The Pennsylvania State University
The Graduate School
College of Engineering

ALGORITHMS FOR ALIGNING AND CLUSTERING GENOMIC SEQUENCES THAT CONTAIN DUPLICATIONS

A Thesis in
Computer Science and Engineering
by
Minmei Hou

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2007

The thesis of Minmei Hou was reviewed and approved* by the following:

Webb Miller
Professor of Biology and Computer Science and Engineering
Thesis Advisor, Chair of Committee

Piotr Berman
Associate Professor of Computer Science and Engineering

Hongyuan Zha
Professor of Computer Science and Engineering

Ross Hardison
T. Ming Chu Professor of Biochemistry and Molecular Biology

Raj Acharya
Professor of Computer Science and Engineering
Head of the Department of Computer Science and Engineering

*Signatures are on file in the Graduate School.

Abstract

Genomes of advanced organisms contain numerous repeated sequences, including gene clusters, tandem repeats, interspersed repeats, and segmental duplications. Among these, gene clusters are the class most frequently of functional importance. Algorithmic processing of regions containing these clusters remains challenging in practice, and its lack of clean solutions has been a big obstacle in sequence analysis in bioinformatics. This thesis includes new methodologies for solving two sets of problems in processing the sequences of gene-cluster regions, particularly methods to properly align gene-cluster regions of multiple species.

Similar sequences sharing the same evolutionary origin are *homologous*. Homologous sequences that differ by speciation are *orthologous*. One set of problems deals with aligning all and only orthologous sequences in a gene-cluster region, between two or more species. A two-way orthologous-sequence identification tool is developed to produce orthologous pairwise alignments. The results are evaluated based on the phylogenetic inference of gene sequences. High specificity is achieved without much loss of sensitivity. Two approaches are designed to create orthologous multi-species alignments. One uses a chosen species to guide the alignment process, and it has been successfully applied genome-wide. The other solves a more difficult formulation of the problem, where all species are treated equally. Its computational difficulty is discussed, and some initial experiments are reported.

Another set of methods deals with the construction of all homologous groups within a single genome. Each homologous group is expected to contain precisely the genomic intervals that are homologous to each other. A mixture of algorithmic and heuristic procedures is designed to maintain a balance between the completeness and purity of each group. We verify the accuracy and efficiency of these methodologies.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I got help from many people in the completion of this thesis. I would like to thank them with my whole heart.

First and foremost, I want to thank my advisor, Dr. Webb Miller, for his guidance throughout my Ph.D program. Dr. Miller led me into the exciting field of bioinformatics, and got me involved in many useful projects from which I learned how to do research. Dr. Miller gave constructive suggestions at different stages of my study, while allowing me much freedom to work independently. His being hardworking has inspired me all the time.

I would also like to thank the other members of my thesis committee, Dr. Piotr Berman, Dr. Ross Hardison and Dr. Hongyuan Zha, for their thoughtful and insightful advice. Dr. Berman inspired and helped me with challenging problems. Dr. Hardison gave valuable comments and advice on the biological insight of my thesis work, which greatly enriched my dissertation. Dr. Zha gave insightful suggestions on extending the application of this thesis to document processing and vice versa.

Many thanks go to my colleagues in the Center for Comparative Genomics and Bioinformatics. They include Rico Burhans, Nate Coraor, Belinda Giardine, Bob Harris, Chih-Hao Hsu, John Karro, David King, Jian Ma, Aakrosh Ratan, Cathy Riemer, Gil Tae Song, Svitlana Tyekucheva, Yi Zhang, and many other people who had short visit at the center. It has always been enjoyable to work with them and learn from them. The wonderful time with them has resulted in friendship at both technical and personal levels. The journey through these several years with them has been one of the best experiences of my life.

I would like to express my deepest gratitude to my fiance Veysel Demir for his patience waiting for me to finish my Ph.D. study and urging me to accomplish my degree step by step. I also owe a huge debt of gratitude to my family: my father Wenke Wang, my mother Zhenggao Hou, and my brother Haobo Wang, for their dedicated and endless love and support not only throughout my Ph.D. study, but also my entire life. Without their guide and encouragement, I could not have achieved anything that I have achieved today.

Dedication

To my father Wenke Wang and my mother Zhenggao Hou.

Chapter 1

Introduction

This thesis is about processing genomic sequences of duplicated regions. In this chapter, we start with an overview of the repetitive structure of genomes of advanced organisms. We next introduce the concepts of homology, orthology, and paralogy, which are the fundamental biological framework underpinning this thesis. We then provide a brief review of current genomic alignment tools and illustrate their deficiencies, which lead to our motivation of doing orthologous alignment in duplicated regions. We conclude this chapter by discussing motivations in more detail.

1.1 Repetitive structure of genomes

A *genome* contains the complete set of hereditary information of an organism. It is encoded by four types of DNA *bases*: adenine (A), cytosine (C), guanine (G), and thymine (T). A genome can thus be simply regarded as a long text string having four types of letters. Genomes of advanced organisms are composed of *chromosomes*, which are actually the biological molecules that carry hereditary information physically and participate in life processes. For purposes of sequence analysis, chromosomes can also be simply regarded as long text strings that are composed of the above four letters. We use "(genomic) sequence" to refer to a segment of such strings in general. A *gene* is a functional segment of a sequence. In this thesis, "gene" usually refers to a functional segment encoding a protein. A gene may be composed of several *exons* (which encode parts of a protein) that are separated by *introns* (which do not encode the protein).

All species have experienced a long history of evolution. The evolutionary events include *substitution* (a DNA base is replaced by one of another type), *insertion* (a new sequence of contiguous DNA bases is added to a genome), *deletion* (a sequence is eliminated from a genome), *translocation* (a sequence breaks off from one chromosome and joins another chromosome), *inversion* (a sequence interval is removed and its reverse complement is inserted at the original location on the chromosome, i.e., the sequence is reversed, G and C are interchanged, and A and T are interchanged), *duplication* (a duplicate copy of a sequence is created and inserted in the

genome), and *transposition* (a sequence interval is moved from one location to another in the genome, or it is copied and the copies are inserted to other locations in the genome). When a gene loses its functions because of mutational events, it becomes a *pseudogene*.

Repetitive sequences occupy a big proportion of the genome of an advanced organism. Among the above evolutionary events, duplication and transposition produce repetitive sequences. Repetitive sequences can be categorized by such properties as length, number of copies, pattern of dispersion in the genome, functionality, and source of the sequences. The largest class is the *interspersed repeats*. They are propagated in the genome through duplication. The length of each copy of an interspersed repeat ranges from hundreds to thousands of bases. Since they are inserted nearly randomly, they are interspersed throughout the genome, and the number of copies ranges from dozens to millions. *Tandem repeats* consist of a certain number of identical or almost identical tandem short segments, as short as several bases. Duplications of large regions are called *segmental duplications*. They range from hundreds to millions of bases. A series of segmental duplications at adjacent regions may lead to *tandem duplicates*. When the copies contain genes and/or pseudogenes, this set of tandem duplicates is called a *(tandem) gene cluster*.

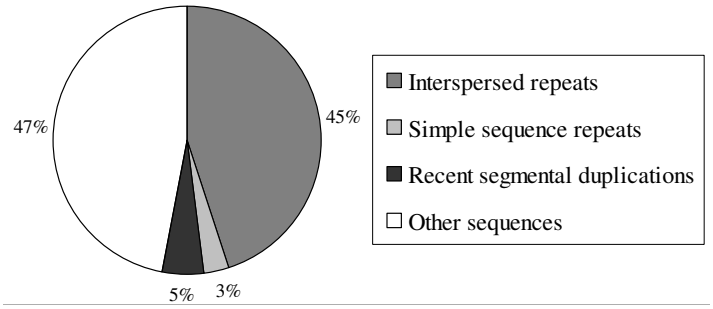


Figure 1.1. Percentages of repetitive sequences in the human genome.

Fig. 1.1 indicates the proportions of different types of repetitive sequences in the human genome [34]. There no doubt are a certain number of ancient segmental duplications that are not included in the above chart. The Figure shows that at least 53% of the human genome is composed of repetitive sequences. 5% consists of regions that exceed 1 kb in length and are at least 90% identical to another genomic segment, ignoring interspersed repeats such as Alus and LINEs [34]. These regions are most probably segmental duplications produced recently because of their high similarity to other regions. There is no clear conclusion about the general mechanisms for segmental duplications. Unequal crossing-over can explain a subset of duplications having the property that duplicated copies are tandemly arranged, such as gene clusters.

It has long been thought that duplications, particularly gene duplications, are an important contributor to evolution [28]. Ohno [45] even argued that duplication is essentially the only way a new gene can arise. Duplication of a gene permits one of the copies to take on new functions, while the other copy performs the original duties. Indeed, an observation that both of the gene copies created by an ancient duplication event have been retained is seen as evidence that their current functions are different, since otherwise one superfluous copy would have been degraded by

random mutations. A small proportion of interspersed repeats might have evolved and became functional genes [34], but most interspersed repeats do not have known functions. Gene clusters, however, are rich in functional elements, and they are thus the focus of this thesis.

1.2 Homology, orthology, and paralogy

Richard Owen introduced the term *homology* in 1849 to refer to physical similarities among species [49]. The proper definition of this term has been debated ever since. When this term is applied to sequence analysis, it suggests that the relatedness of sequences stems from their common evolutionary origin. Any two similar genes or genomic sequences having the same evolutionary origin, no matter whether they are from the same genome or different genomes, are *homologous*, and they are a pair of *homologs*. For example, the members of a given family of intersperse repeats, as described in the last section, are homologous. The difference between the two sequences measures the *divergence* of the two sequences since their nearest common origin. *Speciation* is a complex evolutionary process in which new species arise. A pair of homologs can diverge due to speciation, or duplication, or a combination of both.

The key concept for this thesis is the division of all pairs of homologous sequences into two categories, *orthologs* and *paralogs*, as established by Fitch in 1970 [23]. Two genes or genomic segments in different genomes are *orthologous* if they diverged as the result of a speciation event; two genes or genomic segments (perhaps in the same genome) are *paralogous* if they diverged at an intra-species duplication event. After duplication, the two paralogs have several possible fates. One of the copies might lose its functionality and become a pseudogene, one copy might gain new functions, the two copies might partition the original functions between them, or both copies might keep the original functionality. Except for the last case, functionalities of the two paralogs differ, and hence in general it is not reliable to infer functionality of a gene from its paralog. On the other hand, orthologs are more likely to have similar functionalities.

When there is only a one-to-one relationship of orthologs between two genomes, i.e., one segment of a genome has exactly one ortholog in another genome, it is relatively easy to determine the orthologs. The simple best-hit approach is sufficient to do the job: a segment s in a genome and its most similar sequence s' in another genome are orthologous [64]. But from the definition above, orthology is not necessarily a one-to-one relationship; it can also be one-to-many or many-to-many. Thus, a reciprocal best hit approach does not provide a general solution for orthology detection. Fig. 1.2, modified from [24], shows a complex example of different types of orthologs. A1, A2, B1, B2, C1, C2 and C3 are gene copies. Gene copies with the same letter are from the same species. Horizontal lines indicate duplication events. There is a one-to-many orthology relationship among B2, C2 and C3; B2 is orthologous to both of C2 and C3, since the duplication producing C2 and C3 came after the species split between B and C. If a simple reciprocal best hit approach is applied, the highest scoring hit for B2 might be C2, and the highest scoring hit for C2 might be B2. It would then be concluded that B2 and C2 are orthologous, but C3 would be missed. For another example, since the duplication to produce A1 and A2 happens after the

speciation split between A and other species, A1 and A2 are each orthologous to all of other genes in this tree in the manner of a many-to-many relationship. A reciprocal best hit would not detect such relationships.

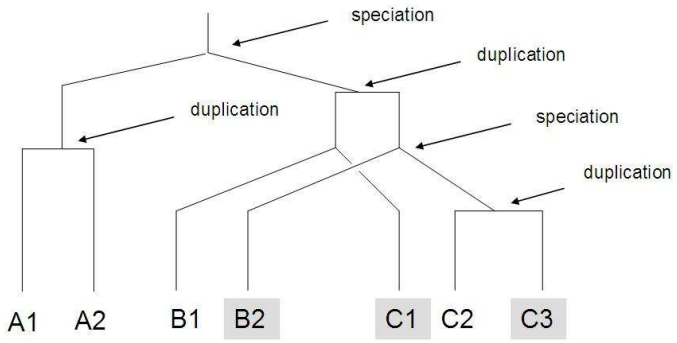


Figure 1.2. A gene tree showing orthology and paralogy relationships. Adapted from [24].

Beside the above one-to-many and many-to-many orthology relationships, many other genetic processes make the problem of orthology assignment more difficult. Horizontal gene transfer, gene loss, genome rearrangements etc. make it sometimes even impossible to perform correct assignment. Fig. 1.3 shows a simple example of lineage-specific gene loss. In such a case, a reciprocal best hit predicts that A1 and B2 are a pair of orthologs, and there is no way to correct this wrong conclusion based only on the sequences of species A and B. This sort of wrong assignment is sometimes acceptable since B2 is still the best counterpart of A1 in the genome of species B, and vice versa. This relationship is called *pseudo-ortholgy* in [38]. An incomplete genome sequence can confuse orthology assignment in a similar way, since a missing sequence is equivalent to a gene loss in this scenario. When the related sequences of more species are available, it is possible to identify duplication and gene loss events by the technique of *tree reconciliation* [50], and further to more reliably predict orthology relationships. However, this technique is computationally very expensive, and many uncertainties make it less practical for application to large-scale genomic analysis. Reference [38] provides a comprehensive review of the issues involving orthologs, paralogs, and homologs, and of the basic techniques to detect orthologs.

Conversion, another kind of genetic process that also complicates the problem of orthology assignment, is frequent in tandem gene clusters such as the alpha globin and beta globin gene clusters. In such a process, a segment of genomic sequence is replaced by homologous sequence from the same genome. It can be regarded as a simultaneous gene loss and a duplication. Such a segment of genomic sequence does not necessarily correspond to an entire gene, which leads to the phenomenon that different parts of a gene may be orthologous to different parts of other genes, and a single gene is no longer the orthologous unit in evolutionary processes. A similar phenomenon is described in the review [38] as resulting from *gene fusion* and *fission*, which are more frequent in genomes of simple organisms.

The one-to-many and many-to-many orthology relationships, entangled with the problems

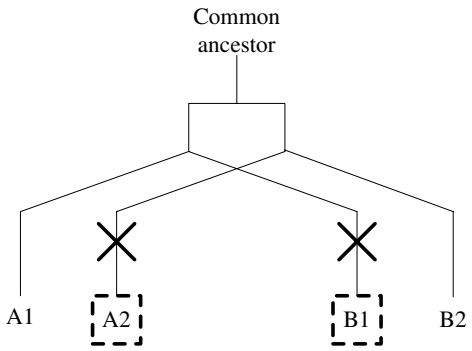


Figure 1.3. A gene tree showing pseudo-orthologs. Genes A2 and B1 are independently lost during the evolution of species A and B. A black cross denotes a gene-loss event.

of gene loss, conversion, and uncertainties coming from sequencing and assembling, make the orthology assignment problem difficult. The popular solution of using phylogenetic inference is limited by the completeness of related sequences, the expensive computation, and the difficulty in interpreting the phylogenetic tree. Thus it is not practical to apply this technique in large-scale analysis. COG [64] initiated the use of sequence similarities to do orthology assignment at a whole-genome scale, but excluded one-to-many and many-to-many relationships. Methods using gene (amino acid or DNA) sequences are also restricted by the availability of annotations of a genome. A major part of this thesis aims to do orthology assignment by genomic sequence alignment specifically for regions containing gene clusters.

1.3 A short overview of genomic aligners

The first step in analyzing a piece of unknown molecular sequence is to align it with a sequence database to find its homologs, or hopefully its orthologs, which can provide the most accurate functional inference. Sequence alignment is thus one of the most frequent and useful computations for a molecular biologist. The alignment can be used in evolutionary analysis, functional inference, and other sequence annotations. In an alignment, the sequences are each padded with zero or more dashes, then stacked vertically, in such a way that the letters (each signifying a nucleotide or an amino acid) in a given column are presumed to all be derived by substitutions from the same position in the most-recent ancestral sequence. This computation explicitly accounts for only a few evolutionary operations, namely substitution (different letters in the same column) and insertion/deletion (indicated by dashes). Since the alignment is done to every position of the input sequences, this method is called *global alignment*. This format of presenting alignment is widely used in aligning either DNA or protein sequences for genes. Since the first dynamic programming pairwise alignment algorithm (the well-known Needleman-Wunsch method) was proposed in 1970 [44], many alignment-generating tools have been developed and improved using various methods. Since a gene may have non-adjacent segments of functional and conserved regions, it is sometimes not reasonable to enforce global alignment of homologous genes. This

observation lead to the development of *local alignment* algorithms, where only conserved regions are aligned, such as the Smith-Waterman algorithm [61]. The design of the very fast heuristic local pairwise alignment tool BLAST [1] in 1990 was a breakthrough in sequence analysis, and it made sequence alignment a routine procedure in a typical molecular biology lab. Most tools after that are derived from it or indirectly use the strategy from it. At the same time, methods to simultaneously align multiple sequences have also been developed. They usually work progressively from pairwise alignments of input sequence pairs. ClustalW [65] is a representative example.

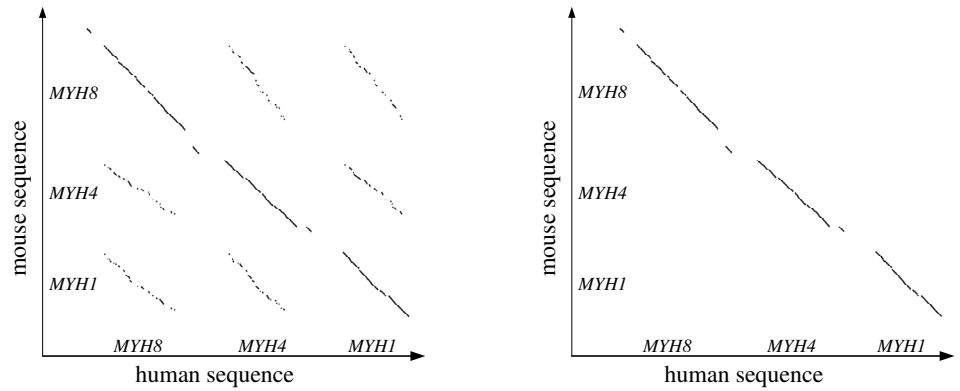
Large-scale genome sequencing projects promote the development of large-scale genomic sequence alignment tools. Since functional regions are frequently shuffled by rearrangements in advanced genomes, they may not be arranged collinearly on different genomes. In typical cases, the desired end-result for large-scale genomic alignment is a set of local alignments. BLASTZ [58] is one example of programs that align two long genomic sequences very efficiently. Recent large-scale multi-sequence genomic alignment tools include TBA [8], MLAGAN [10], Mauve [18], MAVID [9], and ABA [56]. Each has its own approach and features. Most of them start with identifying appropriate regions to align, and then apply certain alignment procedures to obtain final multi-alignments. MLAGAN progressively (i.e., from the leaves to the root in the phylogenetic tree) constructs multiple alignments based on a pairwise global alignment system, followed by an optional iterative improvement stage. Mauve recursively identifies anchors, called "locally collinear blocks", then performs progressive alignment of each such block. MAVID proceeds at an internal node of a guide tree by aligning maximum-likelihood ancestral sequences inferred from alignments of each sub-tree. MAVID also incorporates biological information into alignments by using gene-based anchors. ABA represents alignments by a directed graph, and utilizes a series of graph operations. TBA introduces the concepts of "threaded blockset"; the alignments produced by TBA are composed of blocks (local multi-alignments), and the set of blocks can be projected onto any sequence, which in turn, threads through the alignments so that each position in a given sequence appears in precisely one block. Most of these tools are challenged by existence of sequence rearrangements, such as inversion, transposition, translocation, and duplication, though several of the tools provide partial solutions to tackle complex genome structure. Among the above tools, the original MLAGAN and MAVID required input sequences to be collinear, and thus are not suitable for rearranged genomic sequences. Mauve provides solutions for certain types of rearrangements, but not for duplications. ABA identifies all homologous regions, but does not distinguish between orthologs and paralogs. The original TBA established a conceptual framework for rearrangements, but the published implementation did not handle complex situations.

At the same time as this thesis work was in progress, several of the above tools have been improved to handle more genome rearrangements [43]. MLAGAN added a reshuffle procedure to "rearrange" genome segments to form collinear sequences to apply global alignment. MAVID added a tool to identify orthologous synteny blocks before performing alignment. An updated TBA provided solutions for inversion and translocation [48]. While these programs cope grace-

fully with inversions and chromosomal breaks, they typically do not allow a given position in one genome to be aligned to several positions in another genome, as may be appropriate for duplicated segments. The main remaining challenge is to process duplicated regions appropriately. The most recent versions of TBA and MLAGAN [43] allow different positions of a specified genome to be aligned to the same position in other genomes, and thus have improved their performance in aligning genomic sequences of duplicated regions. But neither of them has completely satisfactory performance in those regions. The deficiency of current methods can be explained using the long-accepted concepts introduced in the last section.

In summary, two sequences are *homologs* if they are evolutionarily related, in which case they diverged at either a duplication event (and are called *paralogs*) or a speciation event (and are called *orthologs*). It is widely appreciated that a basic goal of alignment algorithms is to align sequences if and only if they are homologs. We feel that a better statement of the goal is to align precisely the orthologs. That is, we want the evolutionary relationship among aligned sequences to be the same as the phylogenetic tree relating the species for those sequences. A main (and probably *the* main) use of alignments is to identify intervals within the aligned segments in which the similarity/divergence pattern differs from neutral evolution, and modern methods for detecting such intervals [59, 16, 66] require for their proper functioning that aligned rows be orthologs. In regions of the genome where no intervals have been duplicated, orthology is equivalent to homology, and existing alignment methods are effective. However, for tandem gene clusters, we know of no existing aligner that does a good job. In our opinion, failure to properly handle duplications, and in particular tandem gene clusters, is the most glaring shortfall of current software for aligning genome sequences. It is critical for a genome-comparison program to produce an appropriate set of alignments when applied to a region that contains duplicated segments.

To facilitate discussing the proper aligning of genomic regions that contain duplicated segments, consider Fig. 1.4(a). It gives a graphical overview (called a "dot-plot") of the segments that match between portions of the human and mouse myosin heavy chain gene clusters. The diagonal lines are homologous alignments. There are three pairs of orthologs in this example: *MYH1*, *MYH4* and *MYH8*. The problem with simply creating a six-row alignment containing all the homologs, i.e., the three human genes and three mouse genes, is that it would contain some row pairs that lack the desirable properties enjoyed by other row pairs. For instance, human *MYH4* and mouse *MYH4* are more likely to perform the same biological function than are human *MYH4* and mouse *MYH8*. Moreover, it is much easier to determine how much the pattern of mutations between human *MYH4* and mouse *MYH4* departs from neutral evolution than it would be for human *MYH4* and mouse *MYH8*, because the amount of time since *MYH4* diverged from *MYH8* is difficult to determine. Furthmore, in practical terms, many alignment programs utilize knowledge of the evolutionary relationships among the species from which the sequences came, and they will hence be misled if the relationships among the segments being aligned differ from the species' relationships. The solution, of course, is to create a separate alignment for each of *MYH1*, *MYH4* and *MYH8*, as indicated in Fig. 1.4(b), which contains just the orthologous



(a) homologous alignment (b) orthologous alignment

Figure 1.4. Dot-plot of pairwise alignments of two orthologous regions of human and mouse containing the myosin heavy chain genes *MYH1*, *MYH4*, and *MYH8*.

alignments.

1.4 Motivation

This thesis is motivated by two aspects of sequence analysis, both of which we feel are in need of better solutions. First, as discussed above, orthologs provide the best information for evolutionary analysis and functional inference, and thus it is important to provide accurate orthologous alignments to ensure the correctness of downstream sequence analysis. The other aspect is that it is sometimes necessary to obtain the weakest relationships among homologs so as to determine all evolutionary relationships and functional implications, and this motivates us to construct the complete sets of homologs within a given genome sequence.

When there are no duplications in the input sequences from different species, several aligners such as TBA and MLAGAN provide reasonable alignments, because the homologous alignments are actually orthologous alignments when there are no paralogs. However, gene clusters frequently differ among species in the number of gene copies, which greatly complicates the goal of aligning all and only appropriate interval pairs. To maximize the likelihood that aligned segments have the same biological function, to permit accurate assessment of departure from neutral evolution, and to insure that use of the alignment algorithm is justified, we align segments from two species only after determining that they arose, possibly after substitutions and small insertions/deletions, from the same interval in the most recent common ancestor of those species, i.e., the segments must be orthologous. Stated the other way round, we avoid aligning two segments that were separated by an intra-species duplication event at some point in history. When the number of copies differs between two species, the orthology relationships cannot be a one-to-one. Indeed, they may not be 1-to-1 even when the count is identical. Our approach to aligning genomic sequences of duplicated regions has been implemented in a package that includes several major programs. One is called TOAST, which filters a set of pairwise alignment computed by, say, BLASTZ [58] to retain only

the putative orthologous matches; the putative orthologous alignments are then fed to ROAST and BOAST, which are multi-sequence aligners that produce orthologous multiple alignments.

Because we are trying to use a computational approach that makes biological sense by differentiating orthologous alignments from other alignments, where the alignment properties of orthologs are determined by complex evolutionary processes, there seems not to be a clean algorithmic solution. Therefore, performance of the heuristic methods needs to be examined carefully. However, evaluating the success of our approach proved challenging, due to a shortage of accurate annotation for available vertebrate genome sequences. Furthermore, there is not enough information about which genes are orthologs in complicated gene clusters. To evaluate our aligners, we create a set of benchmarks using phylogenetic inference to determine orthologs.

It initially seemed straightforward to construct the complete sets of homologs of a genome as long as a pair of homologs can be detected by a sensitive pairwise aligner, and a simple linkage strategy could have achieved it. It turned out to be difficult because of the complex repetitive structure of genomes of advanced organisms and the uncertainties of duplication boundaries. Particularly when we want to make a member in a set of computed homologs correspond to a complete gene whenever possible, this problem becomes challenging, and there seems not to be a clean solution either. Our software HomologMiner uses heuristic solutions and proves to be adequate for several mammalian genomes.

The rest of this thesis is composed of details of methods and results related to processing genomic sequences in duplicated regions, motivated by the above two aspects. We provide details for pairwise (Chap. 3) and multi-sequence orthologous alignment. There are two types of multiple alignments: reference-dependent and reference-independent. A *reference sequence* is from is a species that was chosen as a guide to organize the pairwise/multi-species alignment, such that any sequence of another species that is not aligned to the reference is discarded. Reference independent multi-alignment is the ideal one for many biologists, but its difficulty prevents it from a practical use at this time. Thus the details for multi-alignment methods are divided into two chapters. Chap. 4 describes a reference-dependent method, which performs very well and is applied to gene clusters genome-wide. Chap. 5 is for reference-independent alignment, where the problem's difficulty is revealed and discussed. Since we need to evaluate the performance of designed aligners, the very next chapter, Chap. 2, is dedicated to setting up the benchmark system used by the following chapters. At the end, we provide details of constructing homologous groups within a genome in Chap. 6. We summarize our conclusions from this thesis in Chap. 7.

Chapter 2

Creating benchmarks for orthologous alignment tools

2.1 Introduction

The overarching theme of this thesis is the design of orthologous alignment tools. To evaluate the performance of these tools, we need information about true orthologs. Unfortunately, gene annotations are limited, especially for species other than human. Instead, computer simulated genomic sequences have been used to evaluate several aligners [8, 48]. In these simulations, a hypothetical ancestral sequence is first created, and current sequences are then derived from it based on empirical models of substitutions, insertions, deletions, interspersed repeats, and inversions. Since the evolutionary relationships among the generated sequences are known, the orthology relationships are also determined. However, the mechanism of gene duplication is not yet well understood, and thus an accurate simulation of gene duplications is not available. We thus designed a method to obtain orthology information for a sufficient number of gene clusters, and this information will be used as benchmarks for evaluating our orthologous alignment tools.

One approach for orthology assignment of gene sequences is to use phylogenetic analysis: construct a phylogenetic gene tree, compare it with a species tree and reconcile the two, and then infer orthologs from them [21]. A *phylogenetic tree* is a binary tree where each internal node represents the most recent common ancestor of all descendants of this node. When the leaves are homologous genes from more than one species, some branches of this tree can be used to infer orthologs. Fig. 2.1 shows several typical topologies of branches that are useful in inferring orthology for genes from two species. In this figure, genes with the same letter are from the same species. Fig. 2.1(a) indicates that A1 and B1 are descendants of a nearest common ancestor. Fig. 2.1(b) indicates that B1 and B2 are descendants of a nearest common ancestor that descended from an ancestor to A1, that is, the duplication event creating B1 and B2 happened after the species split between A and B. Fig. 2.1(c) indicates that the nearest common ancestor of A1 and

A2 and the nearest common ancestor of B1 and B2 are descendants of a common ancestor, that is, the duplication events creating B1 and B2, and A1 and A2, happened independently after the species split between A and B. Thus these figures illustrate one-to-one, one-to-many, and many-to-many orthology relationships, respectively, between two species. It looks straightforward to interpret these gene trees without much hassle when there are only two species. However, the orthology assignments in these examples can be confused by gene loss, which is described in detail in Sec. 1.2. This deficiency cannot be completely solved when processing sequences from only two species. A tree with sequences from more species may provide more information, but it also becomes more difficult to interpret, as we will see in Sec. 2.2.2. Indeed, orthology assignment for gene clusters is a difficult problem, and it becomes even more difficult when gene conversions (introduced in Section 1.2) are involved. Since gene conversion can theoretically happen anywhere in the gene, for the whole gene, or including more then one gene, it is sometimes impossible to conclude clear orthology among genes. Hence, the phylogenetic reconstruction approach does not necessarily provide absolutely true orthology, but it is does provide a relatively robust approach to infer orthologs. Though the phylogenetic analysis method is not computationally efficient enough to be used in high-throughput orthologous alignment, it still can be used to produce a small yet sufficient set of benchmarks so that the orthologous alignment tools in Chapters 3 and 4 can be evaluated quantitatively.

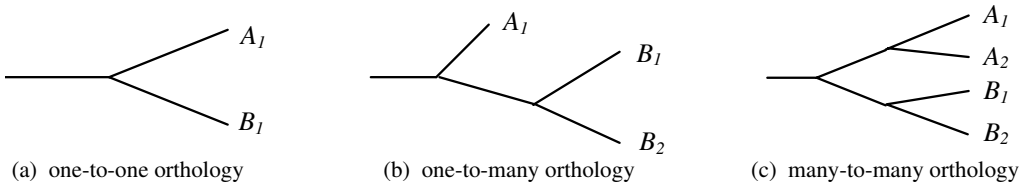


Figure 2.1. Orthology inference from phylogenetic gene trees.

A *gene family* is a group of genes that share similar sequences and related functions. Members of a gene family are frequently located in proximity along the same chromosome, although many large gene families have members dispersed on different chromosomes. For some gene families, duplication events happened very early in evolutionary history. The HOX gene family is one example. Paralogs in this family share low similarity at the DNA level. An ordinary genomic alignment tool is able to do correct orthologous alignment for this case, because of the low similarity between paralogs. More generally, for some gene families, there have been few genomic changes except simple edits since the clusters were formed. The myosin heavy chain gene cluster is one such example (see Fig. 1.4). Pairwise alignments of such gene clusters fall along a diagonal in a dot-plot, as shown in Fig. 1.4(b), and a chaining program or a global alignment tool is probably satisfactory in this case. However, for some gene families, there were new duplications after the speciation split, or there were large deletions. Thus, the same gene clusters in different species can have different numbers of gene copies. We believe the gene clusters of this last type are good examples for evaluating our new alignment strategy.

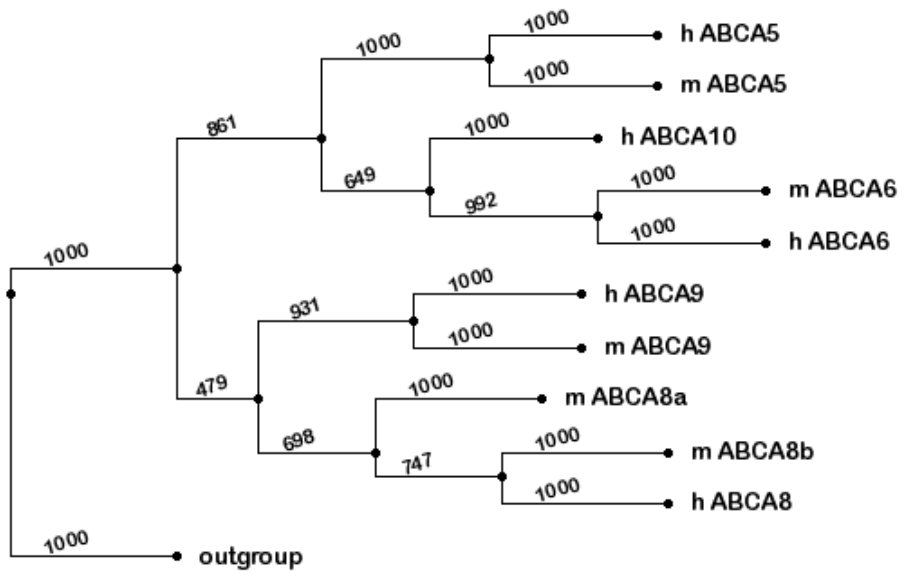
2.2 Results

We here describe 12 such clusters in some detail, each from both human and mouse, including their phylogenetic analysis and orthology assignment results. One of these clusters is the beta globin gene cluster. We consider orthology assignment of this cluster for 19 vertebrate species, because this cluster has been studied relatively extensively [29, 62].

2.2.1 12 selected gene clusters of two species

We found the 12 gene clusters as follows. We divided the human genome into 1-megabase segments, and applied self-alignment for each of them using BLASTZ [58] with default parameters. Many regions with repeated sequences were found. We discard regions with fewer than five copies. The remaining sequences were checked for protein annotations, which were downloaded from the UCSC genome browser [36]. We discarded regions without good annotations in human. The remaining sequences were then manually inspected in the genome browser to determine the orthologous regions of mouse, discarding the sequences without a known orthologous mouse region, those without good annotations in mouse, and those with the same number of gene copies in human and mouse. For gene clusters over 1 megabase in size, we manually determined reasonable cluster boundaries. This process yielded 12 usable gene clusters, having 5 to 31 gene copies in human. This selection process is admittedly crude, but our purpose was to get enough appropriate examples, though not necessarily all.

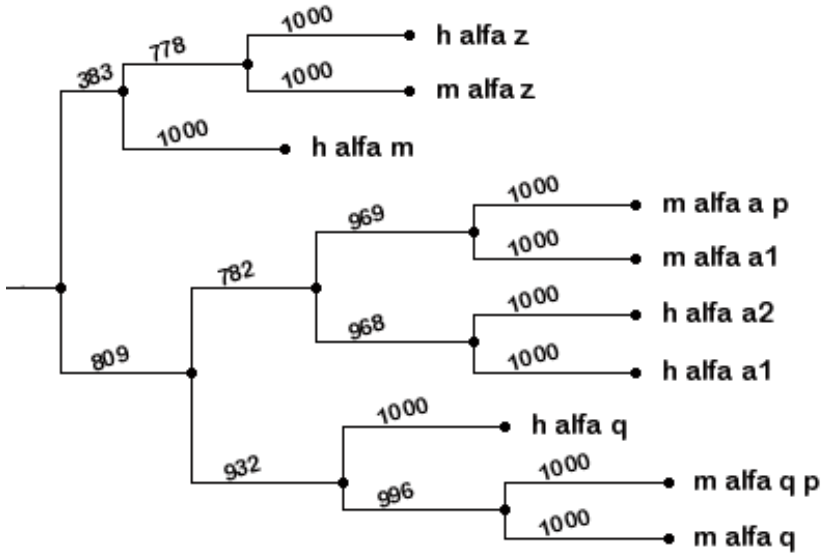
To each gene cluster, we applied Maximum Likelihood phylogeny reconstruction with bootstrapping of 1000 replicates using the Phylip package [22]. "Outgroup" sequences were chosen manually. The bootstrapping value associated with each branch measures how reliable this branch is. Branches with bootstrapping value above a certain number (950, 800 or 500) were taken as trustworthy to infer orthologs. Table 2.1 to Table 2.12 record the phylogenetic gene trees of these 12 clusters. The trees were drawn by [25] that displays phylogenies. The values at each branch do not represent the evolutionary distances. Instead, they are bootstrapping values that show how reliable the branches are. Orthologous pairs inferred from the trees using bootstrapping values above 500 as criteria are summarized in the tables, together with the genomic location of each gene. These inferred orthologous pairs will be regarded as the set of true orthologs in evaluating pairwise orthologous alignments in Chapter 3.



Orthologous pairs of ABCA cluster

human	mouse
ABCA5: 64,754,687-64,823,161	ABCA5: 109,943,375-110,008,800
ABCA6: 64,586,442-64,649,610	ABCA6: 109,847,949-109,922,853
ABCA8: 64,375,028-64,463,087	ABCA8b: 109,605,398-109,666,941
ABCA9: 64,482,684-64,568,731	ABCA9: 109,771,947-109,839,278

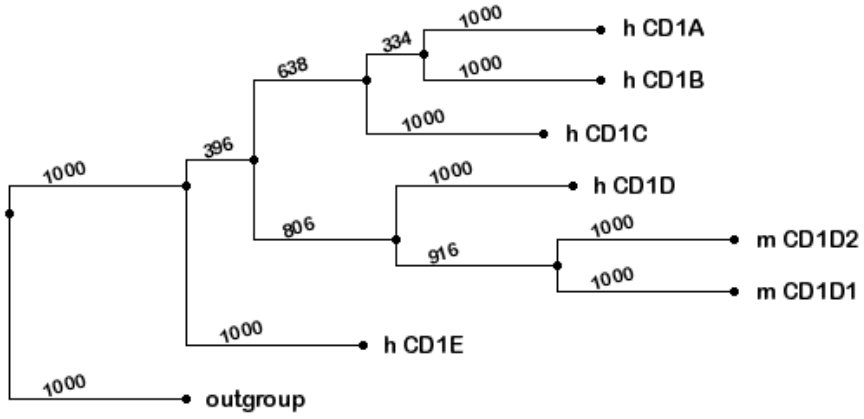
Table 2.1. ATP-binding cassette sub-family A (ABCA) gene cluster. Gene cluster regions are hg17:chr17:$63,987,501 - 65,112,500$ and mm5:chr11:$109,396,702 - 110,319,561$.



Orthologous pairs of alpha globin gene cluster

human	mouse
ζ: 142,854-144,504	ζ: 32,173,175-32,173,594
α_1: 166,679-167,520	α_1: 32,179,272-32,179,898
α_2: 162,875-163,705	α_2: 32,192,089-32,192,715
θ: 170,335-171,178	θ_1: 32,195,640-32,196,444
	θ_2: 32,182,538-32,183,483

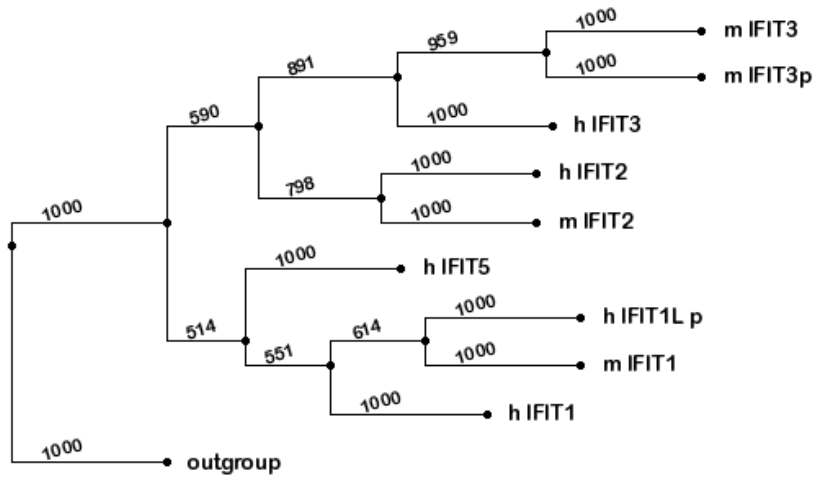
Table 2.2. Alpha globin gene cluster. Gene cluster regions are hg17:chr16:$61,079 - 226,278$ and mm5:chr11:$32,055,614 - 32,242,000$.



Orthologous pairs of CD1 gene cluster

human	mouse
CD1D: 154,962,810-154,967,758	CD1D1: 87,484,188-87,487,687
	CD1D2: 87,474,908-87,478,130

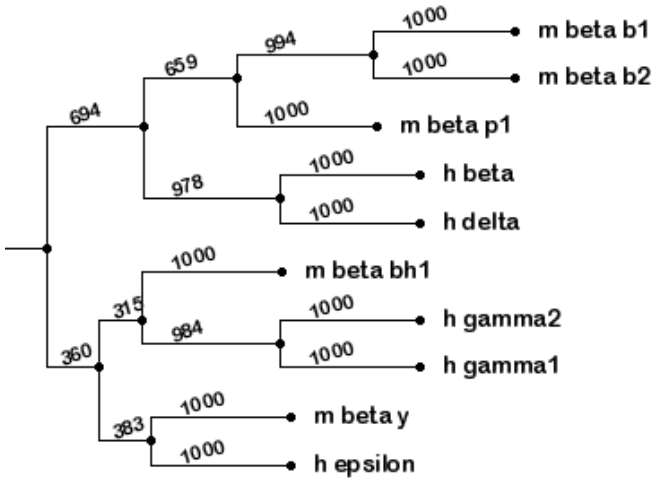
Table 2.3. CD1 gene cluster. Gene cluster regions are hg17:chr1:$154,400,000 - 155,400,000$ and mm5:chr3:$87,310,887 - 87,660,986$.



Orthologous pairs of IFIT gene cluster

human	mouse
IFIT1L_p: 91,127,793-91,134,941	IFIT1: 33,862,150-33,871,015
IFIT2: 91,051,783-91,058,658	IFIT2: 33,779,723-33,785,931
IFIT3: 91,082,283-91,090,294	IFIT3: 33,811,059-33,816,202 IFIT3p: 33,792,990-33,835,310

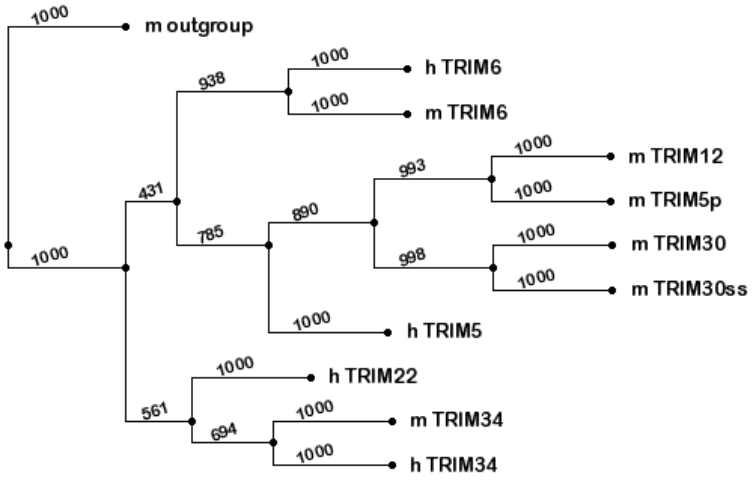
Table 2.4. Interferon-induced protein with tetratricopeptide (IFIT) gene cluster. Gene cluster regions are hg17:chr10:$91,031,111 - 91,208,888$ and mm5:chr19:$33,600,232 - 34,021,565$.



Orthologous pairs of beta globin gene cluster

human	mouse
β: 5,203,272-5,204,877	β_{b1}: 91,344,677-91,346,072
δ: 5,210,640-5,212,289	β_{b2}: 91,325,508-91,326,885
	β_{p1}: 91,357,695-91,358,977

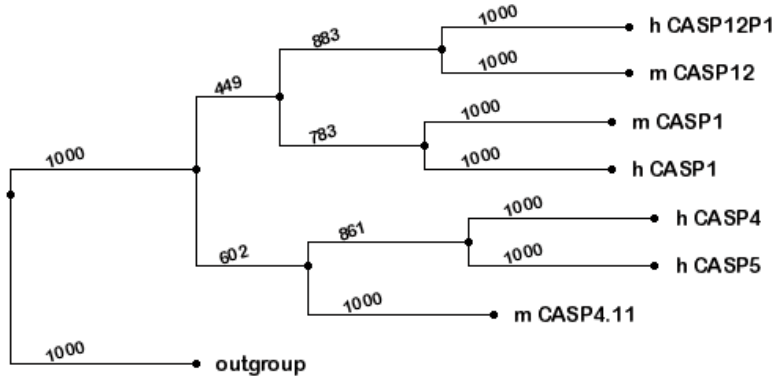
Table 2.5. Beta globin gene cluster. Gene cluster regions are hg17:chr11:$5,167,358 - 5,286,382$ and mm5:chr7:$91,293,611 - 91,386,743$.



Orthologous pairs of TRIM gene cluster

human	mouse
TRIM6: 5,573,923-5,590,764	TRIM6: 91,737,369-91,753,705
TRIM5: 5,641,369-5,662,869	TRIM5p: 91,858,432-91,871,891 TRIM12: 91,818,452-91,834,009 TRIM30: 91,927,581-91,983,664 TRIM30ss: 91,990,404-92,026,378
TRIM34: 5,610,087-5,619,100	TRIM34: 91,766,551-91,780,787

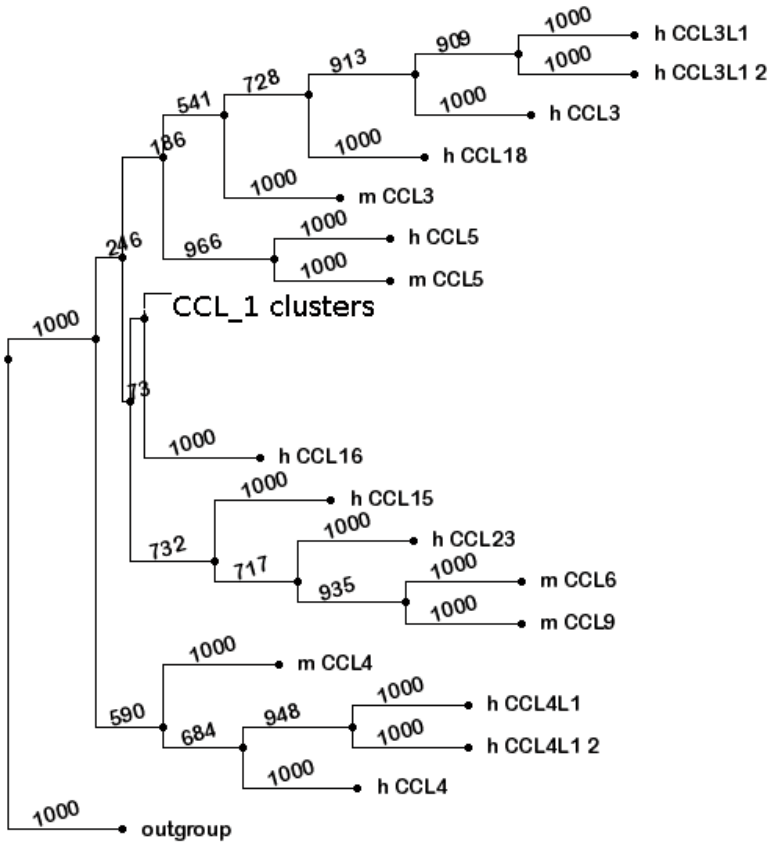
Table 2.6. Tripartite motif (TRIM) gene cluster. Gene cluster regions are hg17:chr11:$5,176,230 - 6,048,779$ and mm5:chr7:$91,117,583 - 92,500,867$



Orthologous pairs of caspase gene cluster

human	mouse
CASP1: 104,401,449-104,411,067	CASP1: 5,227,877-5,236,641
CASP4: 104,318,814-104,344,499 CASP5: 104,370,177-104,384,817	CASP4.11: 5,238,235-5,266,074
CASP12p1: 104,261,876-104,274,370	CASP12: 5,274,850-5,302,006

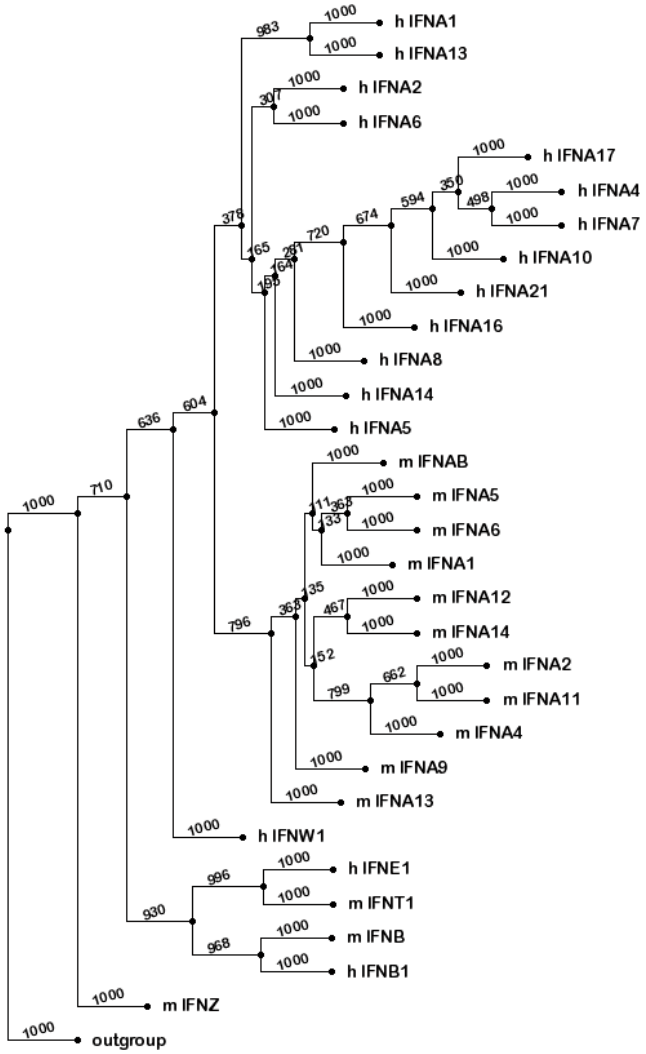
Table 2.7. Caspase gene cluster. Gene cluster regions are hg17:chr17:$102,000,000 - 150,000,000$ and mm5:chr9:$4,693,947 - 5,825,646$.



Orthologous pairs of CCL2 gene cluster

human	mouse
CCL5: 31,222,610-31,231,490	CCL5: 83,127,445-83,132,180
CCL3L1: 31,546,384-31,548,269 CCL3L1_2: 31,647,958-31,649,843 CCL3: 31,439,718-31,441,600 CCL18: 31,415,756-31,422,953	CCL3: 83,249,505-83,251,040
CCL23: 31,364,210-31,369,118	CCL6: 83,189,548-83,194,749 CCL9: 83174579-83180298
CCL4: 31,455,333-31,457,127 CCL4L1: 31,562,581-31,564,385 CCL4L1_2: 31,664,147-31,665,951	CCL4: 83,264,246-83,266,345

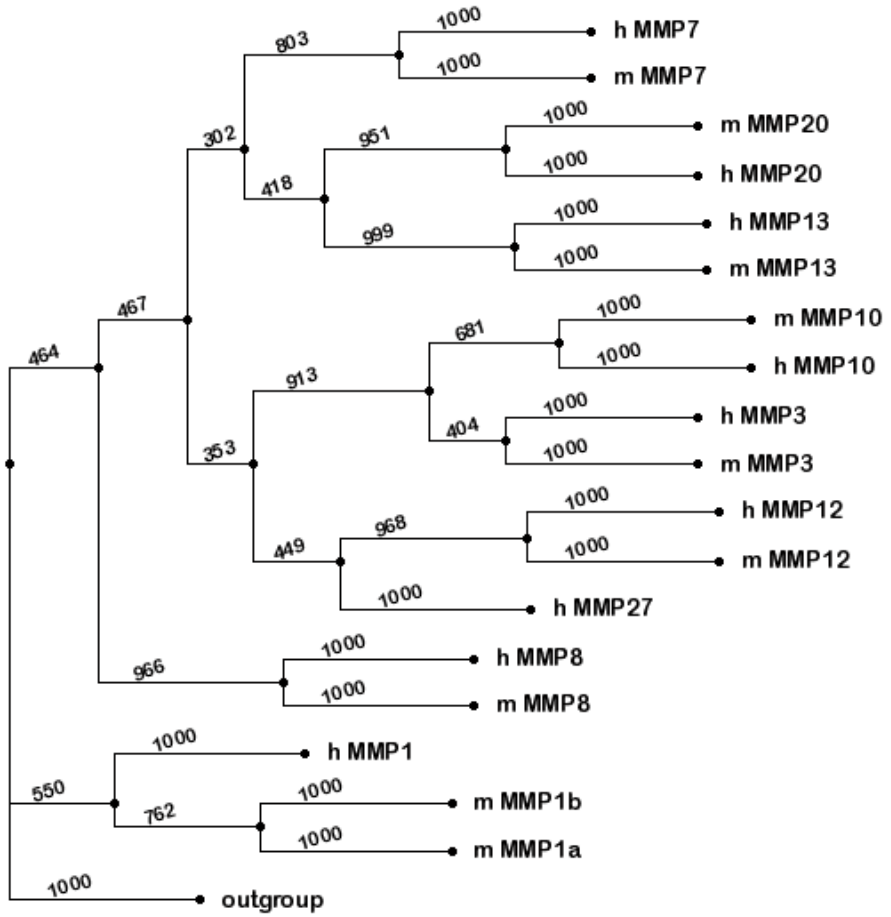
Table 2.8. Chemokine ligand (CCL2) gene cluster. Gene cluster regions are hg17:chr17:$31,046,663 - 31,979,996$ and mm5:chr11:$82,982,470 - 83,925,219$.



Orthologous pairs of IFN gene cluster

human	mouse
IFNE1_p: 21,470,841-21,472,312	IFNT1: 87,188,249-87,188,912
IFNB1: 21,067,105-21,067,932	IFNB: 86,796,207-86,796,976
IFNA1: 21,430,455-21,431,315	IFNA1: 87,158,798-87,159,367
IFNA2: 21,374,429-21,375,387	IFNA2: 86,958,302-86,958,874
IFNA4: 21,176,693-21,177,670	IFNA4: 87,150,572-87,151,132
IFNA5: 21,294,325-21,295,087	IFNA5: 87,144,012-87,144,582
IFNA6: 21,340,317-21,340,886	IFNA6: 87,135,834-87,136,786
IFNA7: 21,191,468-21,192,204	IFNAB: 86,966,353-86,966,928
IFNA8: 21,399,133-21,400,174	IFNA9: 86,865,995-86,866,567
IFNA10: 21,196,180-21,197,142	IFNA11: 87,128,375-87,129,859
IFNA13: 21,357,423-21,358,056	IFNA12: 86,876,762-86,877,558
IFNA14: 21,191,234-21,229,990	IFNA13: 86,917,826-86,918,644
IFNA16: 21,206,372-21,207,310	IFNA14: 86,845,411-86,845,980
IFNA17: 21,217,242-21,218,221	
IFNA21: 21,155,636-21,156,619	

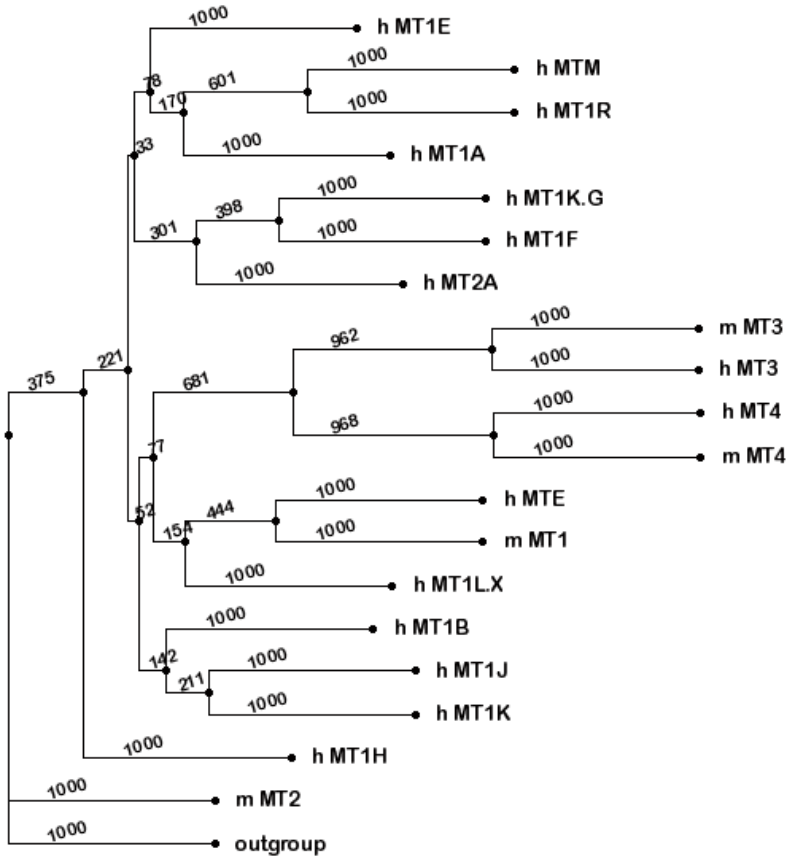
Table 2.9. Interferon (IFN) gene cluster. Gene cluster regions are hg17:chr9:$21,000,000 - 21,500,000$ and mm5:chr4:$86,759,151 - 87,237,150$.



Orthologous pairs of MMP gene cluster

human	mouse
MMP1: 102,165,857-102,174,075	MMP1b: 7,343,902-7,364,454 MMP1a: 7,440,602-7,453,317
MMP7: 101,896,450-101,906,672	MMP7: 7,668,562-7,675,716
MMP8: 102,088,540-102,100,868	MMP8: 7,534,924-7,544,489
MMP10: 102,146,443-102,156,556	MMP10: 7,478,803-7,486,703
MMP12: 102,238,686-102,250,889	MMP12: 7,323,111-7,334,339
MMP13: 102,318,934-102,331,648	MMP13: 7,248,056-7,258,952
MMP20: 101,953,058-102,001,273	MMP20: 7,604,701-7,651,440

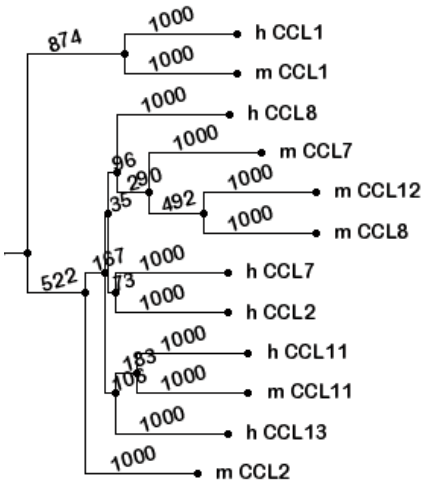
Table 2.10. Matrix metalloproteinase (MMP) gene cluster. Gene cluster regions are hg17:chr11:$100,575,000 - 103,575,000$ and mm5:chr9:$6,014,643 - 8,900,000$.



Orthologous pairs of MT gene cluster

human	mouse
MT3: 55,180,957-55,182,497	93,499,814-93,501,252
MT4: 55,156,542-55,160,345	MT4: 93,484,346-93,485,971

Table 2.11. Metallothionein (MT) gene cluster. Gene cluster regions are hg17:chr16:$55,000,000 - 55,300,000$ and mm5:chr8:$93,340,329 - 93,651,439$.



Orthologous pairs of CCL1 gene cluster

human	mouse
CCL1: 29,711,512-29,714,343	CCL1: 81,789,993-81,793,144

Table 2.12. Chemokine Ligand (CCL1) gene cluster. Gene cluster regions are hg17:chr17:$29,526,113-29,837,223$ and mm5:chr11:$81,566,149-81,834,148$.

2.2.2 Beta globin gene clusters of 19 species

There are several serious deficiencies with using the pairwise orthology assignments as presented in the previous section to evaluate orthologous alignments in later chapters. First of all, only branches with high bootstrapping values are considered. It is reasonable to conjecture that it is relatively easy for aligners to perform well in these "good" cases. Some clusters, such as the metallothionein gene cluster in Table 2.11 and the chemokine ligand gene cluster in Table 2.12, do not have many branches with high bootstrapping values, and the sets of inferred orthologs are not representative of these clusters. However, it is these "bad" cases in which an aligner will most probably make mistakes in doing orthologous alignment. Secondly, the wrong assignment of orthologous pairs in cases of gene loss sometimes cannot be avoided using only two species, as explained in Sec. 1.2 (Fig. 1.3). Thirdly, the available gene annotations are limited. For example the sets of alpha globin genes of mouse in Table 2.2 and beta globin genes of mouse in Table 2.5 are not complete. The missing genes not only decrease the utility of these clusters for evaluation purpose, but also cause problems similar to those from gene loss. Also, as genome projects progress and more annotations are become available, the limitation of gene annotations for less studied species will continue to arise.

One of the ENCODE [43] regions (ENm009) contains the beta globin gene cluster, for which sequences from 19 vertebrate species were available when we started this project. We can use the genomic sequences from this set to perform orthology assignment among beta globin genes. To achieve the most accurate possible orthology assignment for as many genes as possible from these clusters, we try to maximize the usage of available sequences, with input from the literature [29, 62]. We first construct the gene tree from available sequences, and next group genes according to branches of the tree, taking into account both the bootstrapping values as an estimate of reliability and the genomic location information of each gene. We then construct evolutionary event trees using the parsimony principle to infer the relationship of branches with low bootstrapping values. This process leads to orthology assignments for all genes. Below, we start with a brief review of the evolutionary history of beta globin genes before giving details of these procedures.

2.2.2.1 Evolution of beta globin genes

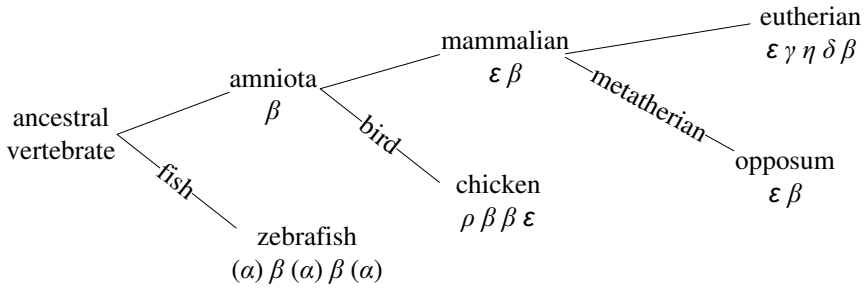


Figure 2.2. Evolutionary history of beta globin genes.

We use β, δ, η, γ and ϵ to denote different types of genes of the beta globin gene cluster. There

is another group of globin genes that are called alpha globin genes. Beta globin genes and alpha globin genes share an ancient evolutionary origin. The ancestor of beta globin genes and the ancestor of alpha globin genes were originally adjacent in a very early ancestral vertebrate before the species split between fish and amniotan (including birds and mammals). This property has persisted in some fish. For example in zebrafish, duplicated beta globin genes and alpha globin genes are interleaved in an array. On the other hand, the ancestral genes for beta and alpha globin genes were spatially separated in an ancestral amniota, before the species split between birds and mammals. We are only interested in the beta globin genes in this chapter, so alpha globin genes are not discussed further.

There is strong evidence that duplications forming the several beta globin genes in birds, such as chicken, happened after the species split between birds and ancestral mammals, as did the copies in mammals, since each beta globin gene in chicken has the same degree of similarity to each copy in mammals, such as human. It is believed that an ancestral β and an ancestral ϵ were already formed in an ancient mammal, from which metatherians and eutherians (i.e., placental mammals) descended. These two copies remained in some marsupials such as opossum. On another hand, in an ancestral eutherian, the ancestral β was duplicated to form the modern β and δ, and the ancestral ϵ was duplicated to form η, γ, and ϵ. These genes formed a tandem array, and their order remained the same within most mammalian lineages. There were subsequent tandem duplications (e.g., γ was duplicated in an ancestral primate), gene losses (e.g., γ in goat was lost), and gene conversions (e.g., δ of galago is replaced by its β). Details of the evolution of the beta globin cluster can be found in [29, 62]. Relevant details will be cited in the following sections where they are needed.

2.2.2.2 Construction of the gene tree of 19 species

A phylogenetic gene tree is constructed based on a multiple alignment, in which all bases from a given column are assumed to descend from the same ancestral base. In other words, an accurate multiple alignment is the most basic assumption in phylogenetic tree reconstruction. An important factor in building a reliable phylogenetic tree is thus that sequences are very conserved, such that the alignment of every column in the multiple alignment is reliable. This requires consistent boundaries of conserved regions. The complete sequences of homologous proteins are good examples of conserved sequences that have consistent boundaries. But many species' protein annotations are not available for the beta globin cluster. In some species, the genomic sequences of the beta globin clusters are not even complete and/or the genomic sequences of these genes do not have consistent boundaries. A feasible solution is to use a sub-region of a beta globin gene that is long enough for phylogenetic analysis, and well conserved among all beta globin genes that are available from ENCODE sequences. Extensive manual processing is required to determine such sub-sequences. In this example, we started by using HomologMiner [32] to obtain rough homologs, followed by extensive manual trimming of each sequence to make sure boundaries of sub-sequences are consistent, and that a global alignment tool such as ClustalW [65] performs well for these sequences. We finally obtained 90 sequences of 19 species, each of around 141 bases,

which corresponds to the second exon of a beta globin gene. The phylogenetic gene tree was then constructed in the same way as the pairwise ones were constructed in Sec. 2.2.1, except that the bootstrapping was performed with 100 instead of 1000 replicates to keep it computationally economical. The numbers of sequences from each species are summarized in Table 2.13.

Table 2.13. Summary of the numbers of gene sequences used to construct the phylogenetic gene tree of 19 species. Each sequence corresponds to the second exon of a beta globin gene. It is the longest conserved region available in the greatest number of beta globin genes from the genomic sequences provided in ENCODE ENm009. For example, rabbit has 4 beta globin genes, but there are only 2 copies in this table, because its other beta globin genes are missing from the sequences provided by ENCODE, or the regions of their second exons are not well sequenced. Although there are relatively fewer sequences from this region in rabbit, the region of the second exon is the most representative for all beta globin genes of 19 species.

species	# of identified beta globin gene sequence
human	6
chimp	6
macaque	4
baboon	6
marmoset	6
galago	5
mouse	6
rat	9
rabbit	2
bat	4
dog	4
shrew	4
armadillo	4
tenrec	5
elephant	4
monodelphis	2
platypus	1
chicken	4
zebrafish	8
total	90

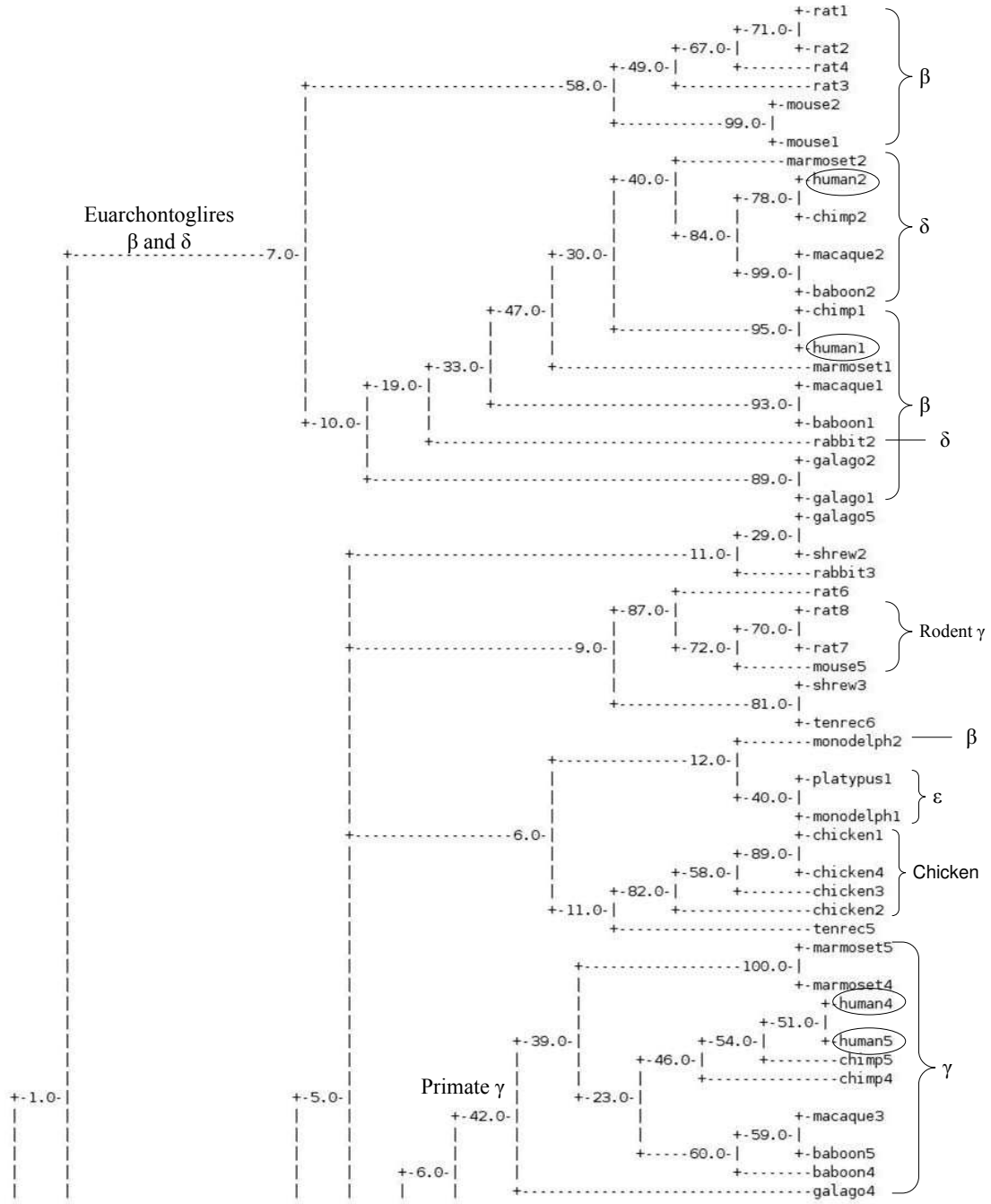


Figure 2.3. Phylogenetic gene tree of the beta globin cluster of 19 species labeled with major gene/species groups (part I). Human genes are circled.

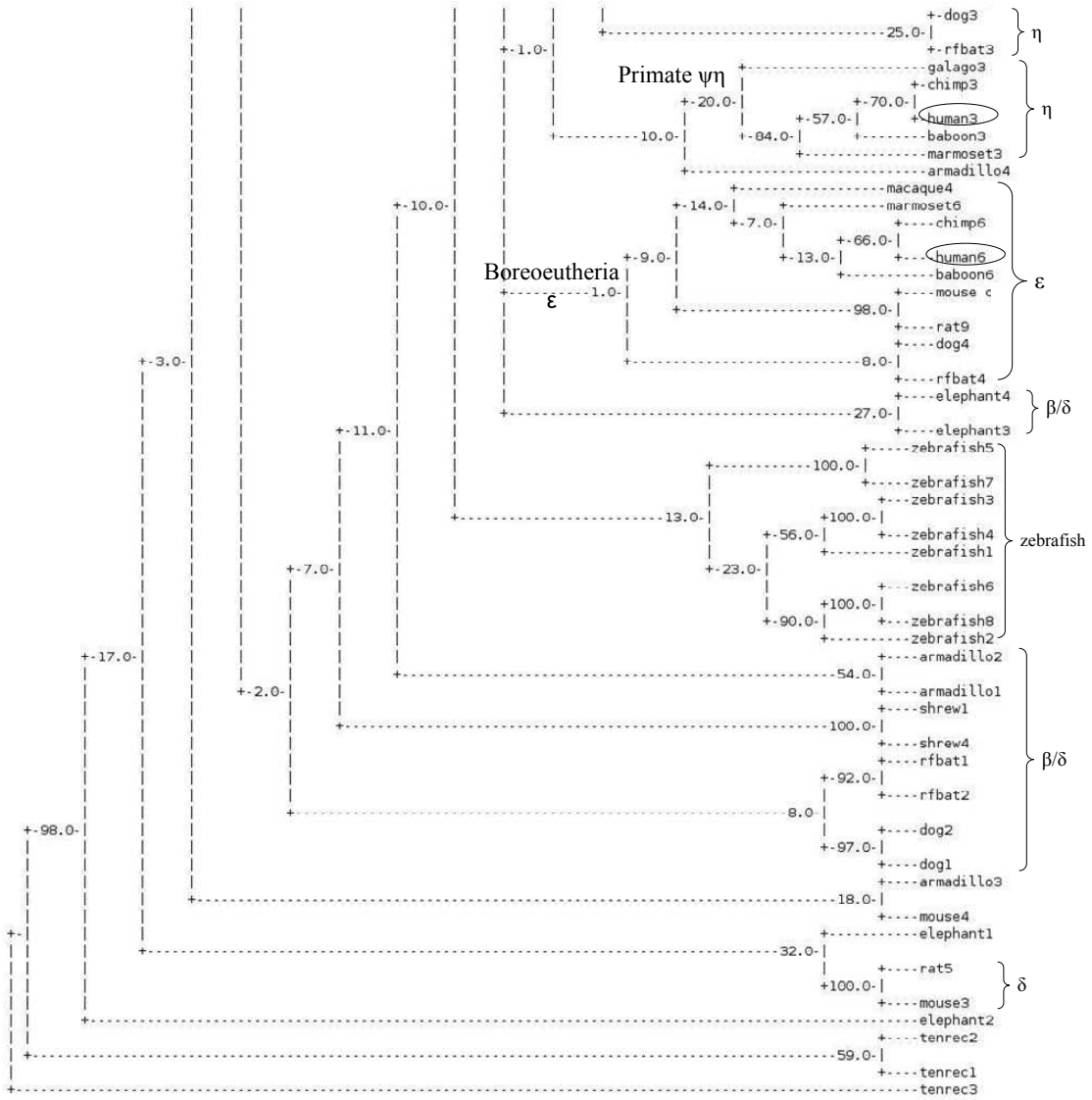


Figure 2.4. Phylogenetic gene tree of the beta globin cluster of 19 species labeled with major gene/species groups (part II). Human genes are circled. The branches of genes of zebrafish contain alpha globin genes of this species.

2.2.2.3 Interpretation of the gene tree and construction of evolutionary event trees

The resulting gene tree in Figs. 2.3 and 2.4 is very complicated. It provides opportunities for reconciliation with a known species tree (Fig. 2.5) determined by the ENCODE project [43], such that the inference of orthologs will be more reliable. However, at the same time it is also more difficult to interpret the tree since the bootstrapping values tend to be small. With results from [29] and [62], we are able to interpret this tree with some confidence. For simplicity, all genes referred to in this section are beta globin genes unless specified otherwise. There are several types of beta globin genes: $\beta, \delta, \eta, \gamma, \epsilon$ which are all derived from a single ancestral gene. This ancestral gene evolved to form two genes, which were the ancestral genes for β, δ and η, γ, ϵ respectively. Denote these two ancestral genes as β^* and ϵ^*. These two genes were formed after the species split between birds and mammals. It is also obvious from the tree that genes of zebrafish and chicken duplicated independently. Hence, all genes of zebrafish and chicken are orthologous to all genes of the other species. It needs to be noted that in the branches of genes of zebrafish, a couple of bootstrapping values are quite low. It turned out that the beta and alpha globin genes of zebrafish are in the same cluster. Its alpha globin genes are mistakenly included in this phylogenetic reconstruction, and their bootstrapping values are low. [62] also suggests that the ancestral genes β^* and ϵ^* were formed before eutherian mammals appeared. From the species tree in Fig. 2.5, platypus and monodelphis are mammals, but not eutherians. The tree in Fig. 2.3 suggests that platypus1 and monodelph1 may be from ϵ^*, and monodelph2 may be from β^*. The descendent of β^* in platypus is missing from the available sequence, but we cannot tell if it is a gene loss or not until we can get a sequence of better quality. It is quite possible that it is missing because of incomplete sequence, since the available sequence breaks around platypus1. All of the types of beta globin genes were already formed in an ancestral eutherian, as shown in Fig. 2.2. [62] suggests that η is missing from rabbit and mouse. Most β and δ genes of euarchontoglires are in a same sub-tree in Fig. 2.3. Within this sub-tree, there is a branch of δ genes of primates, except for galago, whose δ gene is believed to be replaced by its β gene as a result of recent gene conversion. There is another group of β and δ genes in Fig. 2.4 for other boreoeutherians. In this group, it is more difficult to differentiate β and δ genes, since each pair of leaf nodes are from the same species with high bootstrapping values, which suggest many independent gene conversions. According to [62], the ancestral γ gene was duplicated in the ancestor to simian primates, which is consistent with the sub-tree at the bottom of Fig. 2.3, where galago has only one copy. Macaque also has only one γ gene, but most probably because of incomplete sequence, since its sequence breaks around the available γ gene at the side where there is supposed to be one more copy. There is a branch in this sub-tree that has bootstrapping value of 100 between the two copies of marmoset, which suggests a very recent gene conversion between them. Two sub-trees in Fig. 2.4 includes ϵ and η genes respectively.

After identifying major groups of different types of beta globin genes, we try to infer major evolutionary events including duplications, conversions and gene losses, using the principle of parsimony, i.e., the least complex explanation of the evolutionary events. We carefully differen-

tiate gene loss and incomplete sequence based on the quality of the sequences. Since it is quite obvious that $\beta, \delta, \eta, \gamma,$ and ϵ were formed at an ancestor to eutherians, it is reasonable to analyze each gene separately. Since there are frequent gene conversions between β and δ, we discuss them together. Since ϵ remains a single copy in all mammalian species, we skip analyzing this type of gene. The evolutionary events are then labeled in Figs. 2.6, 2.7 and 2.8.

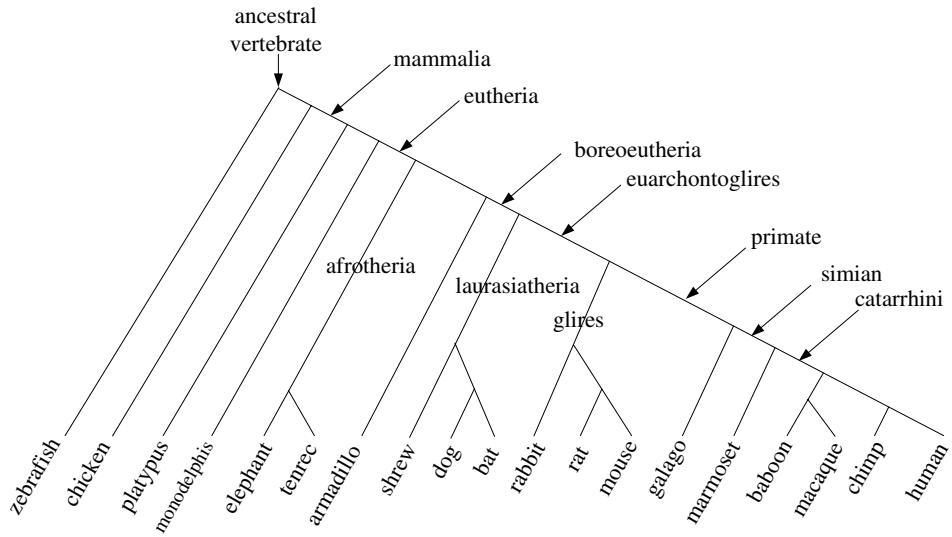


Figure 2.5. Species tree of 19 species (adapted from [43]).

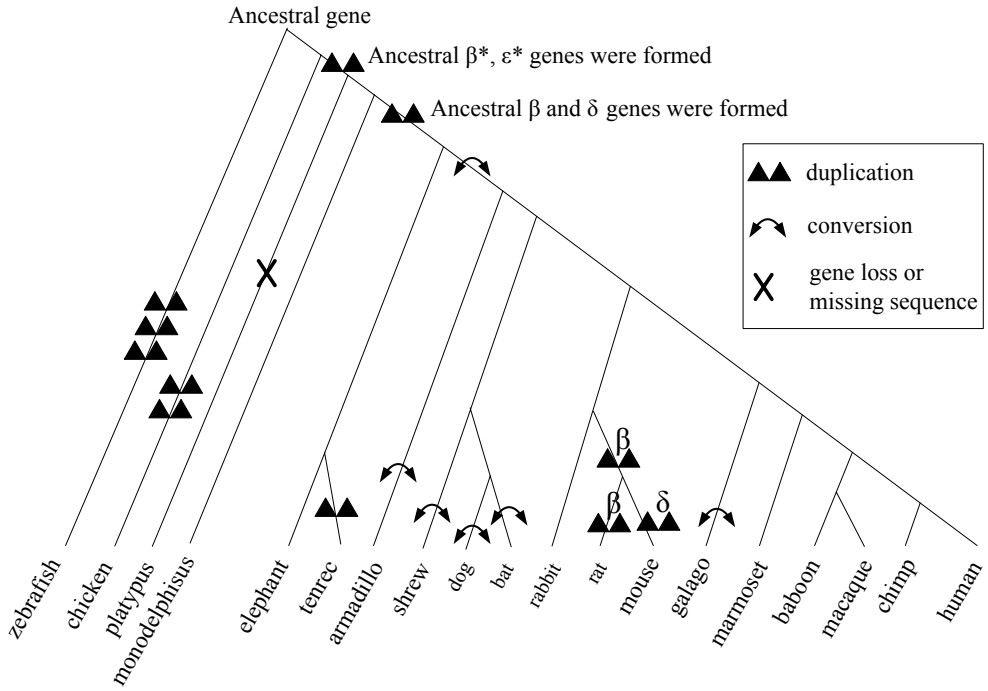


Figure 2.6. The species tree labeled with evolutionary events of β and δ genes.

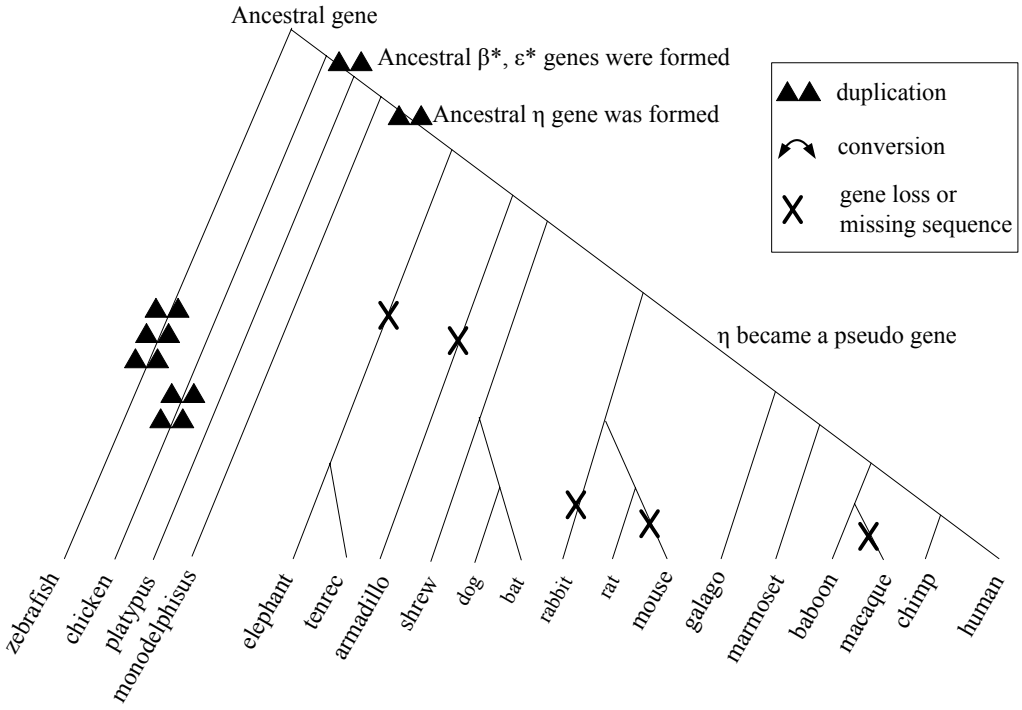


Figure 2.7. The species tree labeled with evolutionary events of η genes.

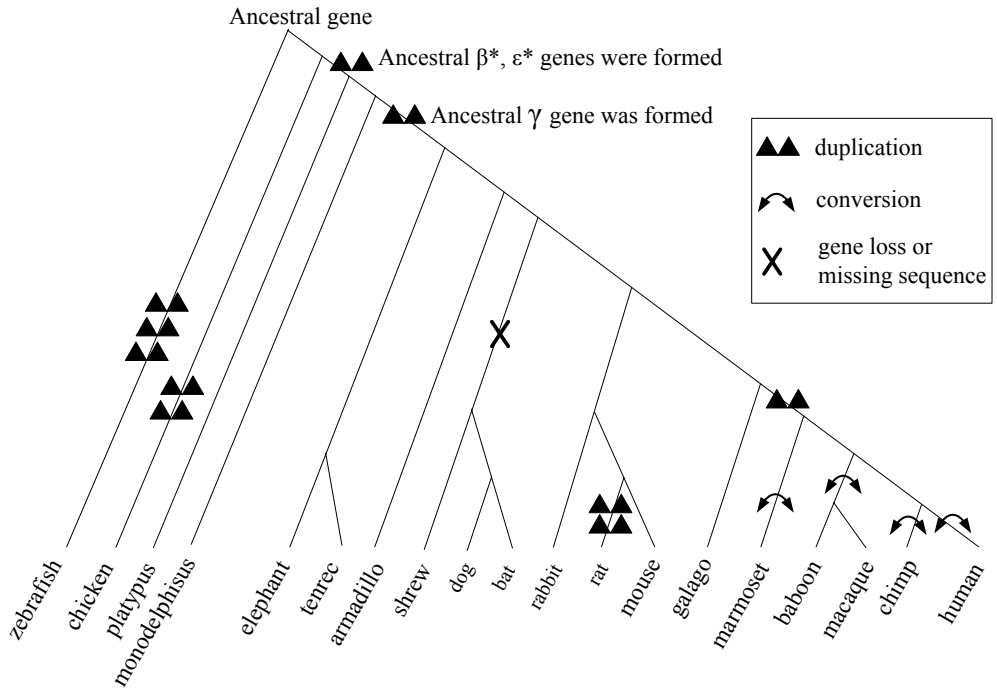


Figure 2.8. The species tree labeled with evolutionary events of γ genes.

2.2.2.4 Orthology assignment

The above analysis helps us to determine the orthology relationships among these genes. For example, since the duplication of the γ gene happened in the ancestor to simian primates, we can conclude that all copies of γ in simian primates are orthologous to the γ genes of non-simian primates, such as galago. However, because of frequent gene conversion events, it becomes extremely difficult to determine true orthologs in some situations. For example, suppose the two ancient copies of γ genes formed in the ancestor to simian primates are γ_A and γ_B. We do not know the direction of gene conversion in the speciation of marmoset, neither do we know it in the speciation of catarrhini (human, chimp, macaque, baboon). If the gene conversion directions are different, then there are no true orthologs between γ genes of baboon and γ genes of human. Nevertheless, γ genes of baboon are still the best candidates for being orthologous to γ genes of human, and there is no difference between the relationship of either copy of γ genes of baboon to either copy of γ genes of human. In this case, we still regard it as a many-to-many orthology relationship. In other words, we try to assign orthologous pairs as long as there is no evidence not to do so based on the gene tree, the species tree, the genomic location of genes, and information from the literature.

From the above analysis, we derive the orthology relationships among beta globin genes of 19 species. To make this task less complicated, we organize orthology assignment of these genes using human as reference. They become benchmarks to which we compare our orthologous multiple alignments in Chapter 4.

Table 2.14. Orthology assignment of beta globin genes of 19 species. ♡ denotes gene loss due to gene conversion; ♢ denotes other kinds of gene loss; ♣ denotes missing sequence; ♠ denotes incomplete sequence. Some missing sequences are not denoted if there is another paralogous sequence of the same species that is orthologous to the human sequence; for example one copy of the γ gene of macaque is missing but not noted in this table.

human	human1 (β)	human2 (δ)	human3 ($\psi\eta$)	human4 (γ_1)	human5 (γ_2)	human6 (ϵ)
chimp	chimp1	chimp2	chimp3	chimp5	chimp5	chimp6
macaque	macaque1	macaque2	♠	macaque3	macaque3	macaque4
baboon	baboon1	baboon2	baboon3	baboon4 baboon5	baboon5 baboon4	baboon6
marmoset	marmoset1	marmoset2	marmoset3	marmoset4 marmoset5	marmoset5 marmoset4	marmoset6
galago	galago1 galago2	♡	galago3	galago4	galago4	galago5
mouse	mouse1 mouse2	mouse3	♢	mouse5	mouse5	mouse6
rat	rat1 rat2 rat3 rat4	rat5	rat6	rat7 rat8	rat8 rat7	rat9
rabbit	♠	rabbit2	♢	♠		rabbit3
bat	rfbat1 rfbat2	rfbat2 rfbat1	rfbat3			rfbat4
dog	dog1 dog2	dog2 dog1	dog3	♢		dog4
shrew	shrew1 shrew4	shrew4 shrew1	shrew2			shrew3
armadillo	armadillo1	armadillo2		armadillo3	armadillo3	armadillo4
tenrec	tenrec1 tenrec2 tenrec3	tenrec1 tenrec2 tenrec3	♢	tenrec5	tenrec5	tenrec6
elephant	elephant2	elephant1		elephant3	elephant3	elephant4
monodelphis	monodelph2	monodelph2	monodelph1	monodelph1	monodelph1	monodelph1
platypus	♣		platypus1	platypus1	platypus1	platypus1
chicken	chicken1 chicken2 chicken3 chicken4	chicken1 chicken2 chicken3 chicken4	chicken1 chicken2 chicken3 chicken4	chicken1 chicken2 chicken3 chicken4	chicken1 chicken2 chicken3 chicken4	chicken1 chicken2 chicken3 chicken4
zebrafish	zebrafish1 zebrafish2 zebrafish3 zebrafish4 zebrafish6 zebrafish8	zebrafish1 zebrafish2 zebrafish3 zebrafish4 zebrafish6 zebrafish8	zebrafish1 zebrafish2 zebrafish3 zebrafish4 zebrafish6 zebrafish8	zebrafish1 zebrafish2 zebrafish3 zebrafish4 zebrafish6 zebrafish8	zebrafish1 zebrafish2 zebrafish3 zebrafish4 zebrafish6 zebrafish8	zebrafish1 zebrafish2 zebrafish3 zebrafish4 zebrafish6 zebrafish8

Chapter 3

Pairwise orthologous alignment

3.1 Overview

Alignments and genomic aligners are introduced and discussed in Section 1.3 of Chapter 1. The ENCODE project [43] uses three major multi-species genomic aligners: TBA, MLAGAN and MAVID. The ENCODE data analysis is based on a combination of the multiple alignments produced by these aligners. All of these multiple alignment tools start with pairwise alignments. We observe below that none of these three tools provides adequate alignments of tandem gene clusters. For example, the human beta globin cluster has 6 genes, while mouse has 7 genes. Fig. 3.1 provides an orthology map of these genes. Genes with the same pattern in Fig. 3.1 from different species are believed to be orthologous.

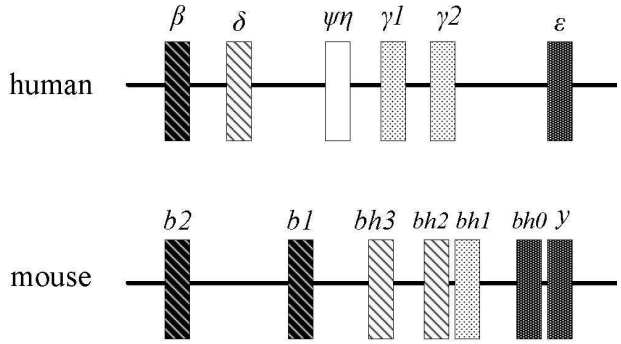
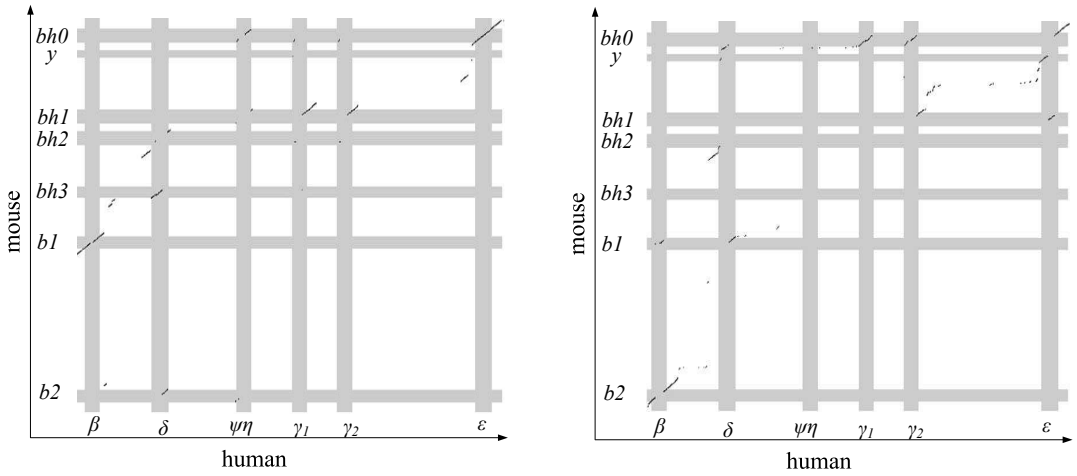


Figure 3.1. Orthology map of beta globin genes of human and mouse. Adapted from [29].

Fig. 3.2 displays the inadequate performance of TBA and MLAGAN. MAVID does not produce alignments within this region and is not shown here. In the TBA alignments (Fig. 3.2(a)), human δ is split into two parts, one part is aligned to part of mouse *bh2*, and another part is aligned to part of mouse *bh3*. Part of human $\psi\eta$ is aligned to mouse y, which is not orthologous at all. In the MLAGAN 3.2(b) alignments, human δ is also split into two parts, one is aligned



(a) Dot-plot of pairwise alignments between human and mouse in the beta globin region, as extracted from TBA multiple alignment of ENCODE region ENm009.

(b) Dot-plot of pairwise alignment between human and mouse in the beta globin region, as extracted from MLAGAN multiple alignment of ENCODE region ENm009.

Figure 3.2. Inadequate performance of major genomic aligners.

to part of mouse *bh2*, and another part is aligned to part of mouse b1, which is not orthologous. The two human γ genes are also aligned to incorrect places in the mouse sequence. Thus, neither of these aligners does a uniformly good job on genomic regions with tandem duplications, though in the beta globin gene cluster the alignments from TBA look better.

A traditional orthology assignment is achieved by phylogenetic analysis, which is quite time-costly, and thus less favorable for our goal of large-scale alignment. Approaches to this type were pioneered by COG [64], followed by INPARANOID [57], OrthoMCL [40] and OrthoParaMap [13]. Among them, INPARANOID limits input to two species. COG first identifies consistent orthologs among three species, which form a triangle; it then fuses any two triangles sharing a common edge. Each COG group is composed of orthologs and paralogs from at least three distinct lineages. OrthoMCL is based on graph flow theory. OrthoParaMap involves phylogenetic reconstruction. All of these methods are based on protein sequences. Orthology assignment is simpler in certain ways for protein sequences than for genomic sequences. A complete protein is a natural unit for orthology assignment. However, duplications can happen at any level of genome structure, from a partial gene to a whole genome, covering both non-coding regions and coding regions, making it unclear which intervals are to be considered. Indeed, when dealing with genomic sequences, the problem of orthology assignment of genes from different species is tightly linked to the problem of finding appropriate regions to align among those species.

The mutational process of *gene conversion*, i.e., when one gene sequence replaces a similar sequence, raises interesting issues for orthology assignment. For example, consider the beta globin gene clusters of galago (a primitive primate) and rabbit. Fig. 3.3 shows the pairwise alignments computed by BLASTZ for genomic intervals containing those clusters. The cluster contains the β, δ, γ, and ϵ genes, as labeled along the axes. The β and δ genes were created (by duplication of

a β/δ precursor) in an ancestor of mammals, which preceded the galago-rabbit split. Therefore the δ genes for galago and rabbit should have been orthologs. However, based on the pairwise alignment scores of galago vs. galago (alignments not shown) and galago vs. rabbit, galago's β and δ genes seem to have resulted from a duplication that occurred after the galago-rabbit split. (That is, galago δ is more similar to galago β than to rabbit δ.) Furthermore, based on more complete evidence, it is believed that galago's δ gene was "overwritten" by its β gene just a few million years ago. This raises the question of what we should take as the ortholog in rabbit of the galago δ. Should it be rabbit β, based on gene locations, or rabbit δ, based on gene content? Here, we regard such a conversion as a recent duplication, and assign orthologs based on gene content. This choice is more consistent with Fitch's distinction between orthology and paralogy, and it provides alignments that are appropriate for functional inference and phylogenetic analysis.

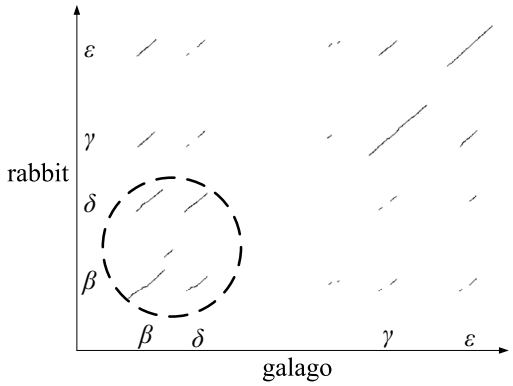


Figure 3.3. Dot-plot of pairwise alignments of beta-globin gene clusters from galago and rabbit. Alignments in the oval illustrate an observation (see text) about gene conversions.

It is difficult to properly align several genomic sequences that contain intra-species duplications, as we see from the inadequate performance of several major genomic aligners on the beta globin cluster, above. With the goal of properly aligning two genomic sequences from species that contain duplications, we have developed a tool, called TOAST (two-way orthologous alignment selection tool), for predicting whether two aligned regions from different species are orthologous, i.e., separated by a speciation event, as opposed to a duplication event. Orthologous pairs constitute the aligning regions that are

1. most likely to share the same biological function,
2. most easily analyzed for evidence of selection, and
3. appropriate for the traditional "progressive" alignment strategy.

Evaluating the success of our approach proved challenging, due to a shortage of accurate annotation for available vertebrate genome sequences. We present specificity/sensitivity measurements for TOAST obtained on 12 human gene clusters for which the orthologous mouse

clusters appear to have a different number of members. The results indicate that our software has a high degree of specificity and an adequate level of sensitivity for the problem of identifying orthologs. This position on the sensitivity/specificity tradeoff curve, i.e., high specificity albeit with modest sensitivity, is optimal for using TOAST in a multi-alignment package, as we will show in Chap. 5.

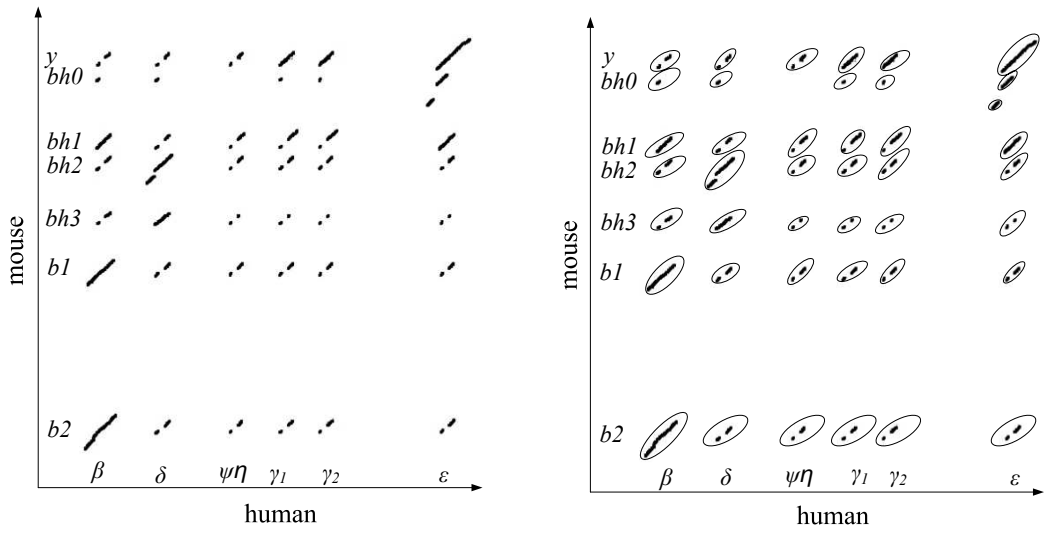
TOAST is essential for our next goal of determining sets of multiple alignments in which two segments are aligned if and only if they are orthologous (Chapter 4 and 5). We describe the details of methods used by TOAST in Section 3.2, and then show the results of evaluating TOAST on 12 human/mouse gene clusters in Section 3.3.

3.2 Methods

We use the program BLASTZ [58] to align each of the input sequences to itself and to each of the others. Each of these steps produces a set of two-way alignments in which a given position can be aligned several times and/or involve the reverse complement of an input sequence. These "raw" pairwise alignments are processed by TOAST, which retains only putative orthologous alignments (e.g., darker lines in Fig. 1.4). In outline, TOAST collects matches (local alignments) found by BLASTZ into "chains", organizes the chains using a form of graph, and selects a subset of the graph's edges that constitute the set of putative orthologous matches. The components of TOAST are described below in detail. We use the pairwise alignments between human and mouse over the beta globin cluster as an example, unless specified otherwise.

3.2.1 Chaining using a k-d tree

In a genome, some parts of a functional unit can be less conserved than other parts (e.g., introns of a gene), and hence may not be detected by a pairwise alignment tool (BLASTZ in our example). Thus, in a dot-plot graph, the detectable matches between homologous biologically meaningful units may be broken into smaller pieces along roughly the same diagonals, as shown in Fig. 3.4(a). We want to chain these small pieces to form complete working units. A k-d tree [6], where $k = 2$ in this thesis, is used to efficiently identify the chains. For a sketch of how this works, consider the matches that do not involve reverse-complemented intervals. (The chaining procedure is applied independently to both orientations of the second sequence.) A k-d tree is constructed for the whole set of BLASTZ alignments, where a fixed number of alignments are placed in the same "bucket". Each alignment has two end-points, a *start point* at its lower left (in the dot-plot) and an *end point* at its upper right. Define a *predecessor* p of an alignment t, which is in bucket b, to be an alignment in b, or a bucket that is located below b or left to b, such that the end point of p is located left of and below the start point of t, allowing a small overlap. To reduce the computation time, the distance between p and t needs to be within a threshold d (e.g. 3000 bases). Among all the predecessors of an alignment, the *best predecessor* maximizes the sum of the two alignment scores minus a specified penalty that reflects the distance between



(a) Dot-plot of BLASTZ alignments of beta-globin clusters of human and mouse.

(b) The same alignment, where each oval encloses a chain.

Figure 3.4. Chaining BLASTZ alignment blocks.

the end-points. The best predecessor is determined for every alignment, unless there is no best predecessor for an alignment when certain thresholds are enforced or there is no predecessor at all. Alignments are connected into chains according to best predecessors; alignments without best predecessors start new chains. Thresholds for block gap penalty, and/or largest gap length are determined empirically. Fig. 3.4(b) shows an ideal chaining result.

3.2.2 Chaining using gene annotations

The purpose of chaining is to organize the fragmented local alignments into appropriate larger chains such that each chain represents the alignment between functional units from two species. Complete genes are natural functional units that are good for our goal of assigning orthologs. The chaining process in Section 3.2.1 uses only the local pairwise alignments as input, without any biological information. Sometimes the above chaining method fails to form chains long enough to include whole genes. For example, when introns of a gene are very long and diverged, the gap between two diagonal lines that are alignments of two adjacent exons may exceed the threshold, and these two diagonal lines are then not chained, though they belong to the alignments between one pair of orthologous genes.

Some genomes, e.g. human, have extensive annotation of genes, in which case it is straightforward to force the chain to contain the fragmented alignment blocks that belong to a complete gene. For example, in Section 3.2.1, the search space for the best predecessor of alignment block t is restricted to be within the threshold of distance d (e.g. 3000 bases). Given the gene annotation, we extend the search space to be the range of the gene that t belongs to, if it is longer than d.

3.2.3 A graph model

A graph is composed of nodes and edges. Let nodes represent genes, and edges represent alignments between genes. This graph is then called the *alignment graph*. Since our alignments are BLASTZ local alignments, edges represent homologous relationships among genes. Let's consider the graph for two species. If we partition the nodes into two parts, such that each part contains nodes (genes) from one species, then the edges within each part represent self-alignments of that species, and the edges crossing between the two parts represent inter-species alignments. Fig. 3.6 is an example of such a graph. We need to select a subset of cross-edges that represent orthologous inter-species alignments. In practice, even if gene annotations are available we still need to make some effort to determine nodes, since a node in the graph that we actually analyze needs to correspond to a chain of local alignments and hence might contain upstream regulatory regions for the gene, or several genes. In other words, TOAST is a genomic sequence aligner, and it aligns more than just genes, including non-coding conserved regions. In the following sections, we provide details about how to construct the graph and select cross-edges that represent orthologous alignments.

3.2.4 Determining graph nodes from pairwise alignments

Fig. 3.4 shows that each of the human genes denoted β, δ, γ_1, γ_2 and ϵ (from the beta globin cluster) is aligned to each of mouse *y*, *bh0*, *bh1*, *bh2*, *bh3, b1* and *b2* genes. The precise boundaries of each gene are not consistently indicated by all of its alignments. For example, although alignments of human β to each of these mouse genes roughly overlap β, their start positions differ, as do their end positions. We need to determine the boundaries of each homologous region. Fig. 3.5 illustrates the method. After projecting the chains onto human, we get a distribution of horizontal lines. These lines can be smoothed by an imaginary curve. Each curve peak represents the central point of a potential working unit. The boundaries of the working unit are determined from the wave hollows. For a fixed species pair, this process is applied separately to each species.

Each working unit becomes a node (Fig. 3.6). As just summarized, TOAST uses both self- and cross-alignments to determine the set of such nodes for each species. For example, human-vs.-human self-alignments and human-vs.-mouse cross-alignments are used to determine the set of nodes for human. In cases where adequate gene annotation exists, the annotated boundary information is also incorporated into forming nodes; an annotated gene cannot be included in more than one node where each node contains a partial gene. It needs to be emphasized that the working unit is not necessarily a gene. It may be part of a gene or a regulatory unit, or may cover several genes or regulatory units. This is in accordance with Fitch's view [24]. Using a homologous region as a working unit is different from approaches taken by most other orthology-assignment tools [68, 13, 57, 40]. These tools target whole genes or gene products. Our approach makes TOAST a more powerful tool to do orthology assignment within genomes.

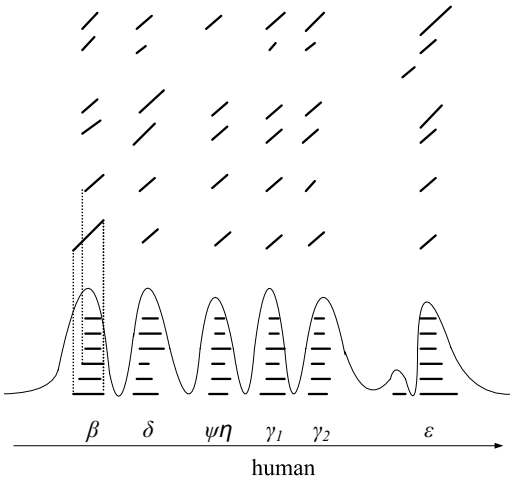


Figure 3.5. Determining boundaries of conserved regions.

3.2.5 Building edges

The chains of self-alignments become edges among the set of nodes for the same species, and the chains of cross-alignments become edges between nodes for two different species, say A and B. The *score* of each edge is the sum of scores of all BLASTZ blocks of the chain associated with the edge. The BLASTZ score penalizes substitutions, insertions and deletions [58]. Each edge is also associated with a parameter of *score per base (spb)*, which is defined to be $2 \times score/(length\_A + length\_B)$, where *length_A* and *length_B* are lengths of the associated chain in species A and B, respectively. (For a self-alignment, A and B are the same species.) To avoid false positive alignments, TOAST ignores chains whose length or *sbp* falls below specified thresholds. Fig. 3.6 depicts the conceptual model of the graph.

3.2.6 Determining in-paralogous alignments

At this point, a graph has been formed (Fig. 3.6). Our goal is to select a set of cross-edges corresponding to orthology relationships. As a preliminary step, we need to select a set of self-edges showing in-paralogous relationships for each species. (The reason for doing this will be obvious when we explain the next procedure.) We use the definition of in-paralog given by Remm [57], namely "paralogs that arose after the species split". (Thus, in-parlogs in species A depend on the choice of a second species.) In a normal evolutionary scenario, the similarity level between two in-paralogs in species A is higher than between one of the in-paralogs in species A and its ortholog in species B, since the orthologous copy in B diverged earlier than the two in-paralogous copies in A. For a given node N, we remove every self-edge whose *spb* is below the highest *spb* among N's cross-edges. The remaining self-edges indicate potential in-paralogous relationships. The figure below shows the graph after selecting self-edges. It indicates that human $\gamma 1$ and $\gamma 2$, as well as mouse *b1* and *b2*, are recent duplicates. The dotted lines between mouse *bh3* and *bh2*, and *bh0* and *y* show self-alignments that are removed by our method, but some literature [29]

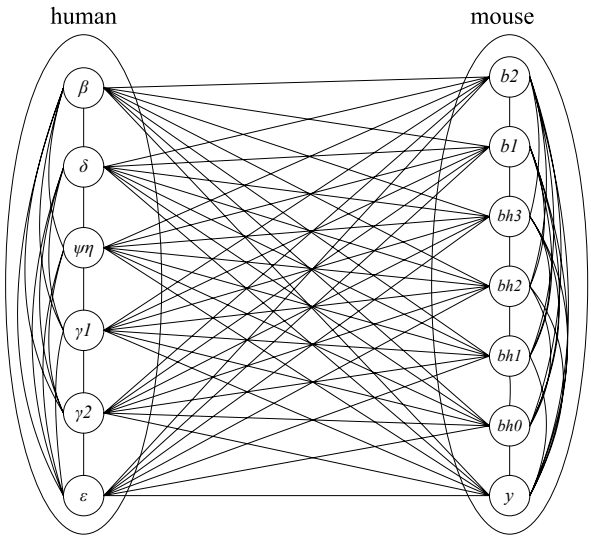


Figure 3.6. Graph structure of pairwise alignments. Nodes represent conserved regions, and edges represent alignments.

indicates they might be in-paralogs.

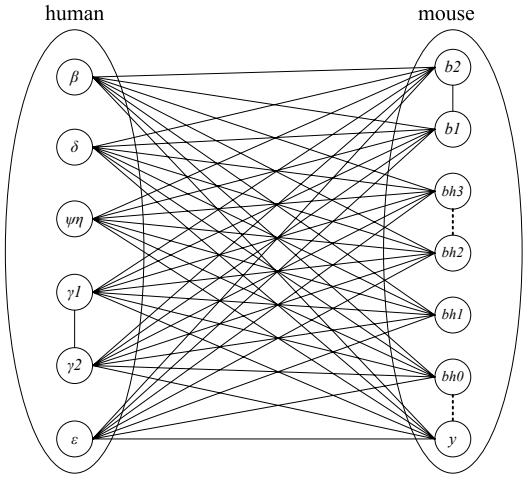


Figure 3.7. The alignment graph after removing self-edges with scores lower than the score of some cross-edge from the same node.

3.2.7 Determining orthologous alignments

Our approach for selecting orthologous alignments is based on the following observation: if a gene s in species A has more than one ortholog in species B, say s' and s'', then s' and s'' must be in-paralogs (relative to A). Suppose the nearest common ancestor of species A and B is C. Since s is orthologous to s', species C must carry a gene copy ss' ancestral to s and s'. Similarly, C must carry a gene copy ss'' ancestral to s and s''. It can be concluded that ss' and ss'' are the

same copy, since s can have only one ancestral sequence in C, and hence ss' is also ancestral to s''. Thus s' and s'' must have been created by a duplication event after the speciation event for A and B, i.e., s' and s'' are then in-paralogs. This observation leads to the procedure: select the set of cross-edges with maximum total of edges scores, subject to the following criterion: a pair of target nodes reached by edges from the same source node (in either species) and the source node itself must form a cycle. For a simple example in Fig. 3.7, mouse *b1* and *b2* and human β form a cycle.

For the problem of determining orthologs, we can prove that it is difficult to get an optimal solution. The problem can be formulated as a Weighted Maximum Clique Connection (WMCC) problem. For a graph $G =< V, E >$, a *cut* is a partition of V to form two sets V_1 and V_2. E_1 denotes the subset of E whose two nodes $n_1 \in V_1$ and $n_2 \in V_1$. E_2 is defined similarly for nodes in V_2. Let G_1 denote $< V_1, E_1 >$ and G_2 denote $< V_2, E_2 >$. An edge $e \in E$ that connects $v_1 \in V_1$ and $v_2 \in V_2$ is a *cut edge*. Let C denote the complete set of cut edges for V_1 and V_2. The WMCC problem is then to maximize the sum of the scores of a subset of cut edges subject to the constraint that if any two such cut edges share a same node, the nodes at their other ends must be connected directly. Since a *clique* is a graph/subgraph where any two nodes are directly connected by an edge, we describe the above constraint as the property of *clique connection* (the selected cut edges connect cliques of G_1 and G_2). To prove its NP-completeness, WMCC needs to be transformed into a recognition problem: given a graph $G =< V, E >$, a cut of it, C, and a score s, is there an WMCC with score of at least s? It is satisfactory to validate a "yes" instance of the problem in polynomial time to prove it is a NP problem. Suppose E' is the yes solution, we need to check

1. $E' \subseteq C$;
2. E' satisfies the above definition of clique connection;
3. score of $E' \leq s$.

The three checks can be done in polynomial time. Next, we need to show there exists a polynomial-time reduction from a well-known NP-complete problem to WMCC. We simplify the WMCC problem to a Maximum Clique Connection (MCC) problem, namely, to maximize the set of cut edges subject to the same constraint as in the WMCC problem. The well known CLIQUE problem is defined as: given a graph $G =< V, E >$, find the largest subset such that for every distinct $u, v \in C, [u, v] \in E$. The CLIQUE problem is known to be NP-complete, and moreover is known not to have an efficient approximation. We show the polynomial-time reduction from CLIQUE to MCC as follows.

Let graph $G =< V, E >$ be an instance of the CLIQUE problem, and fix node u_0. Define $G'' =< \{u_0\}, \emptyset >; V' = \{u_0\} \cup V; E' = \{u_0\} \times V + E; G' =< V', E' >; C = \{u_0\} \times V$. C becomes the graph cut of G' defined previously, which partitions V' into two sets $A = \{u_0\}$ and $B = V$. Suppose $V_s \subseteq V$ is the clique solution of G, then E_s is the MCC solution of G', where V_s and E_s

are shown in solid black in Fig. 3.8. Conversely , suppose $E_s = \{u_0\} \times V_s$ is the MCC solution of G', then V_s is the clique solution of G.

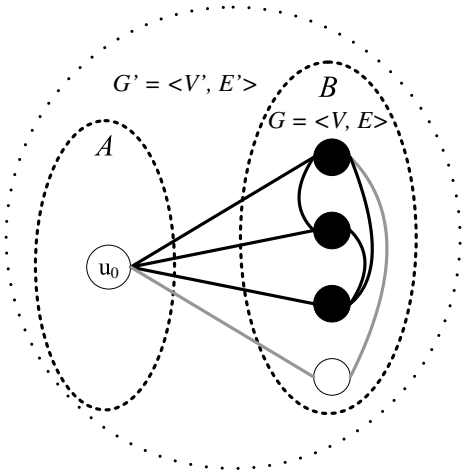


Figure 3.8. The CLIQUE problem.

An efficient approximation for the CLIQUE problem is also known to be difficult [30], and moreover the theoretically optimal solution is not necessarily a biologically optimal solution. We found a simple greedy approach, which we proved to be effective by experiments. Fig. 3.9 shows the orthology assignment generated by cross-edge selection. The dotted cross-edges are not identified by TOAST, though biologists consider them to be orthologs [29]. (However, they are recovered when TOAST is used with our multi-aligner called BOAST. See Chapter 5.) The dot-plot of orthologous pairwise alignments corresponding to the graph in Fig. 3.9 is shown in Fig. 3.10.

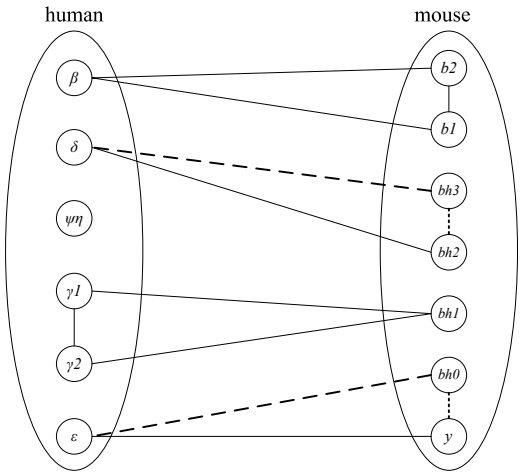


Figure 3.9. Structure of the alignment graph after applying TOAST. Dotted lines are alignments not identified by TOAST, but which are declared to be orthologous in the literature [29, 62].

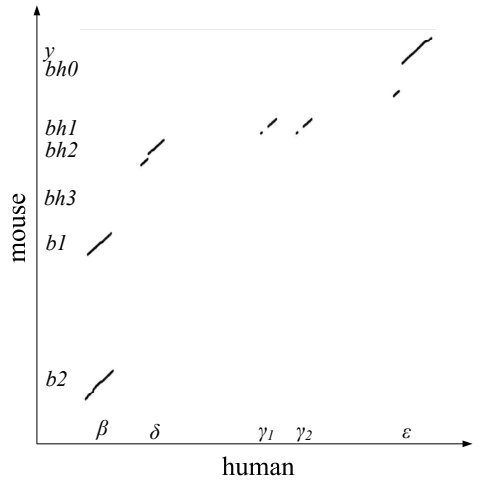


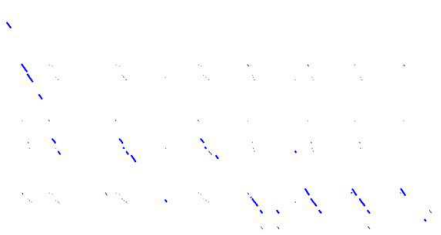
Figure 3.10. The dot-plot of pairwise alignments obtained by TOAST.

3.3 Results and evaluation

In Chapter 2, we described a dozen gene clusters having different numbers of genes in human and mouse, on which we believe most aligners perform inadequately. For each combination of gene cluster and species pair, we applied a traditional maximum-likelihood method for phylogeny reconstruction to the set of all protein sequences in the clusters of the two species. Once we obtain the phylogenetic gene tree, we infer orthology in the following approach, which is based on the tree topology of different orthology relationships (Fig. 2.1). A pair of proteins is considered to be one-to-one orthologous if they are from different species and they are children of an outmost inner node, as shown in Fig. 2.1(a). A protein p of species A is considered to be one-to-many orthologous to a set of proteins S of species B if the node of p is the sibling of the node n of the nearest ancestor of S, and the subtree of n does not contain a protein of species A, as shown in Fig. 2.1(b). A set of proteins S_A of species A is considered to be many-to-many orthologous to a set of proteins S_B of species B if the node of the nearest ancestor (n_A) of S_A and the node of the nearest ancestor (n_B) of S_B are siblings, and the subtree of n_A does not contain a protein of species B, and the subtree of n_B does not contain a protein of species A, as shown in Fig. 2.1(c).

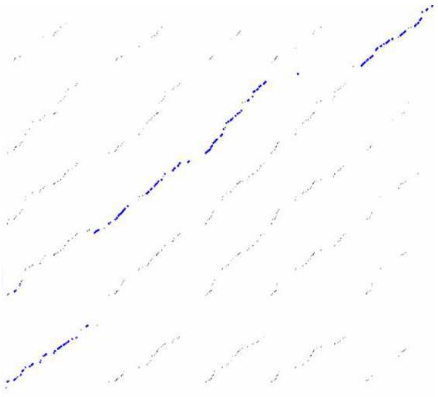
The above inference is related to the criterion that TOAST uses, but it should be more reliable because (1) boundaries of regions under study are clearly defined (not aligned regions that overlap in arbitrary ways), (2) the gene is known to be correctly treated as a unit, (3) comparison at the amino-acid level should be more reliable than DNA-based comparisons and (4) maximum-likelihood is known be more accurate for this purpose than methods based on alignment scores and/or percent identity. These orthology assignments using phylogenetic analysis are the benchmarks for evaluating TOAST. The orthologous alignments in the beta globin cluster were given previously (Fig. 3.10). We first show the TOAST alignment results of other 11 gene clusters (Fig. 3.11), then describe in detail our evaluation protocol, followed by evaluation results.

3.3.1 Alignment results of 11 gene clusters



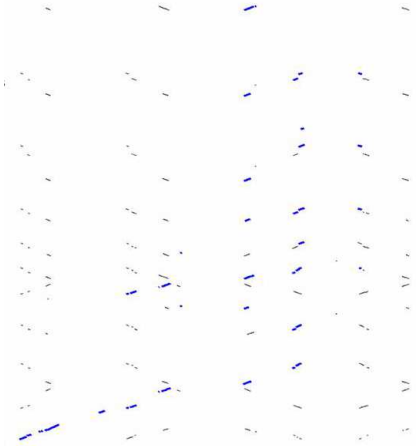
(a) Caspase gene cluster.

(b) CD1 family antige gene cluster.

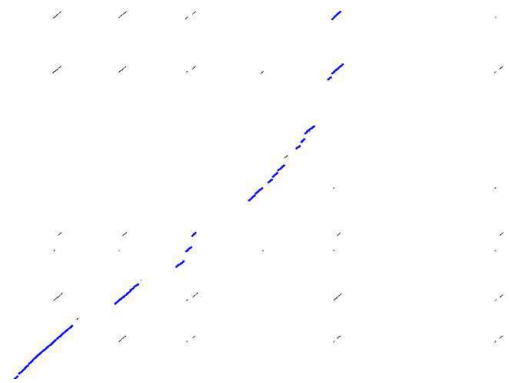


(c) ATP-binding cassette, sub-family A gene cluster.

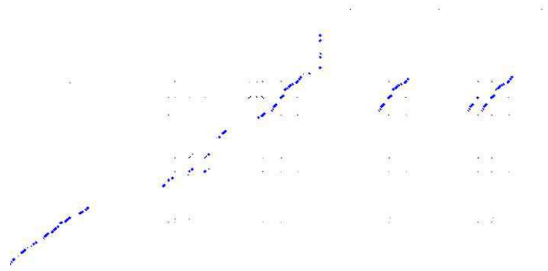
(d) Alpha globin gene cluster.



(e) Tripartite motif gene cluster.

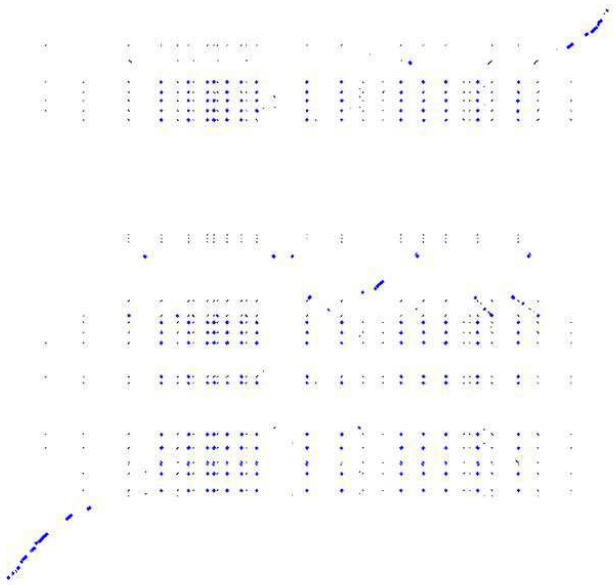


(f) Chemokine ligand gene cluster part I.

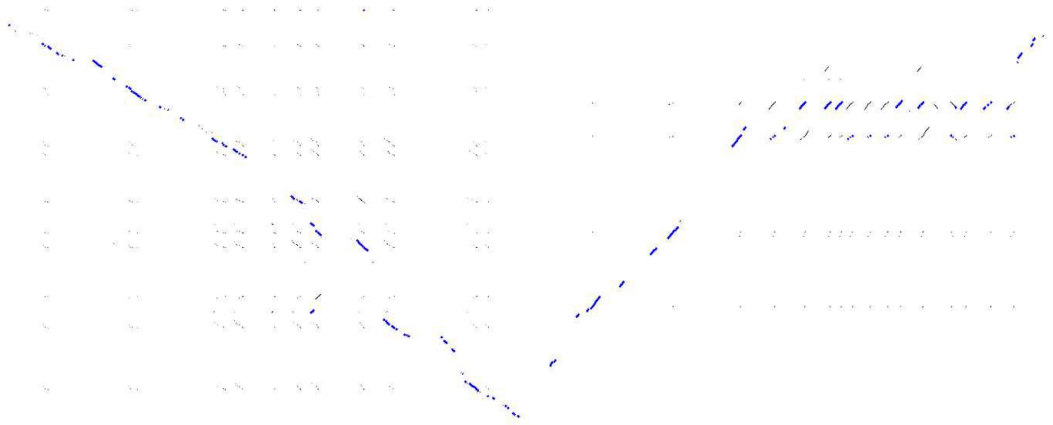


(g) Chemokine ligand gene cluster part II.

(h) Interferon-induced protein with tetratricopeptide gene cluster.



(i) Interferon gene cluster.



(j) Matrix metalloproteinase gene cluster.

(k) metallothionein gene cluster.

Figure 3.11. Pairwise alignments of 11 gene clusters produced by BLASTZ and TOAST. TOAST alignments are shown in blue or darker lines. BLASTZ alignments include all pairwise alignments.

3.3.2 The evaluation protocol

To quantify the evaluation, we define "sensitivity" and "specificity" as follows. Ideally, sensitivity shows the proportion of identified true orthologs among all true orthologs, while specificity shows the proportion of true orthologs among all predicted orthologs.

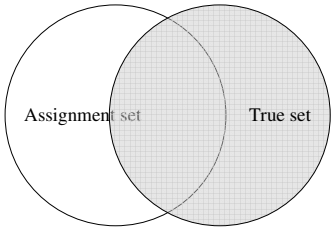


Figure 3.12. A simple view of evaluation for orthologous pairwise alignment.

Let A = predicted ortholog pairs, and let T = true ortholog pairs. They represent the assignment set and true set in Fig. 3.12 respectively. Then specificity = $(A \cap T)/A \times 100\%$ and sensitivity = $(A \cap T)/T \times 100\%$.

In practice, we may not have the complete set of proteins for each cluster, and some of the proteins are not assigned orthologs because of incomplete genome sequence data. Furthermore, we do not have every true orthology, since the bootstrapping values vary greatly and many of the pairs do not satisfy the minimum threshold to be trusted. Besides, the alignment from TBA also includes the flanking areas around the gene cluster. Thus, we cannot directly utilize the above definition of specificity and sensitivity. We need to carefully define the predicted set and true set.

The Maximum Likelihood phylogeny reconstruction on protein sequences gives the relationships of genes from the two species. Proteins without orthologs are denoted as assigned to *null*. The set P denotes the whole set of protein pairs from ML phylogeny reconstruction. Within this set, protein pairs from different species with bootstrap values exceeding a threshold are taken as the true set, denoted as T, a subset of P. TOAST produces alignment blocks of two species. A set of pairwise alignments is denoted as O. A subset A of O is defined for any pairwise alignment block or partial block whose two rows are within any of the sequences in set P, and at least one of whose rows is within a sequence in T. These sets are illustrated in Fig. 3.13. Formulas I, below, can be applied to sets A and T to compute specificity and sensitivity.

There are different measurements to determine whether the DNA pairwise alignment block is within a protein sequence or to determine the number of DNA bases that are within a protein. One approach is to compare the whole genomic interval of the protein's gene with the DNA sequence. This approach might systematically make the sensitivity appear low, since the whole genomic region of a gene includes non-coding segments, such as the UTRs (Untranslated Regions) and introns, and introns are usually very much less conserved and hence often not aligned. An alternative is to compare only the exon positions of the protein with the DNA sequence. We show results for both measurements.

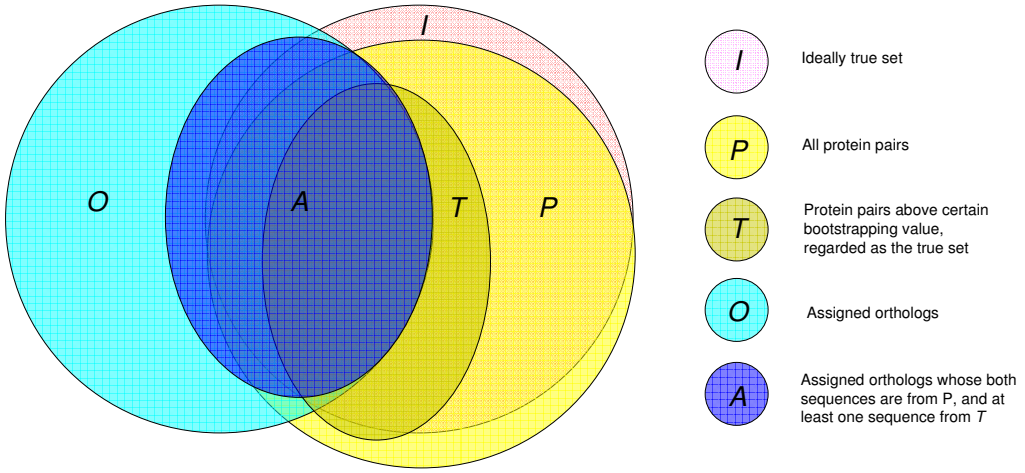


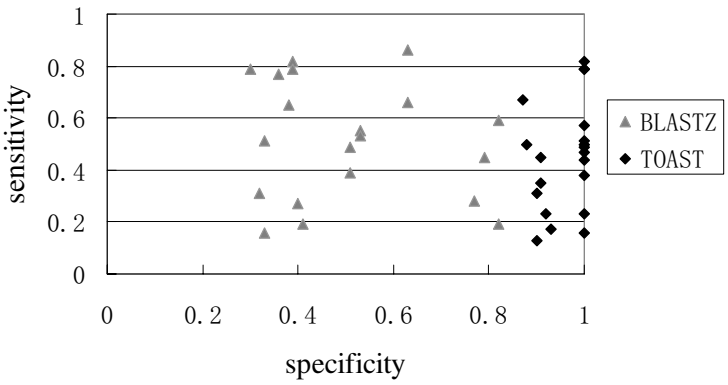
Figure 3.13. Abstract structure of the evaluation protocol.

We need to determine the subset $A \cap T$. Suppose bases b_1 and b_2 from two species are aligned together. If one of them does not fall within the boundary of any gene in the set P, or if neither of them falls within the boundary of any gene in the set T, we discard this pair of bases. Suppose the base b_1 falls within gene copy g_1 in the set P, and the base b_2 falls within gene copy g_2 in the set P. If g_1 and g_2 form a pair in set T, we regard b_1 and b_2 as correctly assigned bases. Let c denote the number of correctly assigned positions in one species S. Let a denote the number of aligned positions of species S within set A. Let t denote the number of positions of species S within set T. We get:

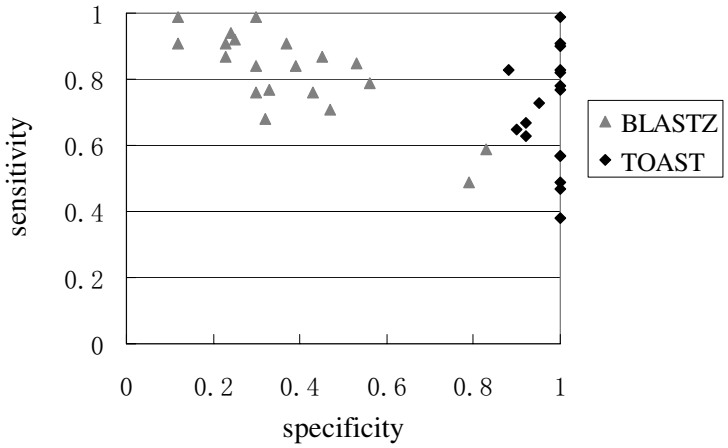
$$\begin{aligned} &\text{Specificity} = c/a \times 100\% \\ &\text{Sensitivity} = c/t \times 100\% \end{aligned} \quad \text{.....(Formulas I)}$$

3.3.3 Comparison between TOAST and BLASTZ

True sets of orthologous pairs were determined by phylogenetic reconstruction. Fig. 3.14 shows that BLASTZ alignments give high sensitivity and low specificity. TOAST improves specificity significantly while maintaining the sensitivity at an acceptable level. Fig. 3.14(b) shows the evaluation result based on exons; Fig. 3.14(a) shows the evaluation based on the whole gene, including exons and introns. Because introns are less conserved than exons, and less detectable by an alignment tool, Fig. 3.14(a) shows generally lower sensitivity than Fig. 3.14(b), as expected. The table of Fig. 3.15 presents the evaluation results based on different bootstrapping values (950, 800 and 500) that are accepted for inferring true orthologs. When bootstrapping value of 950 is the criteria for true orthologs, the set of true orthologs is too small. So the true orthologs used in Fig. 3.14 are from branches with bootstrapping values above 500. Of course, like any alignment program, TOAST has many parameters and internal thresholds that can be adjusted to exchange sensitivity for specificity or vice versa. For reference-dependent multiple alignments (Chapter 4), we can use TOAST with higher sensitivity, but for reference-independent multiple



(a) Specificity vs. sensitivity based on whole genes' boundaries.



(b) Specificity vs. sensitivity based on exons' boundaries only.

Figure 3.14. Comparison of pairwise alignments of 12 gene clusters of human vs. mouse by BLASTZ and TOAST.

alignments (Chapter 5), we may favor TOAST with higher specificity.

			whole gene boundary														exon only											
index	cluster	species	Specificity (%)						Sensitivity (%)						Specificity (%)						Sensitivity (%)					
Thresholds of bootstrapping			950		800		500		950		800		500		950		800		500		950		800		500	
			bz	toast	bz	toast	bz	toast	bz	toast	bz	toast	bz	toast	bz	toast	bz	toast	bz	toast	bz	toast	bz	toast	bz	toast
1	ABCA	human	50	88	49	91	51	91	40	38	39	37	39	35	23	100	23	100	25	96	93	89	93	87	92	86
		mouse	50	88	49	91	51	91	50	48	49	46	49	45	22	86	22	90	24	89	97	93	93	87	94	88
2	CASP	human	–	–	62	100	79	100	–	–	65	65	45	38	–	–	24	100	45	100	–	–	86	86	87	70
		mouse	–	–	61	100	77	100	–	–	28	28	28	23	–	–	25	100	47	100	–	–	52	52	71	57
3	CCL(1)	human	–	–	100	100	–	–	–	–	98	98	–	–	–	–	100	98	–	–	–	–	100	97	–	–
		mouse	–	–	100	100	–	–	–	–	87	87	–	–	–	–	100	87	–	–	–	–	100	96	–	–
4	CCL(2)	human	80	100	–	–	63	88	60	60	–	–	66	50	32	100	–	–	53	92	39	39	–	–	85	67
		mouse	79	100	–	–	63	87	97	97	–	–	86	67	35	100	–	–	56	92	86	86	–	–	79	63
5	CD	human	–	–	46	100	–	–	–	–	71	71	–	–	–	–	36	100	–	–	–	–	67	67	–	–
		mouse	–	–	41	100	–	–	–	–	95	95	–	–	–	–	39	100	–	–	–	–	90	90	–	–
6	AG	human	–	–	17	100	39	100	–	–	96	91	82	82	–	–	17	100	37	100	–	–	94	89	91	90
		mouse	–	–	17	100	39	100	–	–	87	84	79	79	–	–	18	100	39	100	–	–	83	78	84	83
7	BG	human	–	–	–	–	38	100	–	–	–	–	65	44	–	–	–	–	30	100	–	–	–	–	76	38
		mouse	–	–	–	–	36	100	–	–	–	–	77	49	–	–	–	–	30	100	–	–	–	–	99	49
8	IFIT	human	–	–	26	82	32	90	–	–	19	18	31	31	–	–	26	82	30	88	–	–	75	73	84	83
		mouse	–	–	26	82	33	90	–	–	9	6	16	13	–	–	27	100	33	100	–	–	74	74	77	77
9	IFN	human	9	100	–	–	82	100	62	62	–	–	19	16	10	100	–	–	79	100	62	62	–	–	49	47
		mouse	9	100	–	–	82	100	99	99	–	–	59	57	10	100	–	–	83	100	99	99	–	–	59	57
10	MMP	human	53	100	53	100	53	100	50	50	50	49	55	50	20	100	20	100	23	100	97	95	92	91	91	82
		mouse	52	100	52	100	53	100	53	52	54	53	53	47	19	100	18	100	23	100	82	81	81	80	87	78
11	MT	human	–	–	–	–	33	100	–	–	–	–	51	51	–	–	–	–	12	100	–	–	–	–	99	99
		mouse	–	–	–	–	30	100	–	–	–	–	79	79	–	–	–	–	12	100	–	–	–	–	91	91
12	TRIM	human	–	–	33	100	40	92	–	–	48	48	27	23	–	–	25	100	32	90	–	–	99	99	68	65
		mouse	–	–	34	100	41	93	–	–	48	48	19	17	–	–	34	100	43	95	–	–	95	95	76	73

Figure 3.15. Evaluation results for 12 gene clusters, using different bootstrapping values (950, 800 and 500) to obtain true orthologs. '-' represents the case where there is no orthologous pair from phylogenetic reconstruction at the threshold values.

Chapter 4

Reference-dependent multi-species orthologous alignment

4.1 Overview

The topic of multi-sequence alignment is extensively and actively being studied. Several multi-species genomic sequence aligners were briefly discussed in Section 1.3 of Chapter 1, and a published review [4] provides a good summary of this field, especially on genomic sequence alignment. Since the multi-sequence alignment problem is NP-complete [67], i.e., computationally intractable, most multi-species alignment tools, such as TBA [8], MLAGAN [10] and MAVID [9], use heuristic methods. They start with pairwise alignments, and progressively merge pairwise and/or multiple alignments to form multiple alignments of more sequences. The nature of progressive alignment is recursive: for a given guide tree (for the relevant species) and initial pairwise alignments, the multiple alignments at an inner node are obtained by aligning alignments or sequences determined from the two sub-trees. At each inner node, TBA and MLAGAN align alignments from sub-trees, while MAVID aligns reconstructed ancestral sequences determined from the alignments of sub-trees. The guide tree is given as input to TBA or MLAGAN, while it can be automatically computed in MAVID. Since the genomic sequences we are going to align are all from well-known species, we assume that the guide tree is known.

Each of these aligners assumes that none of the aligned sequences contains a duplication, i.e., a pair of intervals in the same sequence that are homologous to each other. Under that condition, any significant alignment is most probably an orthologous alignment. However, this is no longer true when duplications are present. See Section 3.1 for an illustration of the inadequate performance of these aligners on the beta globin cluster. The existence of duplications also raises the question of how to represent the multiple alignment (details in Chapter 5). One way to avoid that problem is to use a chosen species, called the *reference*, such that any position of the reference is aligned to at most one position of each other species. The reference is the species

chosen as a coordinate system for organizing the multiple alignments, and alignment blocks that do not contain a sequence from this species are discarded. This strategy is used in the ENCODE project, where human is the reference since the study is focused on the human genome. We call the approach of creating multi-species alignments using a reference sequence *reference-dependent*, which is the focus of this chapter.

In this chapter, we first explain the logic behind using the "progressive" approach for producing orthologous multi-species alignments that allow duplications in the aligned sequences (Section 4.2.1). We then describe a restricted class of reference-dependent alignment problems where the guide tree is required to be "skewed" (Section 4.2.2). Next, we describe in detail how to select a subset of orthologous pairwise alignments such that any position of the reference is aligned to at most one position in the other species (Section 4.2.3). Section 4.2.4 discusses our implementation of a reference-dependent orthologous multi-species sequence aligner (ROAST) for skewed guide trees. We apply ROAST to the beta globin cluster and compare the alignments with those from TBA and MLAGAN (Section 4.3). Finally, we describe how to create genome-wide alignments of 17 species for gene-cluster regions, using human as the reference (Section 4.4). In that section, we explain how to extend ROAST to handle a guide tree of an arbitrary shape (i.e., not necessarily skewed).

4.2 Methods

4.2.1 Aligning at an inner node of a binary guide tree

Consider an internal node of a binary tree (a tree where each internal node has precisely two children). No reference is specified. Assume that its left sub-tree and right sub-tree have already been processed separately, and that each multi-alignment block identifies correct orthologous relationships. Fig. 4.1 depicts the situation. Blockset BS_1, containing sequences from species S_1, S_2, S_3, and blockset BS_2, for S_4, S_5, need to be aligned. (A "block" is a local multi-alignment, and a "blockset" is a set of blocks.) Pick block B_1 from BS_1 and block B_2 from BS_2 to illustrate our strategy, and let s_1, s_2, s_3, s_4 and s_5 from S_1, S_2, S_3, S_4 and S_5 denote the sequence fragments within the blocks. Suppose there is an orthologous alignment between s_1 and s_4. Ideally, there might be orthologous alignments s_1 vs. s_5, s_2 vs. s_4, s_2 vs. s_5, s_3 vs. s_4 and s_3 vs. s_5. In that case, we can safely fuse the two blocks together. However, in general (and frequently in practice), some of the six orthologous pairwise alignments will be missing. Orthology is not transitive [24]. That is, for three gene copies a, b, c from species A, B, C, we can have a orthologous to b, b orthologous to c, but a not orthologous to c. However, in our context, we argue that as long as at least one correct orthologous alignments has been detected between B_1 and B_2, then each of s_1, s_2 and s_3 is orthologous to each of s_4 and s_5 even though BLASTZ may have failed to detect the alignment or TOAST may have discarded it. Fig. 4.2 illustrates the scenario.

Here, s_4 is orthologous to s_5 since they are within the same block. We argue that s_1 and s_5 are orthologs. Let C_{45} be the *nearest common ancestor* (lca) of S_4 and S_5, and let C_{145} be

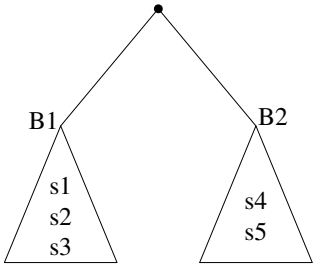


Figure 4.1. Aligning alignments of two sub-trees.

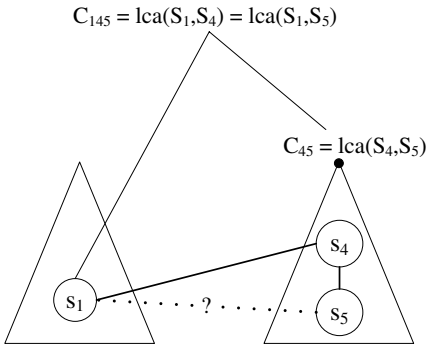
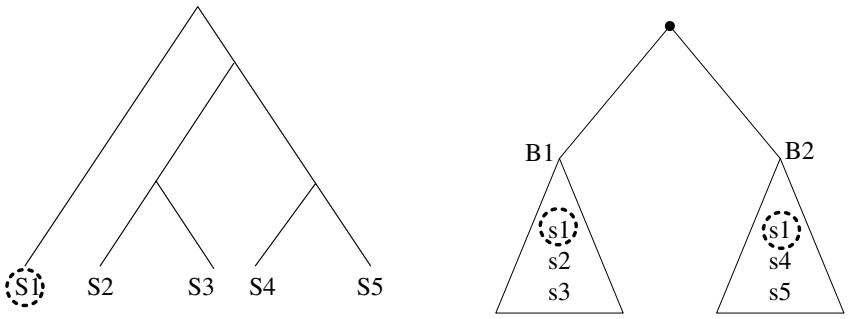


Figure 4.2. A special case for orthology transitivity.

the nearest common ancestor of S_1, S_4 and S_5. Since s_4 and s_5 are orthologs, C_{45} must carry the copy s_{45} which is ancestral to both s_4 and s_5. By the same reasoning, C_{145} must carry the copy s_{14} ancestral to both s_1 and s_4. We can conclude that C_{145} carries the copy s_{145} ancestral to s_1, s_4 and s_5. Since C_{145} is the nearest common ancestor of S_1 and S_5, and s_{145} of C_{145} is ancestral to both s_1 and s_5, s_1 and s_5 must be orthologs. By the same reasoning, s_1, s_2, s_3 in B_1 are orthologous to s_4, s_5 in B_2. We now can safely align B_1 and B_2 in Fig. 4.1 to form a new orthologous multiple alignment block containing s_1, s_2, s_3, s_4 and s_5 that are orthologous to each other.

4.2.2 Aligning with a skewed guide tree

The previous section explains the biological background of progressive multiple alignment, in particular that orthology information is propagated appropriately. In the above example, all species/sequences of sub-trees of the inner node are treated equally, and there is no special species chosen as a reference. When there is no reference to organize the orthologous multiple alignment on genomic regions containing duplications, there is a serious problem of size explosion, which is discussed in more detail in Chapter 5. This chapter discusses the orthologous multiple alignment problem with a specified reference species. At each inner node, the orthologous multiple (or pairwise) alignments of species from each sub-tree together with the reference species are first obtained. Since the multiple alignments of both subtrees contain the sequence of the reference species, the reference sequence can then be used to guide the procedure of aligning alignments of



(a) A species tree where the out-group S1 is chosen as the reference species.

(b) Aligning the two multiple alignment of B1 and B2 may lead to non-orthologous multiple alignments.

Figure 4.3. An example where non-orthologous multiple alignments might result from a reference-dependent approach.

two subtrees at this inner node. This process is repeated at every inner node including the root.

However, depending on whether the reference species occurs in one of the sub-trees of the species guide tree or not, this process may or may not produce orthologous multiple alignments. First, assume the reference species does not occur in either of the sub-trees, for example, that S_1 in Fig. 4.3(a) is chosen as the reference to align sequences s_2, s_3, s_4 and s_5. Then aligning the two multiple alignments from two sub-trees (e.g. B1 and B2 in Fig. 4.3(b)), where each is aligned to the reference sequence, does not necessarily produce orthologous multiple alignments. For example, suppose s_2 and s_4 are descendants of two paralogs that are formed after the species split between S_1 and the ancestor of S_2 and S_4, but before the species split of S_2 and S_4. In this case, both s_2 and s_4 are orthologous to s_1, but s_2 is not orthologous to s_4.

The other case is when the reference species occurs in one of the sub-trees of an inner node of the guide tree. For example, suppose s_1 in Figs. 4.1 and 4.2 is the reference sequence. Then the merge process produces orthologous multiple alignments, as explained in the previous section (Fig. 4.2).

Only a "skewed" guide tree such as (Fig. 4.4) where human is the reference satisfies this requirement for every inner node. Specifically, the requirement for a skewed guide tree in a reference-dependent alignment problem is that for each inner node, one subtree contains the references species and the other consists of a leaf (single species). For the next few sections we assume that the guide tree is skewed, though near the end of this chapter we will show how to remove that restriction.

4.2.3 Threading projection

One property of the reference-dependent approach is that any position of the reference can align to at most one position of another species in the multiple alignment blockset, i.e., that a given reference-sequence position can appear in at most one block. However, the orthologous pairwise alignments produced by TOAST, described in Chapter 3, allow a position of the reference to align

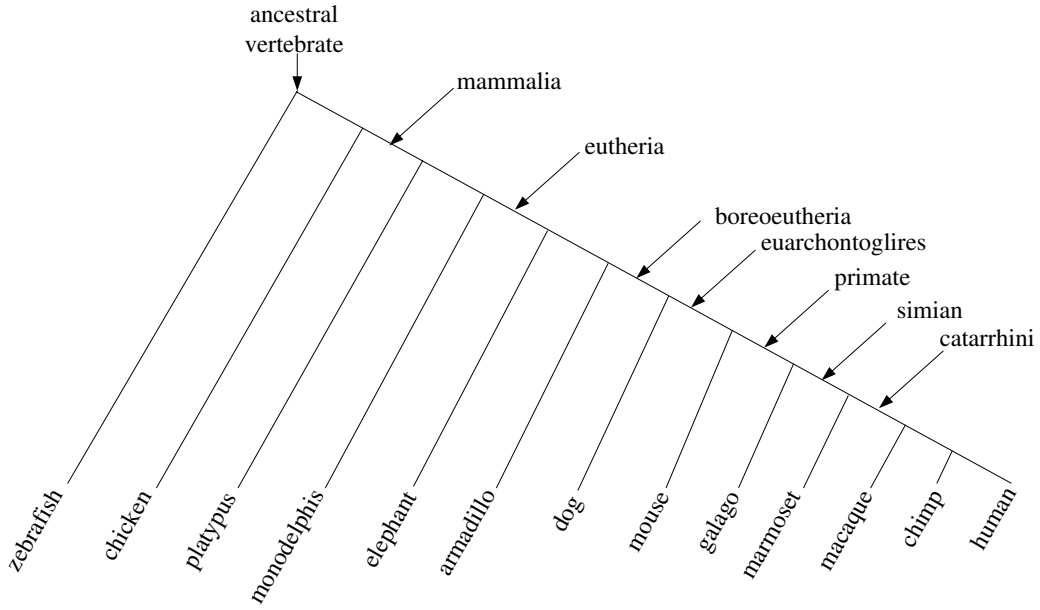


Figure 4.4. A skewed binary species tree with 13 species, assuming human is the reference.

to more than one position of another species, e.g. human β of the beta globin cluster is aligned to two genes in mouse (Fig. 3.10). The process of organizing the set of unordered alignment blocks according to the reference is called *projection*. The process of selecting a set of pairwise alignments such that any reference position is aligned to at most one position in another species is called *threading projection*. There are different approaches to perform threading projection, including several simple greedy approaches that try to maximize different objective functions. For example, one maximizes the number of bases of the reference that are aligned, while another maximizes the total score of the projected pairwise alignments. These approaches work fine in the example of Fig. 3.10, but they often do not provide pairwise alignments that are biologically meaningful. We next consider another gene cluster as an example (Fig. 4.5) to illustrate the deficiency of these simply greedy approaches, then provide a more complicated heuristic approach that combines usage of chaining and annotation. We have shown that the approach performs well.

Fig. 4.5 shows the alignments between the human *SERPINA3* gene and its orthologs in mouse. It indicates that there are many in-paralogs (at least 10) in mouse, and all of them are orthologous to human *SERPINA3*. The *SERPINA3* gene has around 12K bases, including several long introns. These introns are less conserved and are not aligned. Thus the alignments between human *SERPINA3* and one of the mouse genes are broken into several shorter alignment blocks. A greedy approach to select alignment blocks to form a threading projection may lead to the result that partial human *SERPINA3* genes are aligned to partial regions from several mouse genes. Details are discussed below. We use an example of artificial sequences to illustrate the problem (Fig. 4.6).

4.2.3.1 Maximize the number of aligned bases of the reference species subject to the threading condition

One may want to maximize the number of aligned bases when doing threading projection of TOAST pairwise alignment. In other words, one objective is to maximize the alignment coverage of the reference species. The following simple algorithm achieves this objective:

MaximizeCoverage (A: the set of pairwise alignment blocks; R: reference species)
1 $A \leftarrow$ sort alignment blocks in A according to their starting positions on reference R
2 $O \leftarrow \phi$
3 $p_s \leftarrow$ the first position of R in A
4 **while** there exists a block containing p_s
5 Insert the longest block/partial block B starting at p_s to O
6 $p_e \leftarrow$ the end position on R of B
7 $p_s \leftarrow p_e+1$ or the next aligned position on R if the position p_e+1 is not aligned
8 O contains the maximum coverage threading projection of A

Fig. 4.7 illustrates the selection process of the alignment blocks from the example in Fig. 4.6 according to above algorithm. Fig. 4.8 shows the dot-plot of selected alignment blocks. There are maybe more than one solutions that also maximize the alignment coverage, and the above algorithm produces one of such solutions. It actually produces the least number of alignment blocks among all solutions, which can be easily proved.

4.2.3.2 Maximize the alignment score subject to the threading condition

Alternatively, one may want to maximize the alignment score when doing threading projection of TOAST pairwise alignment. The following algorithm achieves this objective:

MaximizeAlignmentScore (A: the set of pairwise alignment blocks; R: reference species)
1 $O \leftarrow \phi$
2 $P \leftarrow$ all end points of all alignment blocks in A on R
3 $\{p_1, p_2, p_3, ..., p_n\} \leftarrow$ sorted distinct elements of P, from small to large
4 For each interval V of $(p_i, p_{i+1}), i \in [1, n-1]$
5 Insert the partial block between V that has the highest score to O if there is one
6 O contains the threading projection of A that has the maximum score

Fig. 4.9 illustrates the process of selecting alignment blocks from the example in Fig. 4.6 according to the above algorithm. Fig. 4.10 shows the dot-plot of selected alignment blocks. Strictly speaking, the alignment blocks selected by this approach have the highest alignment score when the break points are limited to be the end points of input pairwise alignment blocks.

4.2.3.3 Threading projection using chaining and annotation

Each of above two algorithms maximize a certain objective function, but in both cases, a single gene of species A can broken into several pieces that are aligned to partial regions of different genes of species B. Though all alignments are orthologous, they do not make much biological sense. It is preferable to have each gene align to another complete gene. We use the methods from Section 3.2.1 (chaining using a k-d tree) to identify local chains and Section 3.2.2 (chaining using gene annotations) when such annotations are available, and then select chains to form a threading projection. The following greedy approach based on alignment scores of chains turns out to perform well in practice.

ChainingProjection(C: the set of chains of pairwise alignment blocks; R: reference species)
1 $\{c_1, c_2, ..., c_n\} \leftarrow$ sort chains C according to the alignment score from large to small
2 $O \leftarrow \phi$
3 For i in 1 to n
4 Insert the unaligned region of c_i to O
5 Mask the aligned region on R
5 O contains the threading projection of A

4.2.4 A simple reference-dependent multi-species aligner (ROAST)

An updated MULTIZ [8] aligns two alignment blocksets that have the same reference. It uses the reference to find appropriate blocks from each blockset, and then aligns them using an efficient dynamic programming algorithm. A recently developed program, ROAST, is similar to TBA [8] in the sense that both are multi-species aligners working progressively according to a given phylogenetic guide tree, and both use MULTIZ to align alignments at an internal node of the guide tree. ROAST differs from TBA in being reference-dependent, while TBA is reference-independent. For the reference-dependent aligner, any pairwise alignment that is not related to the reference is discarded from the input pairwise alignments. If there are N species to be aligned, ROAST needs $N-1$ sets of pairwise alignments, while TBA requires $N(N-1)/2$ of them. This difference alone leads to a large reduction of the running time for using ROAST. At the same time, any regions of the $N-1$ species that are not aligned to the reference are discarded. The following algorithm describes ROAST.

ROAST (T: phylogenetic species guide tree; E: reference species)
At each node of T, in leaf-to-root order, do
1 $L \leftarrow$ the set of species of the left child
2 $R \leftarrow$ the set of species of the right child
3 $A \leftarrow$ the alignment blockset for the left child referenced by E
4 $B \leftarrow$ the alignment blockset for the right child referenced by E
5 $G \leftarrow$ MULTIZ(A, B, L, R)

G is the alignment blockset for this node

As an aligner aligning two alignments, MULTIZ would presumably fix the alignments from two inputs and align them column by column. But it does something slightly different when L or R contains the reference E (E cannot be in both) than when neither of them contains E. In the first case, suppose L contains E, the alignment between E and other species in A is fixed, and it stays the same in T after being aligned with B, while the alignment between E and other species in B is discarded. In the second case, the alignment between E and other species is not fixed in either A or B, and they are both discarded. The alignment is then actually optimized between $(A - E)$ and $(B - E)$, followed by adding E using a second run of dynamic programming. The purpose of handling the two cases differently is to give preference to alignments among closer species. ROAST guarantees that any pairwise alignments input to ROAST are contained in the final multiple alignment blockset, which TBA cannot guarantee. The base-to-base alignment in the final blockset may be slightly different from its original pairwise alignment input, where the maximum difference depends on the parameter of "radius" in the banded dynamic programming computation.

ROAST guarantees that any non-reference sequence is orthologous to the reference sequence in the same alignment block, which is called the *partial orthology* property of the multiple alignments. As discussed in section 4.2.2, it depends on the shape of the species guide tree whether any two rows of sequences are orthologous or not. Only if the species guide tree is skewed and the reference is at the innermost node are any two rows of a same multiple alignment block guaranteed to be orthologous, in which case the alignment is called *completely orthologous*. With some extension, ROAST can be used to produce completely orthologous multiple alignments for any guide tree. The extensions are described in Section 4.4.1.

4.3 Comparison with other multiple aligners

In Chapter 2, we established the orthology relationships of beta-like globin genes from 19 species. 13 of these species form a skewed tree (Fig. 4.4), which is advantageous for applying the algorithm of Section 4.2.4 directly to obtain orthologous multiple alignments using human as the reference. We present the alignment result between human and several other representative species where the beta globin clusters are completely sequenced and assembled (Figs. 4.11 to 4.16). The alignments of TBA and MLAGAN are shown for comparison purpose. We compare these alignments with the orthology assignment in Table 2.14, which summmarizes the results from phylogenetic reconstruction. In Figs. 4.11 to 4.16, shaded regions are the locations of beta globin genes including intronic regions. From left to right, the human beta globin genes are β, δ, $\psi\eta$, γ_1, γ_2 and ϵ.

We have not attempted to quantify alignment quality, because the dataset is so small and only a small number of species have well-assembled genomic sequences for the beta globin region. Furthermore, MLAGAN is a global aligner, which tries to align as much as possible; ROAST

and TBA are local aligners, which only align conserved regions. These properties of global and local aligners mean that a global aligner usually aligns more sequence positions than does a local aligner. It is thus not appropriate to simply compare the number of aligned bases, like we did in Section 3.3 for TOAST. Thus we have not found a robust and meaningful evaluation function in this case. Instead, we believe side-by-side comparison of alignments in dot-plot format reveals where the alignments are different and how significant the differences are. We discuss the accuracy of the alignments in the caption of each figure, using results of Table 2.14 as benchmarks. ROAST is significantly better than the other two aligners with respect to correctly aligning orthologous regions, though MLAGAN sometimes aligns more positions because it is a global aligner.

4.4 17-way genome-wide alignment of gene clusters

The UCSC genome browser [36] provides a human-referenced genome-wide alignment of 17 vertebrate species (Fig. 4.17). This alignment is a fundamental resource for many aspects of comparative genomics including identifying conserved regions and computing divergence rates. An accurate orthologous alignment is crucial for performing many of these tasks. In genomic regions without duplications, any alignment is orthologous as long as the aligned segments are indeed evolutionarily related, but this is not true when duplications are present, especially within gene cluster regions. Since the previous version of the program producing the 17-way alignment assumes no duplications, it is thus desirable to replace its results with orthologous alignments within gene clusters. The TOAST program from Chapter 3 requires that the input sequences of different species are descended from the same ancestral sequence. The program identifying such sequences relies on the completeness of the genomes [42]. Among the 17 species, six species, viz. human, chimp, macaque, mouse, rat and dog, can be considered as completely assembled for our purposes. Call them *core species*. Most sequences of other species are in the form of "contigs" (contiguous intervals) that are partial regions of those genomes, but are not assembled completely. It is not possible to reliably identify orthologous chunks from these species to perform orthologous alignment. Thus our goal is to create a reference-dependent 17-way multiple alignment where any two sequences from the six core species are orthologous to each other. We first obtain the orthologous pairwise alignments between human and each of the other five core species using TOAST (Chapter 3). The pairwise alignments between human and the other 11 species are obtained from the original 17-way multiple alignment. We then produce multiple alignments for gene clusters using ROAST. However, the species tree of the core species (Fig. 4.18) is not skewed, which is the requirement for ROAST to produce completely orthologous alignment. We first describe how to extend the usage of ROAST to produce completely orthologous alignments for such a species tree (Section 4.4.1). We then describe how to obtain appropriate sequences to be aligned for each gene cluster (Section 4.4.2).

4.4.1 Dealing with a non-skewed guide tree for the 17-way alignments

The smaller tree of the six core species is shown in Fig. 4.18. This tree is not skewed because of the mouse-rat subtree, and thus the ROAST program described in Section 4.2.4 cannot be applied directly to produce completely orthologous multiple alignments. However, human, mouse and rat form a smaller tree that is skewed, and mouse and rat occur at the innermost nodes. Thus using either mouse or rat as the reference, for example we use mouse, we can get completely orthologous alignments of these three species using ROAST. We then project this multiple alignment onto the human sequence, where it becomes a human-referenced orthologous alignment of three species. This set of alignments may also contain some alignments between mouse and human only, where adding a rat sequence would destroy the complete orthology property of the alignment block. It is also possible that a human region does not have an ortholog in mouse, but it has an ortholog in rat. We then add such orthologous pairwise alignments between human and mouse to the set of multiple alignments if the human region is not already aligned to mouse in the multi-alignment. Let's condense the mouse-rat subtree into a single node called "mouse.rat", shown in Figs. 4.20 and 4.21. The orthologous alignments between mouse and rat obtained above are then treated as if they are a sequence of the "species" mouse.rat located in these trees. We they apply ROAST for the tree in Fig. 4.20 using human as reference. Since the tree in Fig. 4.21 of core species is skewed and human occurs at the innermost node, the resulting alignment is a completely orthologous alignment of six core species.

Theoretically, a species tree of any shape can be processed in this way so that ROAST can be used to create completely orthologous multiple alignments. For an arbitrary species tree with reference A, the completely orthologous alignments of its subtree can be formed first using a different reference B (in case the subtree is not skewed either, then its subtree is processed in a recursive manner), then the subtree is treated as if it is a single node and the original species tree is simplified. This process is performed for any branch of the tree where the skewness property is violated, until the tree is simplified to be skewed where the reference A occurs at the innermost node.

4.4.2 Obtaining orthologous sequences of gene clusters genome-wide

We are able to get the genomic locations of gene clusters of the human genome simply by looking for clusters of RefSeq genes with similar names (personal communication from Dr. Jian Ma). We then need to determine the orthologous regions of the other five core species. Using the genome-wide multiple alignment of the six core species that is obtained from the UCSC Brower website, referenced by human, we determine such regions with the following approach. An alignment *anchor* is defined to be a certain number (e.g. 200) of contiguous alignment columns where the core species have no insertion or deletion with respect to each other, regardless of the strand orientations of the sequences. Each alignment row in this anchor is called an *anchor row. A pair of anchors* is defined to be two anchors where each pair of anchor rows of the same species must be from the same chromosome, of the same strand orientation, and within a certain distance.

We looked for the closest such pair of anchors around a human gene cluster region. Fig. **??** illustrates a pair of alignment anchors. The sequences of six species between the middle point of each anchor are considered to descend from a same ancestral sequence. The corresponding sequences of 11 other species are then determined from the 17-way alignment. Applying this approach, we found 75 gene clusters with appropriate anchors. We then applied the method in Section 4.4.1 to each of them.

4.5 Discussion

In the resulting 17-way alignment of 75 gene clusters described above, the six rows of core species are completely orthologous. The analysis of these alignments and a detailed comparison with the original 17-way alignment will be undertaken in the near future.

We perform the completely orthologous alignment process only for the six core species because the genome sequences of the other species are not fully assembled, and it is difficult to find pairs of anchors that are reliable evidence to infer the orthologous regions of gene clusters in these species. While we wait for more species to be completely sequenced and assembled, we can also try to design methods to obtain orthologous sequences without using anchors.

The special handling required to produce completely orthologous multiple alignments for a non-skewed species tree is not automated yet. In the simple case of Fig. 4.18, it was acceptable to manually implement a script to run ROAST for two cycles. However, it will be challenging to create a reliable script that can automatically process a more complicated tree such as Fig. 2.5 and run ROAST properly.

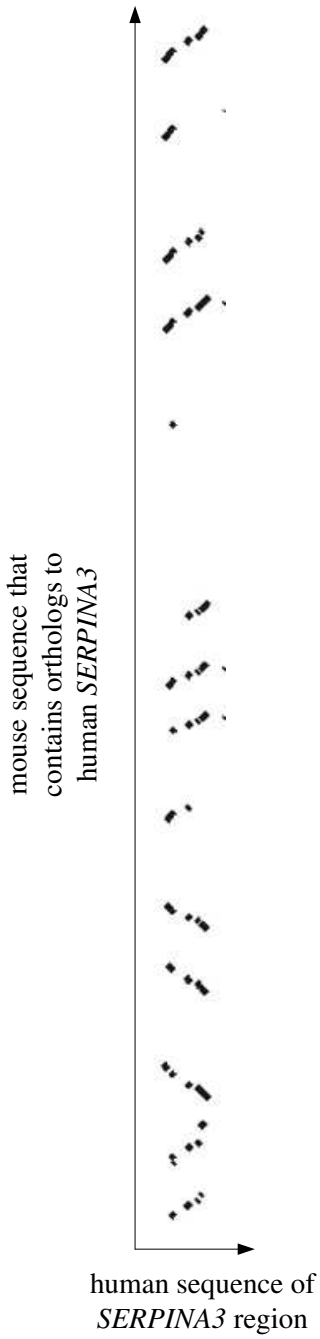


Figure 4.5. Dot-plot of pairwise alignments between human serpin peptidase inhibitor clade A member 3 gene (*SERPINA3*) and its orthologous genes in mouse. This is an example of a one-to-many orthology relationship. The alignment between each pair of orthologous genes is broken into several segments because non-conserved regions such as introns are not aligned.

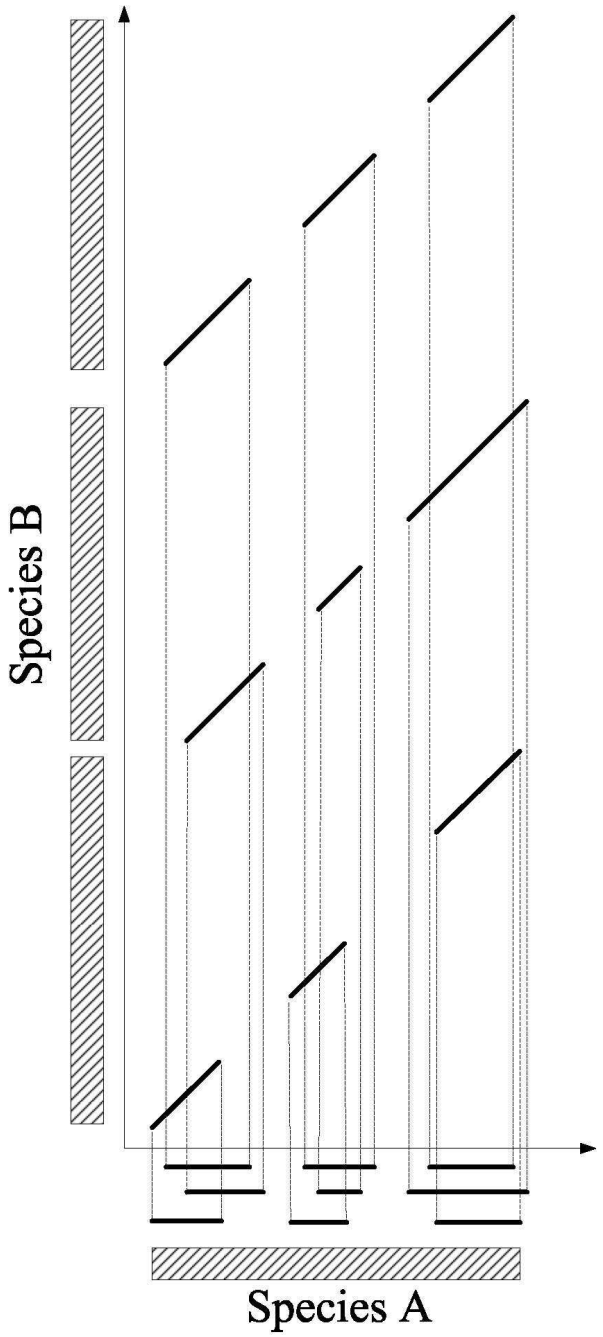


Figure 4.6. Dot-plot of orthologous pairwise alignments between two artificial sequences of two species, illustrating methods of threading projection. A shaded box denotes a gene. The sequence of species A contains one gene, while the sequence of species B contains three genes. The alignment between each pair of genes is broken into 3 pieces. Short line bars below the X axis are projections of diagonal lines of pairwise alignments. They represent the alignments referenced by species A.

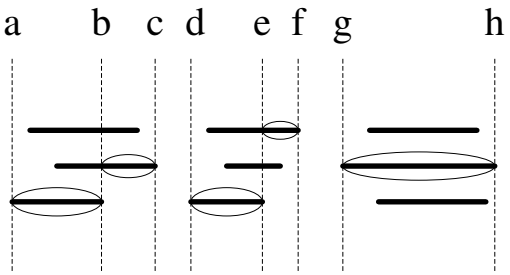


Figure 4.7. Greedy approach for maximizing the number of aligned bases for threading projection. Segments inside circles are chosen as the threading projection.

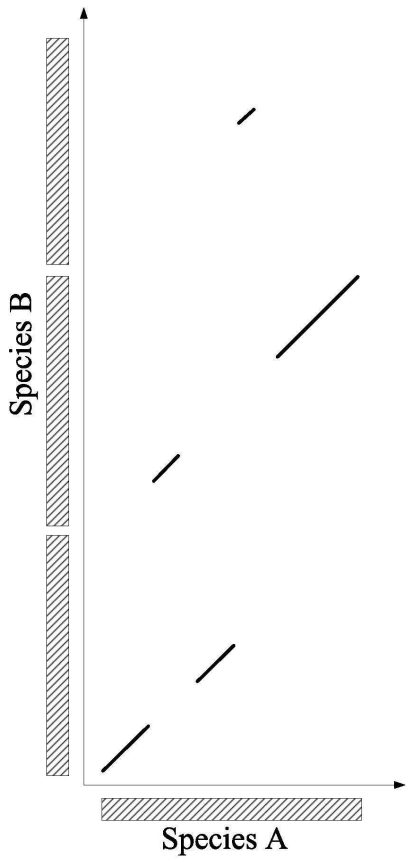


Figure 4.8. Dot-plot of threading projected orthologous pairwise alignment using the greedy approach for maximizing the number of aligned bases (the example of Fig. 4.7).

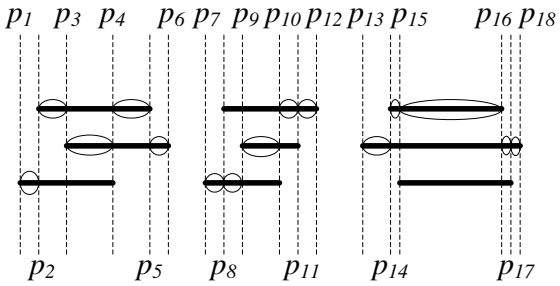


Figure 4.9. Greedy approach of maximizing the alignment score for threading projection. Segments inside circles are chosen as the threading projection.

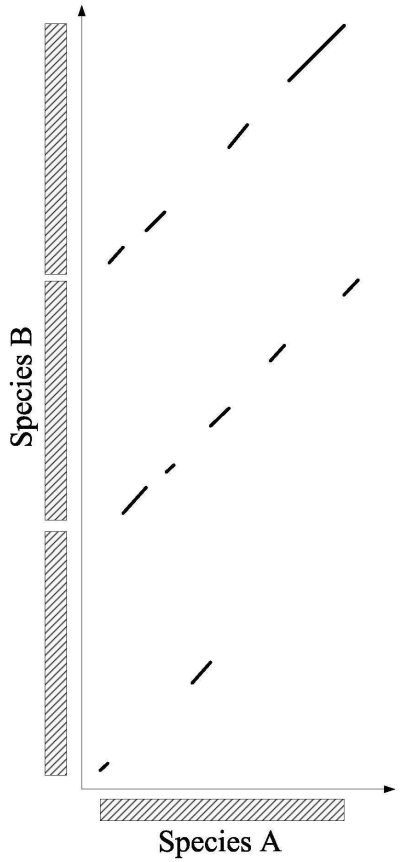
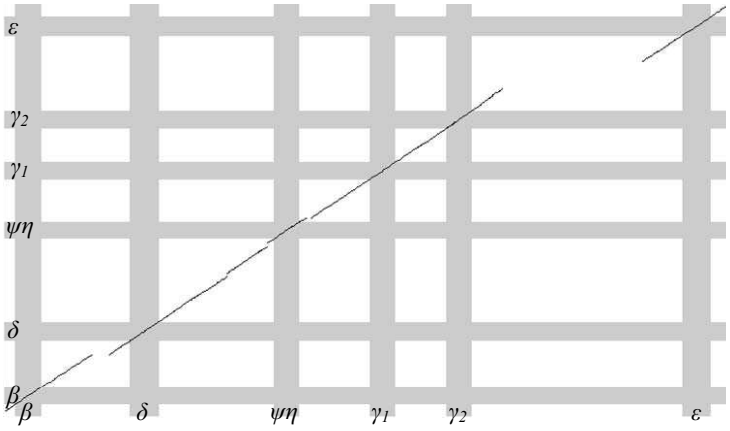
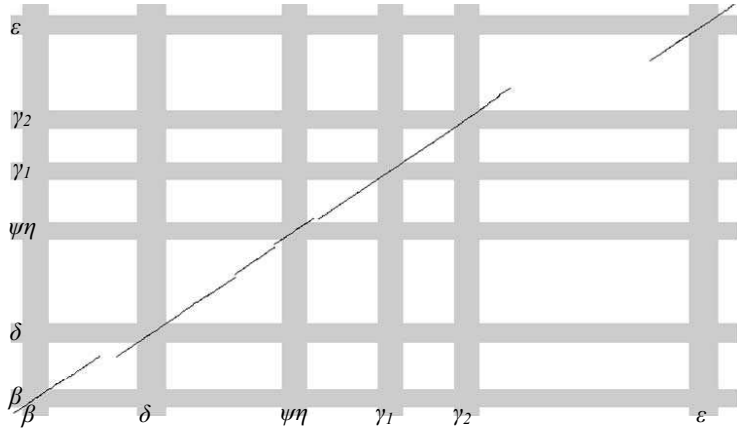


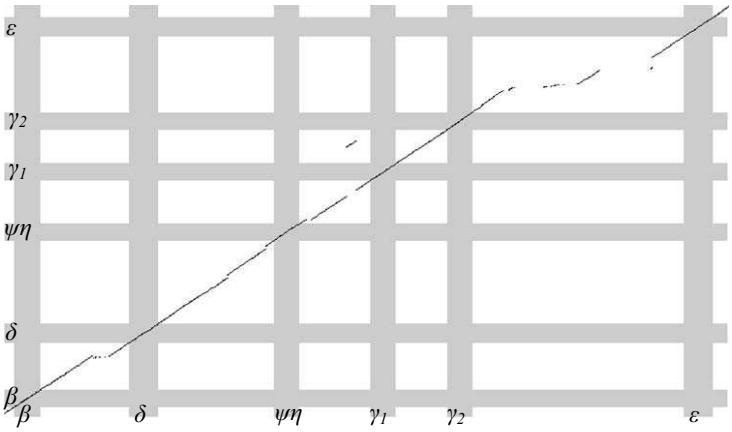
Figure 4.10. Dot-plot of threading-projected orthologous pairwise alignment using the greedy approach of maximizing the alignment score (the example of Fig. 4.9).



(a) Orthologous human-marmoset alignments extracted from the ROAST multi-species alignments.

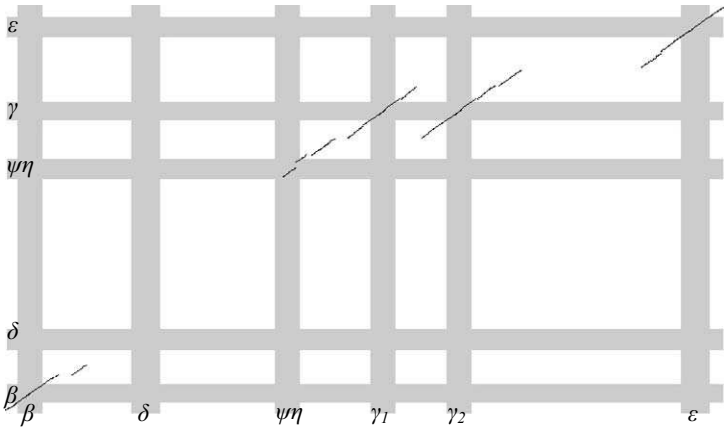


(b) Human-marmoset alignments extracted from the TBA alignments of ENCODE region ENm009.

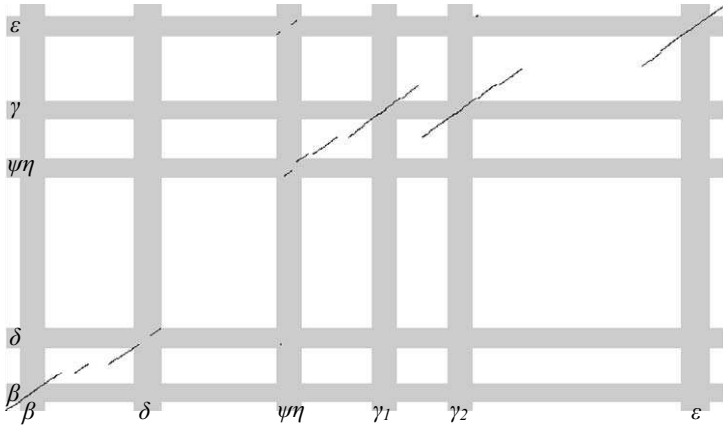


(c) Human-marmoset alignment extracted from the MLAGAN alignments of ENCODE region ENm009.

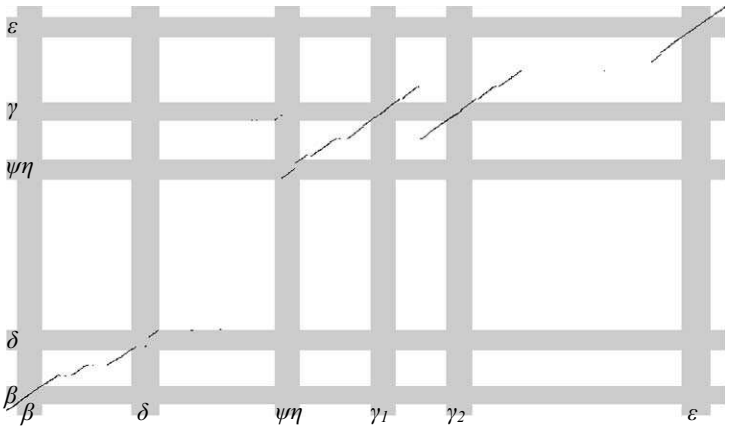
Figure 4.11. Alignments between human and marmoset in the beta globin cluster. All three sets of alignments are similar, largely because of the close evolutionary relationship between human and marmoset. In the MLAGAN alignments, a small region between human $\psi\eta$ and γ_1 is aligned to an obviously incorrect region of marmoset.



(a) Orthologous human-galago alignments extracted from the ROAST multi-species alignments.

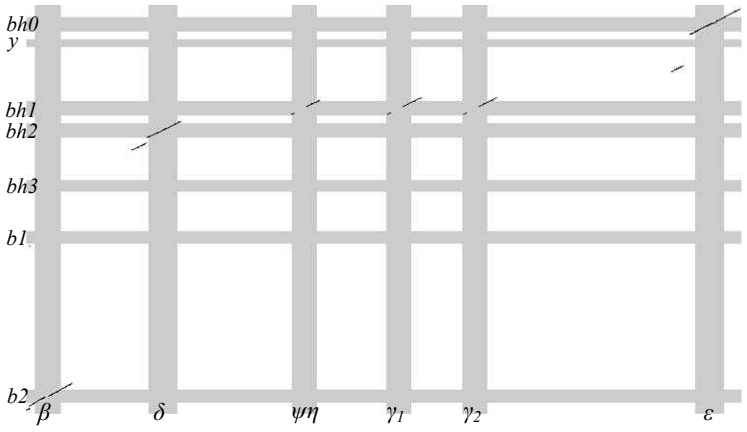


(b) Human-galago alignments extracted from TBA alignments of ENCODE region ENm009.

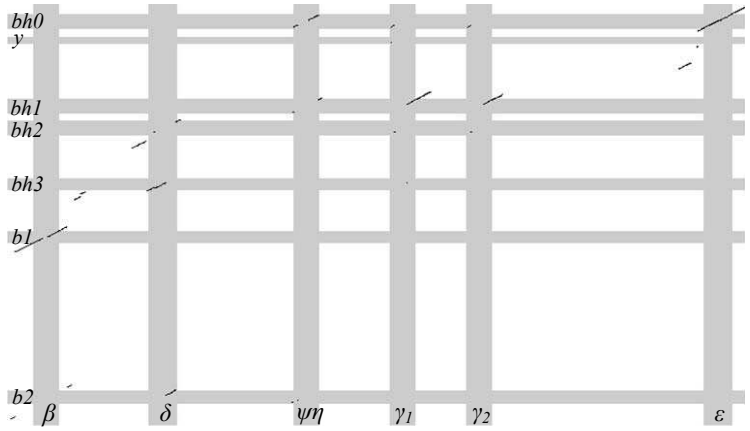


(c) Human-galago alignments extracted from the MLAGAN alignments of ENCODE region ENm009.

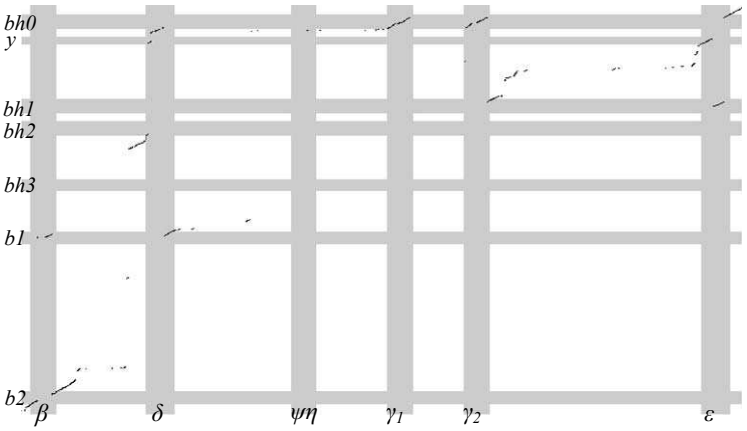
Figure 4.12. Alignments between human and galago in the beta globin cluster. The galago β gene was replaced by δ as a conversion. The major difference among three alignments is that human δ is not aligned to any gene of galago in an orthologous alignment since its orthologous copy in galago is gone. In both TBA and MLAGAN alignments, portions of human $\psi\eta$ are aligned to incorrect regions of galago.



(a) Orthologous human-mouse alignments extracted from the ROAST multi-species alignments.

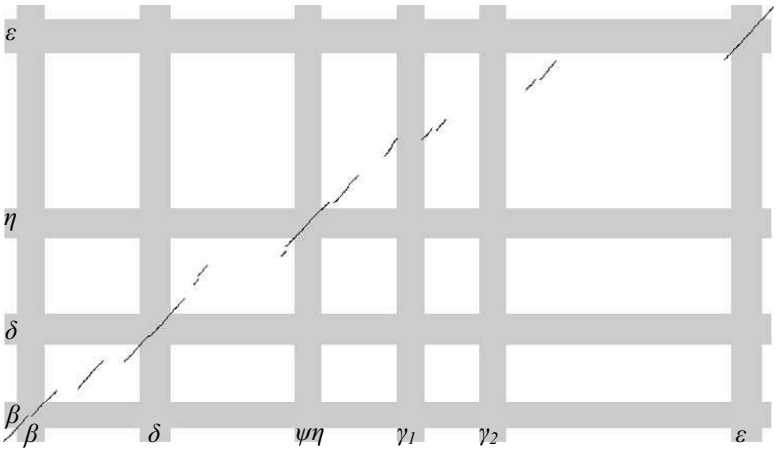


(b) Human-mouse alignments extracted from the TBA alignments of ENCODE region ENM009.

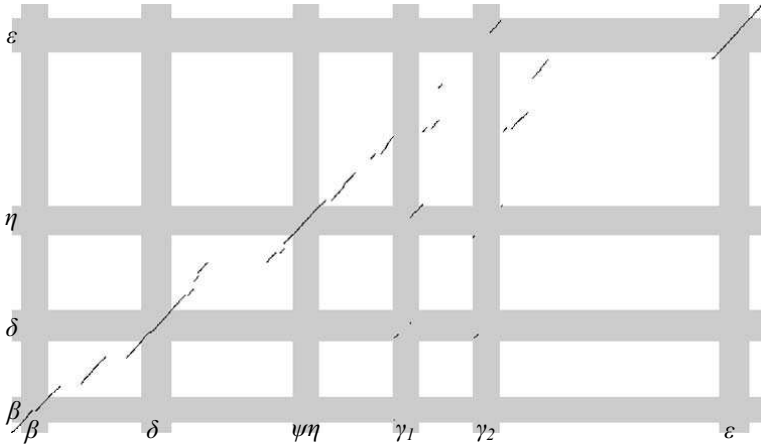


(c) Human-mouse alignments extracted from the MLAGAN alignments of ENCODE region ENm009.

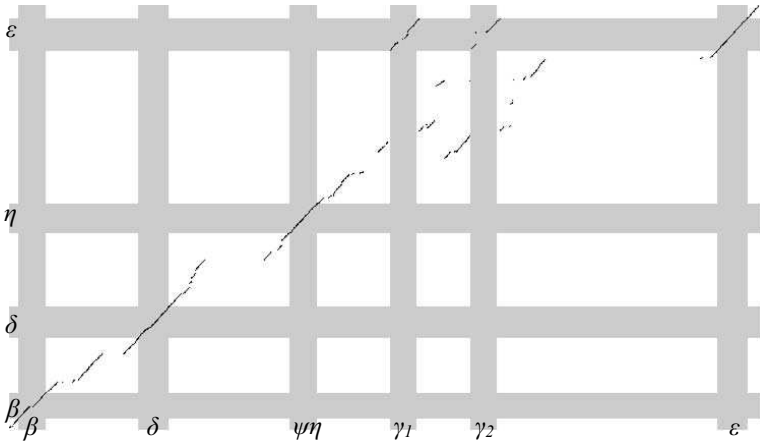
Figure 4.13. Alignments between human and mouse in the beta globin cluster. MLAGAN incorrectly aligns the two human γ genes to mouse ϵ, while other human genes are aligned to other partial regions of different genes of mouse. TBA incorrectly aligns human $\psi\eta$ to mouse ϵ. The alignment in (a) is slightly different from the one in Fig. 3.10 because of different parameter settings.



(a) Orthologous human-dog alignments extracted from the ROAST multi-species alignments.

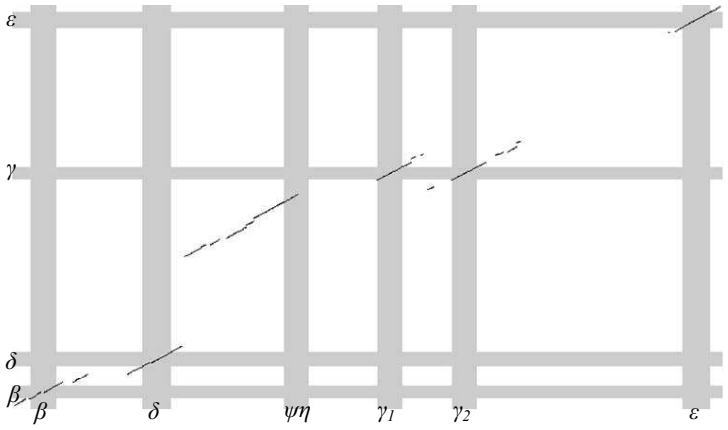


(b) Human-dog alignments extracted from the TBA alignments of ENCODE region ENm009.

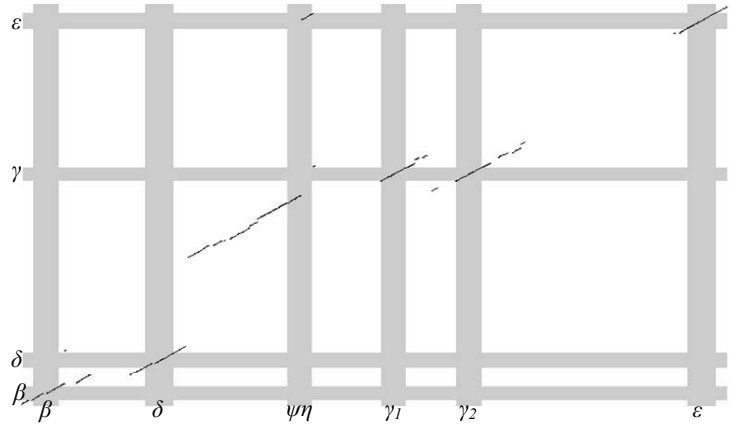


(c) Human-dog alignments extracted from the MLAGAN alignments of ENCODE region ENm009.

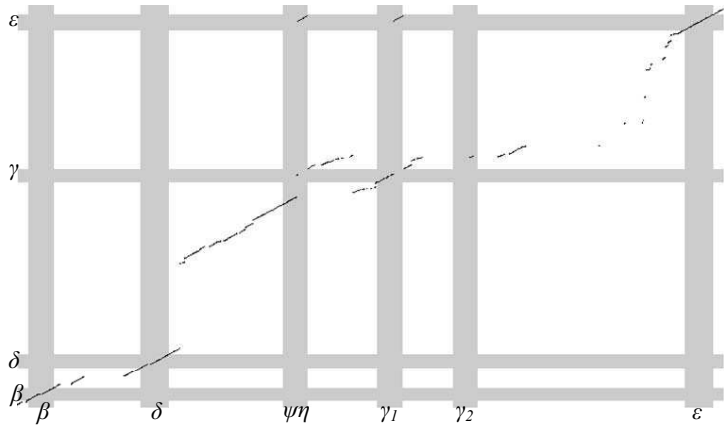
Figure 4.14. Alignments between human and dog in the beta globin cluster. Dog does not have a γ gene. MLAGAN mistakenly aligns two human γ genes to the dog ϵ. TBA mistakenly aligns two human γ genes to portions of dog η and ϵ genes.



(a) Orthologous human-armadillo alignments extracted from the ROAST multi-species alignments.

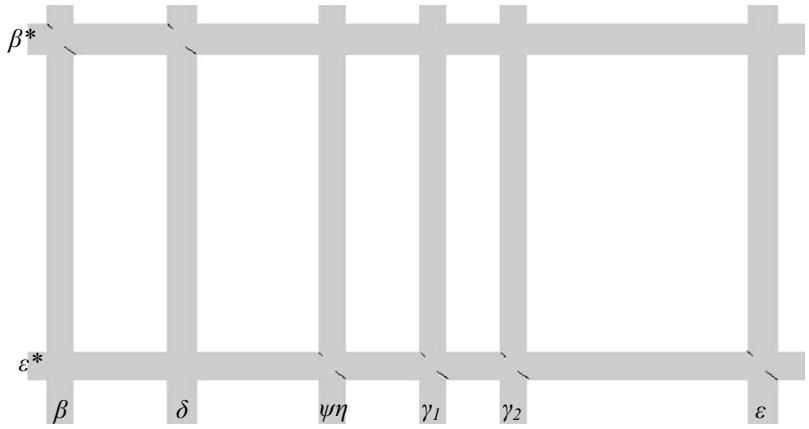


(b) Human-armadillo alignments extracted from the TBA alignments of ENCODE region ENm009.

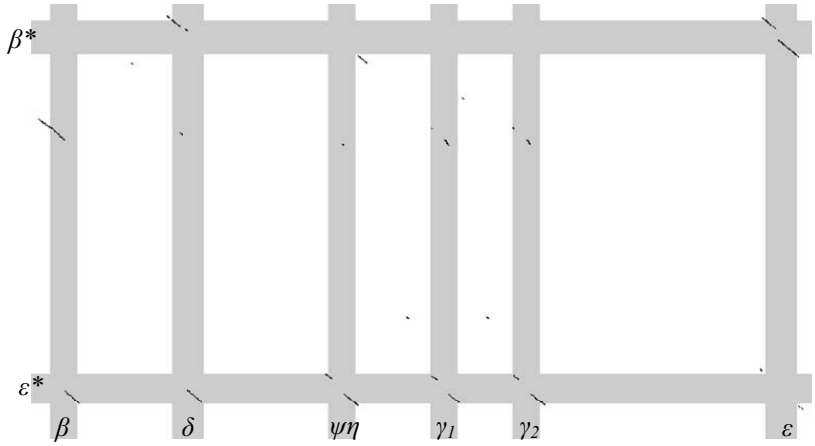


(c) Human-armadillo alignments extracted from the MLAGAN alignments of ENCODE region ENm009.

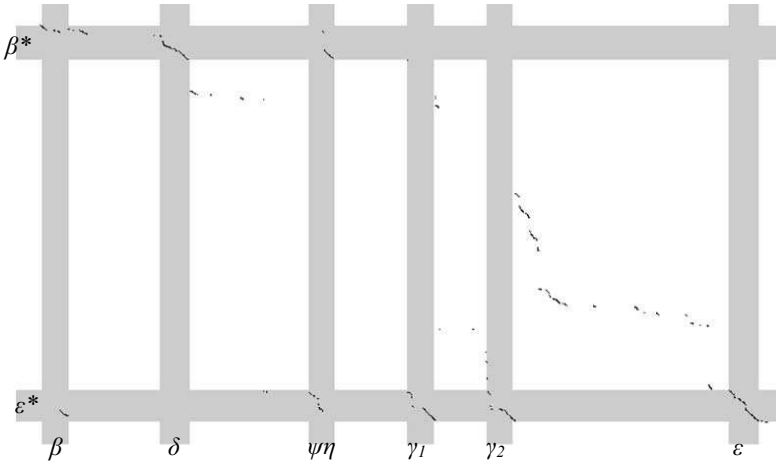
Figure 4.15. Alignments between human and armadillo in the beta globin cluster. Armadillo does not have an η gene. The orthologous alignments and TBA alignment are quite similar except that a small region of human $\psi\eta$ is mistakenly aligned to armadillo ϵ. MLAGAN mistakenly aligns portions of human $\psi\eta$ and γ_1 genes to armadillo ϵ.



(a) Orthologous human-opossum alignments extracted from the ROAST multi-species alignments.



(b) Human-opossum alignments extracted from the TBA alignments of ENCODE region ENm009.



(c) Human-opossum alignments extracted from the MLAGAN alignments of ENCODE region ENm009.

Figure 4.16. Alignments between human and monodelphis in the beta globin cluster. According to Table 2.14, the monodelphis β-related gene is orthologous to human β and δ genes; its ϵ-related gene is orthologous to human $\psi\eta$, γ_1, γ_2 and ϵ genes. This is consistent with the orthologous alignment in subfigure 4.16(a).

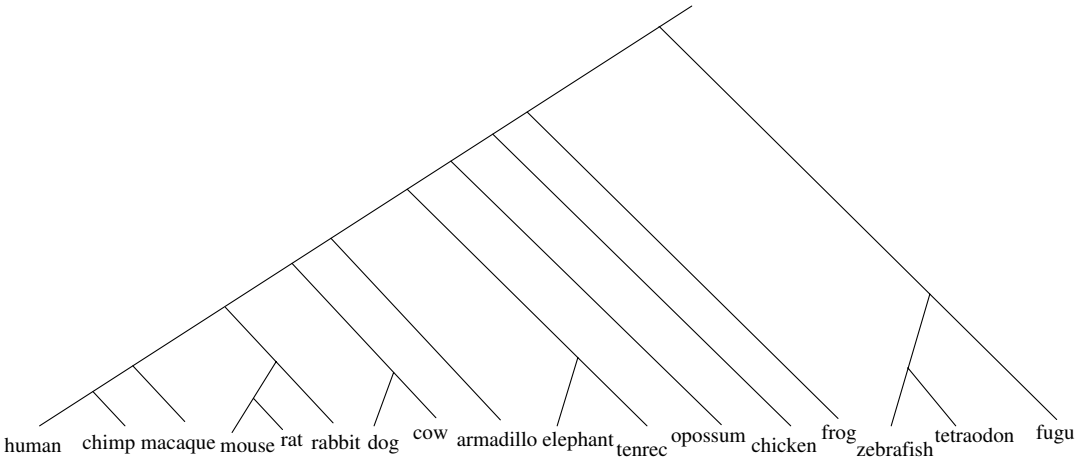


Figure 4.17. The species tree of 17 vertebrates.

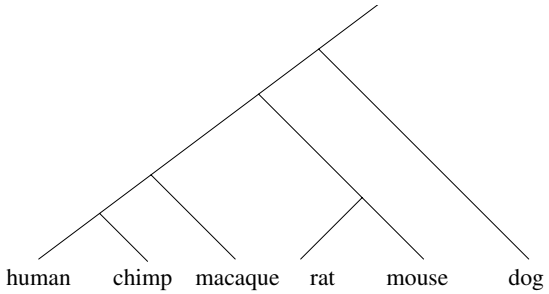


Figure 4.18. The species tree of six completely sequenced and assembled vertebrates.

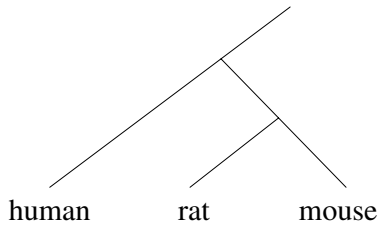


Figure 4.19. The species tree of human, mouse and rat is skewed. Mouse and rat occur at the inner-most nodes.

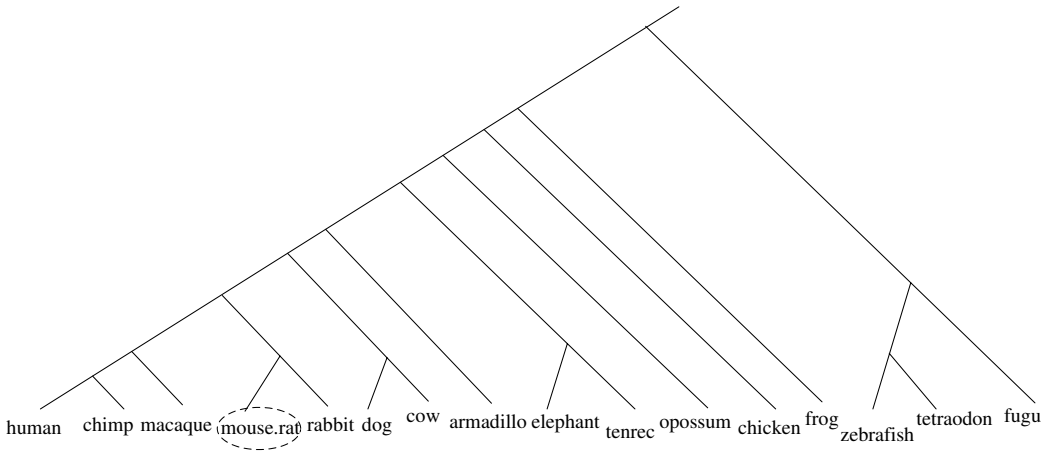


Figure 4.20. The species tree of 17 species, where the mouse-rat subtree is condensed.

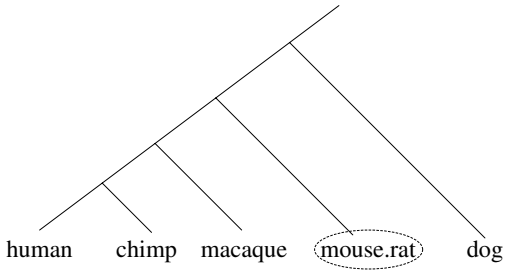


Figure 4.21. The species tree of core species, where the mouse-rat subtree is condensed.

Chapter 5

Reference-independent multi-species orthologous alignment

5.1 Introduction

The previous chapter presents a computational method for orthologous multiple alignment using a reference. In that approach, any sequences of other species that do not align to the reference are discarded. Sometimes it is desired to have all orthologous alignments of all species, which precludes use of a special species as a reference. In another word, the multiple alignment is *reference-independent*. We present our work on this approach in this chapter, which is largely based on a published paper [31], authored by Minmei Hou, Piotr Berman et al.

To represent duplications and other large-scale evolutionary rearrangements, our programs for aligning several genomic sequences produce a set of alignment "blocks", each of which is in essence a traditional alignment of segments from the given sequences or their reverse complements [8]. With duplications, the same sequence position can appear in several blocks. It is useful to note that if rows of a block are pairwise orthologous, then no two rows can be from the same species.

When we tried to build a new alignment program conforming to the requirements that

(1) any two rows of a computed block are orthologous and

(2) any pair of orthologous positions appears together in at least one block,

two hurdles had to be overcome. The first was to distinguish orthologs from paralogs, and for this there was a large literature to draw from. We provide our solution in Chapter 3. The second difficulty was that the number of possible blocks can grow exponentially with the number of sequences and duplications, which is the topic of this chapter. For instance, a straightforward implementation meeting our requirements produced over 900 Mbytes of alignments when applied to intervals containing the alpha globin gene clusters of 20 mammals, where the total length of the original sequences was only 3.9 Mb. We designed and implemented a space-saving strategy that

decreased the amount of output to 8.7 Mb, while still fulfilling requirements (1) and (2). New ideas were required to achieve this savings, and we were led to the development of a theoretical model that turned out, in the general case, to be equivalent to a previously studied NP-complete combinatorial optimization problem, which we will call MinCliqueCov, namely, finding a minimum cardinality set of cliques that contains all edges of a given undirected graph. We show that the graphs we study have special properties, which can be utilized to apply divide-conquer techniques, which would not work well with an arbitrary graph.

Here we describe our graph-theoretic model and derive a theoretical upper bound on the number of blocks that are needed to meet our requirements in an important subclass of problems. Also, using the model, we formulate two heuristic methods, and with the help of our upper bound and some computer simulations, we measure where the two methods lie in the tradeoff between computation time and output size. We also compare our solutions to an existing heuristic method for general graphs [39]. Finally, we describe the performance on the alpha globin gene cluster of our alignment software that is based on the new ideas.

5.2 Methods

5.2.1 A graph-theoretic model

Let $G = (V, E)$ be a graph with vertex set V and edge set E. An m-vertex complete subgraph of G is called an *m-clique*. A *clique cover* of G is a set of cliques whose edges contain every edge $e \in E$. The *clique cover number*, $cc(G)$, of G is defined to be the minimum number of cliques in a clique cover of G.

Assume we align genomic sequences from K species in a genomic region containing a family of tandemly duplicated genes. Suppose that each member of that gene family can be aligned to every orthologous member. In our model, a vertex represents a gene in one of the species, and there is an edge between two vertices if the genes that they represent are orthologous. Thus, we obtain a K-partite graph, which is an *alignment graph* (defined in Section **??**), where each part contains the nodes that represent the gene family members in a given species. A multi-alignment block with pairwise orthologous rows corresponds to a clique, and a set of multi-alignment blocks that contains every pairwise alignment (condition (2) in Section 5.1) corresponds to a clique cover. Thus it would be helpful to solve MinCliqueCov for the alignment graph.

Fig. 5.1 gives an example in which there are three species (A, B and C), each containing two members of a gene family. Each pair of genes from different species is orthologous. In this example, there are eight possible alignment blocks, but four blocks are sufficient to include each orthologous pair in a block.

Unfortunately, MinCliqueCov is NP-hard [47]. The restriction to multi-partite graphs does not make this problem easier since a graph of n vertices is trivially an n-partite graph in which each part has only one vertex. Various techniques have been applied to solve MinCliqueCov and closely related problems [27, 54, 11]. For instance, when the degree of any vertex in G is at

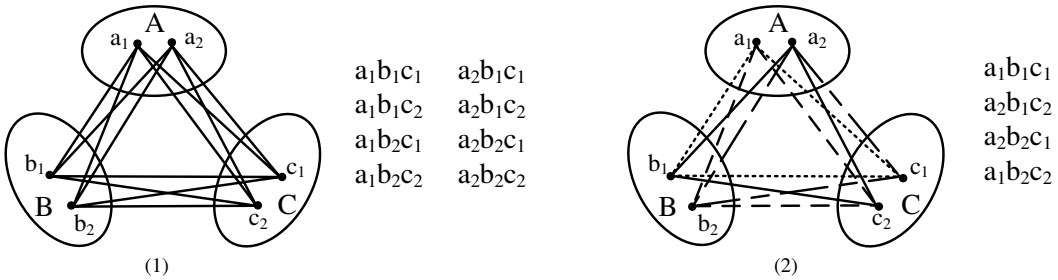


Figure 5.1. A trivial example on minimum clique cover. Panel 1 shows 8 cliques to cover all edges, while panel 2 shows 4 cliques to cover all edges.

most 4, the problem is solvable in linear time [55].

5.2.2 A special case of the minimum clique cover problem and its upper bound

In this section, we investigate the graph structure that arises under certain natural conditions, namely when all duplications have occurred after all speciation events. That is, we suppose that each gene is orthologous to every gene in a different species. Moreover, to keep things simple, we supposed that each of K species has precisely P copies of the gene. The resulting alignment graph is a complete K-partite graph: each part has P nodes, and there is an edge between any two nodes that are in two different parts. We denote such a graph by $G_{K,P}$. Note that the shape of such graph is determined for each pair of K and P. Thus MinCliqueCov restricted to graphs of the form $G_{K,P}$ has $O(n)$ distinct instances with n nodes, where $n = KP$ (because n can be factored into KP in less than n ways). It was shown that problems with polynomially many instances per size cannot be NP-hard (unless P=NP) [7]. But even for the case of $P = 2$, we do not know the exact solution. The technical result of this chapter does not imply the problem is easy either. The purpose of this section is to derive a non-trivial upper bound on $cc(G_{K,P})$, which will help us to interpret the results of simulations that we report below.

First, though, let us mention lower bounds. For $K \geq 2$, $cc(G_{K,P}) \geq P^2$, since there are P^2 edges between any two parts of the graph, and any clique can contain at most one of those edges. Moreover, it was recently proved that $cc(G_{K,P}) \geq \log_b(KP)$, where $b = \frac{P}{(P-1)^{(P-1)/P}}$ [14]. That lower bound is approximately equal to $P(\log_P K + 1)$.

For an upper bound, it has been known for some time that $cc(G_{K,2}) = \Theta(\log_2 \mathrm{K})$ [27], but we seek a bound for general P. Assume $G_{K,P}$ has the following node set and edge set:

$$V = \{u(i,j) : \ 0 \leq i < K \text{ and } 0 \leq j < P\}$$
$$E = \{\{u(i,j), u(k,l)\} \subset V : \ i \neq k\}$$

To present a recursive construction of small clique covers of $G_{K,P}$, we start with two simple observations.

Observation 1. *Let $U \subseteq V$. If $\mathcal{C}$ is a clique cover of G, then $\{C \cap U : \ C \in \mathcal{C}\}$ is a clique cover of $G(U)$. Thus $cc(G(U)) \leq cc(G)$.*

Observation 2. *If $\mathcal{C}_i$ is a clique cover of $< V, E_i >$ for $i = 1, 2 \ldots, k$ then $\bigcup_{i=1}^{k} \mathcal{C}_i$ is a clique cover of $< V, \bigcup_{i=1}^{k} E_i >$.*

The edges of $G_{K,P}$ can be split into two sets

$$E_0 = \{\{u(i,j), u(k,l)\} \in E : \ j = l\}$$
$$E_1 = E - E_0$$

We can cover $< V, E_0 >$ with cliques $C_j = \{u(i,j) \in V\}$, $j = 0, \ldots, P-1$. Now it remains to find a clique cover for $G^1_{K,P} =< V, E_1 >$.

If there are $K = M \times L$ species, the edges of $G^1_{ML,P}$ can be represented as $E_2 \cup E_3$, where

$$E_2 = \{\{u(i,j), u(k,l)\} \in E_1 : \ \lfloor i/L \rfloor \neq \lfloor k/L \rfloor\}$$
$$E_3 = \{\{u(i,j), u(k,l)\} \in E_1 : \ i \ \mathbf{mod}\ L \neq k \ \mathbf{mod}\ L\}$$

To make that split more intuitive, put the ML parts of $G_{ML,P}$ into a matrix with M rows and L columns. An edge between different parts will either connect parts from different rows, or parts from different columns. Denote the set of edges connecting parts from different rows by E_2, and the set of edges connecting parts from different columns by E_3. Note that E_2 and E_3 are not necessarily disjoint.

Lemma 1. $cc(< V, E_2 >) \leq cc(G^1_{M,P})$.

Proof. Consider a clique cover $\mathcal{C}$ of $G^1_{M,P}$. Obtain $\mathcal{C}'$ by transforming each clique $C \in \mathcal{C}$ into

$$C' = \{u(i,j) \in V : u(\lfloor i/L \rfloor, j) \in C\}.$$

C' is still a clique, because if $\lfloor i/L \rfloor \neq \lfloor k/L \rfloor$ than $i \neq k$. Now an edge $e = \{u(i,j), u(k,l)\}$ of E_1 is covered by $\mathcal{C}'$ unless $\lfloor i/L \rfloor = \lfloor k/L \rfloor$, hence $e \notin E_2$. ❒

Lemma 2. $cc(< V, E_3 >) \leq cc(G^1_{L,P})$.

Proof. Similar to Lemma 1. ❒

Lemma 3. $cc(G^1_{P,P}) \leq P(P-1)$ *if P is prime.*

Proof. We construct a set of cliques $C_{a,b} = \{\{u(i, ai + b \ \mathbf{mod}\ P) : \ 0 \leq i < P\}$ where $0 < a < P$ and $0 \leq b < P$. Given an edge $\{u(i,j), u(k,l)\} \in E_1$ we can find the parameters a, b of the clique that covers it by solving linear system

$$\begin{aligned} ai + b &= j \ \mathbf{mod}\ P \\ ak + b &= l \ \mathbf{mod}\ P \end{aligned}$$

This system yields the following equation for a: $a(i-k) = j - l \ \mathbf{mod}\ P$. Because $i \neq k$, $i - k$ has a reciprocal $\mathbf{mod}\ P$, and because $j \neq l$, the a computed from this is non-zero. ❒

Theorem 1. $cc(G_{K,P}) \leq P + P(P-1)\lceil \log_P K \rceil$ *if P is a prime.*

$$\overbrace{\{a_0,b_0,c_0\}}^{E_0} \quad \overbrace{\begin{array}{cccc}\{a_0,b_1,c_2\} & \{a_1,b_0,c_3\} & \{a_2,b_0,c_1\} & \{a_3,b_0,c_2\}\\ \{a_0,b_2,c_3\} & \{a_1,b_2,c_0\} & \{a_2,b_1,c_3\} & \{a_3,b_1,c_0\}\\ \{a_0,b_3,c_1\} & \{a_1,b_3,c_2\} & \{a_2,b_3,c_0\} & \{a_3,b_2,c_1\}\end{array}}^{E_1}$$
$$\{a_1,b_1,c_1\}$$
$$\{a_2,b_2,c_2\}$$
$$\{a_3,b_3,c_3\}$$

Figure 5.2. Clique cover of $G_{3,4}$ with 16 cliques.

Proof. By Observation 1, it suffices to prove that $cc(G_{K,P}) \leq P + aP(P-1)$ for $K = P^a$, where a is an integer. Because we can cover E_0 with P cliques, it suffices to prove that $cc(G^1_{K,P}) \leq aP(P-1)$. We can show it by induction on a. For $a = 1$ this is proven in Lemma 3. Assuming it is proven for $a-1$, we have $K = ML$ where $M = P$ and $L = P^{a-1}$. This allows to apply Lemmas 1 and 2 to show that $cc(G^1_{K,P}) \leq (a-1)P(P-1) + P(P-1) = aP(P-1)$. ❐

When P is not a prime, we can use the above result for P', where P' is the smallest prime larger than P. For example, when $P = 6$ and $K = 2$, we have a trivial solution with 36 cliques (indeed, edges), but when $K = 3$, we can use a solution for $P' = 7$ that has 49 cliques, and this solution works for $K \leq 7$, while for $K \leq 343$ we have a solution with $7 + 2 \times 42 = 91$ cliques.

Of course, there may exist better solutions. For example, it is easy to find a solution for $G^1_{3,4}$ with 12 cliques; see Fig. 5.2. We can apply the reasoning of Theorem 1 to show that $cc(G_{K,4}) \leq 4 + 12\lceil \log_3 K \rceil$ which, except for $K = 4, 5$, is better than $5 + 20\lceil \log_5 K \rceil$.

Thus, $cc(G_{K,P})$ is at most $P(P-1)\log_P K + P$ when P is a prime number and K is in power of P. In cases that P and K do not satisfy these conditions, the values can be approximated by the nearest prime number and its power. This upper bound is conjectured to be tight, but this has not been proved.

5.2.3 Heuristic solutions

MinCliqueCov is not only NP-hard, but also has no efficient approximation algorithm unless NP=P [41], so we have to use heuristics. We propose two heuristic methods for generating clique covers for complete multi-partite graphs studied in the previous section. It is then relatively straightforward to adapt these methods for the multi-alignment problem in tandem gene clusters, even in the general case of arbitrary orthology relationships; the next section discusses some of the issues that arise. However, in the idealized setting of $G_{K,P}$, with the upper bound derived above, we can evaluate how close they come to an ideal reduction in output size.

Both heuristic methods follow a divide-and-conquer strategy: Partition G into two graphs, G_1 and G_2, each having about $K/2$ parts, find clique covers CC_1 and CC_2 of G_1 and G_2 respectively, and merge them to obtain a clique cover CC of G. While these methods do not give the fewest cliques, they efficiently find a relatively small clique cover. The merge procedures can be described as follows, where "uncovered" refers to an edge not currently in a clique in CC.

Heuristics for MinCliqueCov were studied already in 1978 [39]. Recently an exact solution was also proposed [26] for the general graph. Unfortunately, our complete multi-partite graph is quite dense, and the reduction rules from [26] cannot be applied here. Thus we only compare our heuristic methods' performance with [39]'s method.

```
Merge I (CC_1, CC_2)
1   CC ← φ
2a  while there exists an uncovered edge
3       For each pair of cliques c_i and c_j from CC_1 and CC_2 respectively
4           u_ij  ← the number of uncovered edges of c_i ∪ c_j
5       (c_maxi, c_maxj) ← the pair with maximum u_ij
6       insert c_maxi ∪ c_maxj to CC
7   Output CC

Merge II (CC_1, CC_2)
1   CC ← φ
2b  while there exists an uncovered edge (u, v) between subproblems
3       c_1  ← a clique from CC_1 that contains u
4       c_2  ← a clique from CC_2 that contains v
5       insert c_1 ∪ c_2 into CC
// We still have to incorporate unused cliques from each subproblem
6   while there exists an unused clique c_1 from CC_1
7       c_2  ← an unused clique in CC_2; if none, then any clique in CC_2
8       insert c_1 ∪ c_2 into CC
9   while there exists an unused clique c_2 from CC_2
10      c_1  ← an unused clique in CC_1; if none, then any clique in CC_1
11      insert c_1 ∪ c_2 to CC
12  Output CC
```

Merge I forms a new clique from the pair of cliques that maximizes the number of additional covered edges. Merge II processes cliques of CC_1 and CC_2 in a random order and forms any new clique that covers at least one new edge. Running times are dominated by the number of executions of the loops $2a$ and $2b$, respectively, which are essentially the number of cliques generated. The lowest-level operation inside loop $2a$ is to examine whether an edge is covered or not; for loop $2b$ it is to decide whether a clique contains a certain edge or not. Both operations involve an array access and can be regarded as taking unit time. The number of unit operations inside loop $2a$ is $|CC_1| \cdot |CC_2| \cdot K_1 \cdot K_2$, where K_1 and K_2 are the number of partitions (species) of G_1 and G_2. The number of unit operations inside loop $2b$ is $|CC_1| \cdot K_1 + |CC_2| \cdot K_2$. The performance of these two methods, in terms of both actually running time and the number of cliques generated, is analyzed below.

5.2.4 Application in the aligner program

The above divide-and-conquer methods can be adapted to so-called "progressive" multiple alignment programs, which work leaves-to-root in the phylogenetic tree for the given species. At a tree node, these aligners merge multiple alignments from the sub-trees, which is analogous to merging cliques for subgraphs G_1 and G_2, except that the split into subproblems might not be balanced. The process of merging blocks from the left and right subtree is guided by pairwise alignments between a species in the left subtree and a species in the right subtree. The set of

multi-alignment blocks corresponding to the tree's root constitutes the multiple alignment of the original K species.

Thus, the use of the guide tree by the aligner can be viewed as the recursive partition of the alignment graph, and such a partition can be used both by MERGE I and MERGE II. In our current reference-independent orthologous multiple sequence aligner, BOAST, we have decided to apply MERGE II since our tests indicated that MERGE I could be about 1000 times slower (see Fig. 5.4). While MERGE I produced fewer cliques(see Fig. 5.3), i.e., alignment blocks, MERGE II produced a number that was acceptably small.

We need to address the following issue. The alignment graph generally has fewer edges than a complete K-partite graph, since some speciation events are preceded by duplications. Consider what we should expect if we have a perfect alignment graph, with all ortholog/paralog relationships properly diagnosed. We have two siblings T_1 and T_2 with common ancestor T and let c_i be a clique in the subgraph of T_i, $i = 1, 2$. One can see that either $c_1 \cup c_2$ is also a clique, or there are no edges between c_1 and c_2 (see 4.2.1 for the analysis of orthologous inference). In this case, when MERGE II selects a pair of cliques with union that covers at least one new edge (lines 3-4), this union is a clique. When MERGE II processes an unused clique c_1 in line 7 and 10, we first look for a clique c_2 in another sub-tree that is connected to c_1, and the union of them becomes a new clique; if none exists we simply add c_1 to the output.

Clearly, this adaptation may be incorrect if we have an imperfect alignment graph, ie., with some edges established wrongly and some correct edges missing. Consequently, it may happen that a pair of selected cliques, c_1 and c_2, has some connections, but not all. If only a minority of the possible edges between c_1 and c_2 are present, we behave as if c_1 and c_2 were not connected at all, and if the majority is present, we behave as if $c_1 \cup c_2$ was a clique. This majority criterion can have the effect of correcting some errors created by incorrect identification of orthologous pairwise alignments from TOAST (chapter 3). Note that we have to choose between two kinds of discrepancies in the output. One is that we do not include all actual orthologous relationship in a blocks. The second is that we contaminate a block with a paralogous relationship. If one kind of discrepancy is more harmful than the other, we can replace the majority criterion with some other ratio.

5.3 Results

5.3.1 Simulations

Methods Merge I and II were tested on graphs of the form $G_{K,P}$, i.e, complete K-partite graphs, where each part has P nodes. The results are plotted in Fig. 5.3(a) and 5.3(c), which shows that Merge I substantially outperforms Merge II. Values of upper bound $P + P(P-1)\lceil \log_p K \rceil$ are shown in Fig. 5.3(b). The bounds for P=4 are determined by $cc(G_{K,4}) \leq 4 + 12\lceil \log_3 K \rceil$ discussed below Theorem 1. The lower bound from [14] is lower than the trivial P^2 bound in most of our cases, so we do not show it in the figure.

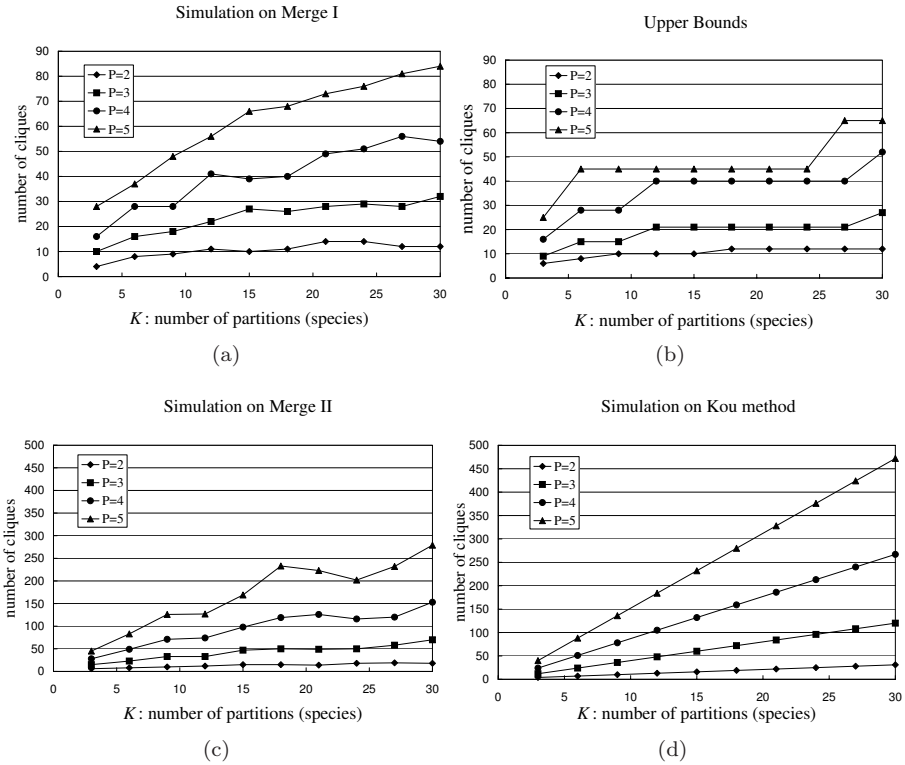


Figure 5.3. Simulations of three heuristic procedures together with plots of upper bounds, with different values of K and P. The curves, top to bottom, refer to P=2,3,4,5.

Though MERGE I produces fewer cliques, it requires a longer running time. To estimate the difference in CPU requirements, we counted the numbers of previously described unit operations, so the impact of different number of cliques produced by the two methods is included. Although it is possible to improve the time efficiency of MERGE I by designing better data structures, MERGE I will always be slower than MERGE II.

The heuristic method from [39] takes even more time than Merge I, so its running time analysis is not shown here. However, we plot its clique numbers. As shown in Fig. 5.3(d), it produces more cliques than Merge II for any instances.

5.3.2 The alpha globin gene cluster

The alpha globin clusters of a number of mammals have been extensively studied recently [33]. There are four types of genes in those clusters: ζ, αD, α and θ. Each species discussed below has exactly one αD-related gene, so those genes are not pertinent to this analysis. The studied species have 1 to 3 ζ-related genes, 1 to 4 α-related genes, and 0 to 3 θ-related genes (counts include pseudogenes). Table 5.1 shows details. Each gene copy is regarded as a node in our graph.

Our earlier TBA program [8, 43] guarantees that every position in the reference sequence

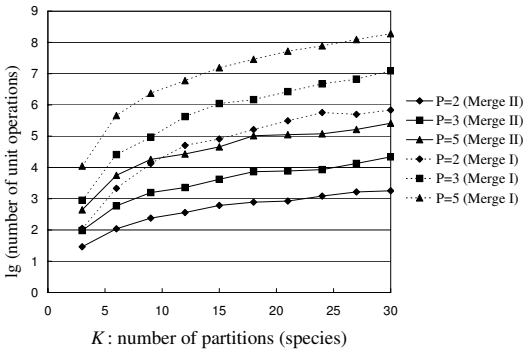


Figure 5.4. Comparisons of running time on methods Merge I and II.

(human in this case) is in exactly one multiple alignment block, and thus TBA is not able to capture all pairwise orthologous relationship in a tandem gene cluster. For example, each human α gene is aligned to only one rat α gene, despite the fact that a human α is actually orthologous to two more rat α genes, and those alignments are lost. Our new aligner, called BOAST, which implements MERGE II, captures all pairwise orthologous relationship.

We aligned sequences containing the alpha globin clusters from 20 mammals. Each sequence is around 200K bases. Both TBA and BOAST utilize pairwise alignments computed by blastz [58]. For BOAST, the pairwise alignments are filtered by TOAST (chapter 3), which retains only the putatively orthologies (i.e., deletes paralogous matches). After computation of pairwise alignments, TBA produces 7.5 Mb of alignments after 170 CPU seconds, while BOAST produces 8.7 Mb of alignments in around 112 CPU seconds.

Each aligner outputs a set of blocks, whose endpoints do not in general correspond to gene or exon boundaries. Moreover, each functional globin gene has three exons, and many alignments do not extend from one exon to the next. We estimated how many cliques were formed for each type of genes as follows. For a given gene sequence, we manually determined three positions distributed roughly evenly throughout the gene, and counted the number of times each position appeared in the multi-alignment blocks; the maximum of the three counts was used to estimate the number of times the gene copy appears in the blocks. In this example (alpha globin clusters of 20 species), the clique sizes (i.e., the number of rows in a multi-alignment block) vary from 4 to 20, with most between 16 and 20. Alignment blocks containing θ-related genes have at most 17 rows since three species do not have a θ-related gene. Other blocks with less than 20 rows result from a number of factors, including inconsistent pairwise alignments, pseudogenes, or retention of a non-orthologous alignment. With many-to-many orthologous relationships, there will be combinatorial increase on the number of alignment blocks. Table 5.1 shows that utilizing MERGE II, we reduce the number of alignment blocks to 26 for ζ-related alignments and 58 for α-related alignment. It means that 26 and 58 multiple alignment blocks contain all pairwise orthologous relationships of a certain region for ζ-related and α-related genes respectively.

The BOAST alignments are reference-independent, which means that no sequence data from any of the species is missing from the alignment. One of our tools extracts a reference-sequence-

Table 5.1. Number of copies for each type of alpha globin genes of 20 mammals, and number of cliques formed for each type of genes in the multiple alignment employing the heuristic Merge II. When the number of genes is given in the form *x-y*, *x* refers to genes, and *y* refers to genes together with pseudogenes.

gene	ζ-related	α-related	θ-related
armadillo	1	1	0
baboon	2	1-2	1
cat	2	1	0
chimp	2	1-3	1
colobus	2	2-4	1
cow	2	2	1
dog	2	1	1
dusktit	1	2	1
galago	2	2	1
hedgehog	2	2-3	1
human	1-2	2-3	1
lemur	1	2	1
macaca	2	2	1
marmoset	1	1-2	1
mouse	1	2-3	1-3
owlm	1	2	1
pig	1	1	1
rat	1	3	3
rfbat	1	1	0
sqmonkey	2	2	1
#cliques	26	58	7

based alignment from the BOAST alignment for any specified reference. This gives an alignment similar in size to the output of a typical reference-based multiple aligners, for example MULTIZ [8]. Moreover, the BOAST alignments capture complete and accurate orthology information, which is currently lost by other aligners.

5.4 Discussion and further work

In gene clusters having a significant level of lineage-specific duplications (i.e., producing many-to-many orthology relationships), it is not practical to enumerate all possible multi-alignment blocks having pairwise orthologous rows. However, we have shown here that it is still frequently feasible to produce a set of blocks with the property that every pair of orthologs appears together in one of the blocks. The essence of the situation is captured by the problem of finding a minimum-cardinality clique cover. Both the problems of the finding the minimum number of cliques to cover all edges and the minimum number of cliques to cover all nodes (see below) are NP-complete. However, our simulations show that sizes can in practice be reduced (especially by Merge I) to be close to the upper bound. Though our alignment program has only incorporated Merge II, it still dramatically reduces the alignment size, and makes it feasible to align tandem gene clusters from many species. In the future, we hope to implement Merge I in our alignment program.

Though the clique cover problem for an arbitrary graph is NP-complete, it is open whether the problem's restriction to alignment graphs is intractable. The structure of the graphs is constrained by the phylogenetic tree for the species in question and by properties of the orthology relationship, and these restriction might be helpful for determining clique covers.

It also remains to investigate other criteria for aligning regions containing duplicated segments or genes. For instance, one could loosen the requirement that each orthologous pair of positions occur in two rows of the same block, and ask only that a position of one species that has an ortholog in a second species must appear in the same block as some (perhaps different orthologous) position in that second species. In essence, this can be modeled as seeking the minimum number of cliques to cover all *nodes* in the graph constructed above, which in general requires fewer cliques than the problem studied here. We think that the problem of aligning tandem gene clusters is sufficiently important that a variety of approaches should be investigated.

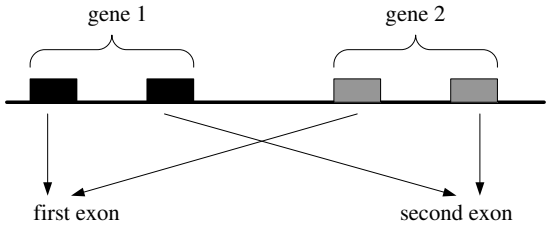
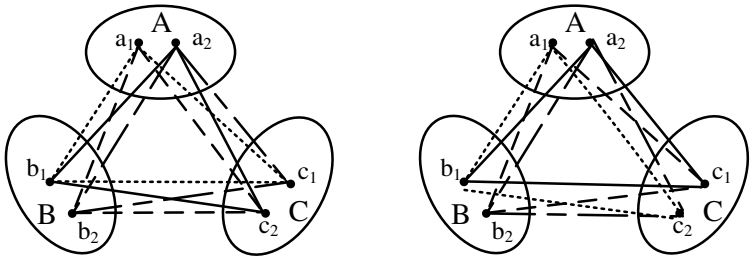


Figure 5.5. A sequence with two genes. These genes are duplicated recently. Each gene has two exons.

However, there is a serious problem of reducing the alignment size, which prevents BOAST being as useful as ROAST. One advantage of reference-independent orthologous multiple alignment is that it is convenient to create reference-dependent orthologous alignments by threading projection (Section 4.2.3) using any of the species as the reference. While it is desirable to have one gene align to another complete gene, it can be shown that it is sometimes impossible to achieve this goal using the strategy of minimum clique cover to reduce the alignment size. For example, shown in Fig. 5, a certain type of gene has two exons, separated by a large intron, and the gene has been independently duplicated in three species. There is a many-to-many orthology relationship between each pair of these species. Because the intron is large, the chaining procedure of TOAST fails to chain the alignment blocks containing the two exons. BOAST produces the multiple alignment blocks as if the two exons are unrelated regions, and two clique covers are formed separately for them, each containing alignments for one exon. It is very possible that these two clique covers are different (Fig. 5.6). Using species A as the reference in this example, no threading projection exists to align a gene of species A to a complete gene of species B and a complete gene of species C at the same time.



$a_1b_1c_1$
$a_2b_1c_2$
$a_2b_2c_1$
$a_1b_2c_2$

(a) Clique cover of the first exon.

$a_1b_1c_2$
$a_2b_2c_2$
$a_2b_1c_1$
$a_1b_2c_1$

(b) Clique cover of the second exon.

Figure 5.6. The two clique covers for the two exon regions are different. A, B, and C are three species. Each has the sequence structure as in Fig. 5.5. There is a many-to-many orthology relationship between each pair of species, as shown in Fig. 5.1(1). The two sub-figures are for two exons separately.

Chapter 6

Establishing homologous groups within a genome

6.1 Introduction

All previous chapters are related to orthologous alignment of gene clusters of different species. This chapter deals with another set of problems also related to genomic sequences containing duplications. We construct homologous groups of intervals with a given genome, where a group can be a complete gene family. This chapter is largely based on a published paper [32].

A main obstacle for DNA search methods is that on one hand a large portion of a complex genomes is composed of highly repetitive sequences called *repeats*. Repeats are divided into two types based on their patterns of dispersion: tandem repeats and interspersed repeats. These highly repetitive sequences are less likely to contain functional elements. On the other hand, certain functional regions, including gene clusters, are also of a repetitive nature; they are derived from duplication events, and we call them *duplicates*. We are more interested in the latter type, especially gene families. A gene family or subfamily may form a cluster. When such a cluster has the form of a tandem array, its members are called tandem duplications. Fig. 6.1 shows examples of tandem repeats and tandem duplications. Our goal is to exclude tandem repeats and extract homologous gene groups. Some of the identified groups are un-annotated interspersed repeats. Additional post-processing can use annotations, dispersion patterns etc. to differentiate interspersed repeats from other groups.

Several tools exist to detect repetitive sequences in a whole genome. They include RepeatMasker [60], RECON [3], RepeatGluer [52], RepeatScout [53] and PILER [19]. Besides masking low-complexity DNA sequences, RepeatMasker uses an extensive library of known repeats to search for interspersed repeats, requiring a separate library for each genome. The others use a variety of de novo methods; each has its own limitations and advantages. RepeatScout uses high frequency seeds, and thus is less likely to find lower frequency repetitive sequences, e.g. moderate

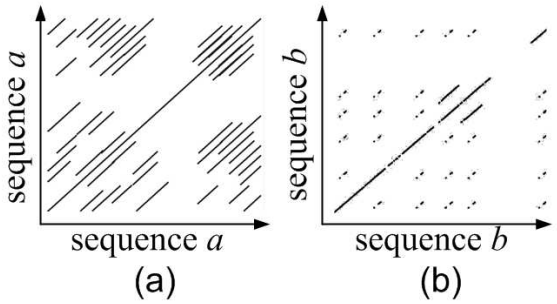


Figure 6.1. Dot plot of self-alignments. (a) Sequence a is from a partially degraded tandem repeat region; (b) Sequence b is from a tandem duplication region of beta-globin genes.

and small gene families. PILER aims for high specificity and thus obtains low sensitivity. One drawback of RECON is its low speed. [5] looks for clusters specifically in non-coding regions. Many of these papers mention the problem of identifying the correct boundaries for each repeat copy. This problem is even more severe for us since, in alignments, diverged genes tend to have less consistent boundaries than repeats. Besides the above tools which look for DNA repeat sequences, other tools cluster protein sequences to classify protein families [12, 20, 37]. Many of them tried to solve a problem caused by the presence of multiple domains in a protein: the same domain exists in many families and thus joins different families into a larger family. We face a similar problem, called alignment drifting, caused by adjacent regions (details in Section 6.3.2.1 and [5]). In Bejerano's method, a graph representing similarities among genomic regions is recursively split until the sub-graphs are sufficiently dense. We apply a different approach aiming to group all homologs together in a consistent manner, as we not only split, but also merge to undo improper splits. Our method must be very efficient to process whole genomes. There is also much research on segmental duplications. In some studies, only duplicates of sequence identity >90% and length >10K bases are considered as segmental duplications [2]. We want to consider more diverged and shorter duplicates.

Duplication is believed to be a main evolutionary mechanism to create new functionalities. Identification of homologous groups provides an opportunity for function inference if some of the group members happen to have known attributes. Clustering methods based on protein sequences have limitations, because the availability of protein sequences depends on the gene expression and annotation extent for a particular genome. Moreover, clustering on the basis of a protein database may be misled by redundant entries [12, 51]. Whole-genome DNA clustering complements existing methods of finding families of genes. Genes sharing high similarity might share similar functionalities. Thus establishing homologous groups can speed up genome annotation. Even if sequences with high similarity have different functionalities, homologous grouping provides resources for gene phylogeny and genome structure research. Unlike methods based on protein sequences, HomologMiner groups non-coding regions as well, the importance of which is increasingly recognized [15]. Furthermore, while contemporary research increasingly depends on orthologous alignment, most orthology assignment tools assume a one-to-one relationship. Orthology assignment is aided by the complete set of paralogs.

We aim to identify all duplicates excluding tandem and annotated interspersed repeats in a genome and to construct proper homologous groups. Our main contributions include:

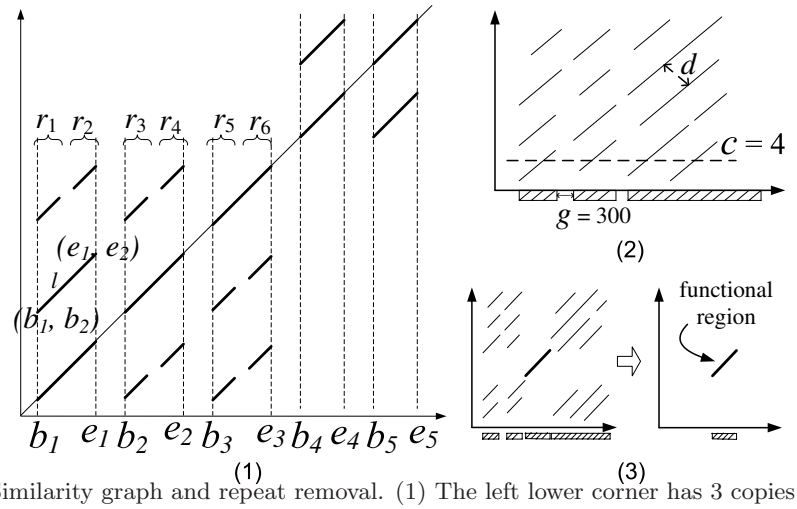


Figure 6.2. Similarity graph and repeat removal. (1) The left lower corner has 3 copies of homologous genes, and they produce 9 diagonal shortChains. The right upper corner has 2 copies of another type of homologous genes, and they produce 4 diagonal shortChains. (2) shows parameters defining a tandem repeat region; (3) shows the result of repeat removal when repeats and functional regions are mixed.

1. We address the problem of establishing correct boundaries of duplicates. We define the notion of a consistent duplicates family and have verified that such families tend to provide meaningful biological units, e.g. complete genes.

2. We design heuristics to find families of duplicates that satisfy the definition described above.

3. We test the validity and usefulness of the obtained duplicate families for several mammalian species (particularly human).

4. We demonstrate our tool's ability to identify gene families, tandem gene clusters, and un-annotated interspersed repeats.

6.2 Definitions, rules and properties

Several sources [3, 19] point out that it is relatively easy to detect repetition, but more difficult to define biologically reasonable families. We tackle this problem with the goal of constructing homologous groups with high accuracy. A *(homologous) group* (HG) is composed of homologous group members. We try to ensure two properties for each HG:

1. **Purity** Every pair of members in a group is homologous;

2. **Completeness** All homologous members are in the same group, unless no reliable alignment exists to support the homologous relationship.

Though sequences with similarity are not necessarily homologs, sequence similarity is still the best and most convenient criterion to identify homologs. Our method starts with a dot-plot of the self-alignment of the entire genome. A dot-plot shows local similarity between two sequences (Fig. 6.2). Such a plot is basically a set of diagonal (*similarity*)*lines*, each representing a gapless

alignment of *region* $[b_1, e_1]$ with region $[b_2, e_2]$. Regions corresponding to a line are potentially homologous. We define the following terms:

- *High Score Pair* (HSP) is a gap free local alignment between two sequences, represented by HSP$<b_1, e_1, b_2, e_2>$ corresponding to a line in a dot-plot from point (b_1, b_2) to (e_1, e_2). The regions $[b_1, e_1]$ and $[b_2, e_2]$ are *projections* of the HSP. Two HSPs *overlap* if either of their projections overlap.

- *shortChain* is a sequence of HSPs, where two HSPs are within a distance threshold T_{short}. More formally, shortChain=HSP$<b_1^i, e_1^i, b_2^i, e_2^i>, i = 1, 2, ..., k$ such that $0 \leq b_a^{i+1} - e_a^i \leq T_{\text{short}}, a = 1, 2, i = 2, 3, ..., k$. A chain is represented by chain$<b_1, e_1, b_2, e_2>$ where b_1 and b_2 are the starting positions of the first HSP, and e_1 and e_2 are the ending positions of the last HSP. The regions $[b_1, e_1]$ and $[b_2, e_2]$ are *projections* of this chain. Two chains *overlap* if either of their projections overlap.

- *longChain* is a sequence of shortChains, where the distance between two shortChains is within a threshold $T_{\text{long}} > T_{\text{short}}$.

Given similarity lines, our goal is to identify regions of each duplicate unit, and establish homologous groups. In Fig. 6.2(1), the output homologous groups include $HG_1(<b_1, e_1>, <b_2, e_2>, <b_3, e_3>)$ and $HG_2(<b_4, e_4>, <b_5, e_5>)$. Ideally, whenever two HSPs h_1 and h_2 overlap, one projection of h_1 is also a projection of h_2. In other words, the boundaries of the HSPs would be consistent. In this situation, we would form a graph; regions would form nodes, HSPs would form edges, and the connected components would form the homologous groups. Such perfect consistency is very rare, however. Usually HSPs' boundaries differ because of evolutionary events such as sequence divergence and genome shuffling. As many sources [5, 37] point out, simply constructing connected components does not achieve our objective. The completeness and Purity postulates may be contradictory unless we carefully determine the correct boundaries. Incorrect boundaries cause two kinds of problems: joining unrelated groups when non-homologous sequences are glued together, and cascades of splits, resulting in tiny useless fragments (details in section Methods).

Though the boundaries are ambiguous, we need to clearly define a *duplicate unit* (DU) to process. We did not find an objective definition that determines the precise limits for a DU; instead we formulate four rules that a consistent set of DU boundaries and homologous groups should satisfy:

1. **Acyclicity** (Acyclicity) A DU shall not contain significant repetitions. A significant repetition is a pair of chains with substantial overlap. For example in Fig. 6.2 (1), region $[b_1, e_2]$ contains significant repetition; regions $[b_1, e_1]$, $[b_2, e_2]$ do not.

2. **Separation of accidental neighbors** (Accidental Neighbor) Different parts of a DU should not align to DUs of different groups. If two adjacent regions forming a DU would mix two groups, they are accidental neighbors. In Fig. 6.2(1), region $[b_3, e_4]$ contains accidental neighbors: subregion $[b_3, e_3]$ belongs to HG_1, while subregion $[b_4, e_4]$ belongs to HG_2.

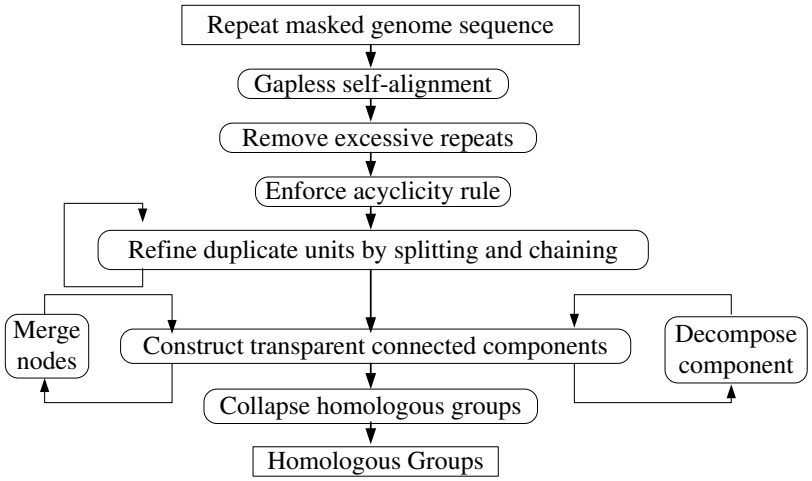


Figure 6.3. Program flowchart of HomologMiner.

3. **Coalescing persistent neighbors** (Persistent Neighbor) If most DUs of a group are consistently followed immediately by most DUs of another group within a distance threshold, we coalesce such pairs of DUs. Each member of HG_1 in Fig. 6.2 (1) could be divided into two regions by gaps. They can form two homologous groups (r_1, r_3, r_5) and (r_2, r_4, r_6). There are 3 pairs of persistent neighbors: r_1 and r_2, r_3 and r_4, r_5 and r_6, and they shall be coalesced to form HG_1.

4. **Elimination of excessive repetitions** (Anti-Excess) A certain number of consecutive DUs are deemed to be tandem repeats; all their similarity lines are removed.

Accidental Neighbor and Persistent Neighbor may conflict. For example, there is no clear parameter to define adjacency. If it is too large, we may include several unrelated regions in the same unit and violate Accidental Neighbor; if it is too small, we may end up with a partial gene in a unit and violate Persistent Neighbor. We solve this heuristically. Anti-Excess eliminates non-functional tandem repeats, and vastly decreases the running time. The screened out sequences can be separately tested for novel repeats.

6.3 Methods

We have described six rules: two for *HG* (Purity and Completeness) and four for DUs. Of the latter, we take care of Anti-Excess and Acyclicity in the preliminary stage. Accidental Neighbor and Persistent neighbor are not achieved in specific stages, but are assured by iterating until *HG*s and DU boundaries stabilize (Fig. 6.3).

6.3.1 Preliminary demarcation of duplicate units

6.3.1.1 Gap-free alignments

Sequences are first masked by RepeatMasker to remove low complexity repeats and annotated interspersed repeats. We then compute all-to-all similarities. Speed is vastly improved by using gap-free alignments, avoiding dynamic programming. We use the initial stage of BLASTZ [58] to identify HSPs with substitution score >3000 (roughly equivalent to 70% nucleotide identity). Only HSPs with length larger than 50 are considered in this paper.

6.3.1.2 Removing excessively repetitive regions

Tandem repeats locate in the form of arrays, while interspersed repeats disperse in a genome. Tandem repeats are usually short and highly repetitive, with a few or hundreds of nucleotides per copy, and hundreds or even millions of copies. They are characterized by multiple contiguous copies. Interspersed repeats are usually longer with fewer copies. Some repeats of the same family are almost intact, but some may be less identical; they may be contiguous, and also can be degraded such that two copies are separated by a diverged region; their end points may be inconsistent, and they may also have indels. [19] provides more detail.

Repeats not masked by RepeatMasker hinder the subsequent procedures. In particular tandem repeats can be mistaken for tandem duplicates. We separate tandem repeats and duplicates based on their dot-plot properties. We tolerate interspersed repeats and identify them as *HG*s. The dot-plot of some tandem repeat may look similar to tandem duplications (Fig. 6.1), so we do not want to be "overzealous" in identifying tandem repeats. A tandem gene cluster tends to have diverged intergenic regions that create longer gaps between two copies, thus we identify tandem repeats by using only the fact that they are usually packed contiguously, or separated by short regions due to sequence decay. Our initial DUs are formed from projections of HSPs that are connected together if gaps are smaller than a threshold g (300). We consider another parameter that may diagnose a tandem repeat cluster: c for the maximum number of crosses (4 by default) a horizontal sweep line creates in this preliminary DU (Fig. 6.2(2)). This c is the number of consecutive DUs described in the Anti-Excess rule. It is possible that functional elements and repeats are intermixed. Because we only remove lines, functional duplicates are not removed even if they are contained in the same preliminary DUs as repeats (Fig. 6.2 (3)). The effectiveness of applying Anti-Excess in this step strongly depends on species (Table 6.1).

6.3.1.3 Enforcing Acyclicity

Since the similarity lines do not contain gaps, some lines are separated by small indels. We do *first stage chaining* before enforcing acyclicity: connecting HSPs to form shortChains ($T_{\text{short}} = 500$). A preliminary DU may still contain significant repetitions, so we need to split it to uphold Acyclicity. The cut point has to be chosen carefully, because a wrong placement may lead to a cascade of unwanted cuts (Fig. 6.4). We choose the cut point using an optimization

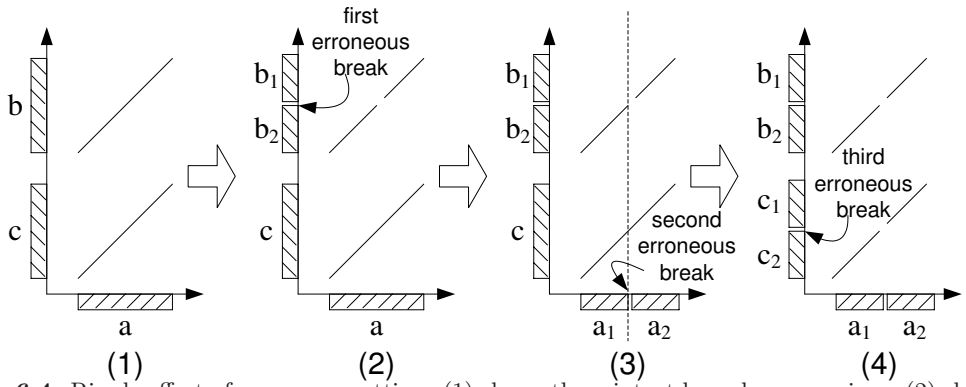


Figure 6.4. Ripple effect of erroneous cutting. (1) shows three intact homologous regions; (2) shows an erroneous cut resulting from other regions; (3) and (4) show subsequent cuts which are not necessary.

described below. A region may contain multiple copies, so this process is carried out recursively until there is no significant repetition in each divided region. Because we attempt to make a duplicate unit consistent with its functional unit, e.g. we do not want a gene be split into many pieces because of introns, we do *second stage chaining* to capture larger gaps: connect shortChains to form longChains ($T_{\text{long}} = 3000$). We re-join adjacent DUs resulting from the splitting procedure if their shortChains can be further chained in this stage and do not violate Acyclicity.

6.3.1.4 The piecewise quadratic optimization

We need to cut a region violating Acyclicity. Because some regions in a genome have chaotic similarity lines, a simple strategy may lead to the wrong choice of cutting points. To reduce the possibility of choosing a bad cutting point, we use a more sophisticated method (Fig.6.5). Denote lines as $(b_1, e_1), ..., (b_k, e_k)$, where b_i and e_i are the starting and ending positions of the ith line on x-axis, and $b_i \leq b_{i+1}$. If there exists i, such that $e_i < b_{i+1}$, then there is a gap, and we can split this region at the gap directly. Otherwise, we optimize as follows. Given a window $[l, r]$, we have a penalty for cutting line (b, e) at $x \in [l, r]$:

$$\text{PEN}_{b,e}(x) = \begin{cases} (x-e)^2 & \text{if } b < l < e < r \text{ and } l \leq x \leq e \\ (x-b)^2 & \text{if } l < b < r < e \text{ and } b \leq x \leq r \\ 0 & \text{otherwise} \end{cases}$$

and we want to choose $x \in [l, r]$ to minimize the objective function: $\sum_i \text{PEN}_{b_i,e_i}(x)$. The ending points of lines such that $b<l<e<r$ and the starting points of lines such that $l<b<r<e$ are *change points*, because the above formula changes at every such point. We look for the minimum in each interval between them. The window is chosen so that there must exist i and j, such that $b_i<l<e_i$ and $b_j<r<e_j$. The search window is determined by the pair of chains that overlap most on the y-axis (Fig. 6.5). When such penalty summation is 0, it indicates that there must exist an interval $[l', r']$, $\forall i$ $\text{PEN}_{b_i,e_i}(x) = 0$ for $x \in [l', r']$. In this case, we simply choose either l', r', or $(\sum_i e_i + \sum_j b_j)/N$, where $b_i < l < e_i < l'$, $r' < b_j < r < e_j$, and N is the number of such e_i's and

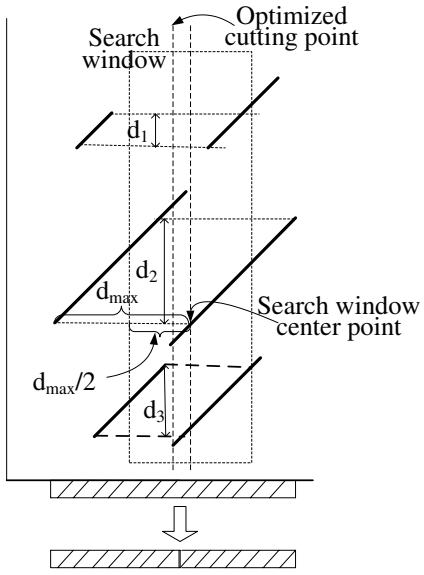


Figure 6.5. Cutting a region with two copies. Lines are chains. Between a pair of overlapped similarity lines, there is an overlapped distance (e.g. d_1, d_2 and d_3). One pair has the maximum overlapped distance (d_2 in this figure). The distance between this pair is d_{max}. A search window is defined to be $\pm\frac{1}{2}d_{max}$ centering at the search window center point shown in the figure.

b_j's.

6.3.2 Iterative refinement of duplicate units

6.3.2.1 The drifting effect and bad cut effect

[37] described a drifting problem in clustering protein sequences with multiple domains. A similar problem exists in clustering genomic sequences because of adjacent regions [5]. These regions may code adjacent domains of a protein, or they may be other functional regions that are accidentally adjacent (Fig. 6.6(1)). The problem is solved by cutting b into two pieces b_1 and b_2. But more generally, because of alignment drifts, it is possible to have regions $r_1, r_2, ..., r_k$ such that r_i is highly similar to r_{i+1}, but with r_1 totally unrelated to r_k (Fig. 6.6(2)). It is not obvious how to choose the cutting point in this case. If the cut point is not chosen properly, it causes a cascading effect and leads to further unnecessary cuttings (Fig. 6.4). The cutting point is determined similarly to section 6.3.1.4, except that the search window is determined by the middle points of two chains that do not overlap (Fig. 6.7 (1)). Besides carefully choosing the cutting point, we also employ an iterative approach to make further cuts and to coalesce results of erroneous cuts.

6.3.2.2 Iteratively enforcing Accidental Neighbor and Persistent Neighbor

We need to split a preliminary DU violating Accidental Neighbor. A DU is associated with many alignments with other DUs. After splitting a region, we must update its alignments. Hence, once we cut a region and update its connections to other regions, more regions may be affected. It is impossible to maintain all such relationships at once. Similarly to our handling of Acyclicity, we need to carefully decide to cut, choose the cutting point, and recover erroneous

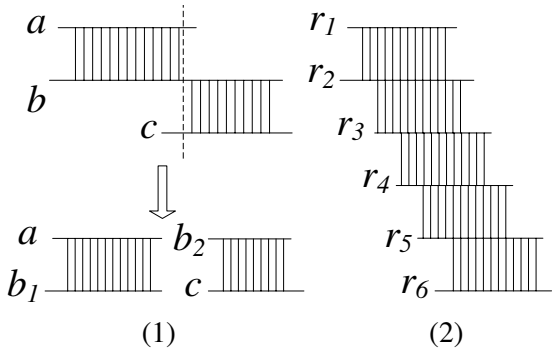


Figure 6.6. Drifting effects. (1) a and c are aligned to different regions of b. They will be mistakenly grouped together by single linkage. (2) Accumulated drift: the cutting point is more difficult to choose in the general case.

cuts. Accidental Neighbor guides the decision to cut; Persistent Neighbor guides rejoining of adjacent regions to correct erroneous cuts. We split a region if the majority of chains do not overlap (Fig.6.7 (1)). We observe that if a region has been mistakenly broken into two, shortChains from the first region and shortChains from the next region should form longChains without violating Acyclicity. So we re-join two adjacent regions if the majority of chains can be further chained allowing bigger gaps (Fig. 6.7 (2)). We continue splitting as long as Accidental Neighbor can be applied, then apply Persistent Neighbor exhaustively. We repeat such phases until the number of DUs becomes stable or an iteration limit (currently 15) is met.

6.3.3 Construction of homologous groups

Our goal is to build *HG*s from DUs that we have computed. We define a graph with DUs as nodes. Node u is connected to v if u and v contain projections of the same chain. Because u and v may be the same node for a certain chain, some nodes may not have edges, and they become singletons. Other connected components identify diverged homologs, but they may also join unrelated groups because of the drifting effect. We describe the following heuristics to prevent such effect, while taking advantage of the connected component approach. To enforce Accidental Neighbor and Persistent Neighbor, we modify both graph and DUs.

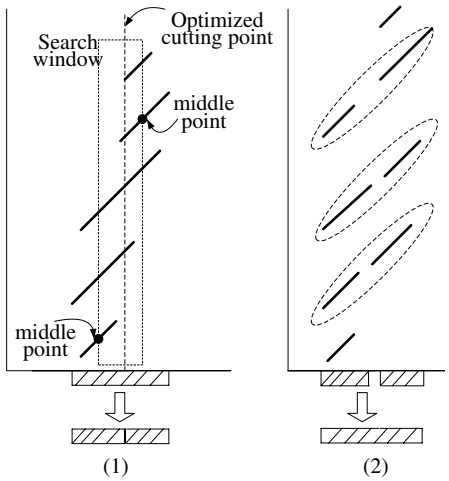


Figure 6.7. Splitting accidental neighbors and coalescing persistent neighbors. Each diagonal line is a chain with small gaps ($\leq 500 bases$). (1) The search window is determined by a pair of non-overlap chains. (2) The second stage chaining to reunion adjacent regions.

6.3.3.1 Construction of transparent connected components

Our drift preventing heuristic uses the notion of a *transparent path*. If two connected nodes n_i and n_j are not directly connected, then either they are so diverged that an aligner (e.g. BLASTZ) fails to identify their similarity, and they indeed should be in the same component; or they are unrelated at all, being grouped together by mistake. In either case, n_i and n_j are connected by a path, and this path provide clues to decide if n_i and n_j are related. We define the following:

- An *alignment* is a chain of HSPs.
- *alignment transitivity*: If position a aligns to position b, and position b aligns to position c, then a aligns to c.
- A path n_i,...,n_j is *transparent* if there exists a sub-region s_i on n_i and a sub-region s_j on n_j such that s_i and s_j can be aligned via other nodes on this path by alignment transitivity.

n_i and n_j are homologous if there is a transparent path between them; otherwise, they were placed in the same component by the drifting effect. When we construct connected components with breadth-first iteration, we add a node to a component only if it has a transparent path from the root. The resulting component is a *transparent connected component* (*tcc*).

6.3.3.2 Iteratively merging nodes and decomposing components

Two nodes are *genome neighbors* if they locate within a certain distance (3000 bases) without nodes in between. Two nodes are *graph neighbors* if they are directly connected. From the last section, two nodes are mistakenly placed in a group if there is no transparent path between them. For two such nodes n_i and n_j which are genome neighbors, it suggests two mistakes: (i) there is a node containing accidental neighbors on the path, and we should decompose the component; (ii) n_i and n_j are persistent neighbors, they have been mistakenly split, and we should merge them. Let N_i and N_j denote the sets of graph neighbors for n_i and n_j. In case (i), $|N_i \bigcap N_j|$ is small relative to $|N_i|$ and $|N_j|$. In case (ii), $|N_i \bigcap N_j|$ is relatively large, or some of their members are genome neighbors where the two nodes in each neighbor pair are from N_i and N_j. Let N' denote the number of such pairs of genome neighbors and N'' denote $|N_i \bigcap N_j|$. *graph neighbor agreement* (*gna*) is defined to be $\max((N'+N'')/|N_i|, (N'+N'')/|N_j|)$. If *gna* is low ($\leq 20\%$), a min-cut is determined by the Ford-Fulkerson algorithm for max flow [17] and the cut edges are removed to decompose the component. Otherwise we merge n_i and n_j. This process is repeated until no change is made, or an iteration limit is met (currently 10).

6.3.3.3 Collapsing nodes

Two groups are adjacent if nodes of one group are all adjacent to all nodes of another group. If two adjacent groups have the same number of nodes, and the longest distance is short, then the groups are collapsed: the boundaries of new nodes are combined from the pair of adjacent nodes. The distance threshold (3000 bases) is arbitrary. If it is too small, no groups will be fused. If it is too large, it might fuse different genes into the same region.

6.4 Results

6.4.1 Summary of homologous groups of 3 mammalian genomes

The choice of genomes in this paper is to a degree arbitrary, because HomologMiner is designed to be applicable to any genome. The parameters were adjusted to give efficient performance for the human genome, and then tested on two other genomes. The availability of a new set of repeats for the macaque genome made it possible for us to test the effectiveness of HomologMiner in identifying interspersed repeats (section 6.4.3). We also applied HomologMiner to mouse which is a more diverged mammal and may have different repetitive structures. The results of these species are summarized in Table 6.1. We provide preliminary analysis of overall characteristics of homologous groups. Some interesting large groups are summarized in Table 6.2. It can be seen that mouse does have more larger repetitive families (# groups ≥ 50 members in Table 6.1) which might be gene families or interspersed repeats. The olfactory gene family is an example (Tables 6.2 and **??**). We also show several applications based on the homologous groups found.

Table 6.1. Summary of output. (* non-singleton groups)

species	human	macaque	mouse
# HSPs from BLASTZ	33.4M	11.3M	60.6M
# bases covered	170.5Mb	104.8Mb	180.9Mb
# reduced HSPs	8.8M	8.9M	52.3M
# bases covered	152.5Mb	102.8Mb	168.7Mb
# non-singleton groups	27,471	22,043	24,538
ave. size of groups *	3.51	3.46	5.69
ave. length of final DUs *	1232	869	880
ave. transparency (%) *	98.3	97.5	92.6
# groups ≥ 50 members	91	81	223
# groups ≥ 10 members	1085	757	1652
# groups uniformly dist.	6065	6553	5970

6.4.2 Consistency test

The transparent path defined in Section 6.3.3.1 provides a estimate of whether two members in the same group are homologous or not. Recall that when we construct a *tcc*, we assure transparency of all nodes with the root, but the existence of a transparent path is not transitive (Fig. 6.6). Therefore we test *tcc*s for pairwise transparency. In particular, path transparency is evaluated for every pair of nodes in a *tcc*. Suppose there are N nodes in a group, there are $C = N(N-1)/2$ pairs of connected nodes. Denote C' as the number of pairs of nodes which have transparent paths, the ratio C'/C gives an estimate of the proportion of true homologs. On average the transparency is high; in other words, the average specificity is high. The average and the ratios for some interesting large groups are listed in Tables 6.1 and 6.2. Note that a transparency score below 100% also implies that different choices of root nodes may lead to different *tcc*s; the transparency score indicates the proportion of nodes that are stable when different root nodes

Table 6.2. Representative large homologous groups and their attributes.

	# copies	transparency(%)	ave. length	attributes
human	825	99	1393	olfactory receptor gene family
	353	93	495	significant for known genes
	301	88	1123	zinc finger family
	207	91	1626	keratin family
	199	99	921	zinc finger family
	161	99	756	non-coding, uniform dst
	136	97	1544	zinc finger family
	109	93	947	significant for known genes
macaque	621	99	1067	olfactory receptor gene family
	490	95	672	zinc finger gene family
	308	95	503	non-coding, uniform dst
	128	98	673	zinc finger gene family
	124	96	1564	newly identified LTR
	92	99	400	non-coding, uniform dst
mouse	1369	99	1622	olfactory receptor gene family
	334	93	390	non coding and uniform dst
	254	98	813	non coding and uniform dst

are chosen.

6.4.3 An interspersed repeat family

We obtained a new library of interspersed repeats for macaque, including types of ALU, L1, and LTR. Among them, ALU and L1 related new repeats are very similar to the ones already annotated, so we did not find significant homologous groups associated with them. However, we identified a group homologous with LTRs. It has 124 DUs, with transparency score of 96%, average length of 1,564 bases and uniform distribution test p-value 0.07, most of whose members align to several new LTR repeats. After running RepeatMasker without the new library using the default search, there are 50 LTR/ERVK repeats matched, with 6,512 bases masked. After running RepeatMasker with the new library using the quickest search, there are 127 LTR/ERVK repeats matched, with 190,186 bases masked. This shows that HomologMiner is useful in looking for new interspersed repeats.

6.4.4 Gene clusters in genomes

Many homologous genes locate close to each other; we call them tandem gene clusters. We looked for boundaries of tandem gene clusters for human, using the locations of duplicate units. We obtained 512 clusters, with average length of 113K bases, and minimum distance of 50K bases between two clusters. Each identified cluster does not necessarily correspond to a gene family, nor does it necessarily contain only one type of gene. But the region of a cluster is enriched with duplications, and a multiple sequence aligner assuming one-to-one orthologous relationship does

not perform well in such a region [8]. Identifying these regions will allow different alignment methods to be applied to them.

6.4.5 Running time and memory consumption

After the initial phase in which we obtain HSPs, HomologMiner is very efficient. Memory consumption is linear to the number of HSPs, while running time grows slowly. It took around 1, 1, and 10 CPU hours on a single processor computer (64-bit 3GHz) to process human, macaque and mouse genomes using around 3, 1, and 6 GB of memory respectively. The different behavior for mouse is due to a much larger number of HSPs, possibly because of larger gene families, and more un-annotated interspersed repeats.

6.5 Discussion

We used the first 10Mb of sequence of chr11 from hg17 to compare performance of HomologMiner with three other repeat identification tools: RECON, RepeatScout and RepeatGluer. There are 122 OR genes or pseudogenes with average length of 937 bases in this region from HORDE database [46], together with a beta globin gene cluster of 6 copies. Among the 4 tools, RepeatScout differs from the others since it works from sequences directly, while the other three rely on alignments produced by independent pairwise alignment tools. This program found both gene families in the sense that it found one consensus sequence for each of them. While it is easy to find the original sequences that match the consensus, extra post-processing would still be required to construct the whole families properly. To be consistent in comparing the other three tools, we use the same pairwise alignment produced by the initial stage of BLASTZ. The biggest family produced by RECON has 166 members with average length of 538 bases, 164 of which intersect with 99 olfactory annotations. RepeatGluer fragmented the olfactory genes into 8 parts, with 80 bases and 103 copies on average. It also provided a consensus sequence for each part. HomologMiner produced a homologous group of 121 members with average length of 1005 bases intersecting 121 annotations. HomologMiner also correctly identified the beta globin cluster as a single group, while both RECON and RepeatGluer fragmented the beta globin genes into many parts. We suspect this might be because of the nature of HSPs being short and not covering the whole genes, while HomologMiner is designed to chain short segments properly. This preliminary comparison shows that different tools are designed for different purposes. HomologMiner is efficient in building homologous groups with high accuracy while using only low quality pairwise alignments. RepeatScout and RepeatGluer perform well in obtaining consensus sequences.

For the human genome, the gap-free alignment procedure produces HSPs covering 171Mb, ∼5.7% of the genome (Table 6.1). Applying Anti-Excess produces HSPs covering 153Mb, ∼5.1% of the genome. Though the total covered length is not reduced much (5.7%⇒5.1%), the number of HSPs is reduced dramatically (33.4⇒8.8 million). This indicates that the removed regions are a relatively small portion of the genome, but largely redundant copies. These re-

peats might be high-complexity tandem repeats or degraded tandem repeats. Perhaps further investigation may be necessary.

The identified HGs from the three genomes include gene families, interspersed repeats and segmental duplications. We plan to categorize each of the above types. If more newly identified interspersed repeats (e.g. for macaque) become available, we can test whether our uniform distribution testing is differentiating interspersed repeat families from the rest of the groups. We can also apply spectral clustering techniques to classify subfamilies of HGs. When applied to a gene family, they provide better functional inference. Using the extra input of pairwise alignment of two species, we can build up the HGs between two species. The complete set of paralogs provide a good starting point to do orthology assignment at the level of whole genomes.

From [3], it seems there does not exist any clean algorithmic approach for repeat classification. We observe that this is especially true for large sequence input. For example, our methods in section 6.3.2 suffice to construct HGs when processing a single chromosome of human, while applying 6.3.2 alone to the whole genome does not produce meaningful HGs of large size. We've developed a set of procedures to work well with primates and a rodent. As heuristic software, HomologMiner has a set of parameters, adjusted to work efficiently and effectively with the human genome. It is difficult to accurately describe the effect of different combinations of parameters. The effect of parameters on different genomes needs further study. We expect more work to be done including parameter adjustment, and even designing new procedures when working with genomes with more complex repetitive structures. For example, the parameter d in Fig. 6.2 which is the distance between two adjacent diagonal lines is not used in HomologMiner, because the other two parameters are effective enough so far. We may need to incorporate d to identify tandem repeats in the future to improve performance in genomes like mouse. As another possible improvement, we implemented a minCut algorithm (modified from [63]), and applied it to the group that contains most of the olfactory receptor regions. We observed that minCut tends to find small adjacent regions which are wrongly grouped, and minimum weight minCut tends to find weak copies of pseudogenes. Before including it in HomologMiner, we need better annotations for other gene families to test it.

Chapter 7

Summary and discussion

7.1 Summary

We have described several sets of new methods to process genomic sequences that contain duplications, especially gene clusters. One general problem that we attacked was to produce *completely orthologous alignments* of several sequences, i.e., such that any two rows of an alignment block are orthologous to each other, which is crucial in many comparative genomic analyses. If this is regarded as identifying the best homology between different species, another problem that we addressed was to find the furthest homologies within a single genome, by establishing the complete set of homologs, the methodology of which can be easily extended to identify remote homology between different genomes.

We manually determined the human-mouse orthology relationships of genes from 12 gene clusters, and all gene orthologies in the beta-like globin gene clusters of 19 species. These datasets were then used as benchmarks for our orthologous alignment tools. The two-way orthologous selection tool (TOAST) was designed to produce orthologous pairwise alignments that capture one-to-one, one-to-many, many-to-one and many-to-many orthology relationships. Two multi-sequence orthologous alignment tools were developed. One is reference dependent (ROAST), and one is reference independent (BOAST).

To create reference-dependent multiple alignments, the orthologous pairwise alignments are first projected onto a reference sequence such that any position in the reference sequence is aligned to at most one position in each other species. A traditional progressive aligner then merges these pairwise alignments using the reference to form the multiple alignment. Only for a certain shape of guide tree and an appropriately chosen reference does this approach produce completely orthologous alignments. Extra processes are necessary to handle a species tree of an arbitrary shape to create completely orthologous alignments. A human-referenced genome-wide multiple alignment of gene-cluster regions was obtained by this approach.

It is desired by many biologists that the multiple alignment be reference independent. How-

ever, there is then a serious problem of representation of the alignment, since the alignment size may become very large when the aligned regions contain many-to-many orthology relationships. We designed a method to reduce the alignment size while capturing all pairwise orthology relationships. However, after this alignment is threading projected onto a reference, the resulting alignments may be less attractive, e.g. different regions of a gene are aligned to different genes of another species. Further study is needed to develop methods that can solve the two problems at the same time.

HomologMiner was developed to establish the complete groups of homologs within the same genome, where each group contains all and only homologous sequences, using pairwise alignments that involve typical amounts of "noise". Computational difficulties when identifying comprehensive sets of homologs come from the unknown boundaries of each repetitive unit, inconsistency of alignments, lack of alignment transitivity, and the complicated repetitive structure of the genome itself. A mixture of algorithmic and heuristic procedures was designed to maintain a balance between the completeness and purity of each group. Such a group may be a gene family, or a family of interspersed repeats.

7.2 Discussion

As pointed out in Chapter 1, it is sometimes not possible to identify correct orthology relationships using phylogenetic analysis when there are only two species, because of gene loss (Fig. 1.3). The same reasoning also applies to orthologous alignments between two species, which implies that some of our putative orthologous pairwise alignments (produced by TOAST) may be wrong. In the reference independent approach, we reject some orthologous pairwise alignments when evidence shows they may be wrong, and we also add orthologous pairwise alignments that are ignored by TOAST when the evidence suggests them. However, in the reference dependent approach, all the orthologous pairwise alignments are assumed to be accurate, and there is no way to identify or correct any mistakes. The problem of correcting incorrect pairwise alignments based on information obtained while creating a reference-dependent multi-species alignment deserves further investigation.

It turned out to be difficult to systematically evaluate the orthologous alignment tools. Doing so required two efforts. A fair amount of time and effort was spent on building the benchmarks for orthologous alignments, because for most species there was not enough useful information about orthology relationships among genes in gene clusters, and it was difficult to tell whether our tools aligned correctly or not. We definitely need more such benchmarks. On the other hand, with these benchmarks, we need to systematically evaluate the performance of the aligners using different sets of parameters. This turned out to be a non-trivial issue because of the lack of a good objective function and the small size of the benchmarks. This is another area where more progress is needed.

We created the 17-way multiple alignment for 75 gene clusters. The alignments of 6 species – human, chimp, macaque, mouse, rat and dog – are completely orthologous. A more detailed

analysis of these alignments such as the number of in-paralogs in human, the number of large indels, or the divergence rates based on these alignments, will reveal important biological properties of these clusters. It will also be useful to investigate the differences between these alignments and the original UCSC 17-way alignments so as to measure the effectiveness of the orthologous alignment tools when applied genome-wide.

HomologMiner successfully constructed homologous groups for human, macaque and mouse, respectively. It thus found the weakest paralogy relationships within these genomes. The methodology of HomologMiner can be easily extended to include cross-species homology. It can then be used to look for the furthest homology between distantly related species. For each homologous group, some clustering techniques can be explored to reveal the substructures of the group, and thereby provide better biological insights about the homologs.

Bibliography

[1] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., Lipman, D. J., 1990. Basic local alignment search tool. *J. Mol. Biol.*, **215**:403-10.

[2] Bailey, J. A., Church, D. M., Ventura, M., Rocchi, M., Eichler, E. E., 2004. Analysis of Segmental Duplications and Genome Assembly in the Mouse. *Genome Res.*, **14**:789-801.

[3] Bao, Z., Eddy, S. R. 2002. Automated De Novo Identification of Repeat Sequence families in Sequenced Genomes. *Genome Res.*, **12**:1269-1276.

[4] Batzoglou, S. 2005. The many faces of sequence alignment. *Briefings in Bioinformatics*, **6**:6-22.

[5] Bejerano, G., Haussler, D., Blanchette, M., 2004. Into the heat of darkness: large-scale clustering of human non-coding DNA. *Bioinformatics*, **20**:i40-i48.

[6] Bentley, J., 1975. Multidimensional binary search trees used for associative searching. *Comm. of ACM*, **18**:509-517.

[7] Berman, P. 1978. Relationship between density and deterministic complexity of NP-complete languages. *Lecture Notes in Compute Science*, **62**:63-71.

[8] Blanchette, M., Kent, J. W., Riemer, C., Elnitski, L., Smit, A. F., Roskin, K. M., Baertsch, R., Rosenbloom, K., Clawson, H., Green, E. D. et al., 2004. Aligning multiple genomic sequences with the threaded blockset aligner. *Genome Res.*, **14**:708-715.

[9] Bray, N., and Pachter. L., 2004. MAVID: Constrained Ancestral Alignment of Multiple Sequences. *Genome Res.*, **14**:693-699.

[10] Brudno, M., Do, C., Cooper, G., Kim, M. F., Davydov, E., Green, E. D., Sidow, A. and Batzoglou, S., 2003. LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res.*, **13**:721-31.

[11] Cacceta, L., P. Erdos, E. T. Ordman and N. J. Pullman, 1985. On the difference between clique numbers of a graph. *Ars Combinatoria*, **19A**:97-106.

[12] Cameron, M., Bernstein, Y., Williams, H. E., 2006. Clustering Near-Identical Sequences for Fast Homology Search. *LNBI*, **3909**:175-189.

[13] Cannon, S.B. and Young, N.D., 2003. OrthoParaMap: distinguishing orthologs from paralogs by integrating comparative genome data and gene phylogenies. *BMC Bioinformatics*, **4**:35-49.

[14] Cavers, M., 2005. Clique partitions and coverings of graphs (Masters thesis, University of Waterloo).

[15] Claverie, Jean-Michel, 2005. Fewer genes, more noncoding RNA. *Science*, **309**:1529-1530.

[16] Cooper, G.M. et al., 2005. Distribution and intensity of constraint in mammalian genomic sequences. *Genome Research*, **15**:901-913.

[17] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C., 2001. Introduction to Algorithms, 2nd edition.

[18] Darling, A. C., Mau, B., Blattner, F. R., and Perna, N. T. Mauve, 2004. multiple alignment f conserved genomic sequence with rearrangements. *Genome Res.*, **14**:1394-1403.

[19] Edgar, R., Myers, E. W., 2005. PILER: identification and classification of genomic repeats. *Bioinformatics*, **21**:i152-i158.

[20] Enright, A. J., Dongen, S. V., Ouzounis, C. A., 2002. An efficient algorithm for large-scale detection of protein families. *Nucliec Acids Research*, **30**:1575-1584.

[21] Eulenstein, O., Mirkin, B., Vingron, M., 1998. Duplication-based measures of difference between gene and species trees. *Journal of Computational Biology*, **5**:135-148.

[22] Felsenstein, J., 2005. PHYLIP (Phylogeny Inference Package) version 3.6. Distributed by the author. Department of Genome Sciences, University of Washington, Seattle.

[23] Fitch, W.M., 1970. Distinguishing homologous from analogous proteins. *Syst. Zool*, **19**:99-113.

[24] Fitch, W.M., 2000. Homology, a personal view on some of the problems. *Trends Genet*, **16**:227-231.

[25] Fredslund J., 2006. PHY-FI: fast and easy online creation and manipulation of phylogeny color figures. *BMC Bioinformatics*, **7**:315-321.

[26] Gramm, J. et al., 2006. Data reduction, exact, and heuristic algorithms for clique cover. *ALENEX* , 86-94.

[27] Gregory, D. A., and N. J. Pullman, 1982. On a clique covering problem of Orlin. *Discrete Math.*, **41**:97-99.

[28] Haldane, J. B. S., 1932. The Causes of Evolution. Longmans and Green, London.

[29] Hardison, R., Miller, W., 1993. Use of long sequence alignments to sutdy the evolution and regulation of mammalian globin gene clusters. *Mol. Biol. Evol.*, **10**:73-102.

[30] Hastad, J., 1999. Clique is hard to approximate within $n\ 1-\epsilon$. *37th Annual IEEE Symposium on Foundations of Computer Science*, 627-636.

[31] M. Hou, P. Berman, L. Zhang, W. Miller, 2006. Controlling size when aligning multiple genomic sequences with duplications. *Lecture Notes in Computer Science*, **4175**:138-149.

[32] Hou, M., Berman, P., Hsu, C., Harris, R., 2007. HomologMiner: looking for homologous groups of whole genomes. *Bioinformatics*, **23**:917-925.

[33] Hughes, J. R., et al., 2005. Annotation of cis-regulatory elements by identification, subclassification, and functional assessment of multispecies conserved sequences. *Proc. Natl. Acad. Sci. USA*, **102**:9830-9835.

[34] International Human Genome Sequencing Consortium, 2001. Initial sequencing and analysis of the human genome. *Nature*, **409**:860-921.

[35] The ENCODE Project Consortium, 2004. The ENCODE (ENCyclopedia of DNA Elements) Project. *Science*, **306**:636-640.

[36] Kent, W.J., Sugnet, C.W., Furey, T.S., Roskin, K.M., Pringle, T.H., Zahler, A.M., Haussler, D., 2005. The human genome browser at UCSC. *Genome Res.*, **12**:996-1006.

[37] Kim, S., Lee, J., 2006. BAG: a graph theoretic sequence clustering algorithm. *Int. J. Data Mining and Bioinformatics*, **1**:178-200.

[38] Koonin, E.V., 2005. Orthologs, paralogs, and evolutionary genomics. *Annu. Rev. Genet*, **39**:309-338.

[39] Kou, L.T., et al., 1978. Covering edges by cliques with regard to keyword conflicts and intersection graphs. *Communications of the ACM*, **21(2)**:135-139.

[40] Li, L., Stoeckert, C. J., Jr and Roos, D. S., 2003. Identification of ortholog groups for eukaryotic genomes. *Genome Res.*, **13**:2178-2189.

[41] Lund C. and M. Yannakakis, 1994. On the hardness of approximation minimization problems. *J. Assoc. for Comput. Mach.*, **41**:961-981.

[42] Ma, J., Zhang L., Suh, B.B., Raney, B. J., Burhans, R. C., Kent, W. J., Blanchette, M., Haussler, D., Miller, W., 2006. Reconstructing contiguous regions of an ancestral genomes. *Genome Research*, **16**:1557-1565.

[43] E. H. Margulies *et al.*, Analyses of deep mammalian sequence alignments and constraint predictions for 1% of the human genome, accepted by *Genome Research*.

[44] Needleman, S. B., Wunsch, C. D., 1970. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *J. Mol. Biol.*, **48**:443-453.

[45] Ohno, S., 1970. Evolution by Gene Duplication. Springer-Verlag, Berlin.

[46] Olender, T., Feldmesser, E., Atarot, T., Eisenstein, M., Lancet, D., 2004. The olfactory receptor universe–from whole genome analysis to structure and evolution. *Genet Mol Res.*, **30;3(4)**:545-53.

[47] Orlin, J., 1977. Contentment in graph theory: covering graphs with cliques. *Indag. Math.*, **39**:406-424.

[48] I. Ovcharenko, G.G. Loots, B.M. Giardine, M. Hou, J. Ma, R.C. Hardison, L. Stubbs, W. Miller, 2005. Mulan: Multiple-sequence local alignment and visualization for studying function and evolution. *Genome Research*, **15**:184-194.

[49] Owen, R., 1849. On the Nature of Limbs. van Voorst, London.

[50] Page, R. D., Charleston, M. A., 1997. From gene to organismal phylogeny: reconciled trees and the gene tree/species tree problem. *Mol. Phylogenet. Evol.*, **7**:231-240.

[51] Park, J., Holm, L., Heger, A., Chothia, C., 2000. RSDB: representative protein sequence databases have high information content. *Bioinformatics*, **16**:458-464.

[52] Pevzner, P. A., Tang, H., Tesler, G., 2004. De Novo Repeat Classification and Fragment Assembly. *Genome Res.*, **14**:1786-1796.

[53] Price, A. L., Jones, N. C., Pevzner, P. A., 2005 De novo identification of repeat families in large genomes. *Bioinformatics*, **21**:i351-i358.

[54] Pullman, N. J., and A. Donald, 1981. Clique coverings of graphs - II: complements of cliques. *Utilitas Math.*, **19**:207-213.

[55] Pullman, N. J., 1984. Clique coverings of graphs IV: algorithms. *SIAM J. on Computing*, **13**:57-75.

[56] Raphael, B., Zhi, D., Tang, H., Pevzner, P., 2004. A novel method for multiple alignment of sequences with repeated and shuffled elements. *Genome Research*, **14**:2336-2346.

[57] Remm, M., Storm, C. E., and Sonnhammer, E. L., 2001. Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *J. Mol. Biol.*, **314**:1041-1052.

[58] Schwartz, S., Kent, W.J., Smit, A., Zhang, Z., Baertsch, R., Hardison, R.C., Haussler, D., and Miller, W., 2003. Human-mouse alignments with BLASTZ. *Genome Res.*, **13**:103-107.

[59] Siepel, A., et al., 2005. Evolutionarily conserve elements in vertebrate, insect, worm, and yeast genomes. *Genome Research*, **15**:1034-1050.

[60] Smit, AFA, Hubley, R. and Green, P., 1996-2004. RepeatMasker Open-3.0, http://www.repeatmasker.org.

[61] Smith, T. F., Waterman, M. S., 1981. Identification of common molecular subsequences. *J. Mol. Biol.*, **147**:195-197.

[62] Steinberg, M.H., Forget, B.G., Higgs, D.R., Nagel, R.L., editors, 2001. Disorders of Hemoglobin: Genetics, Pathophysiology, and Clinical Management. Cambridge University Press.

[63] Stoer, M. and Wagner, F., 1994. A simple min cut algorithm. *LNCS*, **855**:141-147.

[64] Tatusov, R. L., Koonin, E. V., Lipman, D. J., 1997. A genomic perspecitve on protein families. *Science*, **278**:631-637.

[65] Thompson, J. D., Higgins, D. G., Gibson, T. J., 1994. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignments through sequence weighting, position specific gap penalties and weight matrix choice. *Nucl. Acids Res.*, **22**:4673-4680.

[66] Wakefield, M. J., Maxwell, P., Huttley, G. A., 2005. Vestige: maximum likelihood phylogenetic footprinting. *BMC Bioinformatics*, **6**:130.

[67] Wang, L., Jiang, T., 1994. On the complexity of multiple sequence alignment. *J. Comput. Biol.*, **1**:337-348.

[68] Yuan Y. P., Eulenstein, O., Vingron, M. and Bork, P., 1998. Towards detection of orthologues in sequence database. *Bioinformatics*, **14**:285-289.

Vita

Minmei Hou

Ms. Minmei Hou received her B.S. degree in Microbiology from Fudan University, Shanghai, China, in 1999, and her M.S. degree in Computer Science from Syracuse University, NY, in 2002. In summer 2003, she joined the Ph.D. program in the Department of Computer Science and Engineering at the Pennsylvania State University. Since then she has been a research assistant working with Dr. Webb Miller at the Center for Comparative Genomics and Bioinformatics. After her Ph.D. study, she will work as an Assistant Professor in the Department of Computer Science at Northern Illinois University, DeKalb, IL.