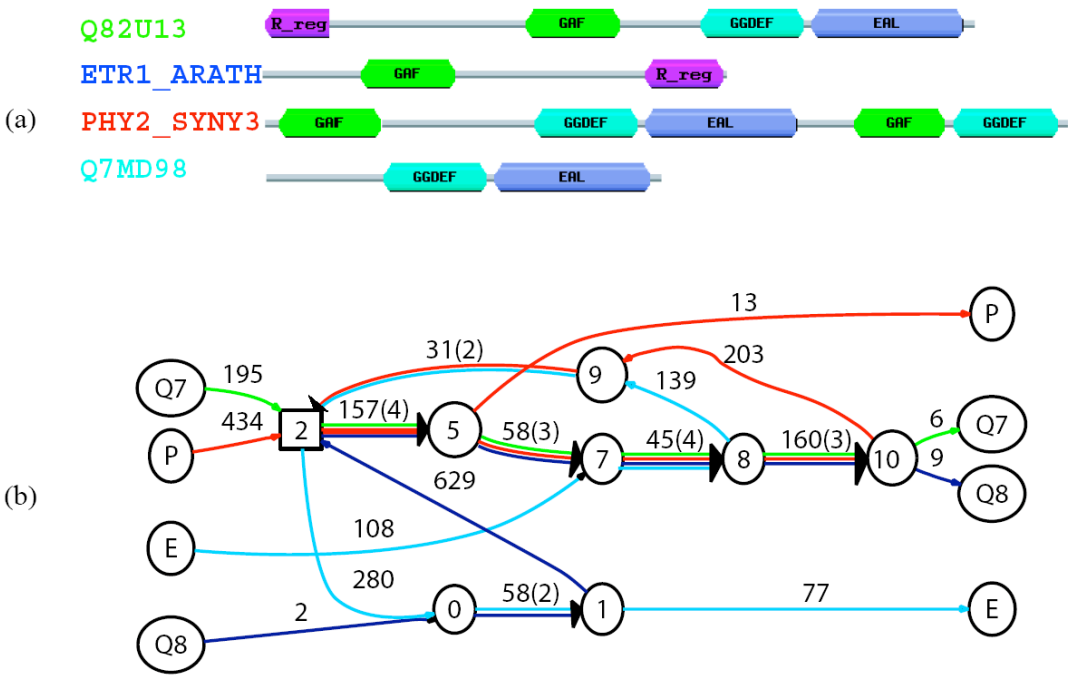



**Figure 3.7: Four proteins with shuffled domains and their ABA graph. (a) The domain structures are derived from the SwissPfam database (Bateman et al. 2004). We show only well-annotated PfamA domains. Domains that appear in only one of the four sequences are not shown. (b) Cycles in the ABA graph reveal the extensive domain shuffling in these sequences.**

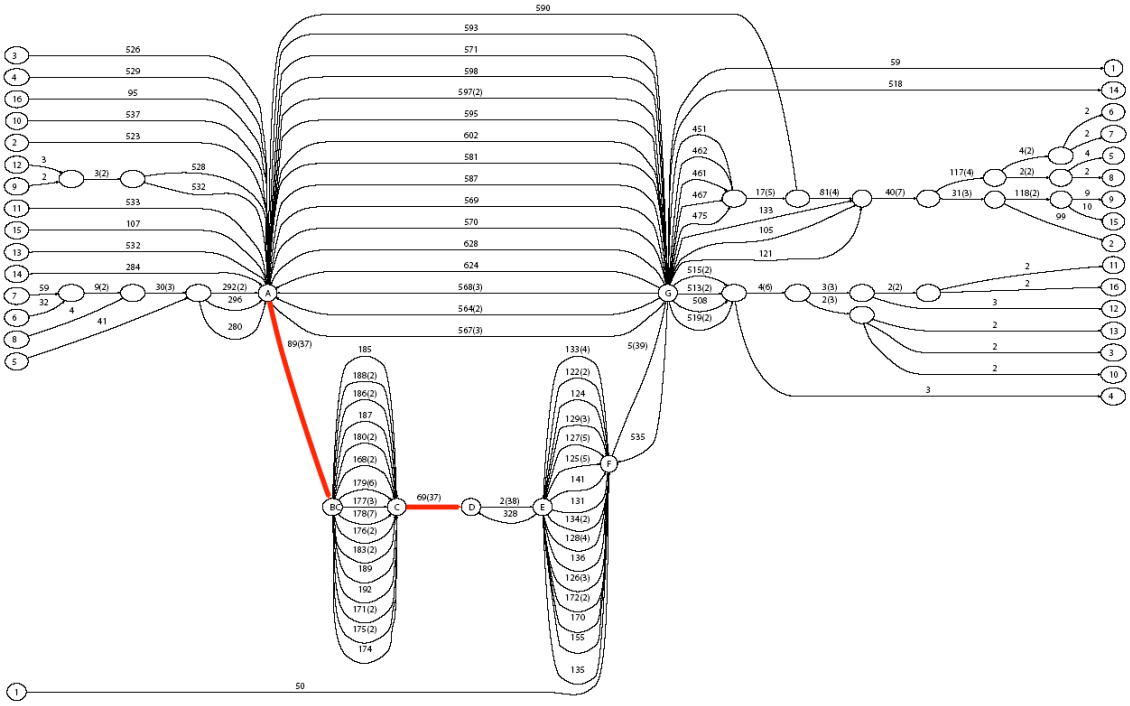



**Figure 3.8: ABA graph of 16 proteins, each containing a condensation domain. Edges *A –> B* and *C –> D* (highlighted) indicate well-conserved parts of the AMP-binding domain. A long directed cycle (*A –> B –> C –> D –> E –> F –> G –> A*) indicates repetition these well-conserved sequences.**

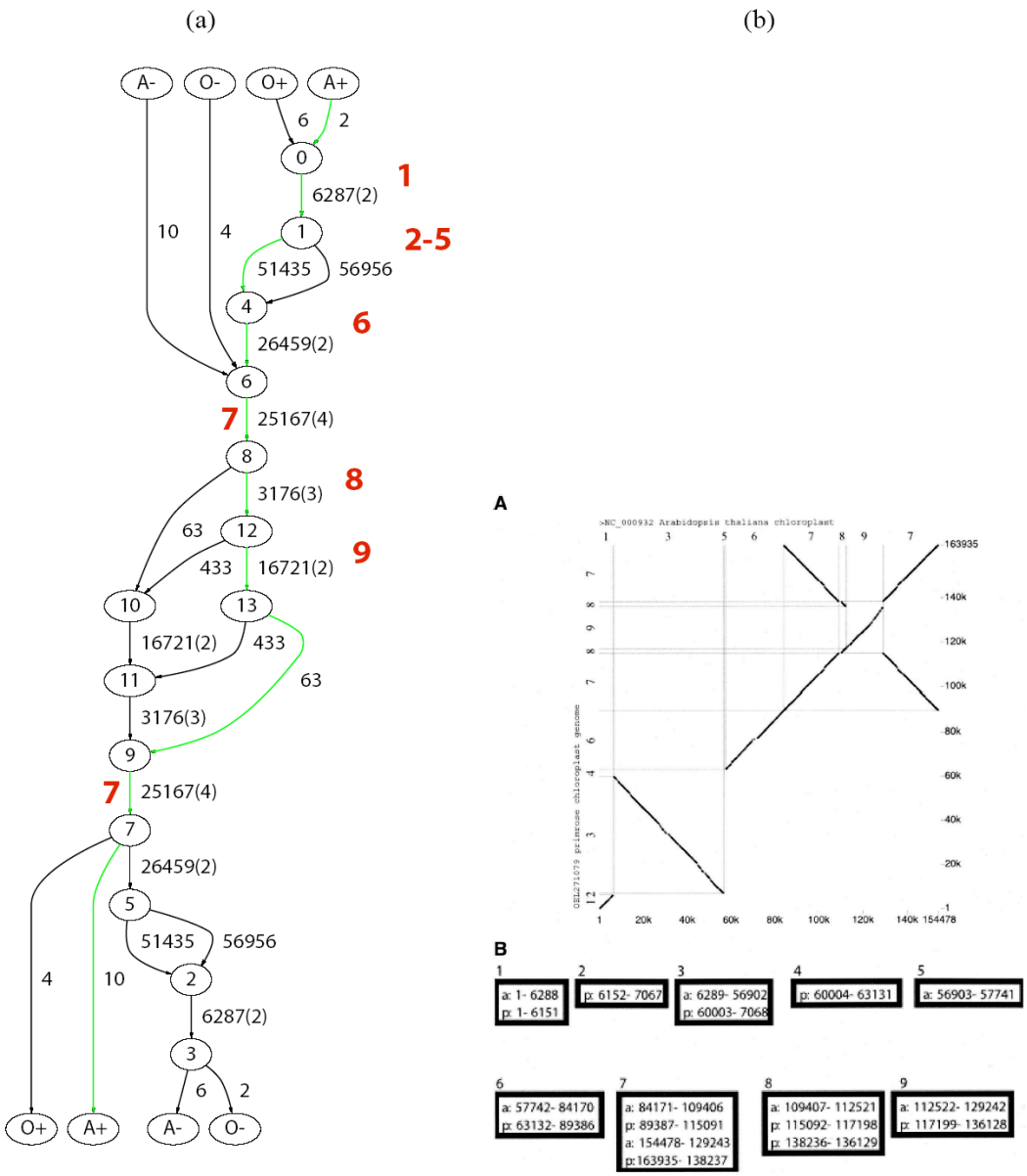



**Figure 3.9: Alignment of genomes of chloroplasts *Arabidopsis thaliana* and *Oenothera elata*. (a) ABA graph. We use BLASTZ [75] with parameter "B=1 C=2" to generate pairwise local alignments. Gray path corresponds to the direct strand of *Arabidopsis* DNA. The number in bold font close to an edge corresponds to the block number in (b) the blockset of Blanchette et al. [63] (Reproduced from Figure 3.2A in [63]). (b) [A] A dot plot of the *Arabidopsis* genome (horizontal axis) and *Oenothera* genome (vertical axis). [B] Nine "alignment blocks"; each block contains sequence segments that labelled by the genome: *Arabidopsis* (a) or *Oenothera* (p), and the coordinates of the segment.**

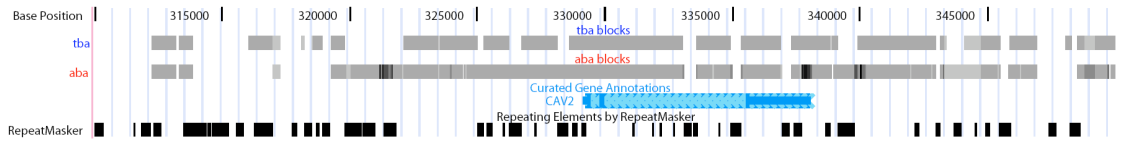



**Figure 3.10: Comparison of blocks produced by TBA and ABA. A region on human genome surrounding the CAV2 gene is displayed on the "zoo genome" (NISC target region T1) of the UCSC genome browser [68]. We use BLASTZ with parameters "B=1 C=2" to include reverse strand matches. Blocks of multiplicity greater than 1 are shown with shades of grey indicating the multiplicity of the blocks. Most of the blocks have multiplicity 3, corresponding to 3-way alignments. Darker grey blocks indicate higher multiplicity, include duplications and inversions (matches on the reverse strands).**

# 4 Repeat domains and composite repeats in repeat libraries

## 4.1 Introduction

Repetitive elements form a major fraction of eukaryotic genomes. Though once dismissed as mere junk DNA, they are now recognized as "drivers of genome evolution" [76] whose evolutionary role can be "symbiotic (rather than parasitic)" [77]. Examples of potentially beneficial evolutionary events in which repetitive elements have been implicated include genome rearrangements [76], gene-rich segmental duplications [78], random drift to new biological function [79, 80] and increased rate of evolution during times of stress [81, 82]. For these and other reasons, the study of repeat elements and their evolution is now emerging as a key area in evolutionary biology.

Individual repeat elements can be grouped into repeat families, each defined by the consensus sequence of its diverged copies. Repeat family libraries, such as Repbase Update libraries [30, 83] and RepeatMasker libraries [84], contain consensus sequences of known repeat families. Repeat families often contain shared subsequences, which we call *repeat domains*. Repeat domains can occur more than once within the same repeat family; for example, the ubiquitous human *Alu* family is dimeric [29]. There are a number of cases of repeat families whose repeat domains are known to have different biological origins, i.e., from repeat families with different modes of replication or from distinct retrovirus families. These repeat families and the domains they share are worthy of special attention, since they are assumed to result from interesting evolutionary events. We define a repeat family to be a *composite*

*repeat* if it contains at least two repeat domains of different biological origin. Of course, discerning the biological origin of a repeat domain is a challenging endeavor. Nevertheless, human Repbase Update documents more than 10 repeat families as composite repeats, including the RICKSHA and Harlequin families. Many other composite repeats contain fragments from different retroviruses. Since composite repeats which contain only fragments of retroviral origin are probably products of retroviral recombinations, these are documented in Repbase Update as retroviral recombinations (see [85] for a review). Composite repeats are likely more than a mere curiosity: one composite repeat, SVA, is the third most active retrotransposon since the human/chimpanzee speciation [86]. An additional example is found in the eel where a composite SINE repeat family borrowed a repeat domain from a different LINE family; this borrowed domain was experimentally shown to greatly enhance the retrotransposition rate of the SINE family [87].

Shared repeat domains yield important insights into repeat evolution, in the same way that multidomain protein organization yields insights into protein evolution [88, 89]. However, while the study of protein domains is a well-established research area, the study of repeat domains is still in its infancy. Indeed, RepeatGluer [41] is the only existing algorithm for repeat domain analysis. While RepeatGluer shows promise as a tool for repeat domain analysis, it is computationally intractable for large genomes. For large genomes, we propose that instead of identifying repeat domains *de novo* from genomic sequence, we identify repeat domains by analyzing repeat family libraries that are obtained via other means.

The main challenge in the analysis of repeat domains is that repeat family consensus sequences typically form a complex mosaic of shared subsequences. This mosaic structure is reminiscent of the mosaic structure of segmental duplications in mammalian genomes (see [63, 90]). Standard sequence comparison tools are unable to capture mosaic structure. These tools reveal local similarities between different repeat families, but do not reveal the structure of shared repeat domains between different families. For example, although a dot plot of the sequences of the 11 *C. elegans* and *C. briggsae* repeat families sharing repeat domains (Figure 4.1) contains essentially all the information about these repeat families, it is not well-organized and leaves one puzzled about what the repeat domains are. Thus, identifying repeat domains is an important and unsolved problem.

In this chapter, we propose a new framework for analyzing a library of repeat families to identify the mosaic structure of its shared repeat domains. Our main idea is to represent a repeat library by a *repeat domain graph* which reveals all repeat domains as edges (lines linking between nodes) of the graph, and indicates the order(s) in which those domains appear in the corresponding repeat famili(es). For example, Figure 4.2 illustrates the domain structure of a selected subset of repeat families sharing repeat domains with the RICKSHA family, and the corresponding repeat domain graph. We describe a method to construct the repeat domain graph from a set of repeat sequences, and we demonstrate methods for analyzing the topology of the repeat domain graph that lead to hypotheses about repeat biology. We apply our method to single-species analyses of human and *C. elegans* repeat family libraries. Our method recovers documented composite repeats in Repbase Update [30, 83] and

suggests a number of additional putative shared repeat domains in human and *C. elegans*. In addition, we use our method to perform a cross-species comparative analysis of *C. elegans* and *C. briggsae* repeat libraries, and we find a putative ancient repeat domain shared between *C. elegans* and *C. briggsae*. We also demonstrate the application of our method in assisting annotation of repeat libraries that are generated *de novo* from genomic sequence. As numerous new genomes are sequenced and repeat family libraries are automatically constructed, the applications of our method will multiply.

## 4.2 Results and Discussion

Finding all repeat domains is a difficult problem, since repeat family consensus sequences can share subsequences with themselves (e.g. *Alu* repeats) or with other repeat families (e.g. the composite repeat families described above). Thus, we argue that the ideal method for comparing repeat families and identifying repeat domains should allow for both self-similarities and shared similarities that appear in different orders in different sequences. Such similarities are difficult to capture in traditional multiple alignments that either attempt to align sequences other their entire length (e.g global alignment) or show small conserved regions of similarity (e.g. local alignment) with no information about the location of these regions in the original sequence. Recently, several software programs including Partial Order Alignment (POA) program [32], Threaded Blockset Aligner (TBA) [63], and the A-Bruijn Aligner (ABA) [91] were developed to address these shortcomings. ABA seems particularly well-suited both to the alignment of repeat family consensus sequences and to the

decomposition of a library of such consensus sequences into repeat domains since ABA was designed for the alignment of sequences with both repeated and shuffled segments. However, we found that ABA could not automatically generate a repeat domain graph from a repeat library because repeat libraries frequently contain a large number of diverged sequences including palindromic sequences. Below we describe how to overcome these difficulties.

In addition, since a repeat library typically contains several hundred to several thousand sequences, and the annotation of repeats is typically incomplete, the analysis of a repeat domain graph is a nontrivial task. Below we show several examples illustrating how particular queries in the repeat domain graph can provide powerful systematic analysis of repeat families in a repeat library, how topology of the repeat domain graph can help elucidating evolutionary history, and how to deal with contaminants which are common in *de novo* generated repeat libraries.

### 4.2.1 Applying the A-Bruijn graph to repeat library analysis: methodology and new algorithms

We represent an alignment of sequences in a repeat library as a directed graph called the *repeat domain graph*. The repeat domain graph of $n$ sequences contains $2n$ source vertices and $2n$ sink vertices. A directed path in the graph from a source to sink vertex represents a sequence or the reverse complement of a sequence in the repeat library. The repeat domain graph typically contains several connected components. Each component corresponds to groups of repeat families with shared repeat domains, and can be analyzed individually. Edges in the repeat domain graph with multiplicity

greater than one represent repeat domains which are shared between different repeat families, while single-multiplicity edges correspond to domains unique to a single family.

We construct repeat domain graphs using the framework of A-Bruijn graphs, which were first introduced and applied to the problems of DNA fragment assembly and *de novo* repeat classification in [41], and later extended to the alignment of protein sequences and genomic DNA sequences [91]. The A-Bruijn graph is a general framework for handling sequences with repeated or shuffled domains and is constructed from a set of sequences and a set of pairwise alignments between these sequences. In practice, the A-Bruijn graph of a set of pairwise alignments often contains numerous short cycles, due to inconsistencies among the input alignments. These short cycles obfuscate the identification of the shared domains among these sequences and thus a series of graph heuristics is used for removing short cycles due to inconsistent alignments while retaining longer cycles due to shared domains. We discovered that these approaches were not sufficient to handle two complications that arise in repeat library analysis: namely, the need to align a large number of diverged sequences and the existence of palindromic sequences. The shortcomings of the method were not anticipated or addressed in earlier work because these issues did not arise in the problems addressed there: namely fragment assembly [41], where the input is a large number of very similar (greater than 95%) DNA sequences (reads), and the problems of multiple sequence alignment of a relatively small number of protein sequences or genomic DNA sequences [91]. We developed new algorithms for the construction of the repeat domain graph that are modifications of the methods used to

construct and simplify the A-Bruijn graph. These algorithms are described in the Materials and Methods section.

### 4.2.2 Analysis of repeat domains in human Repbase

We first build a repeat domain graph of the Repbase library [30, 83] of human repeat sequences – the most well annotated repeat library available – in order to test the ability of our method to reveal shared repeat domains and the structure of composite repeats. The resulting repeat domain graph of the 620 sequences in Repbase update contains 9774 edges and has a complicated topology with 410 connected components, 168 of them containing shared repeat domains. (See [92] for the entire repeat domain graph, and a list of repeat families contained in each connected component). The largest connected component contains sequences in the library corresponding to the L1 retrotransposon, including the consensus sequences of different families, subfamilies, and partial copies of L1 present in Repbase. Many repeat domains identified in the graph are domains shared by such derivative sequences of a single repeat type. However, other repeat domains are shared sequences between repeat elements of different biological origin. We found 624 such domains by choosing edges in the graph that have minimal length 20, multiplicity greater than one and contain sequences whose Repbase annotations suggest different biological origin of the sequences. As there is no ontology of repeat families, we identify "different biological origin" with a very loose definition: by the first two characters of the repeat family name. We also identify for each repeat family, the length of shared domains and the fraction of its total length containing repeat domains

of different biological origin. Table 4.1 lists the repeat families with the greatest length of such shared domains. (Appendix B.2 contains the full list). Near the top of the list are Harlequin and PABL_AI, two repeat families documented in Repbase Update as products of retroviral recombinations. In addition, there are a large number of repeat families with prefix MER- and HER-, consistent with the observation that retroviral recombination is a dominant feature among repeat families in large mammalian genomes [85]. We remark that the 624 repeat domains shared across repeat families is much higher than what is documented in Repbase, suggesting that composite repeats are a rather common phenomenon. However, this conclusion is tempered by the simple criterion that we used to determine biological origin.

In addition to repeat domains, the repeat domain graph also reveals known composite repeats in Repbase. Figure 4.3 shows one connected component in the repeat domain graph containing the families RICKSHA, RICKSHA_0, a number of subfamilies of *MLT2* and the sequences containing the internal part of the endogenous retroviral element *HERVL*. Repbase annotates RICKSHA as a composite repeat that contains 79 bp terminal inverted repeats and a 3'-portion of *HERVL* endogenous retrovirus including its LTR (*MLT2B*) (see Figure 4.3). It is believed that RICKSHA replicated before it obtained the retroviral component, and the Repbase entry RICKSHA_0 contains the terminal inverted repeats and different internal sequence from RICKSHA. The repeat domain graph contains two basic paths: the path in the middle containing the edge of length 855, and the path on the left (or right, since they are reverse complement of each other) containing a sequence of red edges. The path in the middle corresponds to the RICKSHA_0 element. The path on the left corresponds

to the retroviral elements represented by *MLT2* and *ERVL*. Interestingly, the path of RICKSHA (sequence number 304) starts and ends in the middle path (the edge of length 72 corresponds to the inverted terminal repeats), but jumps to the path on the left traversing the edges of lengths 74 and 386. This graph vividly illustrates the sequence structure and putative evolutionary history of the RICKSHA element.

We remark that the subtle structure of shared repeat domains in this example are not clearly revealed by traditional row-column multiple alignment programs such as CLUSTALW [49], which align all sequences over their entire lengths. The repeat domain graph removes the restriction of aligning sequences over their entire length, and strikingly reveals the mosaic structure of these repeat families. We further remark that the correspondence between edges and repeat domains is only approximate. Determining the exact boundaries of repeat domains is a challenging problem, similarly to the difficulty in defining the boundaries of protein domains. The ambiguity in boundary definition is manifested by complicated structures of short edges in the repeat domain graph. We ameliorate this ambiguity by contracting very short edges (length less than 20).

### 4.2.3 Discovering new composite repeats: repeats in *C. elegans*

We build the repeat domain graph of *C. elegans* repeat family library generated by Stein et al. [93] with the RECON program [94]. This library contains 377 sequences of total length 251,168 bp. The resulting repeat domain graph contains 2725 edges which organized into 464 connected components. Of these, 300 components represent 150 repeat families (and their reverse complements) that have neither self-

similarities nor similarities with other repeat families. Another 109 connected components represent self-similarities among 86 repeat families that share no similarities with any other families. The remaining 55 connected components reveal the similarities and complex evolutionary relationship between the remaining 142 repeat families.

We examine one of these 55 connected components that is formed by 7 repeat families (Figure 4.4). We make the following observations.

1. The complex evolutionary history of these repeat families is reflected in the mosaic structure of repeat domains. For example, repeat family E6 (path from source E6 to sink E6 in Figure 4.4) is decomposed into 5 repeat domains. Among them, the repeat domain with length 41 is shared with 3 other repeat families: E1, E2, and E4.
2. The edges with multiplicity greater than one form two paths – plus the two reflections of the paths resulting from the symmetry of repeat domain graph (red in Figure 4.4). We refer to these paths as the long path (containing 5 red edges) and the short path (containing 3 red edges). These paths delineate important parts of the repeats and may correspond to domains important for the propagation of the repeat elements [31].
3. These repeat families contain different combinations of edges on two red paths in Figure 4.4. These structures illustrate how one repeat family may borrow repeat domains from another repeat family. Repeat families E6, E2, and E4 contain only the short path; repeat families E5 and E7 contains only the long path; repeat family E3 contains two (partial) copies of the long path, with

opposite strand configurations. Interestingly, repeat family E1 contains both paths: part of the long path followed by the short path.

These observations provide a more detailed description of the relationships among repeat families than the simple annotations that are part of the RECON library (Figure 4.4(b)). Specifically, the graph reveals a complicated relationship between these repeat families and suggests putative annotations of still unannotated repeat families in the RECON library (e.g., those in Figure 4.4(b)).

### 4.2.4 Comparative repeat domain graph analysis

Comparing repeats across different species is a non-trivial task. Zhang and Wessler [95] compared the transposable elements (TEs) in *Arabidopsis thaliana* and *Brassica oleracea* via TBLASTN searches of the most conserved coding regions for each type of TE. They found nearly all TE lineages are shared between the two plants. Without complete repeat libraries, they are unable to compare repeats on the repeat family level. Stein et al. [93] performed a similar study, comparing repeat family libraries from *C. elegans* and *C. briggsae*, and report that "… despite their general similarities, we were not able to systematically identify ortholog pairs among the *C. briggsae* and *C. elegans* repeats … we found no simple one-to-one mapping between them".

We compare two repeat family libraries by building a *comparative repeat domain graph* in the following way. Given two libraries X and Y, we first pool the sequences from both libraries into a single union library, then construct the repeat

domain graph of the union library, and color the edges in the repeat domain graph according to whether they are (i) from X only, (ii) from Y only, or (iii) from both X and Y. We call the resulting edge-colored graph the comparative repeat domain graph. Note that alternatively one could construct separate repeat domain graphs for X and Y then compare the two graphs, but this approach would introduce additional complexity in comparing graphs and should give essentially same results. We further analyze repeat domains shared by both libraries (ancient domains), and repeat domains present in a single sequence (young domains), and study the evolutionary relationship between them.

We form the comparative repeat domain graph using the *C. elegans* and *C. briggsae* repeat family libraries generated by Stein et al. [93] using the RECON algorithm [94]. Indeed, because *C. elegans* and *C. briggsae* diverged roughly 100 million years ago, it is not surprising that only certain repeat domains present in a common ancestor are still present in both species. We are particularly interested in the discovery of these shared ancient repeat domains, whose conservation are suggestive of a role in repeat propagation, or alternatively may be due to horizontal transfer.

The *C. elegans* library contains 377 sequences (with an average length of 666 bp) and the *C. briggsae* library contains 466 sequences (with an average length of 520 bp). We generate pairwise alignments between these 843 sequences and construct the comparative repeat domain graph. We annotate each edge in the graph as "*C. briggsae* (only)", "*C. elegans* (only)", or "both". Our comparison reveals that only 1810 bp are shared between the two repeat family libraries. These 1810 bp form 9 edges in the comparative repeat domain graph, comprising 4 connected components (Table 4.2).

Each component is a simple path. These edges match to *Mariner*, *CEREP5* element, and *PALTTAA2/PiggyBac* repeat families.

We analyze each of these four connected components. The two shared edges with lengths 61 and 309 are in the same connected component in the comparative repeat domain graph. A translated sequence search reveals that they match to essential parts in the transposase-coding sequence of the *Mariner* element. The edge of length 309 matches to a set of hypothetical protein in *C. elegans*, at residues 117-219. Those hypothetical proteins are all closely similar to transposases of other organisms including *Adineta vaga*, human, and *Stylochus zebra*. The edge of length 61 (translated into a 20 amino acids sequence) does not have a significant BLAST result by it own. A BLAST search of the entire repeat family consensus sequence of Cb000007, which contains both edges, gives a result similar to what was obtained by searching the edge of length 309 alone.

Figure 4.5 shows part of the component of the comparative repeat domain graph containing the edge of length 34 in Table 4.2. The blue edges correspond to the connected component in the *C. elegans* repeat domain graph shown in Figure 4.4. The red edges demonstrate four *C. briggsae* repeat families with shared domains. The green edge of length 34 is shared across the two species. We have a conservative estimate for the statistical significance of the 34 bp edge. Between the 5 sequences from *C. elegans* and the 5 sequences from *C. briggsae* (Figure6.6(b)), the closest pair across two species (e.g., B1 and E5) has only 1 bp mismatch, for which BLAST reports an E-value (P-value) of 8E-16. Thus, with the correction of the database size

(2.5E5 for *C. elegans* and 2.4E5 for *C. briggsae*), the matching between the two sequences has an E-value of 5E-6.

The comparative repeat domain graph vividly depicts the complex evolutionary history of these repeat families: subtrees split by the green edge (indicated in Figure 4.5 by dashed boxes) separate repeat families from the two species, and suggest that the repeat domain shared by both species is an ancient repeat domain from a common ancestor, rather than the result of horizontal transfer. Each of these two subtrees induces a phylogeny of the included repeat families. We checked whether these phylogenies were consistent with a phylogeny derived from nucleotide substitutions in the segment of length 34 shared by these sequences (green edge in Figure 4.5). A phylogenetic tree (Figure 4.6) of the ten sequences of length 34 constructed by CLUSTALW gives a phylogenetic tree that is remarkably consistent with the two subtrees in the comparative repeat domain graph. In particular, all three trees group *C. elegans* and *C. briggsae* families together. In addition, sequences –B2 and –B3 share few domains in the trees from the comparative repeat graph, consistent with their long separation on the CLUSTALW tree, while sequences E5 and E7 are close on all three trees. The similarity of the three trees validates the use of the comparative repeat domain graph to infer evolutionary history.

The structure of the comparative repeat domain graph raises a number of interesting and still unresolved evolutionary questions. For example, can we distinguish shared repeat domains between two species that arise from common ancestry from those that arise from horizontal transfer? How have such ancient repeat domains evolved in both genomes, and which repeat domains acquired independently

in these genomes have contributed to the evolutionary success of some repeats over the past 100 million years? Finally, we remark that the repeat domain graph shown in Figure 4.5 is generated from the alignments shown in Figure 4.1. While Figure 4.1 contains essentially the same information about local similarities between these repeat families, the graph in Figure 4.5 organizes this information into a much more interpretable structure.

### 4.2.5 Analysis of *de novo* repeat family libraries

We now demonstrate how the repeat domain graph overcomes certain imperfections found in automatically constructed repeat family libraries and directly reveals composite repeats. Repeat family libraries have historically been constructed via manual curation. Recently, algorithms such as RepeatFinder [96], RECON [94], RepeatGluer [41], PILER [97] and RepeatScout [98] are increasingly automating the process of identifying repeat families from genomic sequence. For example, RECON has aided the construction of a library of chicken repeat families [99], and RepeatScout has been used to construct human, mouse and rat repeat family libraries which are nearly as thorough as manually curated libraries. However, the resulting *de novo* libraries (particularly for mammalian genomes) are frequently contaminated by sequences resulting from segmental duplications [100]. We analyze a human repeat family library which was automatically constructed by RepeatScout, and show how the repeat domain graph helps remove these contaminants and reveals composite repeat families.

We generate a repeat domain graph of a human library generated by RepeatScout containing 1139 sequences of total length 0.68M bp. Surprisingly, the resulting graph contained a large connected component which contained more than half of the input sequences. Upon close inspection, we found that this large component was connected by a small number of long edges of single multiplicity. An analysis using BLAT [101] revealed that the instances of each of these long edges in the genome are localized in a small number of narrow genomic regions. This suggests that these long edges do not represent repeat domains, but rather are tandem duplications, a known contaminant of *de novo* repeat identification programs like RECON or RepeatScout.

This discovery revealed an extra benefit of the repeat domain graph to repeat domain analysis: it directly reveals contaminants in automatically generated repeat family libraries. Moreover, the graph suggests a procedure for removing these contaminants. Briefly, we select the longest edge along the path of each repeat family whose total length exceeds 100bp. We BLAT these edge sequences against the genome sequence and select BLAT hits whose length exceeds 80% of the edge length. We combine BLAT hits into clusters if they are less than 5Mb apart on the genome. We compute the ratio of the number of hits to the number of clusters, and classify sequences whose ratio exceeds 2 as tandem duplications. Using this approach, 107 repeat families in the RepeatScout library were thus classified as tandem duplications and excluded from further analysis. We remark that this method can detect tandem segmental duplications, but not the dispersed segmental duplications. Distinguishing

repeats from dispersed segmental duplications is a challenging and unsolved problem. It is possible that the repeat domain graph might be useful for this problem; however, this is beyond the scope of this chapter.

After removing these contaminants, we built a repeat domain graph from the remaining 992 sequences. The graph contains 885 connected components; the largest component contains 184 sequences. Since we do not have immediate biological annotations for each sequence in the RepeatScout library, we wanted to determine if direct analysis of the repeat domain graph would reveal domain recombinations or composite repeats. A "signature" in the graph of such an event is a simple branching, or Y-shape fork where two sequences enter a node, and depart on a shared edge. Unfortunately, repeat libraries (including the RepeatScout library) contain a large number of sequences corresponding to partial copies of the same repeat element, which also create Y-shape forks. To reduce the effect of these partial copies, we apply the additional requirement that all three edges in the Y-shape fork should be at least 100 bp long and have multiplicity at least 2. We find 6 such Y-forks in the repeat domain graph. Furthermore, a single connected component contains 3 such forks. Closer inspection reveals that 2 out of the 3 Y-forks are adjacent (Figure 4.7) and contain a repeat domain of length 543. We compared the sequences along this edge to human Repbase and found that they correspond to repeat families HERVE, HERVI, and Harlequin. Furthermore, Repbase Update annotates Harlequin as a recombination between several repeat families including HERVE and HERVI. Thus, we were able to directly identify a composite repeat in an unannotated library directly from a signature

in the repeat domain graph. The third Y-fork is related to some diverged subfamilies of the MER41 retrovirus. Since the MER41 subfamily has very diverged sequences, accurate subfamily annotation may not be possible. Thus it is difficult to judge whether this Y-fork is due to retroviral recombination or due to artifacts of alignment programs.

We searched the repeat domain graph from RepeatScout for the RICKSHA composite repeat family described in Section 4.2.2, but were unable to find it. We determined the reason is that the RepeatScout library itself does not contain RICKSHA, probably due to the high sequence divergence of this repeat family. In addition, we conducted a comparative repeat domain graph analysis (Appendix section B.4) for the *de novo* mouse and rat RepeatScout repeat libraries. We found the repeat domain graph helps in purging artifacts in *de novo* repeat libraries, in annotating the library, and in suggesting possible scenarios for repeat family evolution.

## 4.3 Conclusion and future directions

The computational analysis of repeats is becoming increasingly important as additional full genome sequences become available, particularly those of repeat rich mammalian and plant genomes. The problem of identifying shared repeat domains is especially critical to the understanding of repeat evolution. This chapter describes the first algorithmic advance on automatic identification of repeat domains in large genomes. We have applied our repeat domain graph approach to the single-species analysis of human and *C. elegans* repeat family libraries and the cross-species analyses between *C. elegans* and *C. briggsae* libraries. In doing so, we illustrated the

discovery of their mosaic repeat domain structure and the revelation of interesting clues about repeat evolution. As numerous new genomes with high repeat contents, such as mammals and plants, are sequenced and repeat family libraries will be typically automatically constructed, we expect that the applications of our method will multiply.

We have only begun to explore the uses of the repeat domain graph in understanding the relationships between different repeat sequences. We demonstrated that the repeat domain graph reveals known repeat domains of different biological origin. Additional candidates of such domains can be directly identified from signatures in the graph. Repeat families with shared domains which represent putative composite repeat families can be further analyzed to check if their repeat domains do in fact have different biological origins. The PILER algorithm [97], which achieves high specificity in distinguishing between different classes of repeat families, may aid this process.

The increasing use of *de novo* repeat identification tools demands careful analysis of the resulting libraries. The repeat domain graph opens up possibilities for the analysis of de novo repeat libraries. Essentially, it provides a method for merging and comparing two (or more) repeat libraries for a genomic sequence. One can construct a comparative repeat domain graph as described in Section 4.2.4. This is useful when one wants to combine two de novo repeat libraries generated by different automatic repeat identification tools; or one wants to absorb new entries from a *de novo* repeat library into an existing repeat library. The comparative repeat domain

graph provides a method for the comparison of de novo repeat libraries for the purpose of evaluating *de novo* repeat identification programs.

Also, the repeat domain graph provides an alternative representation of repeat libraries, which could be useful for the repeat masking task in the presence of composite repeats. The repeat family annotation of a genomic sequence in the presence of composite repeats is a non-trivial task. In fact, the RepeatMasker program [84] has implemented hard-coded rules in order to achieve an accurate annotation for repeat families in human genome (Arian Smit, *pers. comm.*), due to the presence of extensive retroviral recombinations. The repeat domain graph explicitly represents the mosaic structures of repeat families. Therefore, it is possible to implement a network matching algorithm [102] for a more accurate annotation of repeat families.

## 4.4 Materials and methods

The concept of A-Bruijn graph was introduced in Chapter 2. In dealing with the sequences in a repeat library, we designed a different whirl handling strategy. The existing method for A-Bruijn graph construction uses an "apply-all-glues-then-simplify (AAGTS)" strategy. Basically, all glues, i.e. pairs of positions that are aligned in one of the input pairwise alignments, are applied to construct an initial A-Bruijn graph (often full of short cycles), and then a series of graph operations are applied that remove bulges and whirls. In [41], for example, the bulge and whirl removal procedures include an approximate solution to the Maximum Subgraph with Large Girth (MSLG) problem.

When investigating the A-Bruijn approach to the construction of a repeat domain graph, we found the direct application of existing A-Bruijn graph construction algorithms is problematic. The major technical challenge is the internal sequence repeats in repeat consensus sequences. Consensus sequences of repeat families typically contain tandem duplicated subsequences, and directed or inverted terminal repeats. Tandem repeats and directed repeats with repeating unit longer than *girth* are represented as cycles in the repeat domain graph, and those with repeating unit shorter than *girth* are handled by the whirl removal procedure. However, the pairwise alignments between repeat families containing similar repeating units can confound the existing procedure for whirl removal in the A-Bruijn graph. For example, when a tandem repeating unit is duplicated for a modest number $n$ times in a repeat, a large number (up to $n(n\text{-}1)/2$) of pairwise local alignments can be generated just by self-similarities in this repeat. Even worse, different copies of a tandem duplicated subsequence can have slight variations, which may result in an even larger number of inconsistencies among the set of pairwise local alignments, leading to huge whirl-bulge networks in the A-Bruijn graph. We found the existing whirl removal heuristic is insufficient in handling the complexity in the alignments of repeat consensus sequences in a repeat library. As a result, some similar regions among repeat families are obliterated during bulge/whirl removal and left unglued in the simplified graph. For example, three repeat families, Ce000444, Ce000069 and Ce000167, in the *C. elegans* RECON library [93] contain 2, 3, and 5 copies of some 48 bp long repeat domains. The alignments between these repeat families have extensive pairwise inconsistencies. Applying the existing bulge and whirl removal procedure to simplify

the A-Bruijn graph (Figure 4.8(a)) for the *C. elegans* repeat library, the resulting graph loses 312 pairs of gluing positions between Ce000444 and Ce000069 and the two repeat families are separated in two different connected components (Figure 4.8(b)).

In order to handle such complex inconsistent glues in repeat libraries, we design and implement a new strategy for filtration of glues. Instead of applying all glues as in the AAGTS approach, we apply the glues one by one and watch for the creation of potential whirls. Specifically, if a pair of positions (*a*,*b*) is about to create a whirl, i.e., if position *b* is in a node n(*b*), and n(*b*) contains a position that is on the same sequence as position *a* and is less than *girth* away from position *a*, then the gluing pair (*a*,*b*) is discarded. This conservative gluing procedure prevents the formation of whirls during the A-Bruijn graph construction, and thus a later whirl removal procedure is no longer necessary.

We compare our new whirl-filtration procedure to the existing AAGTS procedure used by ABA by computing the fraction of gluing pairs in the input pairwise alignments that are present in the resulting repeat domain and ABA graphs. To measure the differences between the two methods, we compute the ratio of: (i) the number of gluing pairs in the pairwise alignments that are input to each method, and (ii) the number of gluing pairs present in the resulting ABA or repeat domain graph. If there are no inconsistencies in the input alignments, this ratio is 1. If there are inconsistencies in the input alignments the resulting graphs would have fewer alignment positions, since both methods resolve inconsistencies by removing some aligned positions. Thus, the ratio would be less than 1; however the best possible ratio is not known. We compute the number of gluing pairs in the input and in the

resulting graphs in the following way. Gluing pairs in the input may be redundant. For example, for three positions $i$, $j$, and $k$, if pairs ($i$,$j$), ($j$,$k$) and ($i$,$k$) are aligned in the input, then since pair ($i$,$k$) can be inferred from pairs ($i$,$j$) and ($j$,$k$) by transitivity, only pairs ($i$,$j$) and ($j$,$k$) are sufficient to define same set of gluing operations for constructing the graph. Thus, when we count the number of gluing pairs in the input, we only count non-redundant sets of position pairs; e.g. if 3 positions are aligned transitively, we only count 2 pairs. In general, if $n$ positions are aligned transitively, we only count $n$-1 pairs. To count the number of gluing pairs in the resulting graph, we count the number of positions along the edges with a multiplicity higher than 1. For an edge with multiplicity $m$ and length $l$, we count the number of gluing pairs as $l(m-1)$. This count is corrected with consideration of over-counting of the positions at common vertices shared by multiple edges.

We find that the whirl-filtration method shows a definite improvement in retaining gluing pairs from the input alignments for the *de novo* derived repeat family libraries *C. elegans* RECON and human RepeatScout (Table 4.3). Most prominently, for the *C. elegans* RECON library, the repeat domain graph produced by the whirl-filtration procedure retains 97.2% of the input gluing pairs while the graph produced by the AAGTS strategy retains only 89.4%; thus, the new procedure recovers 11553 gluing pairs. In particular, the alignment between the repeat families Ce000444 and Ce000069 is now correctly represented in the repeat domain graph (Figure 4.8(c),(d)). Conceptually the whirl-filtration strategy appears to throw away gluing pairs at beginning and therefore should have produced a graph with fewer glued positions than the previous AAGTS strategy. The explanation for this seeming paradox is the failure

of aggressive whirl removal procedure in the AAGTS approach. In the case of the well-annotated human Repbase, there fewer inconsistent pairwise alignments in the input, and thus the whirl-filtration and the AAGTS strategies give essentially the same results. Thus, we conclude that the whirl-filtration is more effective when the input pairwise alignments contain many inconsistencies, which is often the case for libraries constructed *de novo* from genome sequence.

The second complication in the construction of the repeat domain graph in repeat is the presence of palindromic sequences. The procedure for constructing an A-Bruijn graph of DNA sequences is designed to preserve the intrinsic symmetric structure of the entire graph. Thus when gluing a pair of positions, the reverse complement pair of positions are also immediately glued. With palindromic sequences, the order of gluing needs to be coordinated carefully. The existing A-Bruijn construction algorithm did not consider palindromic sequences, and consequently we found that the direct application of the existing procedure often results in a repeat domain graph containing broken paths for a single sequence. We solve this problem by changing the A-Bruijn graph construction procedure so that all glues between positions from same strands are applied before those from opposite strands. Furthermore, we ensure the bulge removal procedure can correctly remove bulges contained in palindromic regions (See Appendix section B.5 for details).

While the procedure for construction of the A-Bruin graph is independent of the particular local alignment method, in practice, the resulting repeat domain graph will vary with different input alignments. We determined that cross_match (P. Green, unpublished) with the default scoring matrix and the gap penalties used by BLASTN

is an effective tool for generating pairwise local alignment of repeat consensus sequences. In comparison, repeat domain graphs produced from alignments with BLASTN (using default parameters) were generally similar but typically contain more edges, thus artificially fragmenting repeat domains.

In all experiments, we selected local alignments with minimal length of 40 and minimal score of 30 (corresponding to BLAST E-value 1E-3) and input these into our method using default parameters with the minimal girth (-w) 40. We found that the topology of the repeat domain graph was similar when BLASTN was used to determine the input alignments (data not shown). However, we also observed the importance of filtering out low-complexity alignments, which both cross_match and BLASTN perform under their default options, but using different techniques. When low-complexity filtering was turned off in BLASTN with the "-F F" option, the repeat domain graph generally included larger connected components, reflecting the fact that more low-complexity regions were aligned into repeat domains.

We comment that although our method shares A-Bruijn graph framework with the RepeatGluer program [41], these programs have different goals. RepeatGluer is a *de novo* repeat family identification tool which attempts to identify all repeat families in an input genomic sequence. Unfortunately, it is currently not feasible to run RepeatGluer directly on long genomic sequences. Our approach takes an existing repeat family library as input and decomposes it into repeat domains.

We incorporated our new methods for A-Bruijn graph construction and simplification into a modified version of the ABA program, which is available at the

ABA website [103]. The program can also be run online at [104]. Perl scripts used for analyzing the repeat domain graphs are available at web site [92].

The text in this chapter, in part or in full, is a reprint of material as it to appear in Genome Biology. The dissertation author was the primary author of the paper. I thank my co-authors Ben Raphael, Alkes Price, Haixu Tang, and Pavel Pevzner.

## 4.5 Tables and figures

**Table 4.1: 15 repeat families containing domains shared with repeat families of different biological origin. The list is sorted by the total length of shared domains. See Appendix Table B.1 for the complete table.**

| repeat family | #domains | length of domains | percent shared |
|---|---|---|---|
| HARLEQUIN | 34 | 6245 | 0.9 |
| MER52AI | 32 | 5375 | 0.76 |
| HERVL | 4 | 5117 | 0.9 |
| ERVL | 4 | 5117 | 0.89 |
| HUERS-P3 | 31 | 5106 | 0.57 |
| LOR1I | 54 | 4034 | 0.5 |
| HUERS-P3B | 67 | 3791 | 0.51 |
| HERVE | 9 | 3463 | 0.44 |
| HERVG25 | 50 | 3431 | 0.49 |
| HERV35I | 54 | 2933 | 0.42 |
| MER51I | 45 | 2914 | 0.37 |
| MER4I | 48 | 2896 | 0.45 |
| HERVIP10FH | 25 | 2782 | 0.54 |
| PABL_AI | 52 | 2665 | 0.53 |

**Table 4.2: Four connected components formed by shared repeat domains (edges shared between *C. briggsae* and *C. elegans*). Length is the total edge length of a component. Multiplicity (Mul.) refers to the highest multiplicity among all edges in a component. The multiplicity may exceed the total number of *C.elegans* and *C.briggsae* families containing the repeat domain, because some repeat families have self-similarities (e.g., E3 and B4 each contribute 2 to the multiplicity of the green edge in Figure 4.5).**

| # Edges | Length | Mul. | # *C. elegans* + *C. briggsae* families | | | Annotation |
|---|---|---|---|---|---|---|
| 1 | 61 | 2 | 1 | + | 1 | Mariner |
| 1 | 309 | 2 | 1 | + | 1 | Mariner |
| 1 | 34 | 10 | 4 | + | 4 | CEREP5 |
| 6 | 71 | 34 | 2 | + | 16 | PALTTAA2/PiggyBac |

**Table 4.3: Comparison of the AAGTS and whirl-filtration strategies. POAP: Pairs of aligned positions. We measure the difference between the procedures by the fraction of aligned positions in the input pairwise alignments that are retained in the multiple alignments produced by each graph procedure.**

| Repeat Library | Number of non-redundant input POAP | Fraction of input POAP in graph | | Difference in POAP |
|---|---|---|---|---|
| | | AAGTS | Whirl-filtration | |
| *C. elegans* RECON | 148989 | 0.894 | 0.972 | 11553 |
| Human RepeatScout | 378080 | 0.981 | 0.994 | 4898 |
| Human RepBase | 666100 | 0.982 | 0.979 | -2446 |

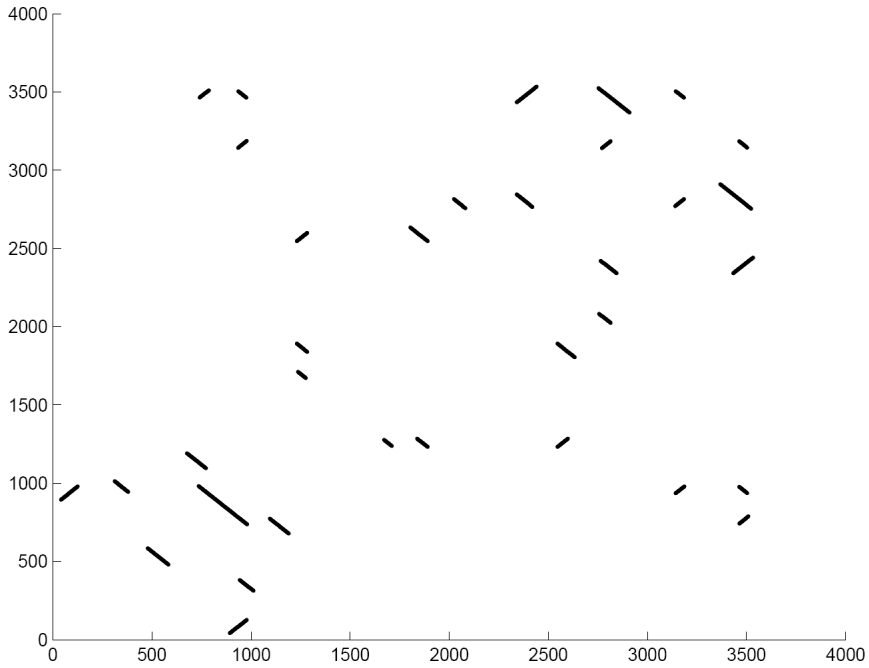



**Figure 4.1: Dot plot of 11 concatenated repeat family sequences from *C. elegans* and *C. briggsae* shows the presence of shared repeat domains. Our repeat domain graph of the same set of sequences is shown in Figure 4.5.**

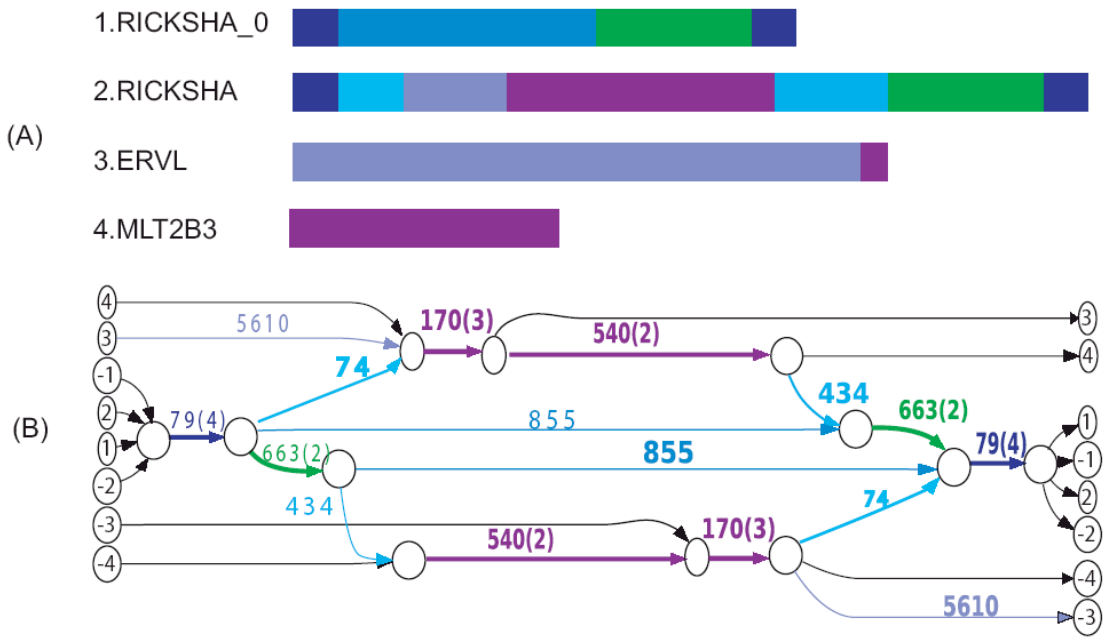



**Figure 4.2: (A) Diagram of repeat domains shared between RICKSHA and other repeat families. RICKSHA and RICKSHA_0 have 79 bp inverted terminal repeats. In addition, RICKSHA shares some sequences from retroviral elements ERVL and MLT2B. (B) Repeat domain graph of the same set of sequences. Each sequence is represented by a path from a source to a sink vertex, where source and sinks are labeled with the ID number in (A). Negative signs refer to the reverse complement sequences (see Results section). Similar parts between sequences are glued into shared edges. Edge with label *l*(*m*) indicates subparts of length *l* from *m* sequences are glued together. Edges labeled simply as *l* indicate no similarity to other sequences and lengths of short edges are omitted.**

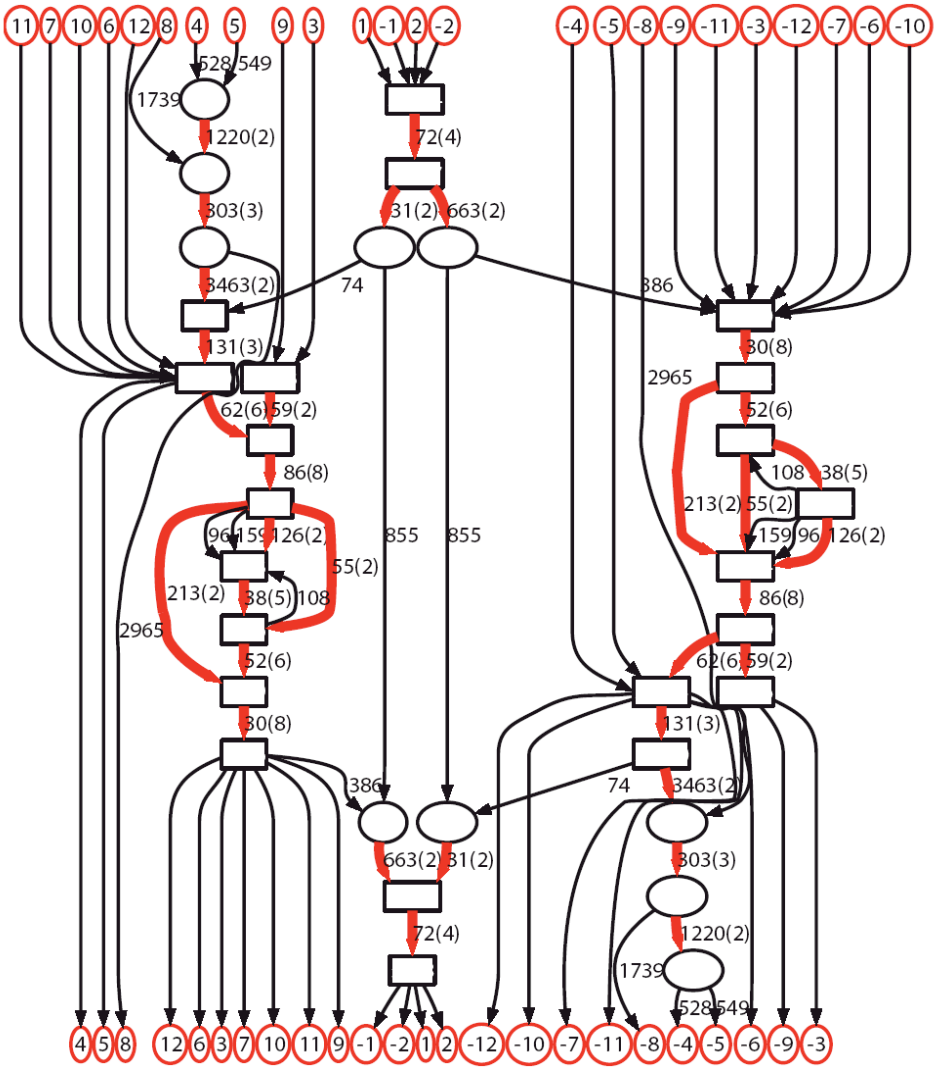
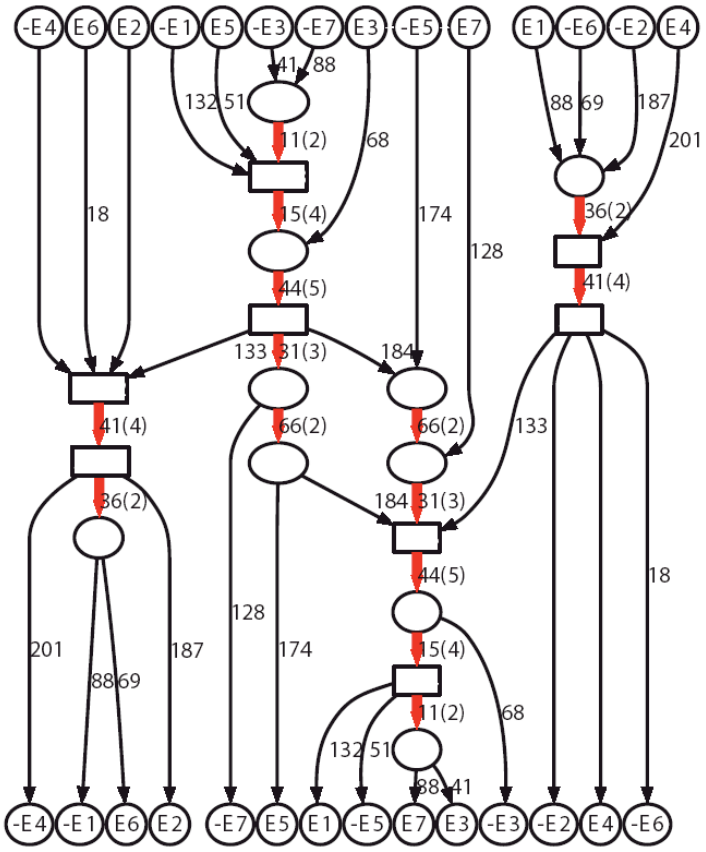



**Figure 4.3: A connected component in the repeat domain graph of the human Repbase. Labeling follows Figure 4.2(B). Edges with multiplicity more than one are highlighted in red. Source/sink labels: 1=RICKSHA, 2=RICKSHA_0, 3-12=various retroviral repeats, including subfamilies of *MLT2* and the sequences containing the internal part of the endogenous retroviral element *HERVL*.**

(a)




(b)

| ID | RECON ID | Annotation from RECON file |
|---|---|---|
| E1 | Ce000442 | CEREP5 |
| E2 | Ce000345 | UNKNOWN |
| E3 | Ce000452 | CEREP5_CEREP5 |
| E4 | Ce000514 | UNKNOWN |
| E5 | Ce000396 | CEREP5 |
| E6 | Ce000413 | UNKNOWN |
| E7 | Ce000211 | CEREP5 |

**Figure 4.4: (a) A connected component in the *C. elegans* repeat domain graph reveals similarities between 7 different repeat families. High-multiplicity edges are colored red. We contract connected subgraphs consisting of edges with a length shorter than 10 (except for edges linked to a source or sink) into boxes to simplify the overall topology of the graph. (b) Annotation of the seven families obtained from [31].**

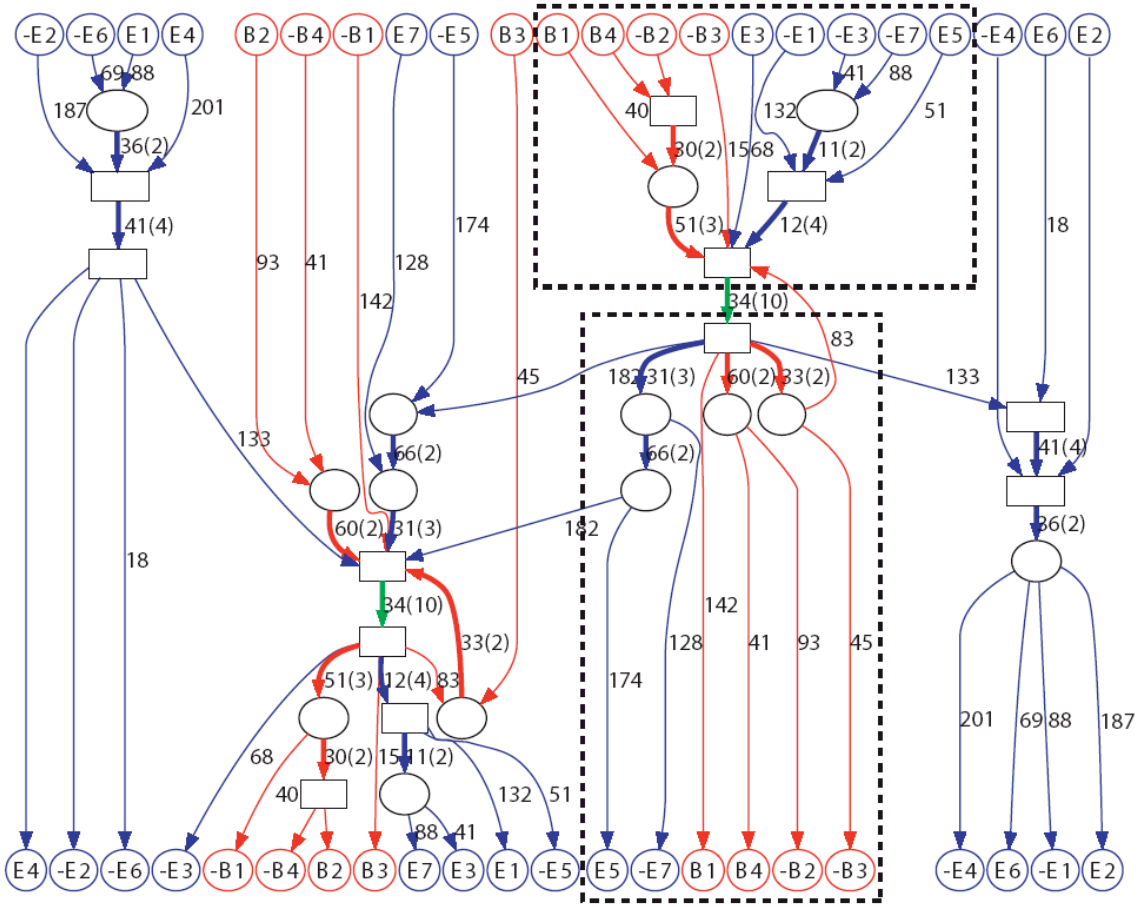



**Figure 4.5: Part of the *C. elegans*/*C. briggsae* comparative repeat domain graph. Labeling follows the legend in Figure 4.2. Edge color codes: blue = *C. elegans*; red = *C. briggsae*; green = both. Thick edges have multiplicity greater than one. Dashed boxes enclose two subgraphs with a tree topology (see Figure 4.6 and text).**

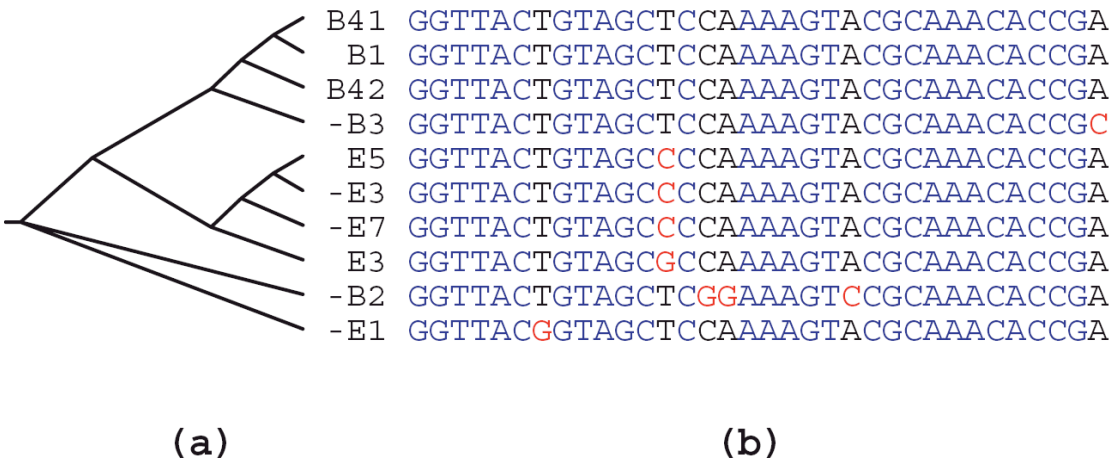



**Figure 4.6: A phylogenetic tree (a) for ten sequences (b) that form the shared green edge in Figure 4.5 (constructed by CLUSTALW). Labeling matches that in Figure 4.5, except that sequence B4 threads through the shared green edge twice, giving two sequences labeled B4_1 and B4_2. We remark that the ten sequences show few substitutions; consequently the topology of this tree is rather reliable despite of the fact that the sequences are very short.**

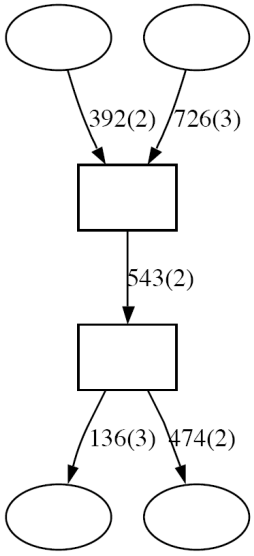



**Figure 4.7: Two Y-forks in a connected component of human RepeatScout library repeat domain graph. The complete graph is available in Appendix B.3.**

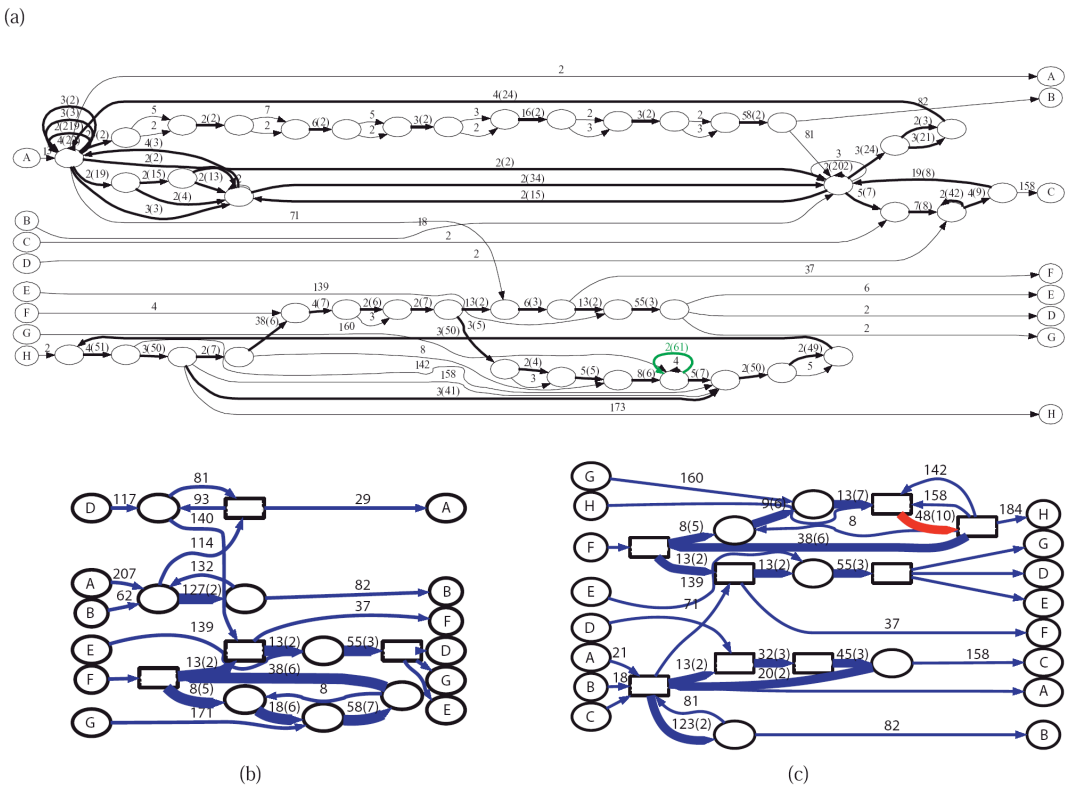



**Figure 4.8: The construction of one connected component in the repeat domain graph of *C. elegans*. (a) In the initial A-Bruijn graph, seq. H (Ce000444) and seq. F (Ce000067) are in a same connected component, but many short cycles fragment repeat domains in this graph. (b) However, in the graph after the standard bulge and whirl removal procedures (e.g. from ABA), seq. H are in a separate connected component, all glues between seq. H and seq. F are lost, due to a whirl removal process starting that the green edge in (b); (c) The repeat domain graph constructed with the new whirl handling algorithm and the bulge removal procedure, now the seq. H and seq. F are shown to share some significant edges; (d) Alignment of 10 sequences along a shared edge (red) demonstrates that sequences H and F belong in the same component with the remaining six sequences.**

# 5 Open problem: discovery of mosaic arrangements in protein structures

Mosaic rearrangement is a major mechanism of protein evolution. In Chapter 3 we have discussed the identification and analysis of protein domains from protein sequences. However, because protein structures can retain their shapes even though their amino acid sequences rapidly diverge, many mosaic arrangements in proteins may only be recognizable by studying their 3-D structures. This chapter reviews current understanding on the mosaic arrangements in protein structures, and explores the possibility of extending the A-Bruijn graph approach to the analysis of mosaic arrangement in protein structures.

## 5.1 Mosaic arrangement in protein structures

Protein sequences fold into 3-D structures to fulfill their functions. Protein structures provide molecular level understanding of protein domain organization. Mosaic arrangements of protein structures exist at both the domain level and the sub-domain level. The domain level mosaic arrangements do not alter the overall folds of individual globular domains, but change what domains are associated and in which order domains are associated. The sub-domain level mosaic arrangements change the internal folds of individual domains, and thus have interesting implications to the evolution of domains [105].

Although usually the mosaic arrangement in a protein refers to those at the domain level, it is not the focus of this chapter for several reasons. Firstly, given a set of 3-D structures, the domain level mosaic arrangements among them are apparent

since the recognition of 3-D domain structures is relatively straightforward. Secondly, the majority of the structures in the PDB are single domain structures. Large multidomain proteins are typically dissected into domains before crystallization. Thus, the remaining of this chapter is devoted to the study of sub-domain level mosaic arrangements of structures.

One type of mosaic arrangement in structures, circular permutation, has drawn considerable attentions from the structure comparison and classification community. A circular permutation of a protein structure can be viewed as an imaginary rearrangement operation that joins its N-terminus and C-terminus, and then breaks the backbone at a new location (See Figure 5.1 for an example). Although it appears dramatic, it is could be explained by tandem gene duplications [106]. As a result of the fast accumulation of solved crystal structures, an increasing number of structure circular permutations [106, 107] are reported. Also, Jung and Lee [107] developed a method for detecting circular permutations.

Another type of mosaic arrangement in structures is structure repeat, or structure duplication. In a sense, the common secondary structure elements, the $\alpha$-helix and the beta-strand, are probably trivial kinds of structure repeats. More interesting structure repeats are those at super-secondary structure level. Major folds such as TIM-barrels, leucine rich repeats (LRR), WD40, and beta-propeller consist of apparent repeating super-secondary structure units. Taylor et al. [108] used the Fourier transform to capture the periodicity of structure repeats.

While the discovery of mosaic arrangements in structures is an important problem, most existing structure comparison tools rely on the concept of rigid body

superposition of entire protein domains, and consequently they are unsuitable for this problem. The work by Jung and Lee [107] and Taylor et al. [108] are good starts. In this work, we explore the possibility of applying the A-Bruijn graph approach in the analysis of mosaic arrangements in structures.

## 5.2 Discovery of mosaic arrangement in protein structures

Although the ABA alignment was initially developed for sequence comparison, the underlying A-Bruijn graph framework has the generality to allow for an arbitrary definition of similarity. Once the similarities are translated into the A-graph, or the correspondence between positions, the sequence information is no longer used. Thus, the A-Bruijn graph approach has the potential to be extended to the analysis of mosaic arrangement in protein structures. We have made some preliminary explorations in this direction, which is presented below.

The first question is: how to extend the ABA multiple alignment approach from sequence alignment to structure alignment? While the local similarity of ABA is typically derived from local sequence alignment tools like BLAST, the local similarities in ABA structure alignment can be derived from the local structure alignment programs. Given a set of protein structures and their pairwise local structure alignments as input, the ABA alignment engine will produce an ABA graph in which the edges of high multiplicity reveal the common (rigid or flexible) segments shared among the structures.

POSA [109] is a pioneering approach to multiple structure alignment that generalized the idea of partial order alignment proven successful in MSA to the

analysis of structure alignment. POSA has the following unique features: (i) it reveals regions that only similar among a subset of structures; (ii) it dissects structures into mosaic arrangements of conserved segments. In our first experiment, we construct an A-Bruijn graph based on a set of pairwise alignments generated by FATCAT for a set of 3 Rossmann fold structures (we call this approach ABA-FATCAT), and compare the resulting graph with the POSA graph (Figure 5.2). The result shows that the A-Bruijn graph is overall in a good agreement with the POSA graph—even though the POSA graph contains more similarities (more edges), since POSA implements additional optimization after graph construction. A further improvement of the ABA-FATCAT approach would be to develop new graph-based operations in the spirit of bulge/whirl removal in the sequence alignment that can simplify the graph topology with the consideration of the 3-D structures.

While the current result is encouraging, there are no repeated or shuffled segments among these three structures. Similar to ABA addressing the limitation of POA in handling rearranged and repeated domains, ABA-FATCAT addresses the limitation of POSA: POSA is based on the notion of partial order. ABA relaxes the restriction of the partial order, and is able to handle repeated or shuffled structural segments (as directed cycles in ABA graphs).

In our second experiment, we investigate if an A-Bruijn graph can be constructed for a single structure with repeated segments based on a set of self-similarities. As repeated segments can only be detected if the suboptimal alignments are included, which is not available for FATCAT yet, we use the AFP (aligned fragment pairs) matrix to generate pairwise structure similarities. The AFP matrix was

used in popular structure alignment programs such as FATCAT [110] and CE [111]. The AFP matrix of two structures is in the similar spirit to the dot-matrix of two sequences. For the fragment starting at residue *i* of length l in structure 1 and the fragment starting at residue *j* of length l in structure 2, the (*i*,*j*)-entry in the AFP matrix is 1 if the two fragments can be superimposed well (with a small RMSD), and 0 otherwise. For all (*i*,*j*) pair there is an associated rotation corresponding to the rotation for the best superposition. It is a temptation to directly use the AFP matrix as the similarity matrix for ABA. However, structures typically contain large regions with secondary structure elements and thus the AFP matrix is typically very dense, and thus the corresponding ABA graph would be full of whirls (see Chapter 2). The FATCAT program [110] implements an AFP chaining procedure, which links two AFPs into an AFP-pair if they have similar associated rotations/translations (allowing gaps). Thus, AFPs in pairs represent longer regions with structure similarity; while AFPs not in a pair are more likely to be spurious hits. We implement a filtration procedure of AFP matrix which only keeps the AFPs that are in pairs with other AFPs. It is also possible to apply the more aggressive AFP-triple filter that only keeps AFPs that participate in some AFP triples.

Figure 5.3 demonstrates the effect to the AFP matrix by selecting AFP-pairs or AFP-triples. The structure 1a4y:a is of the leucine rich repeat fold, which consists of repeating beta-strand/α-helix units, each with about 25 amino acids. Thus the ideal AFP matrix of 1a4y:a should contain one main diagonal and multiple parallel minor diagonals with 25 amino acids from each other. It is clear that the AFP-pair filtration effectively removes some false AFPs, while the AFP-triple filtration provides a more

effective filtration. As a result, the AFP-pair/triple filtration is an effective way for removing noise in the AFP matrix. Admittedly, the case of leucine rich repeat is relatively straightforward. Although the AFP-triple filtration in this case is likely to be sufficient, more sophisticated applications of the AFP matrix are to be explored for structures with more complicated repeat structures. The promise in the AFP matrix similarity is that it would provide more details of mosaic structure arrangements than the Fourier transform analysis [108].

With this AFP-matrix filtration procedure, the filtered AFP-matrix generally contains only a few distinct connected regions. Ideally each region should be a straight line but in the AFP-triple matrix they are thick lines due to the repetitive nature of secondary structures resemble short tandem repeats in sequences. For example, long helix region around 63-88 results in the thick part along the diagonals. Additional works is required to transform the fat diagonals into thin diagonal lines that are suitable as input to ABA.

In summary, we explored the application of A-Bruijn graph to the identification of mosaic arrangements in protein structures. Our preliminary results demonstrated the potential power of A-Bruin graph representation for multiple structure alignment, and revealed the challenges of deriving sub-optimal structure alignments representing structural mosaic arrangements. We plan to continue this exploration with a first goal of producing a program that can automatically identify circular permutations and structural repeats, and an ultimate goal of understanding the mosaic arrangements in protein structures.

Finally, we point out that ABA presents a framework that integrates both sequence alignments and structure alignments, as both types of alignment define a similarity between protein sequences. We can obtain a more accurate alignment by simultaneously taking into consideration the sequence similarity and structure similarity as inputs to the ABA graph construction.

## 5.3 Figures

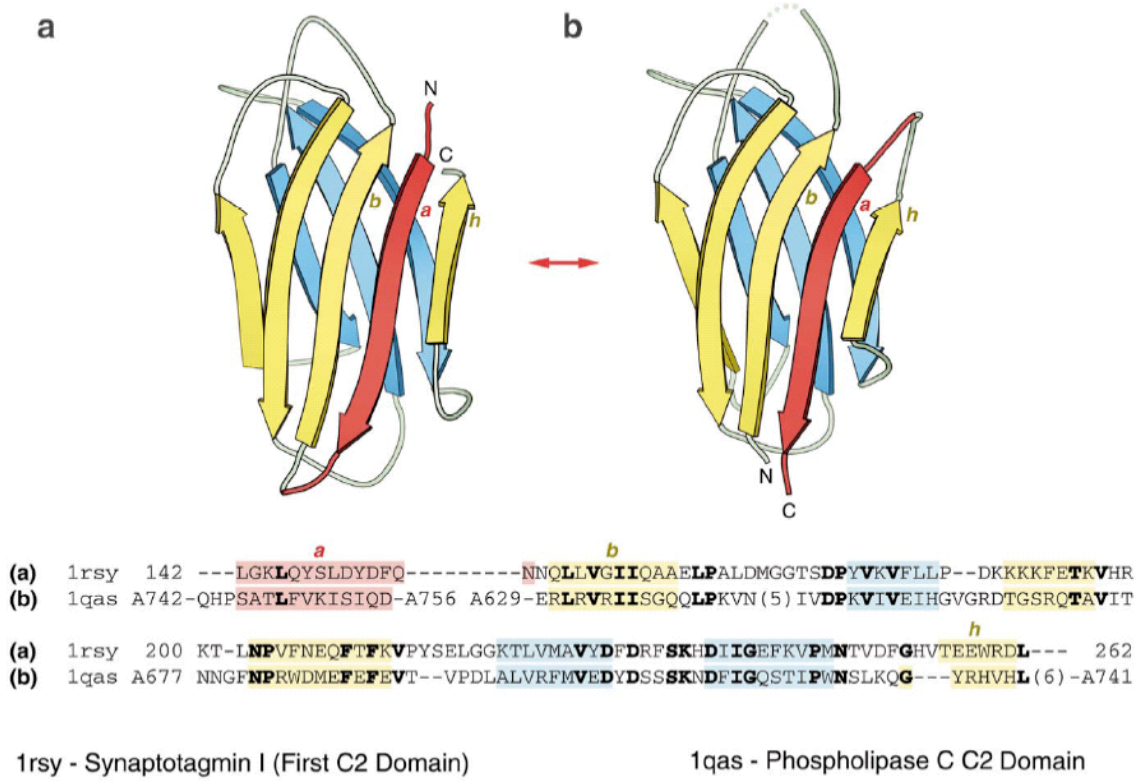



**Figure 5.1: Circular permutation of the C2 domain. (Source: Grishin, N.V., J Struct Biol, 2001. 134(2-3) [92])**

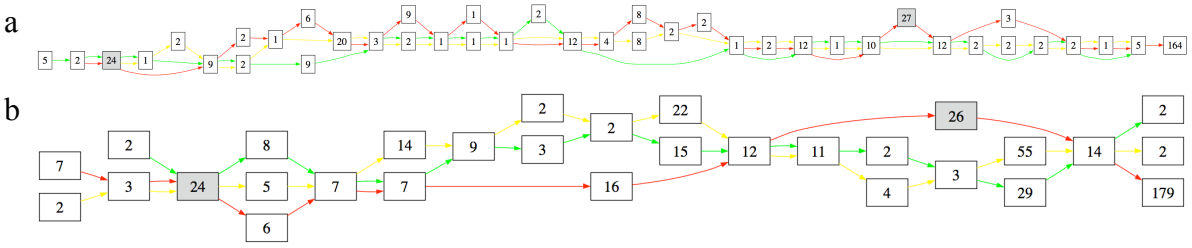



**Figure 5.2: POSA (a) and ABA-FATCAT (b) alignment graph of 3 rossmann-fold structures: SCOP ids: d1ek6a_ d2cmd_1 d1oi7a1. Blocks represent aligned segments labeled with the their lengths. Approximate correspondences between the two graphs are visible, highlighted by color-filled blocks**

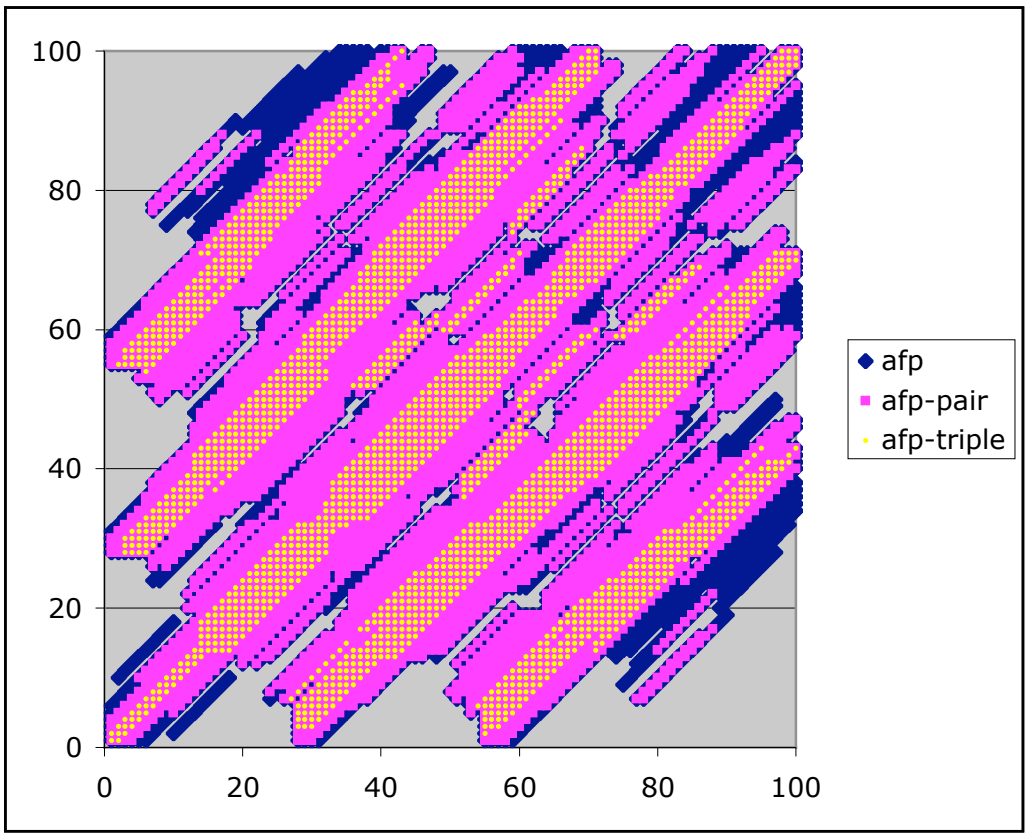



**Figure 5.3 AFP-matrix for leucine rich repeat structure 1a4y:a aligned to itself. Only first 100 residues are shown. Filtering with AFP-pairs removes many weak AFPs. Filtering with AFP-triples seems to be the most effective in removing false AFPs.**

# 6 Representing and comparing protein structures as paths in three-dimensional space

## 6.1 Introduction

Knowledge of protein three-dimensional structure is a prerequisite to understanding its function at a molecular level. With more than 30,000 protein structures in the rapidly growing public repository PDB [112], the importance of computer algorithms that can rapidly compare and find remote similarities between these structures cannot be over-emphasized. The comparison of protein structures has been an extremely important problem in structural and evolutionary biology ever since the first few protein structures became available. Hundreds of algorithms for protein structure comparison have been developed; there are several large databases and WEB resources devoted almost entirely to the problem of comparing and classifying protein structures, such as SCOP [25], CATH [24, 113], and DALI domain dictionary [114].

Typically, different representations of the protein structure are employed for comparing them for different purposes. For example, an all-atom protein model is useful when studying finer details of a protein structure such as the subtle changes of the side-chain conformations of residues in the active site a protein may undergo upon binding of substrate. However, for the rapid comparison of protein structures in order to find global similarities, only one point per residue, often the positions of the $C_\alpha$ atom, is sufficient. Some programs use completely different representations of protein structures, such as distance matrices [115] or secondary structure vectors [116].

All protein structure alignment programs optimize some mathematical definition of structural similarity. The most popular measure of structural similarity is the root mean squared deviation (RMSD) of the matching atoms [117] and its variants [90]. In general, alignments optimizing different measures of structural similarity may be different from each other [118]. Moreover, structural alignment is an NP-hard computational problem [111] and in order to solve it in a realistic time various heuristics have been developed, such as, lowering the dimensionality of the problem by identifying 7x7 interaction patterns in DALI [115], describing the protein as a set of vectors based of secondary structure elements in VAST [116], or using local structural similarities to identify short aligned fragment pairs (AFPs), which are used later to construct the alignment in the CE [111] or FATCAT [110] methods.

Since structure alignment programs that optimize RMSD dominate the field of structure comparison, they create a misconception that only structures that can be superimposed with reasonable RMSD criteria, such as low RMSD over a large length of the proteins, should be considered similar. While this is a pragmatic definition of structural similarity that eliminates an excess of false-positive matches, it also fails to find similarities between structures with extensive conformation changes including structures with internal rearrangements and/or with swapped parts between domains. The recent years have seen advances in algorithms that can align protein structures assuming flexibility of their polypeptide chains [110, 119]. Human curated structure classifications (such as SCOP and CATH) have dealt with this problem indirectly, by using a highly abstracted, but not precisely defined, view of protein structure (fold) and by grouping together protein structures based on a combination of sequence,

structure and functional information. The rapid accumulation of new structures, however, outpaces the human curation efforts, and automatic means of detecting structural similarity that is beyond the scope of RMSD based structure alignment programs are becoming essential.

In this chapter, we propose a very general description of the protein structure that views it as a path in three-dimensional space, and describe a novel algorithm for structure comparison based on this description. Our approach for abstracting the protein structure is inspired by the earlier work in our group [120] and by the earlier U-turn model [121]. We developed a highly simplified description of protein structure that removes all local structural information by "smoothing" the protein backbone, leaving only information about whether a protein chain is locally straight or whether it makes a turn. In particular, we "smooth" the protein backbone by averaging $C_\alpha$ position in a 7-residue window [120]. Chain fragments that remain straight after the smoothing procedure are denoted as *generalized secondary structure* elements. Local secondary structural information is partially lost, and protein structure is abstracted to a path in three-dimension that, for a typical protein structure, winds through space by following a straight line for a 5-12 residues, then turning in a typically 4-5 residue turn only to assume a straight course for another 5-12 residues.

We represent these characteristics by describing such paths as series of turning angles along the (generalized) backbone. Intuitively, the angles are close to 180° along the straight fragments and are smaller where the backbone is changing directions. Turning angle series has several advantages for being a good descriptor of protein structures: (i) It is invariant to rotation and translation. (ii) It is tolerant to hinges,

bending, or other structural distortions that only result in small and well-localized changes of turning angles. (iii) By treating the 1-D turning angle series as a sequence of numbers, one can define the problem of comparing structures as aligning the angle series in the same fashion as the traditional sequence alignment problem, for which an optimal solution can be derived by standard dynamic programming techniques.

The idea of 1-D geometric descriptions of structures and the dynamic programming alignment methods have been explored previously [122, 123], including the use of curvature and torsion angles of spline-approximation of the backbone to describe the local chain structure [124] [93]. However, these methods differ significantly from our method at the level of structure abstraction in that they typically provide a much richer description of the protein. Our method only focuses on turning angles in a generalized (smoothed) protein backbone, thus providing a minimal level of structure description. For instance, it would be impossible to recreate the three-dimensional structure using just our turning angle descriptions, torsion angles would have to be added for that purpose. Interestingly, as we will show in the chapter, in the world of protein structures, this minimal information is often sufficient to recognize similarity between structures.

We implement a dynamic programming algorithm for aligning the turning angle series and test it against the structural similarity defined by the SCOP database. Surprisingly, even at this clearly oversimplified level of protein structure description, our results are in a good agreement with the SCOP classification and existing structure alignment programs. We also test our method on structures with extensive distortion

and structures from distinct SCOP folds. Our results reveal interesting perspectives about structural similarity.

## 6.2 Methods

### 6.2.1 Backbone smoothing and turning angles

Our backbone smoothing procedure follows that of [120]. We assign the center of gravity of every $k$ consecutive $C_\alpha$ atoms as a new pseudo-$C_\alpha$ atom. With a proper choice of $k$, the resulting chain of pseudo-$C_\alpha$ smoothes out the local "wiggles" due to the zigzags in β-strands or the spiral patterns in the α–helices and reveals the global fold of the protein structure as a smooth curve in three-dimensional space. Thus, we refer the chain of pseudo-$C_\alpha$'s as a *smoothed backbone* (Figure 6.1A). Our smoothing procedure suppresses the local high frequency curvature signals that arise from the local periodicity of the backbone, and thus reveals the overall topology of the structure.

We define the turning angle at each pseudo-$C_\alpha$ atom along the smoothed backbone in order to reflect medium level topological features around it. Ideally, this turning angle should be close to 180° in the middle of a long straight segment along the smoothed backbone and small (close to 0°) at a sharp turn such as a β-hairpin. Also following the definition in [120], we define the turning angles at residue $i$ as the angle between the two vectors $[i\text{-}d+1, i\text{-}d]$ and $[i+d\text{-}1, i+d]$. The value of $d$ determines the span of the angle definition, thus $d$ is called the *angle defining distance*. Assuming that $d$ is small, the fragment from residue $i\text{-}d$ to $i+d$ is almost planar and the torsional

angles are negligible, this definition can be interpreted as the integral of the curvature function of the chain in this local interval. As our definition is different from the usual definition of curvature, we use the term “turning angles” instead of “curvature”.

We experimented with different choices of *d* (Figure 6.1B). Small values of *d* make all angles indistinctively large: they only capture local turns and are unable, for example, to describe the 180° turn in anti-parallel β-sheets. Large values of *d*, however, are uninformative for revealing local curvatures. Figure 6.1B demonstrates the effect of value *d* on the shape of the angle series curve: with decreasing *d* values, the first two plateaus in curve *d*=1 dissolve into narrower and lower peaks, while the valley between them becomes deeper and wider. We choose *d*=3 since it is the smallest value which gives a good dynamic range of angle values.

It is worth to note some general features of the angle series description of a protein structure. First, the plateaus and peaks (regions with high angle values) correspond to straight parts after smoothing, often long secondary structure elements or generalized secondary structure elements. For instance, in Figure 6.1B, the first plateau/peak corresponds to the first α−helix in structure and the second peak corresponds to the β-strand after the first α−helix). However, some straight segments do not correspond to classical secondary structure elements [120], we call such regions generalized secondary structure elements. Second, the valleys correspond to points where the path changes direction. Turning angle series is a rich description of the chain topology that also includes detailed turning characteristics, such as the length of the turn and the type of secondary structure (or generalized secondary structure)

elements, with the latter described by the density of points along the smoothed chain [120].

The turning angle series is not the only 1-D representation of 3-D structures. Alternative 1-D representations include secondary structure profile, and solvent accessibility. We choose turning angle series because it is unique in following three aspects. First it is purely geometrical, i.e., it only depends on 3-D coordinates of the $C_\alpha$ atoms. In particular, it does not assume any secondary structure classification, which is sometimes problematic [120]. Thus it gives a minimalist structure description. Second, turning angle series is locally robust, i.e., a global conformation change will not change turning angle series drastically. Solvent accessibility and contact number will change significantly. Third, turning angle series is better interpretable in the context of conformation change, as we will see in the Result section.

In order to uniquely specify a path in 3-D, both curvature and torsion angles would be required. The information about the torsion angles is lost in the representation of the path used here; therefore, it cannot distinguish whether the next straight element after a turn would be to the left or right of the original element. Surprisingly, as we will show below, using only the curvature angle series along, we can still recognize most cases of the structural similarity between actual protein structures.

### 6.2.2 Aligning turning angle series

We treat the turning angle series as a sequence of numbers. A natural way to compare such sequences is via dynamic programming. Protein and DNA sequences are described by discrete alphabet and could be aligned by the well known dynamic programming algorithms ([125, 126]). The alignment of sequences of continuous numbers is rarely used in bioinformatics; however, it is very well studied in computer science as the time warp problem. Essentially, given two series indexed by time, the objective of time warp is to find the optimal matching between the points along the two time series. Typically, mismatches are penalized by the squared deviation of two time points (see [69] for a review).

In this study, we employ a standard time warp setting. Given two turning angle series ($a_i$) and ($b_j$), the goal is to find a maximally scoring gapped local alignment between them. The total score is the sum of scores of matching turning angle pairs with affine gap penalties. We adopt the standard affine gap penalty scheme, and define the score for matching a pair of angles $a_i$ and $b_j$ as of the form $-(a_i - b_j)^2$, i.e., the penalty of aligning two angle values increases quadratically with their angle difference. To avoid over-penalizing a large angle difference, the score has a lower cap. If all matching scores are negative, the optimal alignment would be of zero length. To encourage longer alignments, the matching score is augmented by a default reward $r_0$. Any angle difference smaller than $r_0$ is rewarded, otherwise it is penalized. Thus, the overall score for matching a pair of angles $a_i$ and $b_j$ is:

$$S(a_i, b_j) = r_0^2 - \min[(a_i - b_j)^2, (1.5r_0)^2].$$

Although our simplistic scoring scheme may produce unrealistic alignments (such as creating large gaps in the middle of secondary structure elements), we find that this scheme produces overall structure alignments that while not accurate enough for comparative structure modeling, are yet good enough to discover the overall structural similarity.

$r_0$ and the gap opening and extension penalties are adjustable parameters. Based on parameter-tuning tests (data not shown), we found the alignment is not very sensitive to the choices of $r_0$ and gap penalties as long as the alignment is in the log-phase [127]. In our experiments we choose default parameters to be $r_0$=21 and gap opening/extension penalties 300/100. All these procedures are implemented as a program CURVE, available as a webserver at http://pops.burnham.org/curve.

## 6.3 Results

### 6.3.1 The angle series alignment mostly agrees with existing measures of structural similarity

In our first experiment, we used our program CURVE to align a single structure against all structures in a 40% non-redundant set of SCOP 1.65 with 8666 structures (see http://fatcat.ljcrf.edu/fatcat/struct_neighbor/ for details) to see if the top scoring hits agree with widely accepted protein structure classifications and other structure comparison programs. The top 10 scoring hits of a search using 1dlw:a (chain A of the PDB 1dlw), a six-helical truncated hemoglobin (SCOP classification a.1.1.1) is listed in Table 6.1. The result is not surprising: the top 5 hits include all chains from the SCOP family a.1.1.1, the 6-helices truncated hemoglobin, followed by chains from the

SCOP family a.1.1.2, the 7-helices hemoglobin. Figure 6.2 shows the detailed CURVE alignment for 1dlw:a vs 1gvh:a1. As a comparison, the alignment dotplot generated by FATCAT is overlaid with the alignment dotplot generated by CURVE in Figure 6.2A. The alignment paths of CURVE and FATCAT have an excellent overall agreement. Due to the smoothing procedure, the turning angle series is shorter at both termini, thus CURVE alignment path is shorter correspondingly.

We also get some non-hemoglobin hits in the top 10 list. The 8$^{th}$ and the 10$^{th}$ on the list are all-alpha proteins containing subdomains with some resemblance to the hemoglobin fold. We notice that both these non-hemoglobin hits are easily distinguishable from the other hits in that they have much larger number of gap residues than the hemoglobin hits (see Figure 6.3 for the case of the 8$^{th}$ hit). One could apply a simple criterion limiting number of gap residues to remove such artifacts. However, such a criterion will also limit the search results by scoring low on homologous domains that have large insertions in them, similar to how sequence alignment programs behave.

### 6.3.2 Recognition of similarities between drastically different conformations of same structures

Proteins are intrinsically flexible. Some proteins assume drastically different structural conformation at different conditions (such as binding to different substrates) to fulfill their functions. The similarity between different structural conformations of a protein can go beyond what traditional RMSD-based structure alignment tools can

recognize. Below we demonstrate that our method is particularly suited in identifying similarities between structures with such drastic conformation changes.

It is well known that the APO form and the calcium-binding form of calmodulin have distinct conformations [128]. In its calcium-binding form, calmodulin has two globular domains (N- and C- terminal domains) linked by a central α−helix. In its APO form, the central α−helix is broken into two short α−helices linked by a region with poorly defined secondary structure. In addition, the two globular domains experience some smaller internal changes. Traditional structure alignment programs that are based on rigid body superposition can only align one of the terminal domains at a time and thus are unable to capture the overall conformation change. Angle series alignment produces the correct result. From Figure 6.4, it is clear that CURVE captures the breakage of the central α−helix with a region with large angle deviations, while the twists in both N-terminal and C-terminal domains only result in smaller changes in turning angle among few residues. We notice that flexible alignment programs (FATCAT [110] and FlexProt [119]) can also align these two structures through their entire length. However, they have to introduce four hinges, and they do not have a way to distinguish the major structure changes versus minor structure changes, thus the breakage of the central α−helix and the minor structure internal shifting at both N- and C- terminal domains are both penalized indistinguishably.

Conformation changes are common when monomeric subunits form domain-swapped oligomers. We present two examples of such cases. In the first example, we compare two conformations of the catabolite repression HPr-like protein from *Bacillus*

*subtilis*. The monomer structure 1k1c:a has an anti-parallel β-strand of order 1423. In the dimer structure 1mu4, two subunits swap their N-terminal beta strands. The angle curve overlap graph of the CURVE result is shown in Figure 6.5. Similar to the case of aligning calmodulin structures, CURVE aligns both the main part (alignment positions 13-75) and the N-terminal swapped β-strand (alignment positions 1-7, shortened by smoothing at the N-terminus). Moreover, CURVE captures the subtle conformation change on the main part: notice the angle changes around the alignment positions 44-48.

In the second example we compare the immunoglobulin-binding domain B1 of streptococcal protein G (GB1), a favorite subject of studying protein folding and design. Mutants of GB1 are reported to adapt drastic different structure conformation from the wild type [129]. The wild type structure (PDBID: 3gb1) contains an α–helix and a 4-stranded β-sheet made of two beta hairpins, one N-terminal and one C-terminal to the α–helix. The structure of mutant HS#124F26A (PDBID: 1q10) reveals a domain-swapped dimer that involves exchange of the second β-hairpin. The resulting overall structure is comprised of an eight-stranded beta sheet whose concave side is covered by two α–helices. CURVE alignment reveals that the most significant angle change happens to the region between the α–helix and the second β-hairpin (Figure 6.5); all secondary structures remain mostly unchanged. In both cases, the conformation change results in structures with large RMSD, while the changes on turning angles are only modest. In such cases, aligning structures by directly optimizing RMSD may be not a good choice. CURVE alignment directly captures the

backbone turning angle changes associated with the conformational changes, which, we argue, is a better choice.

### 6.3.3 Revealing similarities between structures from distinct folds but sharing structural (and often functional) similarities

We explore if CURVE alignment can detect similarities between structures from different SCOP folds that share similar functions. We report an interesting case showing the structural similarities between members of two groups of periplasmic binding proteins, a Lysine/Arginine/Ornithine-binding (LAO) protein from *Salmonella typhimurium* (1lst) and a Leucine/Isoleucine/Valine-binding (LIV) protein from *Escherichia coli* (2liv). Apart from their similar functions, both structures consist of two similar intertwined domains, each resembling an a/b/a sandwich from the Rossmann superfold (Figure 6.6). The two domains are linked with different angle at the two structures. Moreover, the two structures have different arrangements of the β-strands. Traditional structure alignment algorithms are unable to detect the similarities between these two structures. The angle curve overlap graph of the CURVE alignment (Figure 6.6A) shows that there are two good aligned regions: 34-103 and 144-303. The region 34-103 corresponds to the alignment between the a/b units[5] 345 of 2liv's N-terminal domain and the a/b units 234 of 1lst's N-terminal domain. The region 144-303 corresponds to similar regions spanning both the C- and N-terminal domains, including the alignment between the a/b units 2345 of the C-terminal domain and the three α–helices in the N-terminal domain of both structures.

[5] An α/β unit is defined as an α-helix followed by β-strand that runs back close to the beginning of the α–helix.

The alignment of structures with similar intertwined multi-domain organization is a difficult problem for structure alignment programs. Above result shows that CURVE is able to find the overall correspondence of two structures with such domain organization. We believe that further refinement of the CURVE program may provide a promising solution for this problem.

## 6.4 Conclusion

In this chapter we introduce the turning angle series along smoothed backbones of structure as a new descriptor of protein structure. We demonstrate its utility in defining structural similarity by implementing and testing a time wrap alignment, CURVE, based on this feature. Our results show that this simple approach works surprisingly well. Although not directly optimizing RMSD, the result of CURVE generally agrees with the SCOP structure classification and traditional structural alignment programs. Moreover, CURVE can reveal similarities between drastically different conformations of the same protein structure, which is beyond the scope of traditional structure alignment programs. In aligning structures from different SCOP folds CURVE demonstrate its potential in identifying remote structural similarity.

## 6.5 Discussion and Future directions

The results of this chapter bring up an interesting question: since turning angle curve similarity is only a necessary condition for structural similarity, why would CURVE alignment work? We postulate that this is because most proteins constraint into a compact shape, thus for a given turning angle series, there are only a small

number of ways to arrange them into a realistic compact shape. For example, turning angle series cannot distinguish between α/β/α units arranged in a right-handed or left-handed fashion. Fortunately, right-handed α/β/αconnections dominate over left-handed ones in naturally occurring proteins.

Our result extends the traditional definition of structural similarity. This brings up an interesting question on the structural constraints of protein evolution. For most structures, changes in their sequences caused by mutations such as substitutions and minor indels only result in subtle changes in structure with the overall three-dimensional shape of the structure largely being preserved. For some structures, such as the GB1 protein, small mutations can result in a drastic change of their structural conformation. Since CURVE is able to detect similarities between structures with a similar turning angle series but drastically different structure conformations, An interesting exercise will be to collect a set of such structure pairs where traditional rigid-body alignments fail to detect a significant similarity but the CURVE alignment can, and see what is in common among these structures. There may be intrinsic characteristics among these structures that account for such conformational changes. Some structures may utilize this flexibility in order to be functional.

The angle series alignment has some interesting implications. Traditional structure alignments have never been like sequence alignment. While sequence alignments typically define an edit distance, a score defined by a procedure via which one can transform one sequence into the other, existing structure alignment programs optimize RMSD of a superimposed subset of residues among structures. The result of

such a structure alignment does not provide a series of operations that transform one structure into the other. Angle series alignment produces a set of angle matches that could be interpreted as a series of operations for structure transformation. Naively, one can bend every angle of one structure to the corresponding angle of the other structure. To derive a set of realistic backbone-bending operations, one needs to consider the stereochemical constraints of the backbone and correlation of the turning angles.

One interesting extension of the pairwise CURVE alignment is the multiple-CURVE alignment. Traditional RMSD based structure alignment programs cannot align more than a handful of structures, due to both the computational complexity and the representation of the multiple-structure alignment. CURVE, however, uses 1-D features and can thus be easily extended to handle multiple structures, in a similar fashion as the multiple sequence alignment. It is thus possible to construct a profile HMM of curves for structures from a family, a superfamily, or even a fold. Such profiles would be useful for remote homology detection, structure classification and comparative structure modeling.

We want to emphasize that the goal of this work is not to produce a structure alignment method that is superior to existing methods. We demonstrated that CURVE is able to find structure similarity that was overlooked by traditional structure alignment programs by case studies. We did not perform a benchmarking test mainly due to that currently there is no collection of structures with extensive conformation changes publicly available. One useful task would be to construct one such collection.The current prototype implementation of angle series alignment certainly can be improved by incorporating more information. For example, the alignment of

angle curves can only give an alignment with a certain resolution due to the smoothing procedure. It is possible to implement an iterative refinement scheme which starts with an overall alignment of angle series based on a large smoothing radius, then iteratively refine the alignment by considering angle series based on smaller smoothing radii. Since angle series is only a "planar" feature, adding 3D features such as handedness information will help (in cases where distinguishing between left and right is important.)

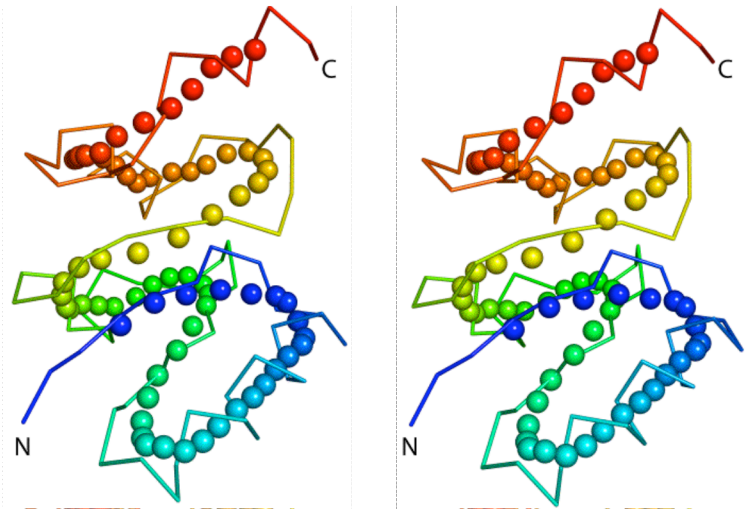
The text in this chapter, in part or in full, is a reprint of material as it to appear in a submitted manuscript. The dissertation author was the primary author of the paper. I thank my co-authors S.S. Krishna, Haibo Cao, Pavel Pevzner, and Adam Godzik.

## 6.6 Tables and Figures

**Table 6.1: 10 top-scoring hits for hemoglobin d1dlwa_ using CURVE.**

| SCOP | Chain id | Score | # gaps | # aligned |
|---|---|---|---|---|
| a.1.1.1 | d1dlwa_ | 44982.0 | 0 | 102 |
| a.1.1.1 | d1idra_ | 41916.9 | 2 | 102 |
| a.1.1.1 | d1dlya_ | 40359.8 | 2 | 102 |
| a.1.1.1 | d1mwba_ | 31841.0 | 15 | 99 |
| a.1.1.1 | d1ngka_ | 31728.6 | 15 | 100 |
| a.1.1.2 | d1gvha1 | 30728.1 | 20 | 98 |
| a.1.1.2 | d1vhba_ | 30711.4 | 17 | 96 |
| a.123.1.1 | d1n46a_ | 29475.5 | 47 | 100 |
| a.1.1.2 | d1jl7a_ | 29430.3 | 31 | 97 |
| a.118.6.1 | d1ld8a_ | 29379.9 | 48 | 100 |

(A)




(B)

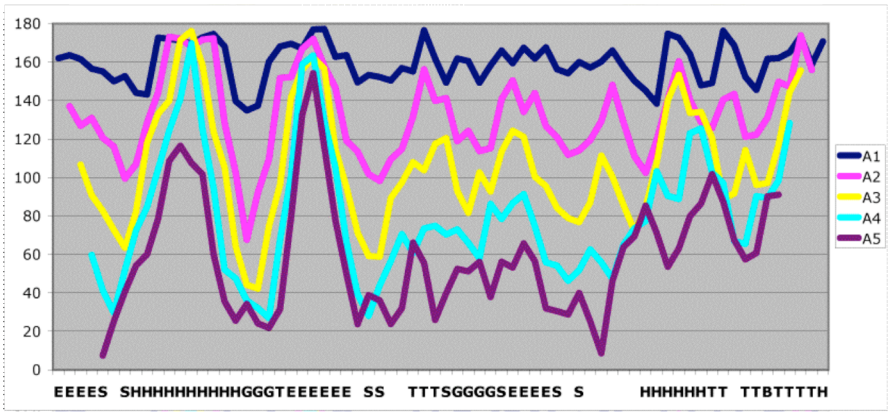



**Figure 6.1: Backbone smoothing and turning angle series of a structure (SCOP id d1b6ra2). (A) Stereo images of overlapping backbone and smoothed backbone with smoothing radius d=3; (B) turning angle series along the smoothed backbone with different angle defining distances, with X-axis labeled by DSSP [130] secondary structure annotation. Data series: A3: angle defining distance d=3.**

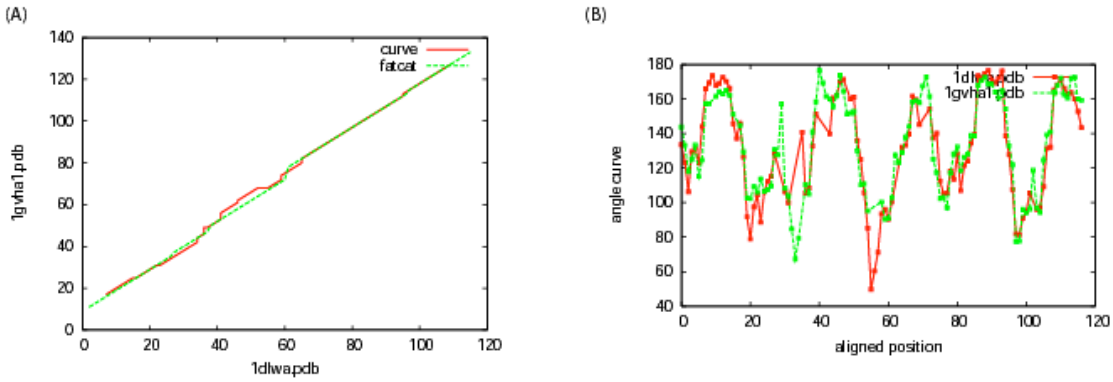



**Figure 6.2: Aligning 6-helices hemoglobin 1dlw:a to 1ghv:a1. (A) dotplot of the alignments generated by FATCAT and CURVE; (B) angle curve overlap graph.**

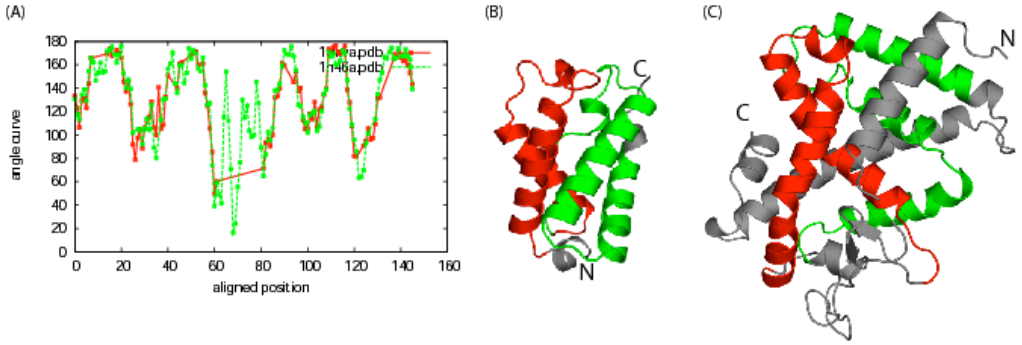
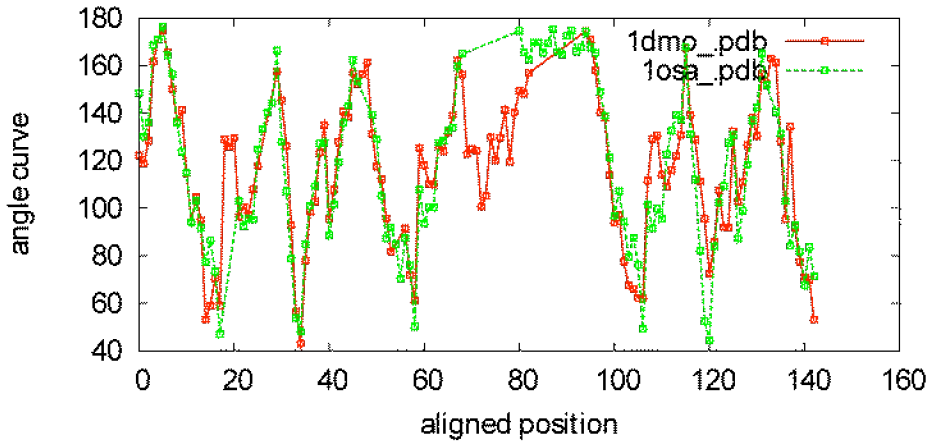



**Figure 6.3: Aligning 6-helices hemoglobin 1dlw:a to 1n46:a. (A) angle curve overlap graph; (B,C) structures of 1dlw:a and 1n46:a with aligned regions highlighted with same colors.**

(A)




(B)

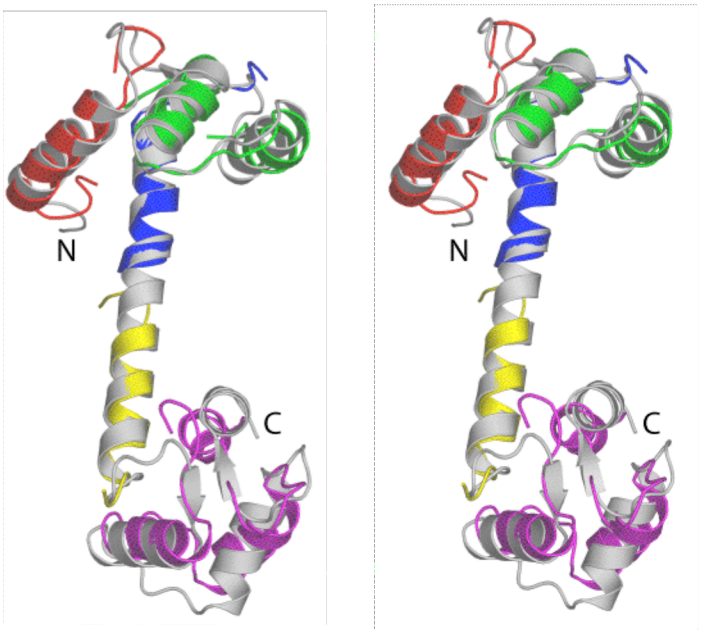



**Figure 6.4: Flexible alignment of different calmodulin structures: 1dmo (APO form) and 1osa (Ca-binding form). (A) Angle curve overlap graph of 1dmo and 1osa. (B) Stereo diagram of the structural superposition of 1dmo (colors) and 1osa (grey) generated by FATCAT. FATCAT breaks 1dmo into 5 rigid body segments (each with a unique color) linked by hinges. Notably, the long α–helix in the middle of 1osa is broken into two smaller ones in 1dmo. See text for details. Higher gap penalties (opening:1000 and extension:333) were used so that the center region (alignment positions 65-95) appears to be "mismatch" instead of parallel gaps in both angle curves. This is only to enhance the presentation of the angle changes associated with the conformation change – the default parameters would produce essentially the same result.**

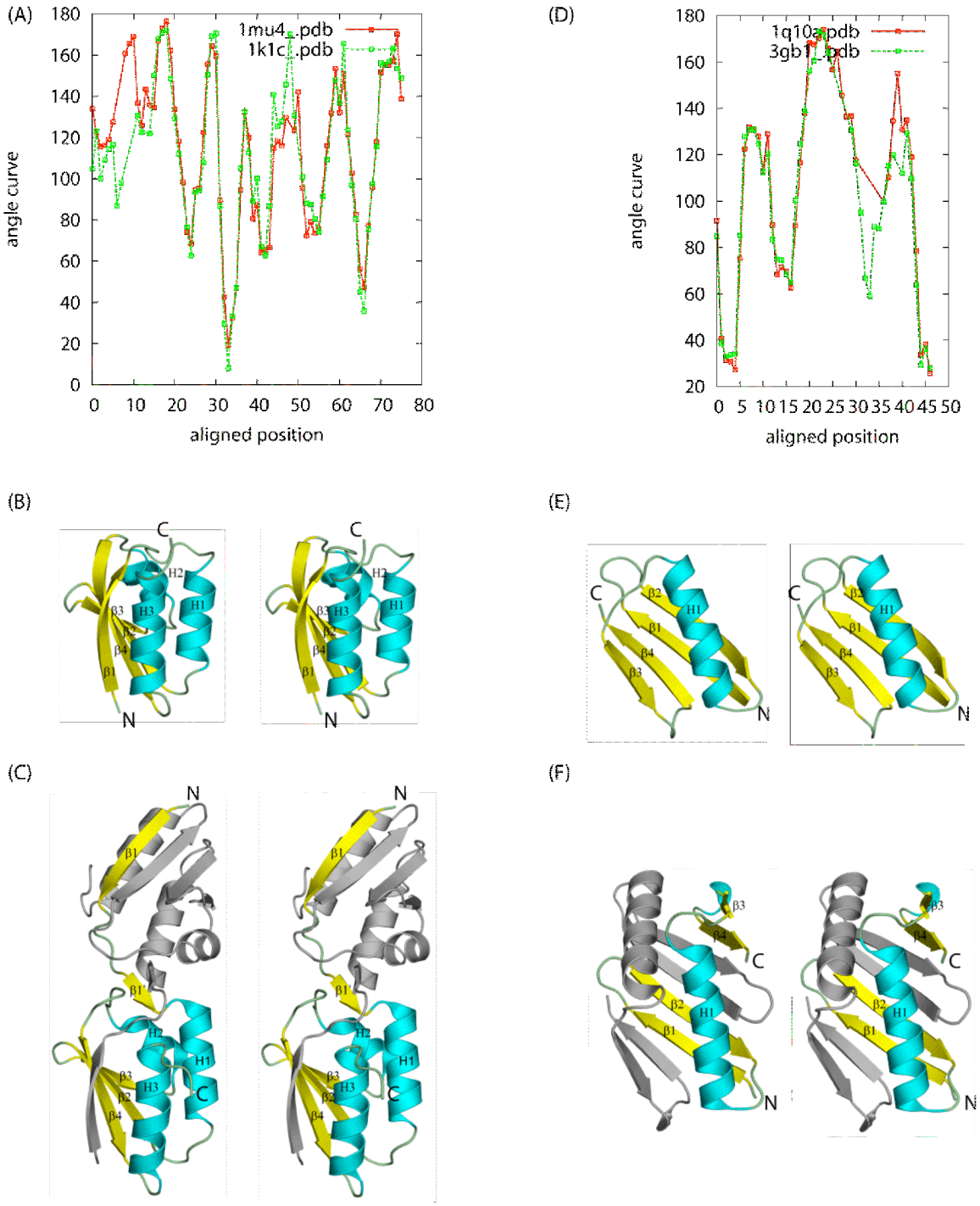



**Figure 6.5: Angle curve overlap graphs for 1mu4 and 1k1c:a (A) and for 1q10:a and 3gb1 (D); (B,C) stereo diagrams of 1mu4 and 1k1c:a with aligned regions highlighted with same colors; (E,F) stereo diagrams of 1q10:a and 3gb1 with aligned regions highlighted with same colors**

.

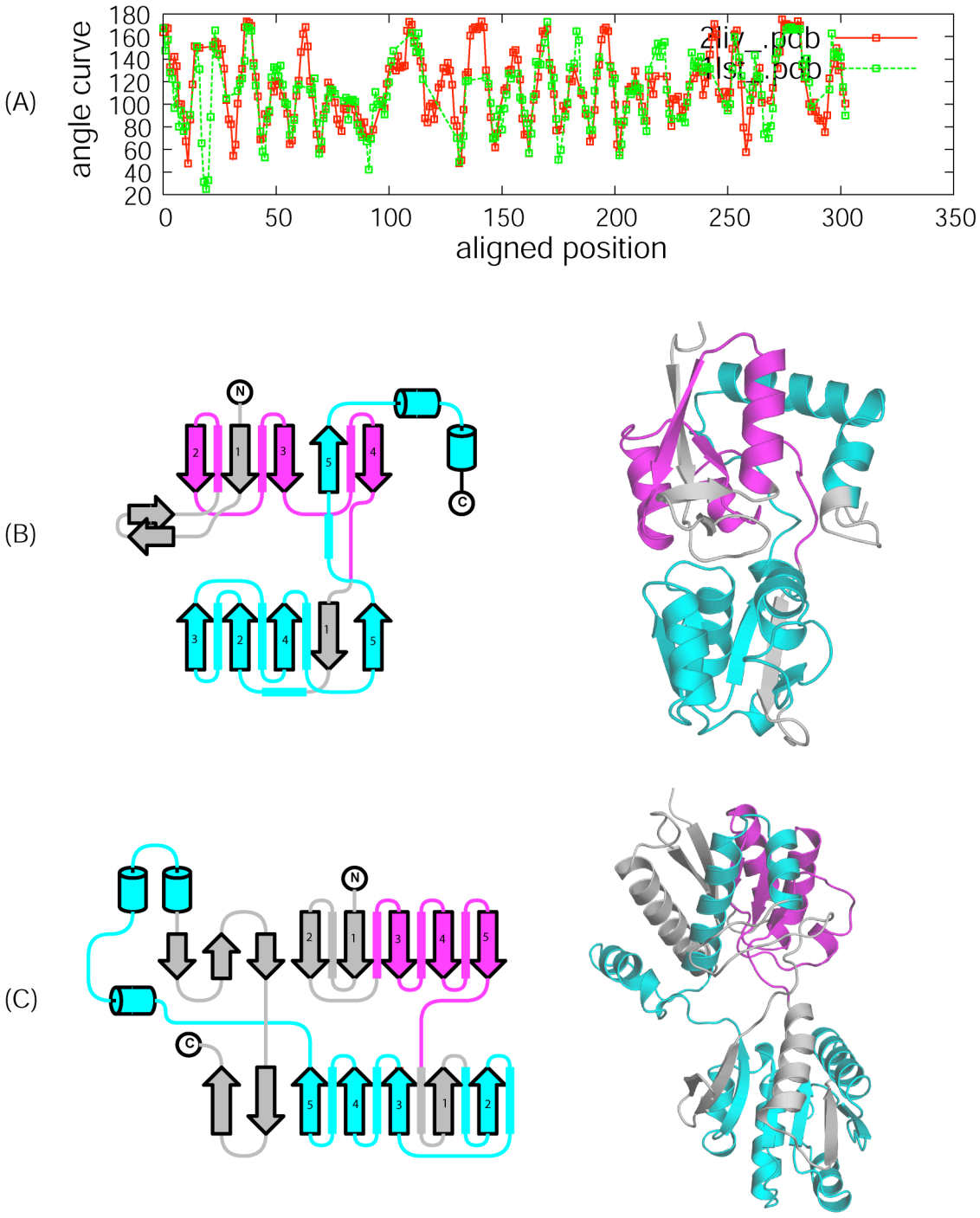



**Figure 6.6: Angle curve overlap graph (A) of structures 1lst (B) and 2liv (C) produced by CURVE. Two similar regions are colored cyan and magenta. The topology graphs are drawn by TOPDRAW [131], and the α–helices in an a/b unit are indicated by thick lines.**

# 7 Conclusions and future directions

With the availability of an increasing amount of biological sequences with limited annotations, bioinformaticians face the grand challenge of developing principled approaches to facilitate large-scale automatic annotation. In particular, since many biological sequences exhibit a mosaic arrangement of elements, the discovery and analysis of the architecture of these mosaic arrangements is a fundamental problem in bioinformatics. This dissertation presented a graph-theoretic framework for the analysis of mosaic arrangements in biological sequences and structures.

In Chapter 3, we developed a new approach to the classical multiple sequence alignment problem based on the A-Bruijn graph. Unlike previous approaches in which multiple alignments are represented in a row-column format or a directed acyclic graph format, our approach represents multiple alignments as a directed graph that may contain cycles. This new approach enables us to identify shared subsequences even though they may appear in different orders among input sequences. We have shown that it can correctly identify repeated and shuffled domain organizations among a set of protein sequences or a set of genomic sequences. We made the ABA software freely available at [103] and [104].

In Chapter 4, we presented a new method for the analysis of repeat families via the construction of repeat domain graphs of repeat family libraries. Our method enables a systematic identification of repeat domains and composite repeats, and is proven to be useful in assisting the elucidation of evolutionary history of repeats and

the annotation of *de novo* generated repeat libraries. We made the software available at [92].

In Chapter 5, we explored the discovery and analysis of mosaic arrangements in protein structures, which inspired the development of a new method for structure comparison presented in Chapter 6. Our structure comparison method is based on a generalized representation of protein structures. Unlike traditional structure alignment programs that typically optimize a measure based on RMSD, our method optimizes a newly defined sum-turning-angle-deviation measure. Our results demonstrate a good agreement with the traditional structure alignment programs and the structural similarity defined by SCOP. More importantly, our method can recognize structure similarities that involve extensive conformation changes, which are beyond the recognition of traditional structure alignment programs. We made this program accessible at http://pops.burnham.org/curve.

While future directions pertinent to individual chapters have been discussed in the corresponding chapters, I will end this dissertation by a discussion on several potential additional applications of the A-Bruijn graph framework, and directions along which improvements of the current A-Bruijn graph method could be achieved.

The A-Bruijn graph approach is best suited for the annotation of a set of biological sequences with little or no annotations. Large scale sequencing projects has generated a large amount of genomic sequence information. For the newly sequenced genomes, *de novo* protein domain identification [28] is an important task for *de novo* function annotation.

The deluge of data from various environmental genome shotgun sequencing projects [132-134] presents a tremendous challenge to practical MSA algorithms for handling a large number of sequences. Currently, fast clustering programs such as CD-HIT [135, 136] are the only tools available for the elucidation of the sequence similarities among these sequences. However, these programs are based on global sequence similarities and thus overlook the domain structures. A-Bruijn graph approach would provide a more detailed analysis of such dataset.

Another possible dataset for the application of A-Bruijn graph is the ultra-conserved non-coding sequences across large evolutionary distances [137]. Experimental and computational [138] analyses are being conducted to understand this dataset. Similar to CDHIT [135, 136], Bejerano et al. [138] clustered the sequences based on global sequence similarity. In addition, their algorithm contains an articulation step that cut a long region into shorten ones. However, the articulation step does not consider the domain structures inside these ultra-conserved sequences. The A-Bruijn graph approach would help elucidating any hidden mosaic structures inside this set of sequences.

The current A-Bruijn graph methods can be improved along two directions. First, applications to large datasets require the scaling up of the current A-Bruijn graph algorithms. In handling such large dataset, generally all pairwise comparisons between the sequences may be too time-consuming. As a result, alternative incremental approach for the construction of the A-Bruijn graph may be desired. Second, when there are a large number of similar subsequences in the input dataset, the boundaries of alignment blocks can be inconsistent and it may not be trivial to derive the true

boundary directly from the A-Bruijn graph. Even though one may derive a statistical test for the optimal placement of the boundaries, the optimization criteria should be motivated by the nature of the underlying biological problems.

# Appendix A: Additional information for ABA

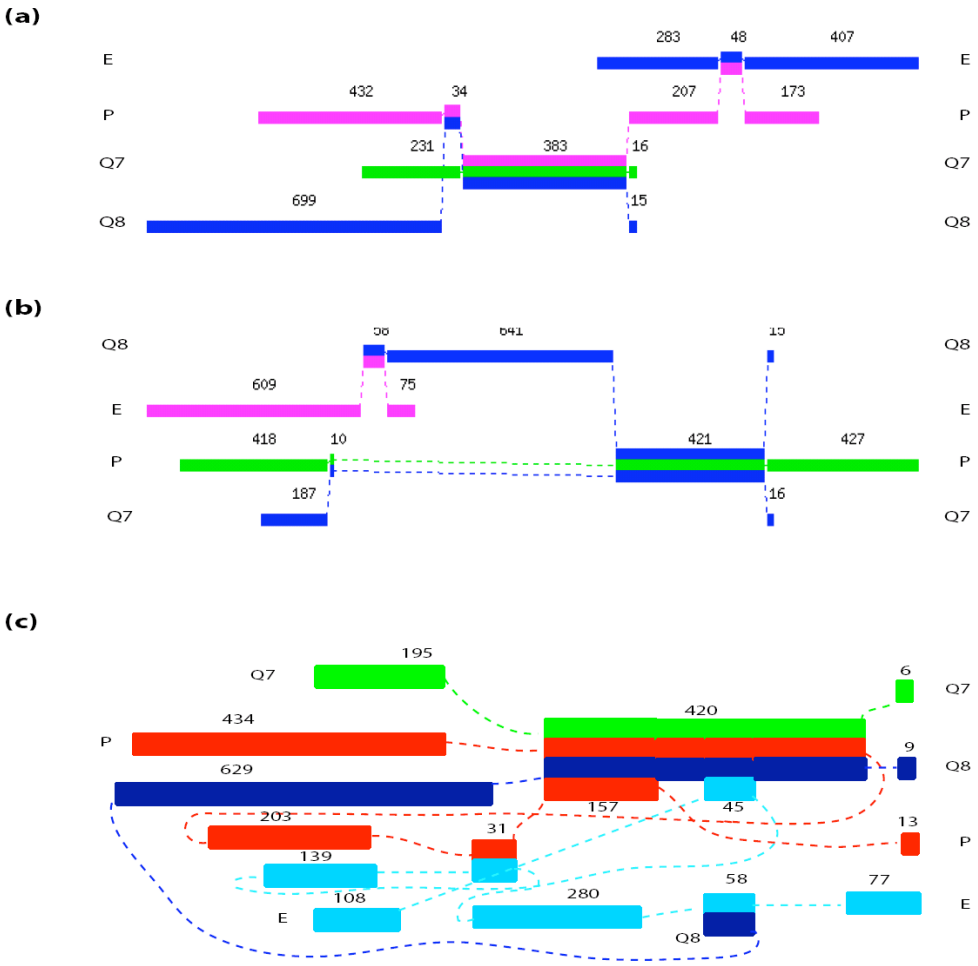



**Figure A.1: POA alignment of four proteins whose domain structure shown in Figure 3.7. These graphs are generated by the web-version of POA [32]. (a) The input order is: ETR1 ARATH, PHY2 SYNY3, Q7MD98, Q82U13. POA detected alignments No. 1 and part of alignment No. 3 in Table 3.1. (b) the input order is Q82U13, ETR1 ARATH, PHY2 SYNY3, Q7MD98. POA detected alignments No. 1 and alignment No. 2 in Table 3.1. (c) Schematic of ABA alignment in the POA visualization format [139].**

**Table A.1: SWISSPROT ID's of proteins in Figure 3.8.**

| Vertex Label | SwissProt ID |
|---|---|
| 1 | HMP2_YEREN |
| 2 | ENTF_ECOLI |
| 3 | PPS1_BACSU |
| 4 | SRF1_BACSU |
| 5 | ACVS_NOCLA |
| 6 | ACVS_PENCH |
| 7 | ACVS_EMENI |
| 8 | ACVS_CEPAC |
| 9 | GRSB_BACBR |
| 10 | PPS2_BACSU |
| 11 | SRF2_BACSU |
| 12 | SRF3_BACSU |
| 13 | PPS3_BACSU |
| 14 | HTS1_COCCA |
| 15 | GRSA_BACBR |
| 16 | TYCA_BREPA |

**Figure A.2: We apply ABA with pairwise BLASTZ alignments of the human, mouse, and rat sequences from the NISC T1 region (discussed in [140]) as input. Here, only the direct strands are glued together and only long alignments are shown. This criterion excludes alignment of some short exons. The 3 direct strands (+) and the 3 reverse strands (-) form 2 connected components in the graph, (anti-)symmetric to each other. Labeling of the nodes indicates the sequence (h=human, m=mouse, r=rat), strand (+ or -) and location in the sequence. For example, "m+:0;" marks the beginning of mouse direct strand; "m+:19442;r+:34977" is a result of gluing position 19442 on mouse direct strand and position 34977 on rat direct strand. The green path in the graph is the human sequence. Long, syntenic regions among the three genomes correspond to edges of multiplicity 3. Mouse and rat share some rodent-specific syntenic regions (e.g., edge 5930(2)). The ABA graph contains two directed cycles: 213(6) -> 30(3) and 172(6) -> 150(3). The former corresponds to a repeat of an ancient alu (aluJb in human, alu B1F in mouse). The latter appears to be an artifact that can be eliminated by different parameter settings.**

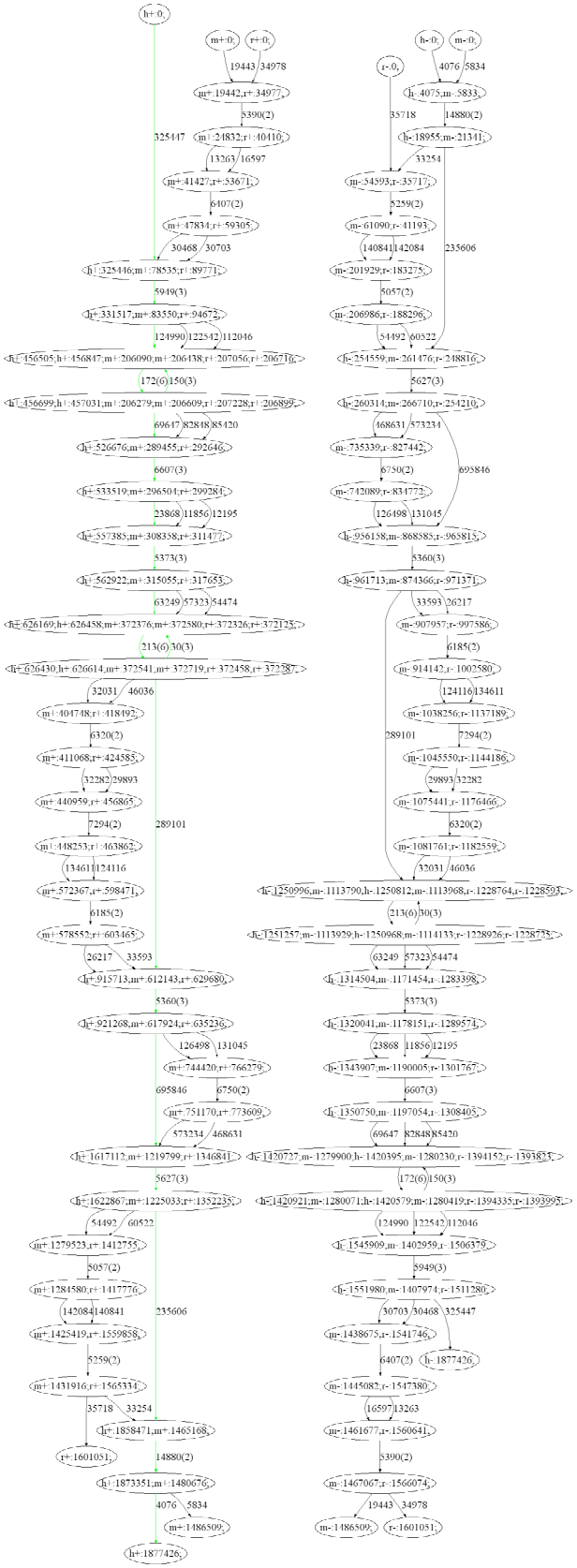
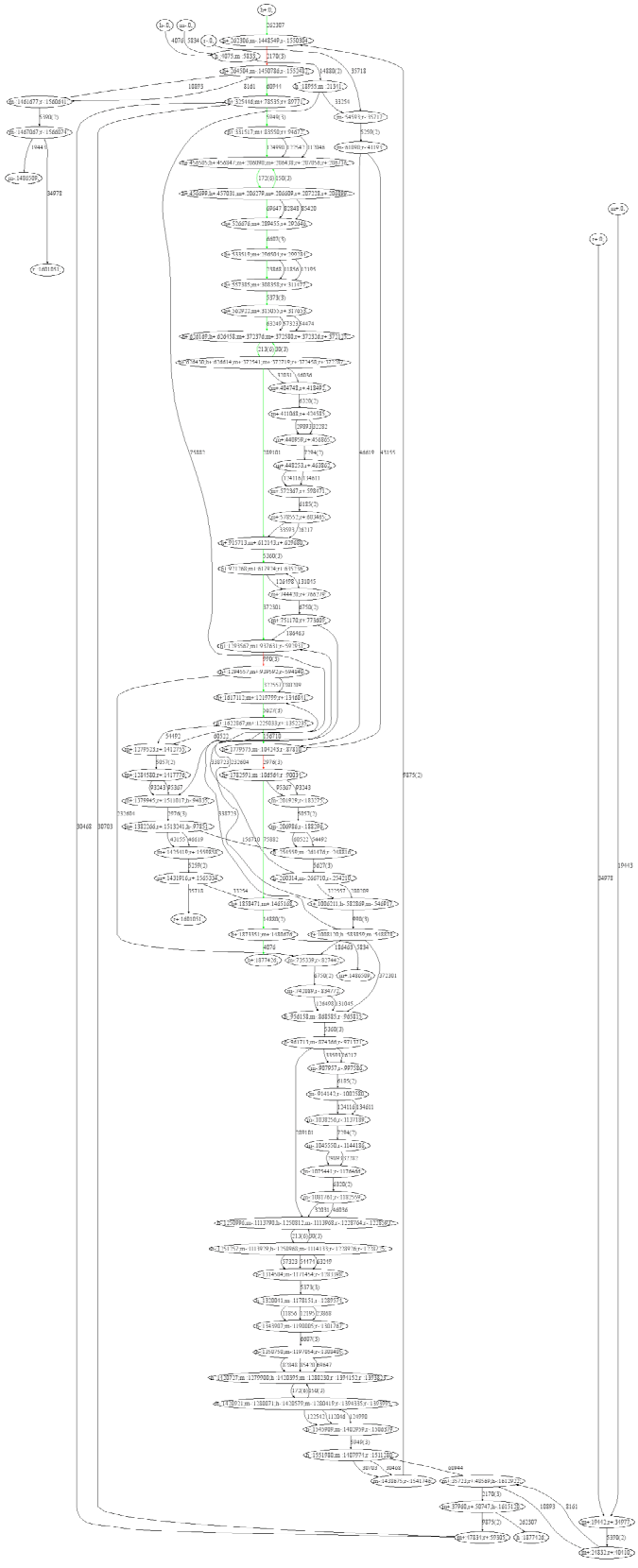

h+ 0;
m+ 0;
r+ 0;
h- 0;
m- 0;
r- 0;
19443
34978
4076
5834
m+ 19442,r+ 34977;
h- 4075,m- 5833;
5390(2)
35718
14880(2)
325447
m+ 24832,r+ 40410;
h- 18955,m- 21341;
13263
16597
33254
m+ 41427,r+ 53671;
m- 54593,r- 35717;
6407(2)
5259(2)
m+ 47834,r+ 59305;
m- 61090,r- 41193;
30468
30703
140841
142084
235606
h+ 325446,m+ 78535,r+ 89771;
m- 201929,r- 183275;
5949(3)
5057(2)
h+ 331517,m+ 83550,r+ 94672;
m- 206986,r- 188296;
124990
122542
112046
54492
60522
h+ 456505,h+ 456847,m+ 206090,m+ 206438,r+ 207056,r+ 206710;
h- 254559,m- 261476,r- 248816;
172(6)
150(3)
5627(3)
h+ 456699,h+ 457031,m+ 206279,m+ 206609,r+ 207228,r+ 206899;
h- 260314,m- 266710,r- 254210;
69647
82848
85420
468631
573234
h+ 526676,m+ 289455,r+ 292646;
m- 735339,r- 827442;
6607(3)
6750(2)
695846
h+ 533519,m+ 296504,r+ 299284;
m- 742089,r- 834772;
23868
11856
12195
126498
131045
h+ 557385,m+ 308358,r+ 311477;
h- 956158,m- 868585,r- 963815;
5373(3)
5360(3)
h+ 562922,m+ 315055,r+ 317653;
h- 961713,m- 874366,r- 971371;
63249
57323
54474
33593
26217
h+ 626169,h+ 626458,m+ 372376,m+ 372580,r+ 372326,r+ 372125;
m- 907957,r- 997586;
213(6)
30(3)
6185(2)
h+ 626430,h+ 626614,m+ 372541,m+ 372719,r+ 372458,r+ 372287;
m- 914142,r- 1002580;
32031
46036
124116
134611
m+ 404748,r+ 418492;
m- 1038256,r- 1137189;
289101
6320(2)
7294(2)
m+ 411068,r+ 424585;
m- 1045550,r- 1144186;
32282
29893
m+ 440959,r+ 456865;
m- 1075441,r- 1176466;
m+ 448253,r+ 463862;
m- 1081761,r- 1182559;
13461
12411
6
h- 1250996,m- 1113790,h- 1250812,m- 1113968,r- 1228764,r- 1228593;
m+ 572367,r+ 598471;
h- 1251257,m- 1113929,h- 1250968,m- 1114133,r- 1228926,r- 1228753;
m+ 578552,r+ 603465;
h+ 915713,m+ 612143,r+ 629680;
h- 1314504,m- 1171454,r- 1288398;
h+ 921268,m+ 617924,r+ 635236;
h- 1320041,m- 1178151,r- 1289574;
m+ 744420,r+ 766279;
h- 1343907,m- 1190005,r- 1301767;
m+ 751170,r+ 773609;
h- 1350750,m- 1197054,r- 1308405;
h+ 1617112,m+ 1219799,r+ 1346841;
h- 1420727,m- 1279900,h- 1420395,m- 1280230,r- 1394152,r- 1393823;
h+ 1622867,m+ 1225033,r+ 1352235;
h- 1420921,m- 1280071,h- 1420579,m- 1280419,r- 1394335,r- 1393995;
m+ 1279523,r+ 1412755;
h- 1545909,m- 1402959,r- 1506379;
m+ 1284580,r+ 1417776;
h- 1551980,m- 1407974,r- 1511280;
142084
140841
235606
30703
30468
325447
m+ 1425419,r+ 1559858;
m- 1438675,r- 1541746;
h- 1877426;
m+ 1431916,r+ 1565334;
m- 1445082,r- 1547380;
r+ 1858471,m+ 1465168;
m- 1461677,r- 1560641;
r+ 1601051;
h+ 1873351,m+ 1480676;
m- 1467067,r- 1566074;
m+ 1486509;
r- 1601051;
h+ 1877426;

**Figure A.3: We apply ABA to the same human, mouse, rat alignments in Figure A.2, but now glue both direct strands and reverse strands. Only long alignments are shown. The ABA graph is more complex than the graph in Figure A.2 due to the tangling between direct strands and the reverse strands. Three alignments between the direct strand of one species and the reverse strand of other species are visible (colored red). Visual inspection in the UCSC genome browser [141] confirm that these alignments are indeed short inversions of length 1-3 kb.**



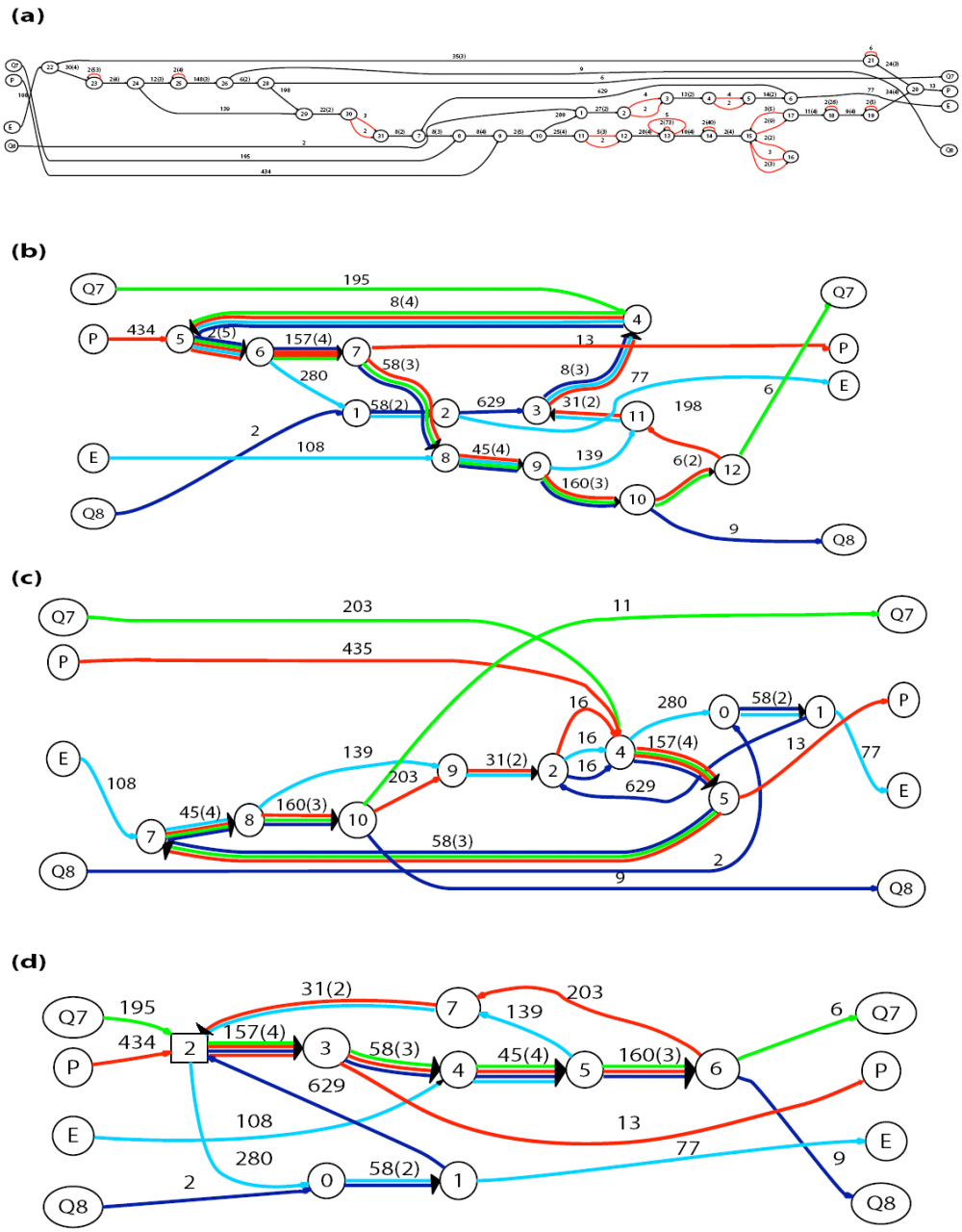



**Figure A.4: Construction of the ABA graph. The input sequences are ETR1 ARATH, PHY2 SYNY3, Q7MD98, Q82U13 from SwissProt [18]. (a) The A-Bruijn graph. In this simple example, only a few bulges and whirls (shown by red edges) are present in the A-Bruijn graph. Long genomic sequences and lower similarity typically produce much more complex A-Bruijn graphs. ABA performs a sequence of steps to simplify the A-Bruijn graph resulting in the intermediate graphs: (b) after bulge/whirl removal; (c) after re-threading; (d) after short edge removal – the ABA graph.**

# Appendix B: Additional information for the analysis of repeat domain graphs

**Table B.1: Repeat families containing domains shared with repeat families of different biological origin. The list is sorted by the total length of shared domains.**

| repeat family | #domains | length of domains | percent shared |
|---|---|---|---|
| HARLEQUIN | 34 | 6245 | 0.9 |
| MER52AI | 32 | 5375 | 0.76 |
| HERVL | 4 | 5117 | 0.9 |
| ERVL | 4 | 5117 | 0.89 |
| HUERS-P3 | 31 | 5106 | 0.57 |
| LOR1I | 54 | 4034 | 0.5 |
| HUERS-P3B | 67 | 3791 | 0.51 |
| HERVE | 9 | 3463 | 0.44 |
| HERVG25 | 50 | 3431 | 0.49 |
| HERV35I | 54 | 2933 | 0.42 |
| MER51I | 45 | 2914 | 0.37 |
| MER4I | 48 | 2896 | 0.45 |
| HERVIP10FH | 25 | 2782 | 0.54 |
| PABL_AI | 52 | 2665 | 0.53 |
| MER61I | 51 | 2485 | 0.48 |
| MER57I | 38 | 2376 | 0.31 |
| HERV49I | 43 | 2146 | 0.34 |
| MER4BI | 28 | 2113 | 0.31 |
| HERV19I | 43 | 2108 | 0.38 |
| MER41I | 40 | 2060 | 0.52 |
| HERVIP10F | 10 | 1715 | 0.22 |
| MSTAR | 12 | 1514 | 0.91 |
| THE1BR | 10 | 1470 | 0.92 |
| HUERS-P2 | 13 | 1413 | 0.46 |
| HSTC2 | 7 | 1298 | 0.68 |
| MER65I | 23 | 1196 | 0.25 |
| MER50I | 31 | 1185 | 0.16 |
| MER83BI | 20 | 1152 | 0.24 |
| TIGGER7 | 8 | 1126 | 0.45 |
| HERVI | 8 | 1090 | 0.14 |
| HERV17 | 9 | 1038 | 0.12 |
| HERV39 | 29 | 982 | 0.11 |
| HERV57I | 18 | 968 | 0.18 |
| PRIMA41 | 8 | 884 | 0.11 |

| repeat family | #domains | length of domains | percent shared |
|---|---|---|---|
| MER34B_I | 19 | 876 | 0.16 |
| MLT1AR | 11 | 848 | 0.49 |
| IN25 | 2 | 832 | 0.96 |
| L1PBB_5 | 2 | 832 | 0.87 |
| MER104B | 6 | 825 | 0.89 |
| MER21I | 4 | 813 | 0.19 |
| HERV23 | 22 | 793 | 0.16 |
| MER31I | 21 | 764 | 0.15 |
| HERV38I | 14 | 744 | 0.41 |
| MER57A_I | 18 | 737 | 0.12 |
| MER44C | 5 | 732 | 0.99 |
| MER104C | 4 | 723 | 0.98 |
| MER44D | 6 | 707 | 0.99 |
| MLT1R | 10 | 705 | 0.52 |
| MLT1CR | 10 | 705 | 0.51 |
| LTR12C | 5 | 668 | 0.42 |
| MER104A | 5 | 663 | 0.88 |
| LTR12E | 4 | 646 | 0.48 |
| PTR5 | 2 | 602 | 0.94 |
| MER44B | 5 | 550 | 0.98 |
| MER1A | 3 | 528 | 0.98 |
| CHARLIE3 | 3 | 528 | 0.19 |
| MER80 | 2 | 508 | 0.98 |
| CHARLIE4 | 2 | 508 | 0.26 |
| HERV30I | 5 | 455 | 0.06 |
| HERV9 | 5 | 455 | 0.05 |
| RICKSHA | 7 | 429 | 0.21 |
| MER28 | 2 | 427 | 0.96 |
| TIGGER2 | 2 | 427 | 0.16 |
| MER4D | 6 | 414 | 0.47 |
| MER4D1 | 6 | 414 | 0.45 |
| LTR5 | 3 | 399 | 0.41 |
| SVA | 3 | 399 | 0.24 |
| MER66I | 6 | 399 | 0.06 |
| MER61C | 9 | 390 | 0.88 |
| LTR20 | 9 | 390 | 0.75 |
| LTR8 | 5 | 382 | 0.54 |
| CHARLIE5 | 4 | 379 | 0.14 |
| MER1B | 2 | 363 | 1.05 |
| MER61B | 9 | 347 | 0.8 |
| MER44A | 4 | 337 | 0.97 |
| LTR12D | 3 | 330 | 0.26 |
| MER33 | 3 | 325 | 0.97 |
| MER5C | 1 | 324 | 0.97 |
| CHARLIE10 | 1 | 324 | 0.11 |
| MER4B | 5 | 310 | 0.5 |

| repeat family | #domains | length of domains | percent shared |
|---|---|---|---|
| HERV16 | 1 | 303 | 0.06 |
| HARLEQUINLTR | 2 | 302 | 0.64 |
| LTR2B | 2 | 302 | 0.6 |
| MER4E1 | 4 | 302 | 0.39 |
| MER4E | 4 | 302 | 0.39 |
| MLT2B3 | 6 | 298 | 0.41 |
| LTR49 | 4 | 292 | 0.48 |
| MLT2C2 | 5 | 268 | 0.58 |
| MLT2B2 | 5 | 268 | 0.52 |
| MLT2B4 | 5 | 268 | 0.47 |
| MER72 | 4 | 268 | 0.36 |
| MER4A | 3 | 262 | 0.39 |
| LTR48 | 5 | 251 | 0.31 |
| HERV3 | 4 | 239 | 0.03 |
| MLT2D | 4 | 230 | 0.58 |
| PRIMA4_LTR | 3 | 223 | 0.37 |
| LTR2 | 1 | 220 | 0.48 |
| LTR2C | 1 | 220 | 0.47 |
| MER112 | 2 | 217 | 0.75 |
| CHARLIE9 | 2 | 217 | 0.08 |
| PRIMA4_I | 2 | 217 | 0.03 |
| MER83AI | 4 | 211 | 0.05 |
| MER46 | 4 | 206 | 0.84 |
| ZOMBI_A | 4 | 206 | 0.84 |
| MER72B | 3 | 205 | 0.26 |
| MER115 | 2 | 204 | 0.29 |
| ZAPHOD | 2 | 204 | 0.05 |
| MER61 | 4 | 198 | 0.54 |
| MER49 | 3 | 195 | 0.21 |
| MER4A1 | 2 | 193 | 0.4 |
| LTR27B | 5 | 189 | 0.31 |
| HERVP71A_I | 3 | 186 | 0.02 |
| MER67C | 1 | 183 | 0.25 |
| LTR70 | 1 | 183 | 0.14 |
| MER104 | 2 | 178 | 0.94 |
| MER39B | 3 | 178 | 0.27 |
| LTR59 | 3 | 175 | 0.29 |
| L1PA17_5 | 1 | 169 | 0.1 |
| LTR20B | 4 | 165 | 0.28 |
| MER3 | 4 | 162 | 0.74 |
| ZOMBI_B | 3 | 160 | 0.33 |
| LTR48B | 3 | 160 | 0.23 |
| ZOMBI | 3 | 160 | 0.06 |
| HAL1C | 1 | 158 | 0.08 |
| L1MC5 | 1 | 158 | 0.07 |
| MSTB | 5 | 156 | 0.36 |

| repeat family | #domains | length of domains | percent shared |
|---|---|---|---|
| MSTA | 5 | 156 | 0.36 |
| MSTB1 | 5 | 156 | 0.35 |
| MSTA1 | 5 | 156 | 0.33 |
| HERV15I | 3 | 154 | 0.02 |
| HERV-K14I | 1 | 141 | 0.02 |
| MER39 | 3 | 139 | 0.19 |
| LTR27 | 4 | 138 | 0.21 |
| LTR28 | 4 | 138 | 0.13 |
| MER52C | 4 | 138 | 0.11 |
| MER52A | 4 | 138 | 0.08 |
| MER52B | 4 | 138 | 0.08 |
| MER52D | 4 | 138 | 0.06 |
| LTR29 | 2 | 136 | 0.22 |
| MLT1FR | 2 | 128 | 0.12 |
| L3 | 2 | 125 | 0.03 |
| MSTD | 4 | 124 | 0.31 |
| MLT1A1 | 4 | 124 | 0.3 |
| MSTC | 4 | 124 | 0.28 |
| MSTA2 | 4 | 124 | 0.27 |
| LTR39 | 2 | 122 | 0.15 |
| MLT2A1 | 2 | 116 | 0.26 |
| MLT2A2 | 2 | 116 | 0.21 |
| PABL_BI | 2 | 106 | 0.01 |
| HERVL68 | 1 | 105 | 0.03 |
| LTR31 | 1 | 93 | 0.15 |
| MLT1A | 3 | 91 | 0.24 |
| THE1B | 3 | 88 | 0.24 |
| THE1A | 3 | 88 | 0.24 |
| THE1C | 3 | 88 | 0.23 |
| THE1D | 3 | 88 | 0.23 |
| LTR8A | 2 | 88 | 0.12 |
| LTR54B | 1 | 85 | 0.17 |
| LTR54 | 1 | 85 | 0.16 |
| LTR51 | 1 | 83 | 0.12 |
| LTR77 | 2 | 76 | 0.12 |
| MLT1G | 3 | 75 | 0.14 |
| MLT1F1 | 3 | 75 | 0.13 |
| MLT1H1 | 3 | 75 | 0.13 |
| MLT1G2 | 3 | 75 | 0.12 |
| LTR1C | 2 | 74 | 0.11 |
| LTR1B | 2 | 74 | 0.09 |
| LTR1D | 2 | 74 | 0.07 |
| CR1_HS | 1 | 71 | 0.15 |
| HERVH48I | 1 | 67 | 0.01 |
| HERVKC4 | 2 | 60 | 0.01 |
| MER8 | 1 | 59 | 0.24 |

| repeat family | #domains | length of domains | percent shared |
|---|---|---|---|
| MER94 | 1 | 57 | 0.4 |
| BLACKJACK | 1 | 57 | 0.03 |
| MLT1B | 2 | 56 | 0.14 |
| MLT1C | 2 | 56 | 0.12 |
| MER4C | 1 | 55 | 0.12 |
| MIR3 | 1 | 54 | 0.23 |
| MLT1G3 | 2 | 53 | 0.1 |
| MLT1G1 | 2 | 53 | 0.09 |
| MLT1H | 2 | 53 | 0.09 |
| MER84I | 1 | 53 | 0.01 |
| MIR | 1 | 51 | 0.19 |
| L2B | 1 | 51 | 0.12 |
| MER34C | 1 | 51 | 0.09 |
| L1MD3 | 2 | 51 | 0.04 |
| L2A | 1 | 51 | 0.02 |
| FORDPREFECT_A | 1 | 49 | 0.09 |
| FORDPREFECT | 1 | 49 | 0.03 |
| MER92B | 1 | 47 | 0.07 |
| HERV70_I | 1 | 47 | 0.01 |
| MER41F | 1 | 43 | 0.11 |
| MER41E | 1 | 43 | 0.08 |
| LTR73 | 1 | 43 | 0.07 |
| MER41G | 1 | 43 | 0.05 |
| LTR5B | 1 | 40 | 0.04 |
| L1M2C_5 | 1 | 40 | 0.01 |
| LTR35 | 1 | 39 | 0.06 |
| MER21 | 1 | 39 | 0.04 |
| MER21A | 1 | 39 | 0.04 |
| LTR20C | 1 | 32 | 0.05 |
| L1MC3 | 1 | 29 | 0.01 |
| PRIMAX_I | 1 | 25 | 0.01 |
| MLT1C1 | 1 | 24 | 0.05 |
| MLT1D | 1 | 24 | 0.05 |
| MLT1H2 | 1 | 24 | 0.05 |
| MLT1F2 | 1 | 24 | 0.05 |
| MLT1F | 1 | 24 | 0.04 |
| MLT1E2 | 1 | 24 | 0.04 |
| MLT1E1 | 1 | 24 | 0.04 |
| MLT1E | 1 | 24 | 0.04 |
| MLT1E1A | 1 | 24 | 0.03 |
| R66 | 1 | 22 | 0.29 |
| LTR12 | 1 | 22 | 0.03 |
| LTR12B | 1 | 22 | 0.03 |

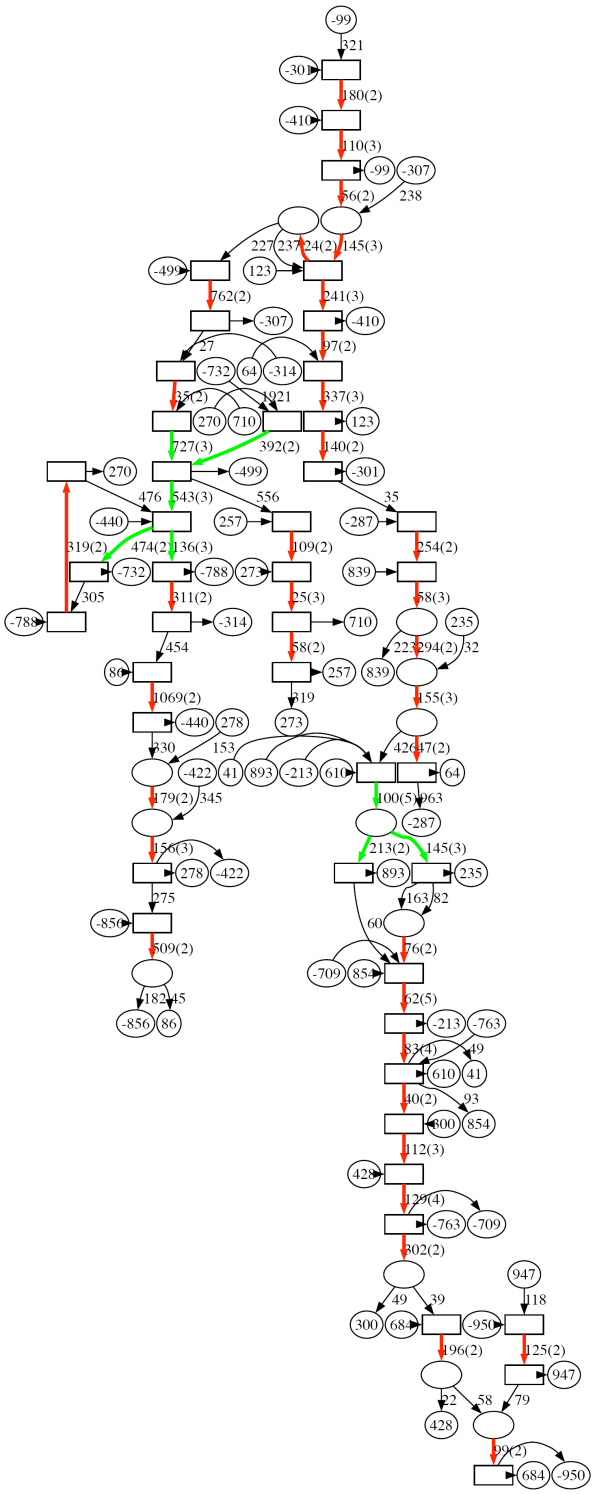



**Figure B.3: A connected component of human RepeatScout library repeat domain graph. This connected component contain 3 Y-forks (green). Two of them are associated with the Harlequin repeat family. (See Figure 4.7).**

**Section B.4: Comparative repeat domain graph analysis of mouse and rat RepeatScout libraries**

While *C. elegans* and *C. briggsae* repeat families have few similarities, mouse and rat repeat families are quite similar, due to the shorter time since speciation. In fact, Repbase Update [30, 83] contains only a single library of rodent repeat families. This library was largely constructed from manual curation of mouse repeat families. Due to the large size and repeat-rich nature of these and other mammalian genomes, *de novo* construction of repeat libraries from genomic sequence is a daunting algorithmic problem. Only recently were separate repeat libraries for mouse and rat generated using RepeatScout [98], enabling a comparison of mouse and rat repeat families.

The input mouse library contains 886 sequences of total length 1.2M bp and the input rat library contains 831 sequences of total length 0.5M bp. We generated a comparative repeat domain graph using our graph-based method. The resulting graph contained a large connected component that contains 59% of the input sequences. Upon close inspection, we found that this large component was connected by a small number of long edges of single multiplicity. As in the analysis of human RepeatScout library (see Section 2.5), we determined that these long edges represented tandem duplications and used the same procedure to remove them. 261 mouse entries and 150 rat entries in the RepeatScout are thus classified as tandem duplications and excluded from further analysis.

We build the comparative repeat domain graph with the remaining entries by selecting all pairwise alignments between the entries with a score cutoff 30. The graph

contains 9446 edges, of which 6598 edges lie in the largest connected component. The largest component contains many edges that match to subparts of the LINE element L1 as well edges that match to some LTRs and SINEs. In order to analyze the long domains in the L1 repeat family, we used a more strict score cutoff of 350 for pairwise alignments, resulting a graph containing smaller number of edges (Table B.4.1).

The largest component (Figure B.4.1) of the resulting mouse/rat comparative repeat domain graph contains 53 sequences, 23 from mouse and 30 from rat. All these entries are related to L1 (with the entire or a significant portion of the sequence aligned to L1).

To identify prominent L1 repeat domains, we weight each edge in the repeat domain graph by the number of times the repeat domain is present in the genome; i.e., we assign a copy number to each edge in the following way. We run RepeatMasker with the RepeatScout libraries against the mouse and rat X chromosome sequences. We create a counter for each edge. For each RepeatMasker hit matching to a sub-region of an entry in the RepeatScout libraries, we identify edges in the repeat domain graph corresponding to the matched sub-region, and increment the counters for these edges.

We observe a number of features in the graph that correspond to known knowledge to L1.

1. The topology of the graph contains only one main path, suggesting that there is mainly one repeat family in this component. Indeed all but two edges in the graph match to subparts of L1.

2. Edge copy numbers on the 3' end (~13,000) are significantly higher than that of the 5' end (~5,000), reflecting the fact that there are frequent 5' end truncations of L1 after insertion.
3. The tree-like structure at the 5' end has a mouse specific branch and a rat specific branch, reflecting that L1 underwent significant expansions after the speciation.
4. The tree-like structure at the 3' end split into three major "sub-trees". One is mouse-specific, one is rat-specific, and one is shared across the two species. The shared "sub-tree" corresponds to the Lx subfamilies, which predate the speciation.
5. There is a set of multi-parallel edges close to the 5' end, representing a region of high variability. Interestingly, the lengths of these edges differ by some multiple of 3. Indeed this region is inside the ORF1 coding region.
6. There are two major mouse specific branches at the 5' end, each contains a short directed cycle consisting of edges with high copy numbers. The two branches correspond to the A-type and the F-type of L1Md subfamilies, which are known to contain distinct domains of tandem repeats at their 5' region that may be serve as alternative promoters [142].
7. There are other directed cycles in this component which could correspond to other interesting features of L1 repeats.

We remark that the graph provides a principled way to extract biological knowledge out of a *de novo* constructed repeat library. During the graph construction

we use zero prior knowledge on the repeat content in mouse or rat. The above results demonstrate the potential of application the repeat decomposition graph approach to post-processing *de novo* repeat libraries: purging tandem duplications, family/subfamily classification of *de novo* repeat families, and gaining biological insights.

**Table B.4.1 The average edge length and multiplicity of mouse/rat comparative repeat domain graph constructed from their RepeatScout libraries.**

| genome | seqs. | # edges | avg. edge len | avg. edge mul |
|---|---|---|---|---|
| Mouse only | 541 | 1542 | 298.9 | 1.2 |
| Rat only | 596 | 1762 | 256.7 | 1.2 |
| both | 85 | 244 | 148.8 | 6.8 |

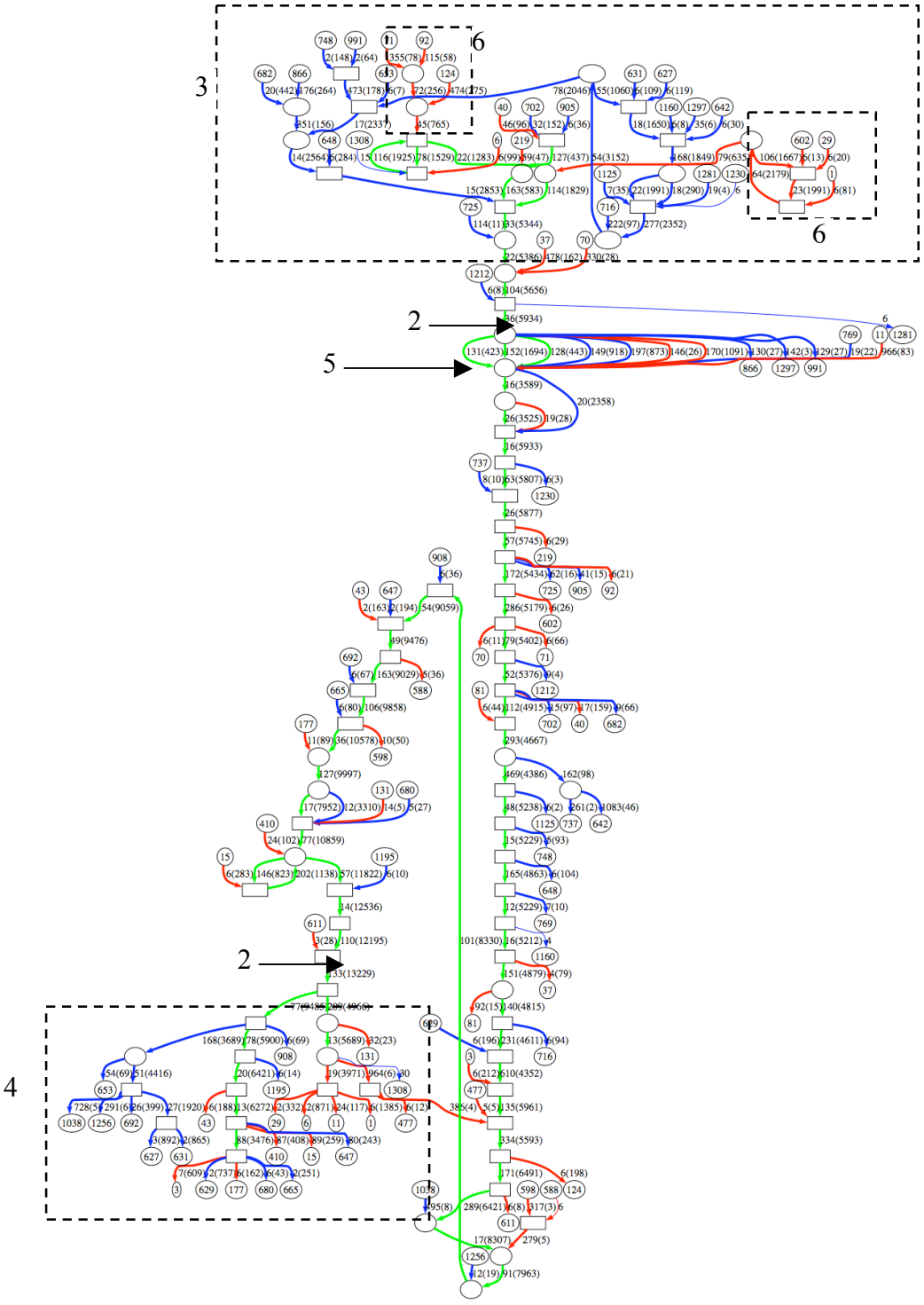



**Figure B.4.1: Largest connected component of mouse/rat comparative repeat domain graph. Each edge is labeled *l*(*c*), where *l* is the length and *c* is the copy number of the sequence in the X chromosomes. Edges with length <10 are contracted. Sources (top of figure) are 5' ends of L1 sequences, while sinks (bottom) are 3' ends. Observations in text are highlighted with boxes and arrows with corresponding numbers.**

**Section B.5: New procedure for handling palindromic regions**

Here we give an example to illustrate the improvement on handling repeat families with palindromic structure. In the *C. briggsae* RECON library, the center part of the repeat family Cb000083 (Stein et al 2003 [93] annotated as AT_rich_Low_complexity___TC1_DNA/Tc1) is palindromic. Part of its sequence aligned with the repeat family Cb000214 (Stein et al 2003 [93] annotated as unknown). Without the new procedure for handling such palindromes, the repeat domain graph (Figure B.5.1 (a)) breaks at the middle of the palindromic sequence of Cb000083. The new procedure (Figure B.5.1(b)) brings them together to make a complete path.

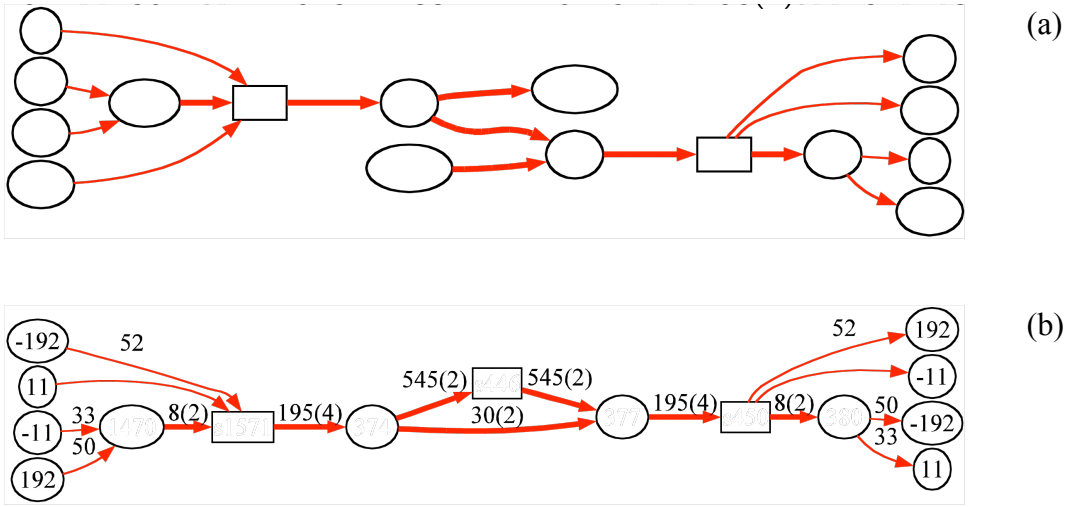



**Figure B.5.1: Handling self-palindromic sequences in the *C. briggsae* RECON library (as part of the *C. elegans*/*C. briggsae* comparative repeat domain graph). Sequences numbered 11 and 192 have corresponding RECON ID's of Cb000083, and Cb000214. (a) Direct use of ABA breaks the path for sequence 11 (and –11) at the nodes labeled 11/-11. (b) The new graph building procedure correctly contains a single path for sequence 11.**

## References

1. Lander, E.S., Linton, L.M., Birren, B., Nusbaum, C., Zody, M.C., Baldwin, J., Devon, K., Dewar, K., Doyle, M., FitzHugh, W., Funke, R., Gage, D., Harris, K., Heaford, A., Howland, J., Kann, L., Lehoczky, J., LeVine, R., McEwan, P., McKernan, K., Meldrim, J., Mesirov, J.P., Miranda, C., Morris, W., Naylor, J., Raymond, C., Rosetti, M., Santos, R., Sheridan, A., Sougnez, C., Stange-Thomann, N., Stojanovic, N., Subramanian, A., Wyman, D., Rogers, J., Sulston, J., Ainscough, R., Beck, S., Bentley, D., Burton, J., Clee, C., Carter, N., Coulson, A., Deadman, R., Deloukas, P., Dunham, A., Dunham, I., Durbin, R., French, L., Grafham, D., Gregory, S., Hubbard, T., Humphray, S., Hunt, A., Jones, M., Lloyd, C., McMurray, A., Matthews, L., Mercer, S., Milne, S., Mullikin, J.C., Mungall, A., Plumb, R., Ross, M., Shownkeen, R., Sims, S., Waterston, R.H., Wilson, R.K., Hillier, L.W., McPherson, J.D., Marra, M.A., Mardis, E.R., Fulton, L.A., Chinwalla, A.T., Pepin, K.H., Gish, W.R., Chissoe, S.L., Wendl, M.C., Delehaunty, K.D., Miner, T.L., Delehaunty, A., Kramer, J.B., Cook, L.L., Fulton, R.S., Johnson, D.L., Minx, P.J., Clifton, S.W., Hawkins, T., Branscomb, E., Predki, P., Richardson, P., Wenning, S., Slezak, T., Doggett, N., Cheng, J.F., Olsen, A., Lucas, S., Elkin, C., Uberbacher, E., Frazier, M., Gibbs, R.A., Muzny, D.M., Scherer, S.E., Bouck, J.B., Sodergren, E.J., Worley, K.C., Rives, C.M., Gorrell, J.H., Metzker, M.L., Naylor, S.L., Kucherlapati, R.S., Nelson, D.L., Weinstock, G.M., Sakaki, Y., Fujiyama, A., Hattori, M., Yada, T., Toyoda, A., Itoh, T., Kawagoe, C., Watanabe, H., Totoki, Y., Taylor, T., Weissenbach, J., Heilig, R., Saurin, W., Artiguenave, F., Brottier, P., Bruls, T., Pelletier, E., Robert, C., Wincker, P., Smith, D.R., Doucette-Stamm, L., Rubenfield, M., Weinstock, K., Lee, H.M., Dubois, J., Rosenthal, A., Platzer, M., Nyakatura, G., Taudien, S., Rump, A., Yang, H., Yu, J., Wang, J., Huang, G., Gu, J., Hood, L., Rowen, L., Madan, A., Qin, S., Davis, R.W., Federspiel, N.A., Abola, A.P., Proctor, M.J., Myers, R.M., Schmutz, J., Dickson, M., Grimwood, J., Cox, D.R., Olson, M.V., Kaul, R., Shimizu, N., Kawasaki, K., Minoshima, S., Evans, G.A., Athanasiou, M., Schultz, R., Roe, B.A., Chen, F., Pan, H., Ramser, J., Lehrach, H., Reinhardt, R., McCombie, W.R., de la Bastide, M., Dedhia, N., Blocker, H., Hornischer, K., Nordsiek, G., Agarwala, R., Aravind, L., Bailey, J.A., Bateman, A., Batzoglou, S., Birney, E., Bork, P., Brown, D.G., Burge, C.B., Cerutti, L., Chen, H.C., Church, D., Clamp, M., Copley, R.R., Doerks, T., Eddy, S.R., Eichler, E.E., Furey, T.S., Galagan, J., Gilbert, J.G., Harmon, C., Hayashizaki, Y., Haussler, D., Hermjakob, H., Hokamp, K., Jang, W., Johnson, L.S., Jones, T.A., Kasif, S., Kaspryzk, A., Kennedy, S., Kent, W.J., Kitts, P., Koonin, E.V., Korf, I., Kulp, D., Lancet, D., Lowe, T.M., McLysaght, A., Mikkelsen, T., Moran, J.V., Mulder, N., Pollara, V.J., Ponting, C.P., Schuler, G., Schultz, J., Slater, G., Smit, A.F., Stupka, E., Szustakowski, J., Thierry-Mieg, D., Thierry-Mieg, J., Wagner, L., Wallis, J., Wheeler, R., Williams, A., Wolf, Y.I., Wolfe, K.H., Yang, S.P., Yeh, R.F., Collins, F., Guyer, M.S., Peterson, J., Felsenfeld,

A., Wetterstrand, K.A., Patrinos, A., Morgan, M.J., de Jong, P., Catanese, J.J., Osoegawa, K., Shizuya, H., Choi, S. and Chen, Y.J., *Initial sequencing and analysis of the human genome.* Nature, 2001. **409**(6822): p. 860-921.

2. Eichler, E.E., *Recent duplication, domain accretion and the dynamic mutation of the human genome.* Trends Genet, 2001. **17**(11): p. 661-9.

3. Samonte, R.V. and Eichler, E.E., *Segmental duplications and the evolution of the primate genome.* Nat Rev Genet, 2002. **3**(1): p. 65-72.

4. Bailey, J.A., Baertsch, R., Kent, W.J., Haussler, D., and Eichler, E.E., *Hotspots of mammalian chromosomal evolution.* Genome Biol, 2004. **5**(4): p. R23.

5. Tuzun, E., Bailey, J.A., and Eichler, E.E., *Recent segmental duplications in the working draft assembly of the brown Norway rat.* Genome Res, 2004. **14**(4): p. 493-506.

6. Bailey, J.A., Church, D.M., Ventura, M., Rocchi, M., and Eichler, E.E., *Analysis of segmental duplications and genome assembly in the mouse.* Genome Res, 2004. **14**(5): p. 789-801.

7. Cheng, Z., Ventura, M., She, X., Khaitovich, P., Graves, T., Osoegawa, K., Church, D., DeJong, P., Wilson, R.K., Paabo, S., Rocchi, M., and Eichler, E.E., *A genome-wide comparison of recent chimpanzee and human segmental duplications.* Nature, 2005. **437**(7055): p. 88-93.

8. Lupski, J.R., *Genomic disorders: structural features of the genome can lead to DNA rearrangements and human disease traits.* Trends Genet, 1998. **14**(10): p. 417-22.

9. Stankiewicz, P. and Lupski, J.R., *Molecular-evolutionary mechanisms for genomic disorders.* Curr Opin Genet Dev, 2002. **12**(3): p. 312-9.

10. Stankiewicz, P. and Lupski, J.R., *Genome architecture, rearrangements and genomic disorders.* Trends Genet, 2002. **18**(2): p. 74-82.

11. Eichler, E.E., Budarf, M.L., Rocchi, M., Deaven, L.L., Doggett, N.A., Baldini, A., Nelson, D.L., and Mohrenweiser, H.W., *Interchromosomal duplications of the adrenoleukodystrophy locus: a phenomenon of pericentromeric plasticity.* Hum Mol Genet, 1997. **6**(7): p. 991-1002.

12. Jackson, M.S., Rocchi, M., Thompson, G., Hearn, T., Crosier, M., Guy, J., Kirk, D., Mulligan, L., Ricco, A., Piccininni, S., Marzella, R., Viggiano, L., and Archidiacono, N., *Sequences flanking the centromere of human chromosome 10 are a complex patchwork of arm-specific sequences, stable duplications and unstable sequences with homologies to telomeric and other centromeric locations.* Hum Mol Genet, 1999. **8**(2): p. 205-15.

13. Horvath, J.E., Gulden, C.L., Vallente, R.U., Eichler, M.Y., Ventura, M., McPherson, J.D., Graves, T.A., Wilson, R.K., Schwartz, S., Rocchi, M., and Eichler, E.E., *Punctuated duplication seeding events during the evolution of human chromosome 2p11.* Genome Res, 2005. **15**(7): p. 914-27.

14. Bailey, J.A., Yavor, A.M., Massa, H.F., Trask, B.J., and Eichler, E.E., *Segmental duplications: organization and impact within the current human genome project assembly.* Genome Res, 2001. **11**(6): p. 1005-17.

15. Bailey, J.A., Gu, Z., Clark, R.A., Reinert, K., Samonte, R.V., Schwartz, S., Adams, M.D., Myers, E.W., Li, P.W., and Eichler, E.E., *Recent segmental duplications in the human genome.* Science, 2002. **297**(5583): p. 1003-7.

16. She, X., Jiang, Z., Clark, R.A., Liu, G., Cheng, Z., Tuzun, E., Church, D.M., Sutton, G., Halpern, A.L., and Eichler, E.E., *Shotgun sequence assembly and recent segmental duplications within the human genome.* Nature, 2004. **431**(7011): p. 927-30.

17. Jiang, Z., Tang, H., Tuzun, E., Pevzner, P., and Eichler, E.E., *Reconstructing evolutionary history of human segmental duplications.* in preparation, 2005.

18. Bateman, A., Coin, L., Durbin, R., Finn, R.D., Hollich, V., Griffiths-Jones, S., Khanna, A., Marshall, M., Moxon, S., Sonnhammer, E.L., Studholme, D.J., Yeats, C., and Eddy, S.R., *The Pfam protein families database.* Nucleic Acids Res, 2004. **32**(Database issue): p. D138-41.

19. Schultz, J., Milpetz, F., Bork, P., and Ponting, C.P., *SMART, a simple modular architecture research tool: identification of signaling domains.* Proc Natl Acad Sci U S A, 1998. **95**(11): p. 5857-64.

20. Servant, F., Bru, C., Carrere, S., Courcelle, E., Gouzy, J., Peyruc, D., and Kahn, D., *ProDom: automated clustering of homologous domains.* Brief Bioinform, 2002. **3**(3): p. 246-51.

21. Marchler-Bauer, A., Anderson, J.B., DeWeese-Scott, C., Fedorova, N.D., Geer, L.Y., He, S., Hurwitz, D.I., Jackson, J.D., Jacobs, A.R., Lanczycki, C.J., Liebert, C.A., Liu, C., Madej, T., Marchler, G.H., Mazumder, R., Nikolskaya, A.N., Panchenko, A.R., Rao, B.S., Shoemaker, B.A., Simonyan, V., Song, J.S., Thiessen, P.A., Vasudevan, S., Wang, Y., Yamashita, R.A., Yin, J.J., and Bryant, S.H., *CDD: a curated Entrez database of conserved domain alignments.* Nucleic Acids Res, 2003. **31**(1): p. 383-7.

22. Mulder, N.J., Apweiler, R., Attwood, T.K., Bairoch, A., Barrell, D., Bateman, A., Binns, D., Biswas, M., Bradley, P., Bork, P., Bucher, P., Copley, R.R., Courcelle, E., Das, U., Durbin, R., Falquet, L., Fleischmann, W., Griffiths-Jones, S., Haft, D., Harte, N., Hulo, N., Kahn, D., Kanapin, A., Krestyaninova, M., Lopez, R., Letunic, I., Lonsdale, D., Silventoinen, V., Orchard, S.E., Pagni, M., Peyruc, D., Ponting, C.P., Selengut, J.D., Servant, F., Sigrist, C.J., Vaughan, R., and Zdobnov, E.M., *The InterPro Database, 2003 brings increased coverage and new features.* Nucleic Acids Res, 2003. **31**(1): p. 315-8.

23. Holm, L. and Sander, C., *Dictionary of recurrent domains in protein structures.* Proteins, 1998. **33**(1): p. 88-96.

24. Orengo, C.A., Michie, A.D., Jones, S., Jones, D.T., Swindells, M.B., and Thornton, J.M., *CATH--a hierarchic classification of protein domain structures.* Structure, 1997. **5**(8): p. 1093-108.

25. Murzin, A.G., Brenner, S.E., Hubbard, T., and Chothia, C., *SCOP: a structural classification of proteins database for the investigation of sequences and structures.* J Mol Biol, 1995. **247**(4): p. 536-40.

26. Wuchty, S., *Scale-free behavior in protein domain networks.* Mol Biol Evol, 2001. **18**(9): p. 1694-702.

27. Ye, Y. and Godzik, A., *Comparative analysis of protein domain organization.* Genome Res, 2004. **14**(3): p. 343-53.

28. Heger, A. and Holm, L., *Exhaustive enumeration of protein domain families.* J Mol Biol, 2003. **328**(3): p. 749-67.

29. Batzer, M.A. and Deininger, P.L., *Alu repeats and human genomic diversity.* Nat Rev Genet, 2002. **3**(5): p. 370-9.

30. Jurka, J., *Repbase update: a database and an electronic journal of repetitive elements.* Trends Genet, 2000. **16**(9): p. 418-20.

31. Stein, L.D., Bao, Z., Blasiar, D., Blumenthal, T., Brent, M.R., Chen, N., Chinwalla, A., Clarke, L., Clee, C., Coghlan, A., Coulson, A., D'Eustachio, P., Fitch, D.H., Fulton, L.A., Fulton, R.E., Griffiths-Jones, S., Harris, T.W., Hillier, L.W., Kamath, R., Kuwabara, P.E., Mardis, E.R., Marra, M.A., Miner, T.L., Minx, P., Mullikin, J.C., Plumb, R.W., Rogers, J., Schein, J.E., Sohrmann, M., Spieth, J., Stajich, J.E., Wei, C., Willey, D., Wilson, R.K., Durbin, R., and Waterston, R.H., *The genome sequence of Caenorhabditis briggsae: a platform for comparative genomics.* PLoS Biol, 2003. **1**(2): p. E45.

32. Lee, C., Grasso, C., and Sharlow, M.F., *Multiple sequence alignment using partial order graphs.* Bioinformatics, 2002. **18**(3): p. 452-64.

33. de Bruijn, N., *A combinatorial problem.* Koninklije Nedderlandse Academie van Wetenshappen Proc., 1946. **A49:**: p. 758–764.

34. Pevzner, P.A., *1-Tuple DNA sequencing: computer analysis.* J Biomol Struct Dyn, 1989. **7**(1): p. 63-73.

35. Pevzner, P.A., Tang, H., and Waterman, M.S., *An Eulerian path approach to DNA fragment assembly.* Proc Natl Acad Sci U S A, 2001. **98**(17): p. 9748-53.

36. Idury, R.M. and Waterman, M.S., *A new algorithm for DNA sequence assembly.* J Comput Biol, 1995. **2**(2): p. 291-306.

37. Pe'er, I., Arbili, N., and Shamir, R., *A computational method for resequencing long DNA targets by universal oligonucleotide arrays.* Proc Natl Acad Sci U S A, 2002. **99**(24): p. 15492-6.

38. Shamir, R. and Tsur, D., *Large scale sequencing by hybridization.* J Comput Biol, 2002. **9**(2): p. 413-28.

39. Heber, S., Alekseyev, M., Sze, S.H., Tang, H., and Pevzner, P.A., *Splicing graphs and EST assembly problem.* Bioinformatics, 2002. **18 Suppl 1**: p. S181-8.

40. Bocker, S. *Sequencing from compomers: Using mass spectrometry for DNA de-novo sequencing of 200+ nt*. in *3rd Workshop on Algorithms in Bioinformatics*. 2003.

41. Pevzner, P.A., Tang, H., and Tesler, G., *De novo repeat classification and fragment assembly.* Genome Res, 2004. **14**(9): p. 1786-96.

42. Morgenstern, B., Frech, K., Dress, A., and Werner, T., *DIALIGN: finding local similarities by multiple sequence alignment.* Bioinformatics, 1998. **14**(3): p. 290-4.

43. Sankoff, D., *Minimal mutation trees of sequences.* SIAM J. Appl Math, 1975. **28**: p. 35-42.

44. Waterman, M.S.a.S., T. F. and Beyer, W. A., *Some biological sequence metrics.* Advances in Math, 1976. **20**(3): p. 367-387.

45. Feng, D.F. and Doolittle, R.F., *Progressive sequence alignment as a prerequisite to correct phylogenetic trees.* J Mol Evol, 1987. **25**(4): p. 351-60.

46. Lipman, D.J., Altschul, S.F., and Kececioglu, J.D., *A tool for multiple sequence alignment.* Proc Natl Acad Sci U S A, 1989. **86**(12): p. 4412-5.

47. Higgins, D.G. and Sharp, P.M., *CLUSTAL: a package for performing multiple sequence alignment on a microcomputer.* Gene, 1988. **73**(1): p. 237-44.

48. Vingron, M. and Argos, P., *Motif recognition and alignment for many sequences by comparison of dot-matrices.* J Mol Biol, 1991. **218**(1): p. 33-43.

49. Thompson, J.D., Higgins, D.G., and Gibson, T.J., *CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice.* Nucleic Acids Res, 1994. **22**(22): p. 4673-80.

50. Kececioglu, J.D. *The maximum weight trace problem in multiple sequence alignment.* in *Combinatorial pattern matching*. 1993. Padova: Springer.

51. Schuler, G.D., Altschul, S.F., and Lipman, D.J., *A workbench for multiple alignment construction and analysis.* Proteins, 1991. **9**(3): p. 180-90.

52. Pei, J., Sadreyev, R., and Grishin, N.V., *PCMA: fast and accurate multiple sequence alignment based on profile consistency.* Bioinformatics, 2003. **19**(3): p. 427-8.

53. Schwartz, S., Elnitski, L., Li, M., Weirauch, M., Riemer, C., Smit, A., Green, E.D., Hardison, R.C., and Miller, W., *MultiPipMaker and supporting tools: Alignments and analysis of multiple genomic DNA sequences.* Nucleic Acids Res, 2003. **31**(13): p. 3518-24.

54. Darling, A.C., Mau, B., Blattner, F.R., and Perna, N.T., *Mauve: multiple alignment of conserved genomic sequence with rearrangements.* Genome Res, 2004. **14**(7): p. 1394-403.

55. Wang, L. and Jiang, T., *On the complexity of multiple sequence alignment.* J Comput Biol, 1994. **1**(4): p. 337-48.

56. Notredame, C., *Recent progress in multiple sequence alignment: a survey.* Pharmacogenomics, 2002. **3**(1): p. 131-44.

57. Notredame, C., Higgins, D.G., and Heringa, J., *T-Coffee: A novel method for fast and accurate multiple sequence alignment.* J Mol Biol, 2000. **302**(1): p. 205-17.

58. Doolittle, R.F., *The multiplicity of domains in proteins.* Annu Rev Biochem, 1995. **64**: p. 287-314.

59. Eddy, S.R., *Multiple-alignment and sequence searches.* Trends Guide to Bioinformatics, 1998: p. 15-18.

60. Neuwald, A.F., Liu, J.S., Lipman, D.J., and Lawrence, C.E., *Extracting protein alignment models from the sequence database.* Nucleic Acids Res, 1997. **25**(9): p. 1665-77.

61. Li, X. and Waterman, M.S., *Estimating the repeat structure and length of DNA sequences using L-tuples.* Genome Res, 2003. **13**(8): p. 1916-22.

62. Zhang, Y. and Waterman, M.S., *An Eulerian path approach to global multiple alignment for DNA sequences.* J Comput Biol, 2003. **10**(6): p. 803-19.

63. Blanchette, M., Kent, W.J., Riemer, C., Elnitski, L., Smit, A.F., Roskin, K.M., Baertsch, R., Rosenbloom, K., Clawson, H., Green, E.D., Haussler, D., and Miller, W., *Aligning multiple genomic sequences with the threaded blockset aligner.* Genome Res, 2004. **14**(4): p. 708-15.

64. Sammeth, M., Morgenstern, B., and Stoye, J., *Divide-and-conquer multiple alignment with segment-based constraints.* Bioinformatics, 2003. **19 Suppl 2**: p. II189-II195.

65. Gansner, E.R.a.N., S.C., *An open graph visualization system and its applications to software engineering*, http://www.research.att.com/sw/tools/graphviz/. 1999.

66. Boeckmann, B., Bairoch, A., Apweiler, R., Blatter, M.C., Estreicher, A., Gasteiger, E., Martin, M.J., Michoud, K., O'Donovan, C., Phan, I., Pilbout, S., and Schneider, M., *The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003.* Nucleic Acids Res, 2003. **31**(1): p. 365-70.

67. Thomas, J.W., Touchman, J.W., Blakesley, R.W., Bouffard, G.G., Beckstrom-Sternberg, S.M., Margulies, E.H., Blanchette, M., Siepel, A.C., Thomas, P.J., McDowell, J.C., Maskeri, B., Hansen, N.F., Schwartz, M.S., Weber, R.J., Kent, W.J., Karolchik, D., Bruen, T.C., Bevan, R., Cutler, D.J., Schwartz, S.,

Elnitski, L., Idol, J.R., Prasad, A.B., Lee-Lin, S.Q., Maduro, V.V., Summers, T.J., Portnoy, M.E., Dietrich, N.L., Akhter, N., Ayele, K., Benjamin, B., Cariaga, K., Brinkley, C.P., Brooks, S.Y., Granite, S., Guan, X., Gupta, J., Haghighi, P., Ho, S.L., Huang, M.C., Karlins, E., Laric, P.L., Legaspi, R., Lim, M.J., Maduro, Q.L., Masiello, C.A., Mastrian, S.D., McCloskey, J.C., Pearson, R., Stantripop, S., Tiongson, E.E., Tran, J.T., Tsurgeon, C., Vogt, J.L., Walker, M.A., Wetherby, K.D., Wiggins, L.S., Young, A.C., Zhang, L.H., Osoegawa, K., Zhu, B., Zhao, B., Shu, C.L., De Jong, P.J., Lawrence, C.E., Smit, A.F., Chakravarti, A., Haussler, D., Green, P., Miller, W., and Green, E.D., *Comparative analyses of multi-species sequences from targeted genomic regions.* Nature, 2003. **424**(6950): p. 788-93.

68. Kent, W.J., Sugnet, C.W., Furey, T.S., Roskin, K.M., Pringle, T.H., Zahler, A.M., and Haussler, D., *The human genome browser at UCSC.* Genome Res, 2002. **12**(6): p. 996-1006.

69. Sankoff, D. and Kruskal, J.B., *Time warps, string edits, and macromolecules: The theory and practice of sequence comparison.* 1983, Reading, MA: Addison-Wesley Publishing Company.

70. Myers, E.W., *Approximate matching of network expressions with spacers.* J Comput Biol, 1996. **3**(1): p. 33-51.

71. Cormode, G., Paterson, M. , Sahinalp, S. , and Vishkin, U. *Communication complexity of document exchange*. in *the eleventh annual ACM-SIAM Symposium on Discrete algorithms*. 2000: Society for Industrial and Applied Mathematics.

72. Erg¨un, F.a.M., S. and Sahinalp, S.C. *Comparing Sequences with Segment Rearrangements*. in *FST TCS: Foundations of Software Technology and Theoretical Computer Science*. 2003.

73. Galperin, M.Y. and Koonin, E.V., *Sources of systematic error in functional annotation of genomes: domain rearrangement, non-orthologous gene displacement and operon disruption.* In Silico Biol, 1998. **1**(1): p. 55-67.

74. Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D.J., *Gapped BLAST and PSI-BLAST: a new generation of*

*protein database search programs.* Nucleic Acids Res, 1997. **25**(17): p. 3389-402.

75. Schwartz, S., Kent, W.J., Smit, A., Zhang, Z., Baertsch, R., Hardison, R.C., Haussler, D., and Miller, W., *Human-mouse alignments with BLASTZ.* Genome Res, 2003. **13**(1): p. 103-7.

76. Kazazian, H.H., Jr., *Mobile Elements: Drivers of Genome Evolution.* Science, 2004. **303**(5664): p. 1626-1632.

77. Holmes, I., *Transcendent elements: whole-genome transposon screens and open evolutionary questions.* Genome Res, 2002. **12**(8): p. 1152-5.

78. Bailey, J.A., Liu, G., and Eichler, E.E., *An Alu transposition model for the origin and expansion of human segmental duplications.* Am J Hum Genet, 2003. **73**(4): p. 823-34.

79. Kidwell, M.G. and Lisch, D.R., *Perspective: transposable elements, parasitic DNA, and genome evolution.* Evolution Int J Org Evolution, 2001. **55**(1): p. 1-24.

80. Brosius, J. *How significant is 98.5% 'junk' in mammalian genomes*. in *European conference on computational biology (ECCB)*. 2003.

81. Capy, P., Gasperi, G., Biemont, C., and Bazin, C., *Stress and transposable elements: co-evolution or useful parasites?* Heredity, 2000. **85**: p. 101-6.

82. Shapiro, J.A., *Transposable elements as the key to a 21st century view of evolution.* Genetica, 1999. **107**(1-3): p. 171-9.

83. Jurka, J., *Repeats in genomic DNA: mining and meaning.* Curr Opin Struct Biol, 1998. **8**(3): p. 333-7.

84. Smit, A., Hubley, R., and Green, P., *RepeatMasker Open-3.0,* http://repeamasker.org. 1996-2004.

85. Negroni, M. and Buc, H., *Mechanisms of retroviral recombination.* Annu Rev Genet, 2001. **35**: p. 275-302.

86. Chimpanzee Sequencing and Analysis Consortium, *Initial sequence of the chimpanzee genome and comparison with the human genome.* Nature, 2005. **437**(7055): p. 69-87.

87. Kajikawa, M. and Okada, N., *LINEs mobilize SINEs in the eel through a shared 3' sequence.* Cell, 2002. **111**(3): p. 433-44.

88. Galperin, M.Y. and Koonin, E.V., eds. *Frontiers in Computational Genomics*. Functional Genomics Series. Vol. 3. 2002, Caister Academic Press.

89. Koonin, E.V., Fedorova, N.D., Jackson, J.D., Jacobs, A.R., Krylov, D.M., Makarova, K.S., Mazumder, R., Mekhedov, S.L., Nikolskaya, A.N., Rao, B.S., Rogozin, I.B., Smirnov, S., Sorokin, A.V., Sverdlov, A.V., Vasudevan, S., Wolf, Y.I., Yin, J.J., and Natale, D.A., *A comprehensive evolutionary classification of proteins encoded in complete eukaryotic genomes.* Genome Biol, 2004. **5**(2): p. R7.

90. Ortiz, A.R., Strauss, C.E., and Olmea, O., *MAMMOTH (matching molecular models obtained from theory): an automated method for model comparison.* Protein Sci, 2002. **11**(11): p. 2606-21.

91. Raphael, B., Zhi, D., Tang, H., and Pevzner, P., *A novel method for multiple alignment of sequences with repeated and shuffled elements.* Genome Res, 2004. **14**(11): p. 2336-46.

92. *Web site for additional files for Zhi, Raphael, Price, Tang, Pevzner 2005.*, http://bioinf.ucsd.edu/~dzhi/comrep/, 2005.

93. Can, T. and Wang, Y.-F. *CTSS: A Robust and Efficient Method for Protein Structure Alignment Based on Local Geometrical and Biological Features*. in *the Second International IEEE Computer Society Computational Systems Bioinformatics Conference, CSB 2003*. 2003. Stanford, CA.

94. Bao, Z. and Eddy, S.R., *Automated de novo identification of repeat sequence families in sequenced genomes.* Genome Res, 2002. **12**(8): p. 1269-76.

95. Zhang, X. and Wessler, S.R., *Genome-wide comparative analysis of the transposable elements in the related species Arabidopsis thaliana and Brassica oleracea.* Proc Natl Acad Sci U S A, 2004. **101**(15): p. 5589-94.

96. Volfovsky, N., Haas, B.J., and Salzberg, S.L., *A clustering method for repeat analysis in DNA sequences.* Genome Biol, 2001. **2**(8): p. RESEARCH0027.

97. Edgar, R.C. and Myers, E.W. *PILER: identification and classification of genomic repeats*. in *Proceedings of the 13th Annual International conference on Intelligent Systems for Molecular Biology (ISMB-05)*. 2005.

98. Price, A., Jones, N., and Pevzner, P. *De novo identification of repeat families in large genomes*. in *Proceedings of the 13th Annual International conference on Intelligent Systems for Molecular Biology (ISMB-05)*. 2005. Detroit, Michigan.

99. Chicken Genome Sequencing Consortium, *Sequence and comparative analysis of the chicken genome provide unique perspectives on vertebrate evolution.* Nature, 2004. **432**(7018): p. 695-716.

100. Bailey, J.A., Yavor, A.M., Viggiano, L., Misceo, D., Horvath, J.E., Archidiacono, N., Schwartz, S., Rocchi, M., and Eichler, E.E., *Human-specific duplication and mosaic transcripts: the recent paralogous structure of chromosome 22.* Am J Hum Genet, 2002. **70**(1): p. 83-100.

101. Kent, W.J., *BLAT--the BLAST-like alignment tool.* Genome Res, 2002. **12**(4): p. 656-64.

102. Gusfield, D., *Algorithms on Strings, Trees, and Sequences*. 1997: Cambridge University Press.

103. Felsenstein, J., *PHYLIP (Phylogeny Inference Package) version 3.6.* Distributed by the author. Department of Genome Sciences, University of Washington, Seattle., 2004.

104. Jones, N. and Zhi, D., *ABA web interface*, http://aba.bioprojects.org/, 2005.

105. Lupas, A.N., Ponting, C.P., and Russell, R.B., *On the evolution of protein folds: are similar motifs in different protein folds the result of convergence, insertion, or relics of an ancient peptide world?* J Struct Biol, 2001. **134**(2-3): p. 191-203.

106. Grishin, N.V., *Fold change in evolution of protein structures.* J Struct Biol, 2001. **134**(2-3): p. 167-85.

107. Jung, J. and Lee, B., *Circularly permuted proteins in the protein structure database.* Protein Sci, 2001. **10**(9): p. 1881-6.

108. Taylor, W.R., Heringa, J., Baud, F., and Flores, T.P., *A Fourier analysis of symmetry in protein structure.* Protein Eng, 2002. **15**(2): p. 79-89.

109. Ye, Y. and Godzik, A., *Multiple flexible structure alignment using partial order graphs.* Bioinformatics, 2005. **21**(10): p. 2362-9.

110. Ye, Y. and Godzik, A., *Flexible structure alignment by chaining aligned fragment pairs allowing twists.* Bioinformatics, 2003. **19 Suppl 2**: p. II246-II255.

111. Shindyalov, I.N. and Bourne, P.E., *Protein structure alignment by incremental combinatorial extension (CE) of the optimal path.* Protein Eng, 1998. **11**(9): p. 739-47.

112. Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., and Bourne, P.E., *The Protein Data Bank.* Nucleic Acids Res, 2000. **28**(1): p. 235-42.

113. Pearl, F., Todd, A., Sillitoe, I., Dibley, M., Redfern, O., Lewis, T., Bennett, C., Marsden, R., Grant, A., Lee, D., Akpor, A., Maibaum, M., Harrison, A., Dallman, T., Reeves, G., Diboun, I., Addou, S., Lise, S., Johnston, C., Sillero, A., Thornton, J., and Orengo, C., *The CATH Domain Structure Database and related resources Gene3D and DHS provide comprehensive domain family information for genome analysis.* Nucl. Acids Res., 2005. **33**(suppl_1): p. D247-251.

114. Dietmann, S. and Holm, L., *Identification of homology in protein structure classification.* Nat Struct Biol, 2001. **8**(11): p. 953-7.

115. Holm, L. and Sander, C., *Protein structure comparison by alignment of distance matrices.* J Mol Biol, 1993. **233**(1): p. 123-38.

116. Madej, T., Gibrat, J.F., and Bryant, S.H., *Threading a database of protein cores.* Proteins, 1995. **23**(3): p. 356-69.

117. Rao, S.T. and Rossmann, M.G., *Comparison of super-secondary structures in proteins.* J Mol Biol, 1973. **76**(2): p. 241-56.

118. Godzik, A., *The structural alignment between two proteins: is there a unique answer?* Protein Sci, 1996. **5**(7): p. 1325-38.

119. Shatsky, M., Nussinov, R., and Wolfson, H.J., *Flexible protein alignment and hinge detection.* Proteins, 2002. **48**(2): p. 242-56.

120. Jaroszewski, L. and Godzik, A., *Search for a new description of protein topology and local structure.* Proc Int Conf Intell Syst Mol Biol, 2000. **8**: p. 211-7.

121. Kolinski, A., Skolnick, J., Godzik, A., and Hu, W.P., *A method for the prediction of surface "U"-turns and transglobular connections in small proteins.* Proteins, 1997. **27**(2): p. 290-308.

122. Krissinel, E. and Henrick, K., *Secondary-structure matching (SSM), a new tool for fast protein structure alignment in three dimensions.* Acta Crystallogr D Biol Crystallogr, 2004. **60**(Pt 12 Pt 1): p. 2256-68.

123. Harrison, A., Pearl, F., Sillitoe, I., Slidel, T., Mott, R., Thornton, J., and Orengo, C., *Recognizing the fold of a protein structure.* Bioinformatics, 2003. **19**(14): p. 1748-59.

124. Kishon, E., Hastie, T., and Wolfson, H.J., *3-D Curve Matching Using Splines.* Journal Robotic Systems, 1991. **6**: p. 723-743.

125. Needleman, S.B. and Wunsch, C.D., *A general method applicable to the search for similarities in the amino acid sequence of two proteins.* J Mol Biol, 1970. **48**(3): p. 443-53.

126. Smith, T.F. and Waterman, M.S., *Identification of common molecular subsequences.* J Mol Biol, 1981. **147**(1): p. 195-7.

127. Waterman, M.S., Gordon, L., and Arratia, R., *Phase transitions in sequence matches and nucleic acid structure.* Proc Natl Acad Sci U S A, 1987. **84**(5): p. 1239-43.

128. Vogel, H.J. and Zhang, M., *Protein engineering and NMR studies of calmodulin.* Mol Cell Biochem, 1995. **149-150**: p. 3-15.

129. Byeon, I.J., Louis, J.M., and Gronenborn, A.M., *A captured folding intermediate involved in dimerization and domain-swapping of GB1.* J Mol Biol, 2004. **340**(3): p. 615-25.

130. Kabsch, W. and Sander, C., *Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features.* Biopolymers, 1983. **22**(12): p. 2577-637.

131. Bond, C.S., *TopDraw: a sketchpad for protein structure topology cartoons.* Bioinformatics, 2003. **19**(2): p. 311-2.

132. *Gordon and Betty Moore Foundation Marine Microbial Genome Sequencing Project*, https://research.venterinstitute.org/moore/, 2005.

133. Venter, J.C., Remington, K., Heidelberg, J.F., Halpern, A.L., Rusch, D., Eisen, J.A., Wu, D., Paulsen, I., Nelson, K.E., Nelson, W., Fouts, D.E., Levy, S., Knap, A.H., Lomas, M.W., Nealson, K., White, O., Peterson, J., Hoffman, J., Parsons, R., Baden-Tillson, H., Pfannkoch, C., Rogers, Y.H., and Smith, H.O., *Environmental genome shotgun sequencing of the Sargasso Sea.* Science, 2004. **304**(5667): p. 66-74.

134. Tyson, G.W., Chapman, J., Hugenholtz, P., Allen, E.E., Ram, R.J., Richardson, P.M., Solovyev, V.V., Rubin, E.M., Rokhsar, D.S., and Banfield,

J.F., *Community structure and metabolism through reconstruction of microbial genomes from the environment.* Nature, 2004. **428**(6978): p. 37-43.

135. Li, W., Jaroszewski, L., and Godzik, A., *Clustering of highly homologous sequences to reduce the size of large protein databases.* Bioinformatics, 2001. **17**(3): p. 282-3.

136. Li, W., Jaroszewski, L., and Godzik, A., *Tolerating some redundancy significantly speeds up clustering of large protein databases.* Bioinformatics, 2002. **18**(1): p. 77-82.

137. Bejerano, G., Pheasant, M., Makunin, I., Stephen, S., Kent, W.J., Mattick, J.S., and Haussler, D., *Ultraconserved elements in the human genome.* Science, 2004. **304**(5675): p. 1321-5.

138. Bejerano, G., Haussler, D., and Blanchette, M., *Into the heart of darkness: large-scale clustering of human non-coding DNA.* Bioinformatics, 2004. **20 Suppl 1**: p. I40-I48.

139. Grasso, C., Quist, M., Ke, K., and Lee, C., *POAVIZ: a Partial order multiple sequence alignment visualizer.* Bioinformatics, 2003. **19**(11): p. 1446-8.

140. Thomas, J.W., Touchman, J.W., Blakesley, R.W., Bouffard, G.G., Beckstrom-Sternberg, S.M., Margulies, E.H., Blanchette, M., Siepel, A.C., Thomas, P.J., McDowell, J.C., Maskeri, B., Hansen, N.F., Schwartz, M.S., Weber, R.J., Kent, W.J., Karolchik, D., Bruen, T.C., Bevan, R., Cutler, D.J., Schwartz, S., Elnitski, L., Idol, J.R., Prasad, A.B., Lee-Lin, S.-Q., Maduro, V.V.B., Summers, T.J., Portnoy, M.E., Dietrich, N.L., Akhter, N., Ayele, K., Benjamin, B., Cariaga, K., Brinkley, C.P., Brooks, S.Y., Granite, S., Guan, X., Gupta, J., Haghighi, P., Ho, S.-L., Huang, M.C., Karlins, E., Laric, P.L., Legaspi, R., Lim, M.J., Maduro, Q.L., Masiello, C.A., Mastrian, S.D., McCloskey, J.C., Pearson, R., Stantripop, S., Tiongson, E.E., Tran, J.T., Tsurgeon, C., Vogt, J.L., Walker, M.A., Wetherby, K.D., Wiggins, L.S., Young, A.C., Zhang, L.-H., Osoegawa, K., Zhu, B., Zhao, B., Shu, C.L., De Jong, P.J., Lawrence, C.E., Smit, A.F., Chakravarti, A., Haussler, D., Green, P., Miller, W., and Green, E.D., *Comparative analyses of multi-species sequences from targeted genomic regions.* Nature, 2003. **424**(6950): p. 788-793.

141. Kent, W.J., Sugnet, C.W., Furey, T.S., Roskin, K.M., Pringle, T.H., Zahler, A.M., and Haussler, a.D., *The Human Genome Browser at UCSC.* Genome Res., 2002. **12**(6): p. 996-1006.

142. Loeb, D.D., Padgett, R.W., Hardies, S.C., Shehee, W.R., Comer, M.B., Edgell, M.H., and Hutchison, C.A., 3rd, *The sequence of a large L1Md element reveals a tandemly repeated 5' end and several features found in retrotransposons.* Mol Cell Biol, 1986. **6**(1): p. 168-82.