Pairwise Comparison Between Genomic Sequences and Optical-maps

by

Bing Sun

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

September, 2006

Bud Mishra

Dedicated to my loving ones

Acknowledgements

This dissertation would not have been finished without the help and support from many people to whom I am greatly indebted.

First, I thank my advisor Bud Mishra, for his continuous support in my Ph.D program. Bud was always there to listen and to give advices. I learned from him how to ask questions and express my ideas. He showed me different ways to approach a research problem and the need of persistence to accomplish a goal. He introduced the CAPO project to me and helped me accomplish it well. I also would like to thank OpGen Inc. for providing experimental data to test the CAPO tool.

A special thanks goes to my co-advisor, Jacob Schwartz, who is responsible for involving me in the COMBAT project in the first place. Jack has been a friend and a great mentor. Without his encouragement and constant guidance, I could not have finished this dissertation. He was always there to meet and talk about my ideas. He took all the patience to correct my English writings and asked me good questions to help me think through my research problems, either philosophical or computational.

Besides my advisors, I would like to thank my thesis reader Prof. Mehryar Mohri for his comments and suggestions. During the course of this work at NYU (2002 $\sim$ 2006), I was supported by the Computer Science Dept. teach-

ing/research assistant scholarship and the Bob Berne research funds. I am also greatly indebted to many teachers in NYU in the past. Also thanks to all the folks at the NYU Bioinformatics Lab for interesting discussions and having fun to be with.

Last, but not least, I thank my family for educating me with aspects from both arts and sciences, for unconditional support and encouragement to pursue my interests, even when the interests went beyond boundaries of language, field and geography. Thanks to my husband Shubin Zhao, for listening to my complaints and frustrations about study and research, for sharing his experience of dissertation writing with me, and for believing in me.

Preface

Writing a dissertation about comparative analysis is a difficult endeavor. I'm glad to have completed it in time to graduate.

Bing Sun

New York, New York

Abstract

With the development and improvement of high throughput experimental technologies, massive amount of biological data including genomic sequences and optical-maps have been collected for various species. Comparative techniques play a central role in investigating the adaptive significance of organismal traits and revealing evolutionary relations among organisms by comparing these biological data. This dissertation presents two efficient comparative analysis tools used in comparative genomics and comparative optical-map study, respectively.

A complete genome sequence of an organism can be viewed as its ultimate genetic map, in the sense that the heritable information are encoded within the DNA and the order of nucleotides along chromosomes is known. Comparative genomics can be applied to find functional sites by comparing genetic maps. Comparing vertebrate genomes requires efficient cross-species sequence alignment programs. The first tool introduced in this thesis is COMBAT (Clean Ordered Mer-Based Alignment Tool), a new mer-based method which can search rapidly for highly similar translated genomic sequences using the stable-marriage algorithm (SM) as an alignment filter. In experiments COMBAT is applied to comparative analysis between yeast genomes, and between the human genome and the recently published bovine genome. The homologous blocks identified by COMBAT are comparable with the alignments produced

by BLASTP and BLASTZ.

When genetic maps are not available, other genomic maps, including optical-maps, can be constructed. An optical map is an ordered enumeration of the restriction sites along with the estimated lengths of the restriction fragments between consecutive restriction sites. CAPO (Comparative Analysis and Phylogeny with Optical-Maps), introduced as a second technique in this thesis, is a tool for inferring phylogeny based on pairwise optical map comparison and bipartite graph matching. CAPO combines the stable matching algorithm with either the Unweighted Pair Group Method with Arithmetic Averaging (UPGMA) or the Neighbor-Joining (NJ) method for constructing phylogenetic trees. This new algorithm is capable of constructing phylogenetic trees in logarithmic steps and performs well in practice. Using optical maps constructed in silico and in vivo, our work shows that both UPGMA-flavored trees and the NJ-flavored trees produced by CAPO share substantial overlapping tree topology and are biologically meaningful.

Contents

Dedication iii

Acknowledgements iv

Preface vi

Abstract vii

List of Figures xiii

List of Tables xv

1 Introduction 1

1.1 Comparative Genomics 2

1.1.1 Motivation: From Single Genomes to Comparative Genomics 2

1.1.2 Problem Statement 5

1.1.3 The Solution 6

1.1.4 Contributions 7

1.2 Optical-Map Comparison and Phylogeny Analysis 8

1.2.1 Introduction 8

1.2.2 Problem Statement 12
1.2.3 The solution 12
1.2.4 Contributions 13
1.3 Thesis Organization 13

2 Prior Work on Comparative Genomics 14
2.1 Whole Genome Pairwise Alignment Methods 14
2.2 Multiple Sequence Alignment Methods 24
2.2.1 Multiple Sequence Alignment Tools for Short Sequences . 25
2.2.2 Multiple Sequence Alignment Tools for Long Sequences . 28

3 Selection of Alignment Anchors 31
3.1 Filters for Selecting Alignment Anchors 31
3.1.1 Longest-increasing-subsequence Approach 32
3.1.2 Dynamic Programming Approach 34
3.1.3 Clustering Approach 35
3.2 Bipartite Graph Matching 35
3.2.1 The Stable Marriage Algorithm 36
3.2.2 Applying the Stable Marriage Algorithm as an Alignment Filter 37

4 The COMBAT Algorithm for Pairwise Genome Comparison 41
4.1 Build Clean Ordered Mer Libraries 43
4.1.1 Scheme I: Generate K-mers Tagged by Genomic Locations 43
4.1.2 Scheme II: Generate K-mers Tagged by Indices of 'J-intervals' 44

4.1.3 Scheme III: Generate 'gapped' Local Mers Containing Two Neighboring Mers . . . 45
4.2 Search For Common Mers . . . 47
4.3 Find A One-to-One Correspondence . . . 49
4.4 Optional Chaining Procedure . . . 49

5 Evaluation of Performance of COMBAT 51
5.1 Yeast Genome Comparison . . . 51
5.2 Human Assembly and Cow Contig Comparison . . . 57
5.3 Human Genome and Cow Assembly Comparison . . . 59
5.4 Implementation and Speed . . . 60
5.5 Error Estimation . . . 62

6 CAPO: Comparative Analysis and Phylogeny with Optical-Maps 65
6.1 Review of Evolutionary Analysis . . . 66
6.1.1 Unweighted Pair Group Method with Arithmetic Mean (UPGMA) . . . 68
6.1.2 Neighbor Joining (NJ) . . . 69
6.1.3 Fitch Margoliash (FM) . . . 70
6.1.4 Maximum Parsimony (MP) . . . 70
6.1.5 Maximum Likelihood (ML) . . . 71
6.2 The Statistical Method for Optical Map Comparison Used by OpGen . . . 72
6.3 Problem Formulation and the CAPO Methodology . . . 73
6.3.1 Heuristic Algorithm for Pairwise Optical Map Comparison 75
6.3.2 Stable Matching Algorithm for Inferring Phylogeny . . . 76

6.3.3 Correction of Sizing Errors 80
6.4 Phylogenetic Tree Comparison Measure 82
6.5 Material 83
6.5.1 Data set I 83
6.5.2 Data set II 85
6.6 CAPO Experiments and Discussion 85
6.6.1 Parameter Optimization 85
6.6.2 Phylogenetic Tree Evaluation 94
6.6.3 Impact of Single-Merge Mode and Multi-Merge Mode . . 94
6.6.4 Experiments with OpGen's Definition of Pairwise Map Similarity 95
6.6.5 Cluster Sizes 98
6.6.6 Implementation and Speed 98

7 Conclusions and Future Work **101**
7.1 The COMBAT Tool for Pairwise Genome Comparison 101
7.1.1 Summary 101
7.1.2 Future Work 102
7.2 The CAPO Tool for Comparative Analysis and Phylogeny with Optical-maps 103
7.2.1 Summary 103
7.2.2 Future Work 103

Appendix: The SOMA Tool for Aligning Optical Maps **105**

Bibliography **108**

List of Figures

3.1 Schematic view of the LIS technique 33

3.2 An example of a procedure for solving the stable marriage problem 38

4.1 Outline of the COMBAT algorithm 42

4.2 Build mer library using mer generation scheme II 44

4.3 Mer generation scheme III. 46

5.1 Histogram of longest continuous matches length in the alignments for the 2420 false negative cases of the comparison between *Saccharomyces cerevisiae* and *Candida glabrata*. 55

5.2 Alignment maps between *chr1* and *cow1* 58

5.3 (1)-(5): Alignment maps between *Hchr1* and *Cchr1*. $T = 5$ for all maps. (6): Specificity and sensitivity measurement of COMBAT from the experiments (1)-(5). 61

6.1 Procedure of selecting an appropriate method to infer phylogeny given single-gene sequences. 67

6.2 An example of building a bipartite graph given a distance matrix. A) A distance matrix M of four items (A, B, C, D). B) The corresponding bipartite graph. 78

6.3 First-degree polynomial fit for restriction fragment sizing error. (a) L vs. $stddev(L)$, $cc = 0.7428$; (b) $\sqrt{L}$ vs. $stddev(L)$, $cc = 0.7562$; (c) $1/\sqrt{L}$ vs. $stddev(L)/L$, $cc = 0.8290$. 81

6.4 View maps of Data set I using MapViewer. A pairwise alignment between *Escherichia coli O*157 : *H*7 *str. Sakai* and *Escherichia coli O*157 : *H*7 *EDL*933 is shown. 86

6.5 View maps in Data set II using MapViewer 88

6.6 Phylogenetic tree for data set I and II ($k = 2$, $\rho = 0.9$) 90

6.7 Phylogenetic tree for data set I and II ($k = 3$, $\rho = 0.8$) 91

6.8 Phylogenetic tree for data set I and II ($k = 4$, $\rho = 0.7$) 92

6.9 Phylogenetic trees constructed by CAPO for data set I and II using default setting and single merge mode. 96

6.10 Phylogenetic trees generated by OpGen for data set I 97

6.11 Phylogenetic trees generated by CAPO for data set I using OpGen's definition of pairwise map similarity 99

6.12 Number of clusters in the iterations of the experiments of data set I and II using CAPO SM-UPGMA/SM-NJ. 100

List of Tables

2.1 The score matrix used by BLASTZ program to evaluate nucleotide substitutions 22

4.1 Parameters involved in COMBAT 50

5.1 Sensitivity and specificity of COMBAT using scheme I assessed by BLASTP or BLASTP/BLASTN search results 53

5.2 Estimated percentages of reduced false negatives by using smaller values of K 56

5.3 Estimated reduced percentages of false negatives if all five yeast genomes are considered with different values of K used. 56

5.4 Estimated reduced percentages of false negatives by using mer generation scheme III. 57

5.5 Exemplary choices of parameters given G and s when $\epsilon = 1/G$. 64

6.1 Data Set I: 11 *Escherichia Coli* Strains 84

6.2 Data Set II: 28 *Enterobacteriaceae* Taxa 87

6.3 Experimented parameter settings 89

6.4 Tree comparison measure by the partition metric using different parameter settings 93

6.5 Tree comparison measure by the partition metric for the same data set using different distance methods 95

Chapter 1

Introduction

The development and improvement of high-throughput experimental technologies have allowed collection of massive amount of biological data for various species. These data include genomic sequences and genomic maps (such as optical, RH, linear/correspondence, genetic, physical (or contig) and HAPPY maps). Alone these data cannot tell us how the genetic information leads to the observable traits and behaviors (phenotypes) that we wish to understand, or how species are related through evolution. Comparing data becomes a central method in investigating the adaptive significance of organismal traits and revealing their evolutionary relations.

1.1 Comparative Genomics

1.1.1 Motivation: From Single Genomes to Comparative Genomics

As discussed earlier, DNA sequences for whole genomes are becoming available rapidly. As of today, many genomes of viruses and bacteria and some much larger genomes of higher organisms have been completely sequenced. These include several yeasts (Saccharomyces cerevisiae, Schizosaccharomyces pombe, Candida glabrata, Debaryomyces hansenii, Kluyveromyces lactis, Yarrowia lipolytica), a nematode (Caenorhabditis elegans), the dog (Canis familiaris), a fruit fly (Drosophila melanogaster), the zebrafish (Danio rerio), chicken (Gallus gallus), the human (Homo sapiens), the house mouse (Mus musculus), chimpanzee (Pan troglodytes), rat (Rattus norvegicus), etc. The available corpus of genomic sequences as well as other biological data including gene and protein expression data, protein-protein interaction data, protein 3-D structures, continue to grow extraordinarily, forming rich sources of challenging computer science problems addressing large, complex biological data sets and domain knowledge. As a field of developing computational techniques for management and analysis of biological data, bioinformatics research has become a driving force in computer sciences. Typical bioinformatics problems include determining the locations and functions of genes, identifying binding sites of proteins, inferring evolutionary history, etc. Numerous algorithms have been designed to address these problems. Early algorithms focused on using information from a single genome. Subsequent consolidated databases (Genbank, protein expression database, protein-protein interaction database, gene expression database,

NCBI Taxonomy database, Gene Ontology, CMR, etc.) partition the space of biological information in different dimensions and help users to manage bioinformatics data-types simultaneously. Many new research areas and countless methods have been developed using these databases. Recently comparative genomic methods have emerged, targeting the analysis of multiple genomes.

Gene prediction methods serve as a good example for the developing sophistication of bioinformatics methods. The goal of gene prediction is to identify protein-coding regions in raw genomic sequences. Another related natural problem focuses on exon finding. Over the last decade many gene prediction programs have been developed. Depending on the nature of the information exploited and the way that information is used, these programs fall into three rough categories as follows:

Intrinsic (ab initio) methods

These methods build hidden Markov models (HMMs) or other probabilistic models that exploit statistical features including exon length, intron length, ORF length, GC content, codon frequencies, presence of upstream CpG islands, locations of potential splice signals, etc to detect genes. Such models are trained on known genes from several closely related species to set up model parameters that reflect their statistical features. These models are then applied to scan genomes of interest to distinguish coding regions from non-coding regions. The most popular tools of this kind are GenScan [11] and HMMgene [32]. This approach is capable of finding genes that lack homologues in protein databases because some of the statistical features might apply for non-homologous genes from the same species. In such cases, however, the accuracy of the method is poor, because known gene measures are insufficient to distinguish true positives

from false ones. A limitation for all these intrinsic approaches is that the statistical models they rely on are derived from limited training sets comprising of known genes. It is difficult for them to predict genes that have unusual, undiscovered features.

Extrinsic (homology-based) methods

These methods identify functional elements by comparing a conceptual translation of nucleotide sequences to databases of known protein sequences ([23], [7]). This approach depends on utilizing homologies between translated genomic sequences and known protein databases. It can find genes that are supported by very strong similarities, but is not able to detect genes of which no homologue are known. Because it is difficult to collect mRNAs comprehensively, only limited number of genes can be detected this way and the observation of a protein match does not guarantee the correct identification of its exon/intron structure. Therefore, in contrast to the intrinsic methods, which have low specificity but high sensitivity, these approaches have high specificity but low sensitivity. Ensembl is a typical tool of this kind [8]. The gene lists compiled by Ensembl contain only a small number of false-positive genes but miss a large number of genes.

Comparative Genomics

These methods exploit the increasing number of completely or partially sequenced genomes to address the limitations of the two kinds of methods discussed above. Comparative genomics includes analysis and comparison of genomes from different species. These approaches depend heavily on the quality

of the underlying sequence alignments, but they do not use any training technique or depend on the existence of cognate known genes in current databases. These approaches exploit a simple biological principle: functional elements tend to be conserved more strongly during evolution than random (or non-functional) genomic sequences. Researchers have learned a great deal about the function of human genes by examining their counterparts in simpler organism models such as the mouse. Genome researchers look at many different features when comparing genomes: sequence similarity, gene location, the length and number of coding regions (called exons) within genes, the amount of noncoding DNA in each genome, and highly conserved regions maintained from simple organisms like bacteria to complex ones like human. One can also infer the path and the local evolution rates based on the degree of conservation of functional elements and underlying alignment deviations.

1.1.2 Problem Statement

Our goal is to develop an efficient and accurate algorithm for large scale pairwise comparative genomics. The complete genomic sequence of an organism consists of a sequence of four nucleotides: adenine (A), thymine (T), cytosine (C) and guanine (G). The length of a genomic sequence may be several MBs, e.g. yeast genomes, or several GBs, e.g. human and mouse genomes. Given two genome sequences x and y, we want to find an optimal global sequence alignment between them. A sequence alignment is a scheme for writing one sequence on top of another where the residues in one position are deemed to have a common evolutionary origin. x and y may have different lengths, so dashes must be used to mark insertions in the alignment. If the i-th residue x_i of sequence x occurs

in the same position p as the j-th residue y_j of sequence y in the alignment, we say x_i is aligned to y_j at p. There are three cases: 1) x_i and y_j are the same residues, indicating that this position p has been conserved in evolution; 2) x_i and y_j differ, indicating that the two residues derive from an ancestral residue (which could be one of the two or neither); 3) one residue may be paired up with dashes in the other sequence to signify an insertion or deletion that has appeared somewhere in evolution.

Whole genome comparison is most difficult for larger mammalian genomes because the homologous loci are scattered in a vast sea of non-genic "noise". In this case we want to find highly similar pairwise local alignments, which often represent conserved biological features. For regions of low similarity no alignment should be given.

1.1.3 The Solution

A variety of comparison algorithms and scoring parameters can be used to evaluate protein or DNA sequence similarity. Two general classes of comparison algorithms are used widely: global alignment algorithms and hash-based heuristic local alignment algorithms. This thesis describes a novel local alignment algorithm, COMBAT (Clean Ordered Mer-Based Alignment Tool). Suppose the two genomes in comparison are referred to as genome A and genome B. The basic COMBAT algorithm works as follows.

1. ***Build Clean Ordered Mer Libraries***

   Nucleotide sequences for the genomes of interest are translated into peptide sequences over all forward and reverse orientations. After choosing a mer-size K, local mers are generated using one of the three mer genera-

tion schemes described below. Mers which occur in repeats annotated by RepeatMasker are ignored. A position index is attached to each mer. Mer libraries for genome A and genome B are built and sorted lexicographically. Three mer generation schemes are discussed in Chapter 4.

2. ***Search For Common Mers***

   The two clean ordered mer libraries produced by step 1 are scanned to search for common mers between them. Peptide compositions which appear more than C times in any one of the two mer libraries are not considered when looking for common mers.

3. ***Find A One-to-One Correspondence***

   Next we apply the modified stable marriage algorithm to find the single best alignment. Details of the stable marriage algorithm are described in chapter 3.

4. ***Optional Chaining Procedure***

   Lastly, a chaining procedure is performed to further remove randomly matching local alignments. The chaining criterion requires that there must be at least F local alignments, each no farther than E intervals from each other. This step is not necessary if strict parameter values are chosen.

1.1.4 Contributions

COMBAT develops an efficient and accurate solution for comparative genomics. The main contributions of COMBAT come from the following points:

- Efficiency: COMBAT hashes genomes into mer libraries and uses mer indexing to search rapidly for highly similar homologous blocks.

- Clean design: COMBAT does not utilize any sophisticated scoring matrix, thus it has few parameters whose values need to be set up by statistical study.

- Fewer constraints: COMBAT makes no assumption of gene order and orientation, so it is capable of detecting translocated conserved elements.

- Novel usage of the stable marriage algorithm as alignment filter: the stable marriage algorithm is used innovatively to find the optimal one-to-one mapping from multimappings.

1.2 Optical-Map Comparison and Phylogeny Analysis

1.2.1 Introduction

An ordered restriction map is a physical map describing the position of restriction endonucleoase cleavage sites (restriction sites) within a DNA sequence. It exploits the fact that a restriction enzyme cuts DNA strands at nucleotide sequence specific locations (48 bp long) by breaking phosphodiester bonds, so that these locations can be used as physical markers. The fragments of DNA generated by cleaving at two consecutive restriction sites are called restriction fragments. Thus a DNA molecule has fixed cutting patterns for a specific restriction enzyme, assuming 100% cutting efficiency and no other experimental errors. In the past decade, single-molecule approaches have proven successful in constructing restriction maps — a primary example of such technologies being the Optical Mapping method originally developed by T.S. Anantharaman, B.

Mishra and D. Schwartz. Optical Mapping has been used in the past to prepare restriction maps of a number of clone types including bacterial artificial chromosomes [12], but also with genomic DNA. e.g., parasite chromosomes [30], and whole bacterial genomes [36].

The physicochemical approach underlying optical mapping is based on immobilizing long single DNA molecules on an open glass surface, digesting the molecules on the surface and visualizing the gaps created by restriction activities using fluorescence microscopy. The molecules retain biochemical accessibility and fragment order after enzymatic digestion. Thus the resulting image, in the absence of any errors, would produce an ordered sequence of restriction fragments, whose masses can be measured via relative fluorescence intensity and interpreted as fragment lengths in base pairs. The corrupting effects of many independent sources of errors affect the accuracy of optical maps created from one single DNA molecule, but can be tamed by combining the optical maps of many single molecules covering completely or partially the same genomic region and by exploiting accurate statistical models of the error sources. To a rough approximation, the density and accuracy of an optical map can be arbitrarily improved by simply increasing the number of enzymes and number of molecules used respectively [40].

Statistical models and Bayesian approaches have been used for accurate optical map construction considering fragment sizing error, and false positive and false negative cutting errors. The Bayesian approach [40] uses a model of the map of restriction sites (Hypothesis, H) and a conditional probability distribution function for the single molecule map data given the hypothesis (conditional *pdf*, $f(D|H)$). The conditional *pdf* models restriction fragment sizing errors in terms of a Gaussian distribution, missing restriction site events (due

to partial digestion) as a Bernoulli trial and the appearance of false restriction sites as a Poisson process. Using the Bayes formula, the posterior conditional $pdf,\ f(H|D) = f(D|H)f(H)/f(D)$, is computed and provides the means for searching for the best hypothetical model given the set of single molecule experimental data. Since the underlying hypothesis space is high dimensional and the distributions are multimodal, a naive computational search must be avoided. An efficient implementation involves approximating the modes of the posterior distribution of the parameters and accurate local search implemented using dynamic programming [4]. The correctness of the constructed map depends crucially on the choice of the experimental parameters (e.g. sizing error, digestion rate, number of molecules). Thus, the feasibility of the entire method can be ensured only by a proper experimental design [40].

Because Optical Mapping requires only small amounts of starting material and maps can be generated at relative high speed and low cost, it enables library construction without associated cloning artifacts and simplifies mapping of microorganisms that are problematic to culture. Optical maps can therefore be resources for numerous scientific study areas: i) facilitating ongoing sequencing efforts [5, 40]; ii) it may uncover new general mechanisms of DNA repair [36]; iii) it may illuminate the basic organization of an entire genome and the presence of extrachromosomal elements [36]; iv) it might identify aberrant DNA structures associated with mechanisms of DNA repair [36]; v) it may reveal polymorphisms due to single nucleotide differences (SNPs), small insertions and deletions (RFLPs), and copy-number variations (CNPs). So Optical Mapping can be useful in studies involving genomic instabilities in cancer, and genetics [5].

Given DNA sequences of various taxa, the standard technique in evolu-

tionary analysis is to first perform a multiple sequence alignment (on DNA sequences or protein sequences). From the resultant distance matrix a phylogenetic tree is built describing the relationship of the various taxa with respect to each other. These distance-based methods compress sequence information into a single number and the two sequences with shortest distance are considered as closely related taxa. However, the high cost of sequencing techniques and the biological diversity among the genomes, together makes it impossible to study phylogeny using detailed sequences of many strains of large-number of related species. The low cost and high speed of Optical Mapping technique provide an elegant solution to this dilemma, provided that one can devise an accurate tool to infer phylogeny from optical mapping data instead of sequence data. At OpGen Inc. of Madison, WI., clusters of optical-maps are generated using their SOMA map aligner and the statistical package *R*. Pairwise map similarity is defined as the percentage of aligned fragment lengths over the sum of all fragment lengths between two maps. These pairwise map similarity values are then fed into *R*, and one clustering method (nearest neighbor, furthest neighbor, or UPGMA) is selected to infer phylogeny. Generally, their unpublished results show that this method performs satisfactorily for small data sets (< 10) where there is adequate similarity variations. When the set sizes grow, this clustering method produces groupings that are no longer biologically meaningful.

To address this concern, we developed CAPO (Comparative Analysis and Phylogeny with Optical-Maps), which is a novel tool that combines the Stable Marriage (SM) algorithm and a distance-based method (either the UPGMA or the NJ method) to infer phylogeny among multiple strains or genomes.

1.2.2 Problem Statement

The problem studied here can be formulated as follows: an optical map can be viewed as an ordered sequence of restriction fragment lengths. Each genome to be studied can be represented as an optical map, H. The algorithmic problem we wish to study is to explore evolutionary relations, given optical maps of multiple species or strains. No full genomic sequences are assumed available.

1.2.3 The solution

The CAPO algorithm is based on pairwise optical map comparison and bipartite graph matching, combined with standard distance methods of phylogeny tree construction. We adapt many of the key ideas from the earlier published COMBAT algorithm [52] for pairwise genome comparison. CAPO consists of two major phases:

1. ***Pairwise optical map comparison***

   First, pairwise optical map comparison is performed to generate a pairwise similarity matrix $S = (s_{ij})$, where s_{ij} is the map similarity between the i-th and j-th map in the input matrix O.

2. ***Inferring phylogeny***

   The similarity matrix S thus constructed is used as input to the second phase of CAPO, which infers phylogeny among input strains or genomes. The stable marriage algorithm is combined with standard distance-based methods (UPGMA or NJ) to infer phylogeny. Output is produced in the Phylip format used by many phylogenetic analysis packages.

The detailed algorithms implemented in CAPO are explained in chapter 6.

1.2.4 Contributions

CAPO develops a fast solution for comparative analysis and phylogeny with optical-maps. The main contributions of CAPO comes from the following aspects:

- Efficiency: CAPO performs optical-map comparison and phylogeny study rapidly, so it can be used in analyzing large data sets.
- Flexibility: Depending on data features users have the option to produce either UPGMA-flavored or NJ-flavored phylogenetic trees and compare the resulting trees.
- Novel usage of the stable marriage algorithm in phylogenetic tree construction: The stable marriage algorithm, combined with a standard distance-based phylogenetic tree construction method, is capable of constructing a phylogenetic tree in $log(n)$ iterations in average cases.

1.3 Thesis Organization

This thesis is organized as follows. Chapter 1 gives an introduction to this thesis. Chapter 2 describes related work on comparative genomics. Chapter 3 focuses on approaches for selecting alignment anchors. Chapter 4 presents the COMBAT algorithm with three mer generation schemes. Chapter 5 shows experimental results obtained using COMBAT. Chapter 6 focuses on phylogeny, prior works on analyzing optical-maps, the CAPO algorithm and experimental results obtained using CAPO. Chapter 7 summarizes this thesis and discusses future applications and improvements.

Chapter 2

Prior Work on Comparative Genomics

Since 1999 there have been quite a few efforts to develop computer programs and algorithms for aligning whole genomes. These are reviewed in the following section.

2.1 Whole Genome Pairwise Alignment Methods

Given the genomic sequences of closely related species one of the first questions bio-scientists ask is how the genomes align. Standard dynamic programming alignment algorithms [Smith-Waterman,1970; Needleman-Wunsch,1981] are not suitable for aligning large genomes due to their quadratic running time and space, and their weak ability to find short regions of good alignments flanked by poor alignments. Accelerated heuristic alignment algorithms, such as BLAST

and FASTA, are also not satisfactory because they were not designed to perform large scale alignments and so only give lists of local alignments ranked by quality. It was not until 1999 that the first software capable of aligning whole genomes, MUMmer [13], was released. Other programs, including GLASS [6], AVID [9], DIALIGN [42], LAGAN [37], BLASTZ [49] and WABA [31], became available shortly thereafter.

There are two basic types of alignment programs, global and local. In global alignment every character in the query sequence is lined up with a character in the target sequence, so gap insertion may be required. These global alignment algorithms embody an assumption that the highly conserved regions in the two aligned sequences appear in the same order and orientation, which is possibly true for closely related organisms. Local alignment algorithms are designed to find optimal similarity between subsequences of the two sequences and do not make such an assumption. Therefore they are able to find conserved regions such as transcription factor binding sites, which are inverted or rearranged with respect to each other. But false positive alignments can result from simple sequence repeats and other sequence artifacts. If a subsequence match appears in a global alignment but not in a local alignment, additional information is needed to determine whether or not the match is significant.

Most programs filter genomic sequences before alignment, for instance, remove lineage-specific interspersed repeats from the sequences being aligned. Some programs mark those repeats and ignore them completely, others use these repeats in alignments if they improve the alignment results.

Anchoring alignments at regions of high sequence similarity can reduce the search space required for an alignment procedure [42], and so most of these global alignment programs use anchor-based algorithms. They work as follows:

first, local similarities between the two sequences of interest are detected and sorted based on length; longest set of anchors are then selected and fixed from the ordered local similarities; finally, interleaving regions are aligned. These programs vary in their underlying data structures, anchor selection algorithms, thresholds, scoring functions and the methods to fill in the gaps of the set of anchors.

These algorithms are reviewed below. For simplicity the two genomes to be compared/aligned are referred to as genome A and genome B.

- **MUMmer**: This program originally integrated generation of suffix trees, detection of longest increasing subsequence (LIS) and Smith-Waterman alignment into a coherent system for large genome alignment. Suffix trees are used to efficiently find all unique common subsequences shared by the two genomes. The algorithm locates single nucleotide polymorphisms, small mutated regions, large insertions, and repeats. The alignment process proceeds as the follows [13]:

  i) Construct a single suffix tree for the two genomes. The leaf nodes of the suffix tree are labeled by start positions of the suffix that they represent. Edges of the tree are labeled with subsequences of the genome such that by following a path from a leaf i to the root, and concatenating the sequences along those edges one produces the suffix starting at position i. Every unique matching sequence is then represented by an internal tree node with exactly two child nodes, whose child nodes are leaf nodes from different genomes. By clever use of sets of pointers a suffix tree can be built in linear time and linear space [38]. The newly released MUMmer 3.0 uses approximately 17 bytes for each base pair in the reference sequence in cre-

ating this data structure (see http://www.tigr.org/software/mummer/).

ii) Sort matches by their start locations in genome A from the suffix tree and extract the set of maximal unique matches (MUM) that occur in the same order in both genomes using a variation of the LIS (Longest-Increasing-Subsequence) algorithm [25]. This variation of the algorithm takes into account the lengths of the MUMs and allows them to overlap. It runs in $O(K \log K)$ time, where K is the number of MUMs.

iii) Generate Smith-Waterman alignments for all the regions between the MUMs.

iv) Output the alignment, including all the matches in the MUM alignment as well as the detailed alignments of regions that do not match exactly.

- **GLASS (Global Alignment System)**: This program uses a hashing technique and computes a global alignment recursively by finding long segments that match exactly and whose flanking regions have high similarity. More precisely it works as follows [6]:

  i) Find all common K-mers for a fixed value of K that appear in both genomes.

  ii) Use a hash technique that maps each matching K-mer to a unique character and convert both sequences of genomes into strings of such characters. The alphabet of these characters must be disjoint from that of the letters in the genomic sequences.

  iii) Apply the standard dynamic programming algorithm to the short flanking regions on both ends of each matching K-mer and compute two scores. Each K-mer receives a score equal to the sum of these two scores.

iv) Take only 'consistent' K-mers whose score exceeds a threshold T. Two K-mers are inconsistent if they correspond to positions that overlap by $i > 0$ bases in genome A but not in genome B. Fix the alignment in the underlying genomic sequences contained in these K-mers.

v) Recursively align the intervening regions using a smaller value of K. The value of K is often chosen from the sequence 20, 15, 12, 9, 8, 7, 6 and 5.

vi) Extend all pairs of aligned segments by short local alignments to the left and right by standard dynamic programming.

vii) Perform standard dynamic programming on the remaining unaligned regions to form the complete global alignment between two genomes.

- **DIALIGN (Diagonal Alignment)**: This program assigns a weight score to every pair of matching segments based on the degree of similarity and selects a consistent set of segments maximizing their total weights. The procedure is summarized below [42]:

  i) Construct collections of gap-free local pairwise alignments by processing a comparison matrix in a column-by-column fashion from left to right. A dynamic programming procedure is used to keep track of the optimal prefix alignments for each column.

  ii) For each matching segment pair f, a weight score is computed at the nucleotide and the peptide levels using the BLOSUM 62 substitution matrix [26] for both possible orientations; The final score of f is chosen to be the maximum of these values.

  iii) Intervals between those fragments are realigned based on the proba-

bility of random occurrence in sequences and the size of the respective intervals.

- **AVID**: This program uses both overlapping repeat matches and non-overlapping clean matches. It works as follows [9]:

  i) Find matches using suffix trees: concatenate two genomic sequences and place the character N between them; a maximal repeat in this string that crosses the boundary between the two sequences represents a maximal match between the two sequences.

  ii) Remove matches that are less than half the length of the longest match. Sort the matches by length with clean matches appearing first.

  iii) A variant of the Smith-Waterman algorithm is used to select anchors. Every match is evaluated based on its length and alignment scores of its two flanking regions (10 bp on each side). This is similar to the idea first employed in the GLASS algorithm. It requires that anchors selected are not overlapping.

  iv) Check each match to determine whether that match is entirely between two sets of anchors. Shorter matches removed in step ii) and repeat matches will be considered if there are not enough matches. Smaller inter-anchor regions are realigned using the anchor selection step recursively. For short such regions use the Needleman-Wunsch algorithm.

- **LAGAN (Limited Area Global Alignment of Nucleotides)**: This program is suitable for aligning distantly related organisms. In outline, this algorithm consists of three main steps [37]:

  i) Use the CHAOS algorithm to generate local alignments between the

two sequences. The CHAOS algorithm [10] is a highly sensitive method that detects local alignments using multiple short inexact words instead of the longer exact words used by MUMmer, GLASS and AVID. Given a maximum distance d and maximum shift s, two local alignments that are x and y letters apart in the two sequences, can be chained together if $x \leq d$, $y \leq d$, and $|x - y| \leq s$. Assign a weight to each pair of chained local alignment.

ii) Construct rough global map by maximizing the weight of a consistent chain of local alignments using the LIS algorithm which is also used by MUMmer. A local alignment is chained to the previous one that produces the highest scoring chain among all chains that end with this alignment. Apply step i) and ii) recursively between every pair of anchors that are separated by more bases than a threshold.

iii) Compute the optimal Needleman-Wunsch global alignment within the subset of cells at most r from cells of the anchors.

- **BLASTZ**: It is an independent implementation of the Gapped BLAST algorithm [1]. Currently the BLASTZ algorithm is used by PipMaker, a visualization tool for cross-species sequence alignment [Schwartz et al. 2000]. In contrast to other local alignment algorithms, BLASTZ requires that the matching regions must occur in the same order and orientation. The BLASTZ algorithm can be summarized as follows ([48], [49]):

  i) Remove lineage-specific interspersed repeats from both sequences. This strategy was utilized earlier by Lee et al [34].

  ii) Look for all pairs of identical 12-mers except for at most one transition

shared by the two genomes. These form a collection of seeds. Remove the seeds that align to many different regions of another genome.

iii) Extend each seed in both direction without gaps. Stop extending if the score drops below some threshold X. If the gap-free alignment score is more than some threshold Y then extend again allowing gaps. Consider the alignment valid if it scores above some threshold Z.

iv) For the regions between the zones aligned by the preceding steps, repeat these steps using a more sensitive seeding procedure (e.g., 7-mer exact matches) and lower score thresholds.

- **WABA (Wobble Aware Bulk Aligner)**: This program attempts to allow for the case in which two divergent species can have conserved gene functions while differing at their third codon positions. The underlying biological concept is the redundancy of the genetic code and the first two positions of a codon are subject to much more selective pressure than the third, "wobble", position.

  i) Look for eight consecutive nucleotides with the pattern $XXoXXoXX$, where X denotes a match and o denotes a possible mismatch.

  ii) Use a seven state pair Hidden Markov Model to produce a detailed alignment of these overlapping regions.

  iii) Stitch the overlapping alignments together.

Some of the above programs focus on alignment at the nucleotide level, others at the peptide level. That is, sequence similarity can be evaluated by comparing segments nucleotide-by-nucleotide or by translating DNA sequences

according to the genetic code and then comparing the resulting peptide segments. It is well known that proteins evolve slower than their coding DNA, so translation into peptide sequences from raw genomic sequences tends to minimize the noise caused by synonymous nucleotide substitutions. Of the possible kinds of nucleotide substitutions in a typical viral coding sequence, synonymous (or so-called "silent") substitutions are the most commonly observed. If scoring matrices are used, a synonymous substitution has a better weight than a non-synonymous substitution (generating amino acid replacements). The PAM family matrices [45] and the BLOSUM family matrices [26] are the two most commonly used types of log-odds substitution matrices for scoring amino-acid alignments. Some programs also use a scoring matrix for nucleotide substitutions. For example BLASTZ uses the scoring matrix shown in Table 2.1:

	A	C	G	T
A	91	-114	-31	-123
C	-114	100	-125	-31
G	-31	-125	-100	-114
T	-123	-31	-114	91

Table 2.1: The score matrix used by BLASTZ program to evaluate nucleotide substitutions

Given a table of scores for matches and mismatches between all amino acids or nucleotides and penalties for insertions or deletions of different lengths, a mathematically 'optimal' alignment is guaranteed. However, the scoring matrix which is most appropriate for aligning a set of sequences is affected by the level of relatedness of sequences, irrespective of whether or not it is a PAM,

a BLOSUM or other matrix. For example, the PAM1 matrix is calculated from the global alignments of sequences with no more than 1% divergence, and the BLOSUM 62 matrix is calculated from local multiple alignments of sequences with no less than 62% identity. The percentage of similarity between two genomes has to be estimated or measured before choosing a scoring matrix. The fact that the neutral rate of evolution varies across genome regions also complicates the application of cross-species sequence comparisons. Between two genomes which diverged about $40 \sim 80$ million years ago the background levels of sequence conservation can vary from one genomic region to another. Therefore it is impractical to pick a single scoring matrix and gap costs (e.g., gap opening penalty, gap extension penalty) used for identifying sequences that are under selection pressure [21]. In addition, attempts at generalizing dynamic programming to multiple alignments are thus far limited to small numbers of short sequences.

Many current global alignment algorithms use an "anchor and stitch" strategy after first building a suffix tree for both genomes in order to efficiently find the maximal set of unique matches. Although suffix trees can be built and searched in linear time using linear space, the algorithm for this is complex and the constants involved in this asymptotic complexity of this algorithm are large. Also, using the "match and extend" strategy many current local alignment algorithms pay a steep cost in extending each short match in both directions. For all vs. all sequence alignment between human genome and mouse genome even after filtering lineage-specific interspersed repeats the number of short matches that must be processed is still huge. Also, sophisticated scoring functions and alignment models result in slow algorithms that are also very memory intensive.

2.2 Multiple Sequence Alignment Methods

As natural extensions of pairwise sequence alignments, multiple sequence alignments are used ubiquitously in computational biology. They find patterns that characterize protein families; detect homology between new sequences and existing families of sequences; help predict the secondary and tertiary structures of new sequences; suggest oligonucleotide primers for PCR; and serve as an essential prelude to molecular evolutionary analysis. Aligning multiple genomes can reveal relationships among several organisms. However, available multiple alignment methods are far from mature. Multiple genome alignment is much more complicated than pairwise genome alignment, and has higher time and space costs. Many existing multiple alignment algorithms are only suitable for aligning short sequences, but not genome level sequences.

Progressive alignment is by far the most widely used approach for multiple sequence alignment, particularly for evolutionarily related sequences. This approach assembles multiple alignments progressively by a series of pairwise alignments, following the branching order in a phylogenetic tree. One first aligns the most closely related sequences, gradually adding in the more distant one. ***Iterative alignment*** uses algorithms to refine alignments through a series of iterations until no more improvements can be made. Iterative methods can be deterministic or stochastic, depending on the strategy used to improve the alignment.

2.2.1 Multiple Sequence Alignment Tools for Short Sequences

Clustal W [Thompson, Higgins and Gibson 1994], T-COFFEE [Notredame et al. 2000] and DIALIGN 2 [Morgenstern et al. 1998], are multiple sequence alignment tools for short sequences.

- ***CLUSTAL W***: This is a general purpose multiple sequence alignment program for DNA or protein sequences. It consists of three main stages [54]:

  (i) Building a distance matrix. All pairs of sequences are aligned separately in order to calculate a distance matrix representing the divergence of each pair of sequences. It uses a full dynamic programming alignments with two gap penalties and a full amino acid weight matrix. The scores calculated are the number of identities in the best alignment divided by the number of residues compared excluding the gap positions. Given n sequences, an $n \times n$ distance matrix is calculated. In the distance matrix each entry is the mean number of differences per residue converted from the percent identity scores.

  (ii) Constructing a phylogenetic tree. A phylogenetic tree is calculated from the distance matrix using the Neighbor Joining method [Saitou and Nei, 1987]. The principle of this method is to find pairs of operational taxonomic units (OTUs[=neighbors]) that minimize the total branch lengths at each stage of clustering of OTUs starting with a starlike tree. This method produces an unrooted tree with branch lengths proportional to estimated divergence along each branch. The weight of a sequence is cal-

culated based on the distance from the root of the tree to the node that represents the sequence and the number of other sequences that share the a common branch with it. In other progressive alignment algorithm, all sequences would be equally weighted.

(iii) Progressive alignment. Sequences are progressively aligned according to the increasing order of branch lengths in a guide tree using a dynamic programming algorithm. Each step aligns two existing alignments or sequences. Gaps present in existing alignments remain fixed. This strategy is justified because the placement of gaps in alignments between closely related sequences should be much more accurate than between distantly related ones. When all of the sequences being aligned are highly divergent (e.g. less than -25-30% identity between any pair of sequences), this progressive approach becomes much less reliable. New gaps introduced later get full gap opening and extension penalties. The new score at each position between two existing alignments is calculated as the average of all the pairwise weight matrix scores from the amino acids in the two sets of sequences. For two alignments with x and y sequences respectively, the score at each position is the average of $x \times y$ comparisons, multiplied by the weights from the two compared sequences. Each gap paired to a residue is scored as zero.

Clustal W includes many specialized heuristics which aim at maximal exploitation of sequence information:

- Automatic gap penalty adjustment based on the weight matrix, degree of sequence similarity, and differences in the lengths of the sequences being aligned.

- Local gap penalties based on residue-specific penalties, distances to existing gaps, and the existence of hydrophilic stretches.
- Automatic substitution matrix choice
- Delaying the alignment of distantly related sequences

- ***T-COFFEE (Tree-based Consistency Objective Function for alignment Evaluation)***: This first computes local and global pairwise alignments and then combines them into a primary library that is extended for use in computing multiple sequence alignments in a progressive manner similar to that of Clustal W. During each alignment step information from all of the sequences is considered, not just those being aligned at that stage. The primary library is extended to produce an 'extended library', a position-specific substitution matrix in which the score associated with each pair of residues depends on the compatibility of that pair with the rest of the library [43]. T-COFFEE uses a procedure reminiscent of Vingron's Dot matrix multiplication and Morgenstern overlapping weights.

- ***DIALIGN 2***: This is a consistency-based algorithm that attempts to use local information in order to guide a global multiple alignment. It preceeds as follows [41]:

  i) For each sequence pair a collection of local matches with maximum sum of weights is calculated. The weight of a match of length l_D is defined to be:

$$w(D) = -\log P(l_D, s_D) - K$$

  where $P(1_D, s_D)$ denotes the probability that a random match of the same length l_D has at least the same sum s_D of similarity values, and K is a

constant that depends on the sequence length. The purpose of including K in this formula for calculating weights is to prevent the program from splitting up long high-scoring matches into several shorter ones.

ii) Each of these matches receives another score proportional to its compatibility with the rest set of matches. This score is called the 'overlapping weight' of the pair. Sort the matches by these overlapping weight scores. The multiple alignment is then progressively assembled by adding the matches one in weight order. Local matches that are not consistent with the matches already accepted are discarded. To decide whether or not a match is consistent with others some tags, called "consistency bounds", must be stored. With addition of every match in the multiple alignment, the consistency bounds must be updated.

iii) Iteratively realign those parts of the sequences that are not yet directly aligned until no additional local matches can be found.

2.2.2 Multiple Sequence Alignment Tools for Long Sequences

Three current code available multiple whole genome alignment are MultiPipMaker [47], MultiLAGAN [37] and EMAGEN [15].

- ***MultiPipMaker***: This is a Web-based multiple alignment tool[1] which used iterative refinement. It first uses BLASTZ to get pairwise alignments between the reference sequence and each of the others. Local alignment overlaps are removed by a pruning process. Internal gaps are penalized in

[1]MultiPipMaker is available at http://bio.csi.psu.edu.

the usual way and the end-gaps lying between aligned segments are not penalized. MultiPipMaker then produces a crude multiple alignment from these pairwise alignments by an iterative refinement procedure, which repeats the following steps for each row of the multiple alignments [47]:

i) Given alignment column positions i and j, and a row index r in the multiple alignment such that there are no end-gaps between i and j of row r, extract the subalignment from i to j without row r.

ii) Reduce the sub-alignment by discarding any column that contains only internal-gap or end-gap symbols. Reduce the segment of row r by removing any internal-gap symbols.

iii) If these steps improve the score, the new alignment is spliced into the large alignment, otherwise, no change is made.

- ***MultiLAGAN***: This is an extension of LAGAN to multiple genome alignment. MLAGAN uses a progressive alignment approach. It compares synteny regions which are gene loci occurring in the same order on chromosomes of different species, and it is designed to efficiently produce detailed alignments of closely related multiple whole genomes. MLAGAN assumes that a phylogenetic tree for the species being aligned is known. It uses the phylogenetic tree to select the two closest genomes. It then uses LAGAN to produce a pairwise alignment between the two closest sequences. The program repeats this step, aligning the closest sequences or multi-sequences (alignment between two or more sequences produced before), until there remains only one multi-sequence. MLAGAN gives its user the option to perform an iterative refinement during which individual sequences are removed and realigned until no significant improve-

ment can be made. During progressive alignment, MLAGAN aligns two (multi-)sequences X/Y and Z, to generate multi-sequence $X/Y/Z$ in the following two steps [37]:

i) Select a collection of anchors for the new alignment as the anchors between X and Z, and between Y and Z. Reweight the anchors between X and Z that overlap an anchor between Y and Z according to the length of the intersection, the original scores and the length of the union of the overlapping anchors.

ii) Compute the rough global map between X/Y and Z as the highest-weight chain of these anchors using the Longest Increasing Subsequence algorithm.

- ***EMAGEN (Efficient Multiple Alignment algorithm for whole GENomes)***: This is an anchor-based alignment system [15]. EMAGEN uses an approach that combines suffix arrays and graph theoretic formulation for multiple whole genome alignment. It first finds conserved regions among multiple genomes by a linear time procedure. Then it calculates a maximum set of conserved regions by a graph theoretic approach. These conserved regions are used as alignment anchors. Short subsequences between the anchors are then aligned using the multiple sequence alignment tool Clustal W.

Chapter 3

Selection of Alignment Anchors

3.1 Filters for Selecting Alignment Anchors

Heuristic sequence alignment approaches start by generating a collection of highly similar subsequences between a target sequence and query sequence. It is not uncommon for one region of one sequence to align with several regions of the other sequence because of genome duplications, incomplete masking of interspersed repeats, or the presence of low-complexity regions. Such phenomena, and also genome rearrangements, can cause local alignments to appear superimposed. A filter for selecting alignment anchors must be used to filter out spurious matching regions. Many filters used by sequence aligners also try to filter out superimposed local alignments. Usually the single best orthologous match for each conserved region gains most biological attention. Therefore, it is reasonable to offer users a 'single coverage' option which selects a single highest-scoring chain of local alignments such that any position in the first sequence can appear in one alignment, at most once.

Tools for pairwise long sequence alignment, such as MUMmer, GLASS,

AVID, LAGAN, and WABA (discussed in the previous chapter) assume that local alignments appear in the same relative order and orientation in the target and query sequences and select a single collinear set of alignment anchors. These anchors are used to construct a rough global alignment that is iteratively refined. Two local alignments can be chained if the end of one precedes the start of the other in both sequences. The longest-increasing-subsequence (LIS) [25] algorithm and a dynamic programming approach similar to the Smith-Waterman algorithm are often used during anchor selection process. We review these techniques briefly:

3.1.1 Longest-increasing-subsequence Approach

The LIS algorithm sorts the local alignments found between Genome A and B by their positional order in Genome A, representing local alignments by integers indicating the order of their B-positions. For example, suppose that the order of B-positions is given by the sequence $\langle 1, 3, 2, 4, 6, 7, 5\rangle$, as shown in Figure 3.1. The top alignments shows all the local alignments. The shift of pairs $\langle 3, 3\rangle$ and $\langle 5, 5\rangle$ indicates transpositions. After running the LIS algorithm the LIS is given by $\langle 1, 2, 4, 6, 7\rangle$. The overall running time of the LIS method is $O(n \log n)$, where n is the size of the integer set representing the order of B-positions. Notice that this approach removes all transpositions from the remaining local alignments, leaving us a collection of non-overlapping, non-crossing local alignments. Therefore it is most appropriately used for comparison of closely related sequences where transpositions are expected to be rare.

The longest increasing subsequence algorithm is well explained by Dan Gusfield [25]. Given a set of integers $\prod$, the LIS technique finds the longest subset

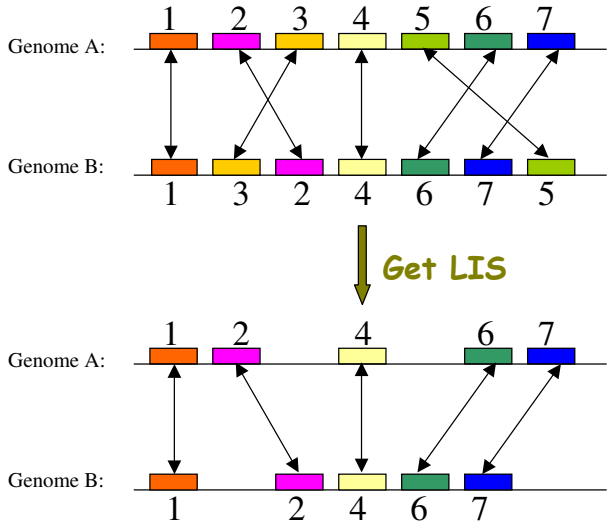


Figure 3.1: Schematic view of the LIS technique

of $\prod$ whose values increase strictly from left to right. The idea is to decompose the set of integers $\prod$ into a greedy cover C such that there is an increasing subsequence I containing exactly one number from each decreasing subsequence in C. A greedy cover of $\prod$ is a set of decreasing subsequences of $\prod$ that contain all the numbers of $\prod$.

An $O(nlogn)$ (n is the size of $\prod$) greedy cover algorithm works in the following straightforward way: Let L be the ordered list containing the last number of each decreasing subsequences built so far. Scan $\prod$ from left to right, for each number x, find the left-most number y in L larger than x using binary search, place x at the end of that subsequence containing y. If no such a y exists in L, start a new subsequence to the right of all the current decreasing subsequences built so far. Update L before examining the next number.

Given the greedy cover the longest increasing subsequence algorithm is as follows:

```
begin
0. Set i to be the number of subsequences in the
   greedy cover. Set I to the empty list; pick any
   number x in subsequence i and place it on the
   front of list I.
1. While i > 1 do
   begin
2. Scanning down from the top of subsequence i - 1,
   find the first number y that is smaller than x.
3. Set x to y and i to i - 1.
4. Place x on the front of list I.
   end
end.
```

Since every number in $\prod$ is only examined once in finding the longest increasing subsequence given the greedy cover, and since it takes $O(nlogn)$ during the running of the greedy cover algorithm, overall running time of the LIS method is $O(nlogn)$.

3.1.2 Dynamic Programming Approach

GLASS implements its dynamic programming approach in several steps: first an initial value of k is chosen, then all matching k-mers are found, then scores are assigned to matches according to their length and the alignment score of their flanking regions. Local alignments whose scores below a threshold T are removed. The related AVID aligner first finds maximal matches using suffix trees, then selects anchors using a variant of the Smith-Waterman algorithm,

similarly to the one used by GLASS. Both these algorithms require the selected anchors to be collinear.

BLAT [29] uses a graph traversal strategy in its protein alignment. Local hits are extended into maximally scoring gap-free alignments (called "high-scoring pairs", or HSPs). The scores are computed by a score function which assigns a match score 2 and a mismatch score 1. Then consider HSPs are considered as nodes in a graph. An edge from node A to B indicates that HSP A proceeds HSP B in both sequence coordinates. The edge is weighted by the score of B minus a gap penalty based on the distance between A and B. After this graph is built, a dynamic programming algorithm traverses the graph, and extracts the maximal-scoring alignment.

3.1.3 Clustering Approach

The Gcomp module of the program MGA [27] clusters local chains together if they are separated in both genomes by no more than a threshold distance. After running a chaining procedure it reports only one chain of maximum score as a representative of that cluster, similarly to the NUCmer module in MUMmerII [14].

3.2 Bipartite Graph Matching

Besides the techniques described above, several other filters can be used for choosing the best match candidates from the lists of ranked matching pairs. One prepares for this by a technique called 'multimapping', which works with a data structure containing many potentially useful mappings as subsets [39] (A multimapping can be viewed as a bipartite graph). One can use the maximum

weight matching (MWM) technique for selecting best match candidates [24]. But the MWM solver maximizes cumulative similarity, and thus might not give us the single best match for any individual region. The alternative so-called stable marriage (SM) problem incorporates a helpful intuition that leads to an improved selection strategy for multimappings.

3.2.1 The Stable Marriage Algorithm

The SM problem was introduced by Gale and Shapley [22]. This problem can be stated as follows: given two finite, equal-sized sets of items, called *men* and *women* ($m_i \in men, w_j \in women$), where each m_i/w_j ranks w_j/m_i in strict order forming his/her preference list, find a one-to-one stable matching M between the two sexes. M is "stable" if there are no two couples (m, w) and (m', w') in M such that m prefers w' to w and w' prefers m to m'. The so-called Gale/Shapley "proposal algorithm" solves this problem (This algorithm was first applied to study the problem of admission of college students to colleges of their choices. In computational biology field it has been used for protein structures classification [53]). Our application of the stable marriage algorithm as an alignment filter is explained in detail in section III, and appears to be novel. The algorithm's complexity is $O(n^2)$ where n is the number of men/women.

The stable marriage problem has a few variants: i) there can be unacceptable partners, or preference lists can be incomplete; ii) there can be ties in the preference lists; or iii) the preference lists can have both ties and incompleteness. The first two cases are still solvable in polynomial time. However, in the third case we get the Stable Marriage Problem with Ties and Incomplete lists (SMTI). In this case, the SMTI decision problem "does a stable matching of size n exist?"

and the optimization problem "find the smallest or largest stable matching" have been proven to be NP-complete [28]. In our application this issue is avoided by randomly selecting one from each group of tied partners.

3.2.2 Applying the Stable Marriage Algorithm as an Alignment Filter

We now describe the application of the stable marriage (SM) problem as an alignment filter for large-scale genome comparison.

Let $P = \{(a, b)\}$ denote the set of all 'local alignment' [1] pairs (a, b) found initially, and let $X = \{a \mid \exists b : (a, b) \in P\}$, and $Y = \{b \mid \exists a : (a, b) \in P\}$. P can be viewed as a bipartite graph, i.e. a multi-valued mapping between these matching pairs. We wish to find a one-to-one stable matching M as a subset of P. Since normally many a in X match to a multi-element subset of Y, and we might have ties in the preference list, this SM problem becomes a case of the Stable Marriage Problem with Ties and Incomplete Lists (SMTI). The preference lists required for application of the stable marriage algorithm are formed using measures of similarity defined in the manner explained in chapter 4. The required procedure uses the following two steps:

1. We count the number of K-mers shared by a local alignment (a, b) in P as $S_{(a,b)}$, and use it as a measure of the absolute similarity of (a, b). Details of the technique used to count common mers are explained in section IV-B.

2. A relative similarity $R_{(a,b)}$ is computed subsequently as fractions of the

[1] In COMBAT scheme I, 'local alignment' refers to a maximal interval in genome A that matches an interval in genome B. In COMBAT scheme II/III, 'local alignment' refers to a J-interval in genome A that matches a J-interval in genome B.

largest absolute similarities between the element a in X and its matching partners in Y. A relative similarity $R_{(b,a)}$ going in the other direction is computed in the same manner. Then each element j in X or Y ranks its match partners in strict order of $R_{(j,i)}$, forming j's preference list.

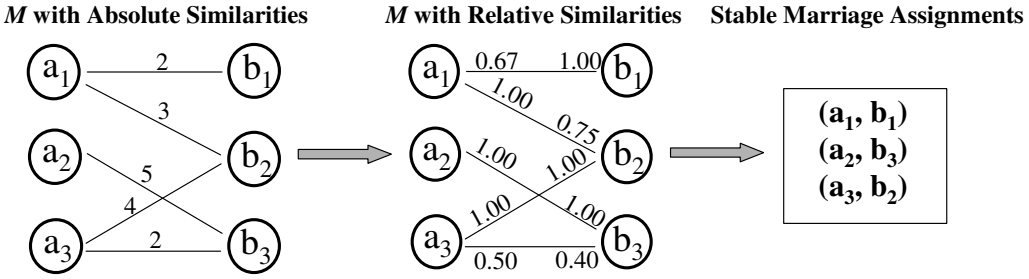


Figure 3.2: An example of a procedure for solving the stable marriage problem

In the example shown in Figure 3.2, we first compute relative similarities from absolute similarities in the bipartite graph, and then use the SM algorithm to find the stable marriage assignments. Lines $(a_1, b_1) - (a_3, b_3)$ denote the local alignments between genome A and B. In multiple mapping M with absolute similarities, the numbers on the edges show numbers of K-mers sharing by the local alignments. Numbers associated with edges in the middle panel show the relative similarities of local alignments. Since b_2 is the best match for a_1, we set $R_{(a_1,b_2)} = 1.00$. The relative similarities of other matches of a_1 are computed as fractions of $S_{(a_1,b_2)}$. Thus, $R_{(a_1,b_1)} = S_{(a_1,b_1)}/S_{(a_1,b_2)} = 2/3 \approx 0.67$. Relative similarities are asymmetric. Under the marriage interpretation, this asymmetry implies that any two match partners may like each other unequally. We modify the proposal algorithm and explain the SM algorithm used by COMBAT as follows:

```
1. X={a},Y={b},M={}. Every a and b has
   an ordered preference list.
2. WHILE X is not empty, LOOP
3.   choose an element a from X
4.   b=the first element on a's list
     (Randomly choose one from ties if such exist)
5.   IF a is not on b's preference list, THEN
6.       delete b from a's list;
7.       IF a's preference list is empty, THEN
8.          delete a from X; goto line 2
9.       ELSE goto line 4
10.  ELSE
11.      IF (x, b) is in M for some x in X, THEN
12.         remove (x, b) from M; add x to X;
13.      add (a, b) to M
14.      FOR each successor x (x ranks after a)
         in b's list, LOOP
15.         delete x from b's list, and b from
            x's list;
16.      END LOOP
17. END LOOP
18. RETURN M
```

This SM algorithm's time complexity is $O(n^2)$, and it is linear in space (n is the number of local alignments). The result returned by this algorithm is an incomplete one-to-one mapping, which means that any fragment in genome A

will map to at most one fragment in genome B, and vice versa.

Chapter 4

The COMBAT Algorithm for Pairwise Genome Comparison

The goal of COMBAT is to identify protein-encoding regions in genomic sequences using a genome comparison approach. The two genomes being compared are referred to as genome A and genome B. This chapter describes the three mer generation schemes used in COMBAT algorithm in detail.

A schematic view of the COMBAT procedure is shown in Figure 4.1. In contrast to those programs that build statistical models based on training data and those programs that utilize very sophisticated scoring functions in dynamic programming, the COMBAT procedure involves relatively few parameters that need to be pre-determined, and these are not hard to guess based on probabilistic computation. Most steps of the algorithm, except for the steps involving sorting, require only linear time and space.

The COMBAT algorithm involves the following steps:

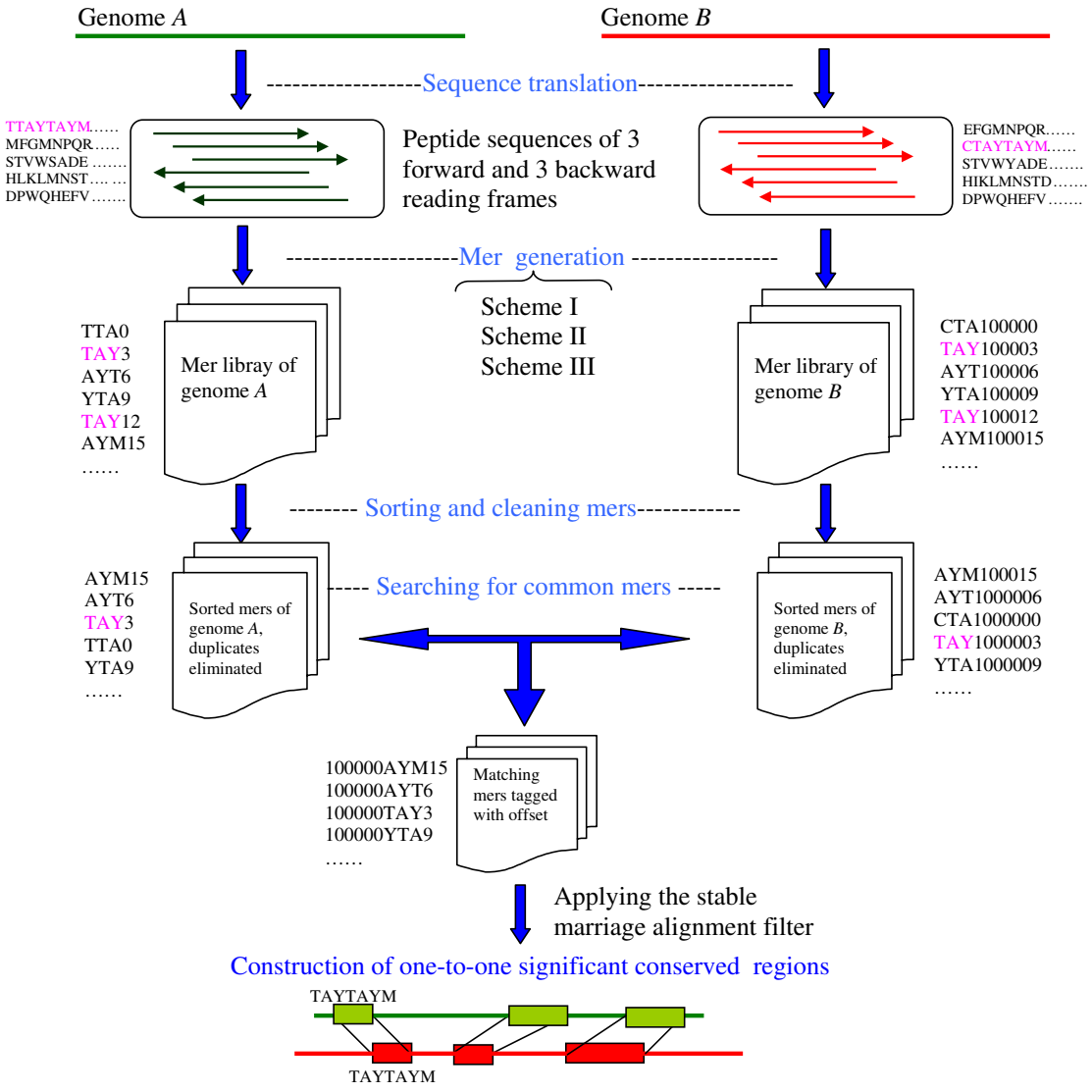


Figure 4.1: Outline of the COMBAT algorithm

4.1 Build Clean Ordered Mer Libraries

First, nucleotide sequences for the genomes of interest are translated into peptide sequences in all six possible reading frames over both forward and reverse orientations. After choosing a mer-size K, K-mers are generated using one of the three mer generation schemes described below. Mers which occur in repeats annotated by RepeatMasker are ignored. A position index is attached to each mer. All the mers generated are then sorted lexicographically by using their peptide sequences. The mer libraries for genome A and genome B are built and sorted separately.

4.1.1 Scheme I: Generate K-mers Tagged by Genomic Locations

This is the simplest of the three schemes we consider. Similar mer generation schemes are widely used in local sequence aligners. In this scheme, our 'mers' are simply polypeptide mers of a certain size, K, typically $3 \sim 9$. K-mers are generated taking all K-mers from the peptide sequence of a genome. Mers are indexed by their genomic locations. We keep up to C copies of duplicate local mers which occur in a mer library.

The following two schemes are variants of scheme I, used to reduce the size of the mer index so as to make it possible to apply the stable marriage algorithm efficiently.

4.1.2 Scheme II: Generate K-mers Tagged by Indices of 'J-intervals'

In this scheme, genomes of interest are covered with 'J-intervals'. A J-interval is a continuous genomic sequence of length J. Adjacent J-intervals are spaced $J/2$ bases apart. These J-intervals cover each base in the genome twice, except for the bases at the beginning and ending regions of the genome. The serial index of a J-interval is called its J-index. The "representation of position" that we attach to each K-mer is the index of each J-interval to which it belongs (named 'J-index'). In Figure 4.2, K_i denotes the ith K-mer; J_j denotes the J-index of the jth J-interval. Most mers (like K_{13}) occur in regions covered by two adjacent J-intervals, and so will appear twice in the mer library. Within each J-interval we keep only one copy of duplicate K-mers if there are any such mers. This makes K-mers unique within every J-interval.

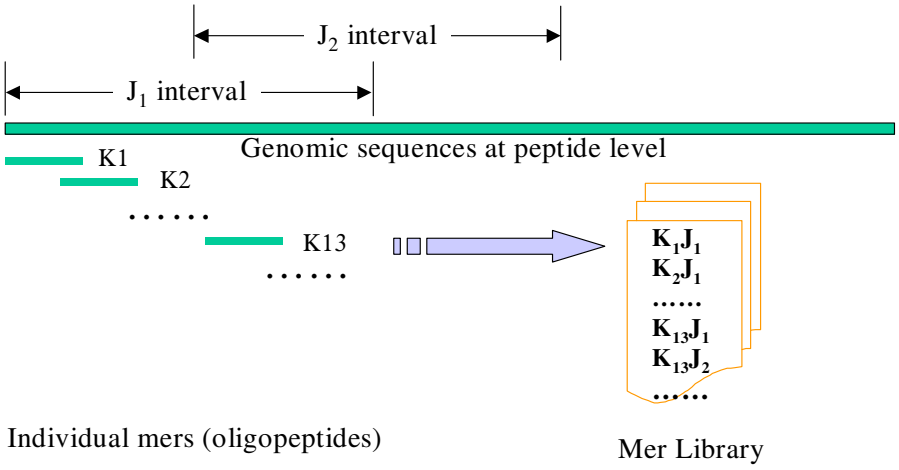


Figure 4.2: Build mer library using mer generation scheme II

Every genomic interval I of length at most $J/2$ is contained in at least one of

these J-intervals. To see this, consider the rightmost interval j which includes the right-hand end point R of I. If this does not include all of I, then clearly R must lie in the left half of J, and the left end of I must lie within $J/2$ positions to the left of J. Hence, by moving j $J/2$ positions to the left, we get another J-interval which does include I.

Suppose then that we take only the tagged A-mers that appear only once, and likewise the tagged B-mers that appear only once. We can then consider the offsets[1] between these. In a random string these offsets would be scattered randomly over the whole range $-2G$ to $+2G$, so the probability of an offset being matched accidentally by another is $1/4G$. Consider one of the J-base long J-intervals, and any one of the J K-mers in it. In the random case, the probability that there should exist another K-mer in the same J-interval which has the same offset is $J/(4G)$, so the probability that two K-mers in the same J-interval should have the same offset is therefore $J^2/(4G)$. This small probability indicates that the number of negative cases which are caused by eliminating duplicates inside the same J-interval will be reasonably few.

4.1.3 Scheme III: Generate 'gapped' Local Mers Containing Two Neighboring Mers

In this scheme, we allow local mers which contain two nearby unbroken K-mers K1 and K2, with K usually equal to 3 or 4, but with a gap between them, the number of amino acids constituting the gap being allowed to vary from 0 to M, where M is usually chosen to be $5 \sim 10$. The peptide sequence assigned to such

[1]The offset d_{ij} = A_i - B_j is the index difference between mer i in genome A and its matching mer in genome B.

a ‘gapped mer’ is then the concatenation of the sequences of K1 and K2, and the location index used is K1’s J-index. For comparing small genomes, instead of using K1’s J-index, we can use the starting genomic location of K1 as a location index of a K-mer. Allowing a few amino acids between nearby K-mers lets us detect imperfect local alignments invisible to scheme II. A schematic view of this procedure is given in Figure 4.3, where green boxes denote two neighboring K-mers, K1 and K2. At every genomic position $M+1$ local mers are generated with fixed K1 but varying K2 for every reading frame.

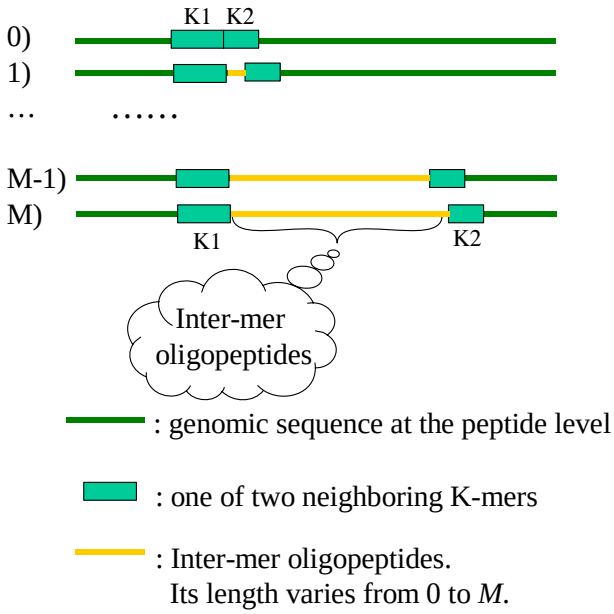


Figure 4.3: Mer generation scheme III.

This scheme is easily extended to generate other ‘generalized local mers’, e.g. those containing three neighboring mers with a few amino acids appearing among these mers. If this extended gapping is allowed, very many objects can be generated, but they are generated rapidly in a linear fashion. The sorting and pruning steps guarantee the order and uniqueness of these objects. So the

computation will not be heavy in time and space.

4.2 Search For Common Mers

Having built cleaned ordered mer libraries for the two genomes of interest in one of the three ways just described, we now search for common mers.

In scheme I, genome position offsets between all matching mer pairs A_i, B_j are computed and attached to the mers A_i take from the 'A' genome. The offset $d_{ij} = A_i$ - B_j is the index difference between mer i in genome A and its matching mer in genome B. Then this list of mer-with-offset pairs is sorted by their offsets and their indexes in genome A. Mers sharing the same offsets within a local window of a specified size are grouped together to form a local alignment. The number of common mers in each local alignment is counted and used as the similarity measure of that local alignment. Refer to section III for the use of these values.

In scheme II, we look for mers having identical peptide sequences and for each such mer place an item in our 'common mer library'. Each item contains the corresponding pair of J-indices (but not detailed genome positions), one from each genome. Our common mer library is then sorted using J-indices in genome A as the primary sorting key and J-indices in genome B as the secondary sorting key. Then for each matching interval pair in the sorted common mer library the number of common mers is counted and used as the measure of absolute similarity of that pair. We set a threshold T, and keep only those matching interval pairs whose number of common K-mers exceeds T for the next step.

Peptide sequences which appear more than C times in any one of the two mer libraries are not considered when looking for common mers. This threshold

C can be estimated by the following simple probabilistic computation:

The number of K-mers in the peptide sequences taking all six reading frames together is:

$$N = 6 * G/3 = 2G$$

where G is the size of the genomes being analyzed, e.g. ~ 3 billion bases for the human genome and ~ 12 million bases for the yeast genome.

If mer generation scheme I is used, we usually set $C = 1$. Since the alphabet size is 21 for protein sequences (20 amino acids plus one stop codon), the probability that at least that one K-mer in a genome matches a random K-mer is:

$$P_1 = N/21^K = 2G/21^K.$$

If $K = 9$ is used in comparing the Human genome to the Mouse genome, then $P_1 \approx 0.008$. Hence, randomly duplicated peptide 9-mers should be relatively rare, and elimination of duplicates should remove less than 1% of the 9-mers on average. In comparing yeast genomes, we can take $K = 7$ and get $P_1 \approx 0.013$.

If mer generation scheme II is in use, the probability that a random K-mer in one genome matches a K-mer in another genome is:

$$P_2 = \frac{2N}{(21^K * (N/J))} = \frac{2J}{21^K}$$

The expected number of identical K-mers in one genome in different J-intervals are therefore:

$$C = \frac{2N}{21^K * J} = \frac{4G}{21^K * J}$$

In comparing the Human genome to the Mouse genome, if $K = 5, J = 1000$, then $P_2 \approx 4.90 \times 10^{-4}$ and $C \approx 3$. In comparing yeast genomes, it is better to take $K = 4, J = 100$, then $P_2 \approx 6.35 \times 10^{-9}$ and $C \approx 3$.

If mer generation scheme III is used, C increases by the factor M. In practice, we found C needed to be increased by $2 \sim 10$ times in order to improve sensitivity, especially when contig sequences are used since these contain many repeated subsequences.

4.3 Find A One-to-One Correspondence

Once having the multimappings of 'local alignments' produced by previous step, we apply the modified SM algorithm described in chapter 3 to find the single best alignment. In COMBAT scheme I, 'local alignment' refers to a maximal interval in genome A that matches an interval in genome B. In COMBAT scheme II and III, 'local alignment' refers to a J-interval in genome A that matches a J-interval in genome B.

4.4 Optional Chaining Procedure

Lastly, a chaining procedure is performed to further reduce local alignments caused by accidental random matches. In chaining, we demand that there must be at least F local alignments, each no farther than E intervals from each other. Our experiment results show that this step is not necessary if we choose suitable values of J and K. The parameters involved in our COMBAT experiments are summarized in Table 4.1.

K	The K-mer size
J	The length of a J-interval
C	The allowed maximum number of copies of a certain peptide composition we keep in each mer library
T	The minimum number of common K-mers required in any matching J-interval pairs
M	The maximum number of amino acids between two nearby K-mers when using mer generation scheme II or III

Table 4.1: Parameters involved in COMBAT

Chapter 5

Evaluation of Performance of COMBAT

We tested the performance of COMBAT as applied to small genome comparison (between yeast genomes) and large genome comparison (between human and cow genomes). Our evaluations focus on the efficiency of COMBAT tool, and the coverage and accuracy of COMBAT experimental results.

5.1 Yeast Genome Comparison

Several complete genomes of yeasts were used for testing the COMBAT algorithm. The reasons for focusing on yeast species are:

i) Yeast is well studied and annotated. Yeast is a superb model for understanding the basic functions of human cells, which must do nearly everything yeast cells do. When a gene in yeast similar to a gene in human is located, its function in human genome can be deduced through experiments with yeast, which is much more amenable to genetic manipulation.

ii) The sizes of different yeast genomes are very similar, around 12 MB to 14 MB which is not too big or too trivial for testing the algorithm efficiently.

iii) Six complete yeast genomes are available[1]. These four hemiascomycete yeasts as well as the most popular yeast Saccharomyces cerevisiae (baker's yeast) and Schizosaccharomyces pombe (fission yeast) represent a broad evolutionary range within a single eukaryotic phylum.

A cross-species sequence comparison program's sensitivity and specificity can be used to measure its success in identifying orthologous regions of two or more genomes. We consider identified conserved regions as true positives (TP) if they belong to annotated exons, and as false positives (FP) if they do not; we treat identified non-conserved regions as true negatives (TN) if they do not belong to annotated exons, and as false negatives (FN) if they do. Mathematically, a matching procedures's specificity Sp is defined as: $Sp = TP/(TP + FP)$, and its sensitivity Sn is defined as: $Sn = TP/(TP + FN)$. So specificity reflects the accuracy of a program and sensitivity measures its coverage of annotated exons.

To assess the COMBAT method we performed a comparison between closely related *Saccharomyces cerevisiae* and *Candida glabrata*. Scheme I was used, first with $K = 9$, to generate peptide mers of length 9. Common mers were detected and merged to form local alignments as explained previously. $10,136$ local alignments between *Saccharomyces cerevisiae* and *Candida glabrata* were found, of which $7,027$ (i.e 69%) actually occur in the protein database, as was

[1]In a comparative genomics project designed to examine eukaryotic genome evolution, the Génolevures Consortium determined the complete genome sequences of the yeast species, including *Kluyveromyces lactis*, *Candida glabrata*, *Yarrowia lipolytica*, and *Debaryomyces hansenii*.

confirmed by searching these regions using BLASTP with its default parameter setting. But 96% of the local alignments found by COMBAT but not found by BLASTP in the protein database were found to belong to parts of functional elements in yeast genomes by doing a BLASTN search in the nucleotide database. This discrepancy may be due to the incompleteness of the protein database or the insensitivity of BLASTP. These local alignments belong to $3,865$ different proteins (61% of the $6,300$ total *Saccharomyces cerevisiae* protein database).

If we use $K = 7$, then $25,915$ local alignments were found, of which $12,482$ (i.e 48%) different proteins were confirmed by BLASTP. 79% of the local alignments found by COMBAT but not found by BLASTP in the *Saccharomyces cerevisiae* protein database were confirmed by BLASTN search in the nucleotide database. These local alignments found by COMBAT belong to $4,733$ different proteins (75% of the $6,300$ total cerevisiae protein database). This experimental data, used to calculate the sensitivity and specificity of COMBAT, are summarized in Table 5.1.

	Sp(BLASTP)	Sp(BLASTP/BLASTN)	Sn
K = 9	69%	99%	61%
K = 7	48%	89%	75%

Table 5.1: Sensitivity and specificity of COMBAT using scheme I assessed by BLASTP or BLASTP/BLASTN search results

There are several situations that might prodece false negative when mer generation scheme I is used:

i) If the corresponding K-mers of every matching region in a local alignment are removed since they have as duplicates in other regions in the genome, then

the corresponding local alignment cannot be found by COMBAT using mer generation scheme I.

ii) The real alignments are composed of tiny aligned pieces of length less than K. In this case we lack a necessary 'seed' that is long enough to be detected by COMBAT using mer generation scheme I. This problem can be ameliorated by using small value of K. However if K is too small then real matches will get lost in the 'noise' of small random matches. A better solution is to use mer generation scheme III, especially in large genome comparison.

We studied the number of false negative cases which arise from using scheme I and removing duplicates, in the following way: BLASTP searches were performed for the false negative cases (2420 *Saccharomyces cerevisiae* proteins COMBAT missed using $K = 9$) of our results against the official database of $\sim 6,300$ *Saccharomyces cerevisiae* proteins; we then collected the longest continuous match lengths in the alignments obtained by BLASTP. Histogram of these longest match lengths is shown in Figure 5.1. It indicates that if a local alignment has a continuous match of length larger than 11 then it is very unlikely to be missed. There are 793 cases missed by BLASTP search due to the limit of BLASTP method itself.

We estimated the extant to which false negatives could be reduced by using a smaller value of K in the following manner: first we searched the protein database using BLASTP for each of those 2420 missed *Saccharomyces cerevisiae* proteins; we then collected the longest continuous match lengths in the alignments between each of these 2420 *Saccharomyces cerevisiae* proteins and the matching protein of one of the other five yeasts (*Kluyveromyces lactis*, *Candida glabrata*, *Debaryomyces hansenii*, *Yarrowia lipolytica*, and *Schizosaccharomyces pombe*). The results of this search are summarized in Table 5.2. For example,

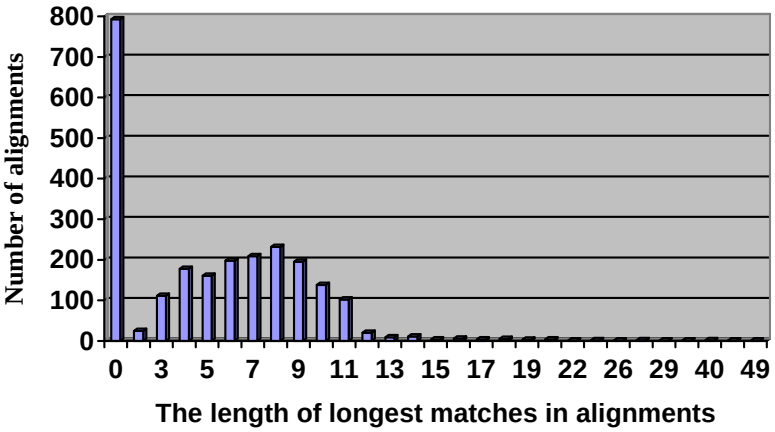


Figure 5.1: Histogram of longest continuous matches length in the alignments for the 2420 false negative cases of the comparison between *Saccharomyces cerevisiae* and *Candida glabrata*.

57% *Candida glabrata* proteins align to some of the missing *Saccharomyces cerevisiae* proteins whose alignments are at least 3 mer long. This empirical analysis implies that by using $K = 3$ COMBAT might be able to find 57% of those missed at $K = 9$ while comparing the genomes of *Saccharomyces cerevisiae* and *Candida glabrata*[2]. Since *Saccharomyces cerevisiae* and *Schizosaccharomyces pombe* are poorly related, comparison between these two genomes generate alignments containing more tiny aligned pieces than other comparison.

Another possible strategy for improving COMBAT's sensitivity is to employ multiple genome comparison. We studied the possibility of finding more proteins by comparing all five other yeasts with *Saccharomyces cerevisiae*. We reasoned that a *Saccharomyces cerevisiae* protein can be found by this variant of COMBAT if it has at least one matching protein of any of these five yeasts, by a match confirmed by BLASTP, and if the alignment contains at least one

[2]These percentages are approximate because if we lower the values of K we may have to deal with more random local mers bringing more noise to the process.

	K=7	K=3
Kluyveromyces lactis	32%	62%
Candida glabrata	33%	57%
Debaryomyces hansenii	15%	52%
Yarrowia lipolytica	11%	42%
Schizosaccharomyces pombe	4%	25%

Table 5.2: Estimated percentages of reduced false negatives by using smaller values of K

match no shorter than K. The estimated reduced percentages of false negatives if all these five yeast genomes are considered are summarized in Table 5.3. This table shows that it is possible to find roughly the same number of proteins using $K = 6$ by multiple genome comparison as using $K = 3$ by pairwise genome comparison. Lineage-specific protein loss, to a large extent, accounts for the differences in gene repertoire between genomes, particularly among eukaryotes. By comparing multiple genomes one is likely to find proteins that are lost in one species but remain in others.

K = 9	K = 6	K = 3
32%	55%	72%

Table 5.3: Estimated reduced percentages of false negatives if all five yeast genomes are considered with different values of K used.

Although the above results show that if we use $K = 3$ we will be able to find a lot more proteins, the large number of small random local mers can bury the real matches. As a possible way of dealing with this problem, we studied the

number of possible proteins COMBAT can find by using scheme III to generate local mers. Table 5.4 shows that by using $K1 = 4$ and $K2 = 3$, 65% out of 2420 otherwise missing proteins might be found by COMBAT, which indicates that its sensitivity is about 86% in this case.

K1 = 4, K2 = 3	K1 = 3, K2 = 3
65%	68%

Table 5.4: Estimated reduced percentages of false negatives by using mer generation scheme III.

5.2 Human Assembly and Cow Contig Comparison

We have applied COMBAT to Human Assembly (hg17, May 2004) and Cow Contigs (bosTau1, Sep. 2004, BCM HGSC Btau_1.0), both from the UCSC Genome Bioinformatics Site. As an example illustrating our results, we take chromosome I from hg17 and the first 33,000 cow scaffolds, and align them by COMBAT using mer generation scheme II. These two sequences are approximately 250MB in size. Let us call the first sequence *chr1*, and the second sequence *cow1*. The resulting alignment maps are shown in Figure 5.2, with the X-axis showing the J-indexes along the *chr1* sequence, and the Y-axis showing those along the *cow1* sequence. Figures 5.2-(1),(2),(4), and (5) are the results produced by COMBAT, with each plus sign representing the index coordinates of a pair of matching intervals found by COMBAT. Figure 5.2-(4) and (5) show results obtained without using the chaining procedure. Figures 5.2-(3) and

(6) show matches produced by BLASTZ, filtered by the axtBest program [48] (downloaded from the UCSC Genome Bioinformatics Site and transformed to fit our J-intervals context), with each dot representing the index coordinates of the starting positions of two matched regions. The BLASTZ result is transformed twice according to two values of J used. The BLASTZ result contains the best alignments in the genome with gaps in the best alignments filled in by next-best alignments where possible.

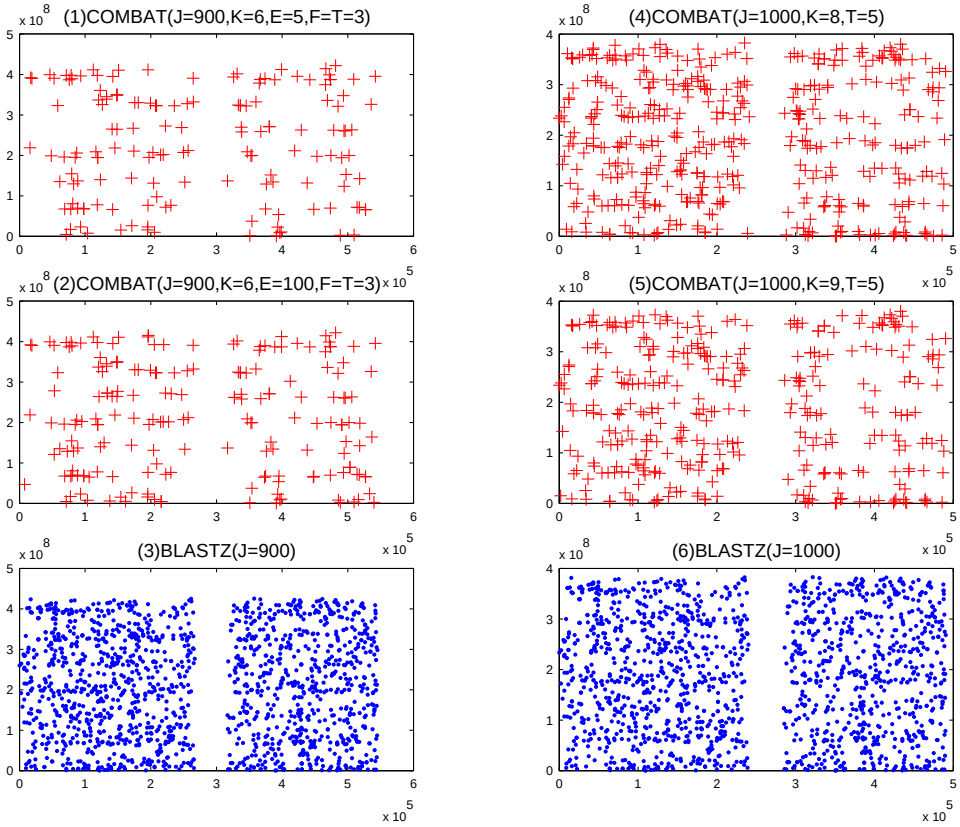


Figure 5.2: Alignment maps between *chr1* and *cow1*

The chaining criterion used by COMBAT turns out to be relatively insensitive to the value of E used (see Figure 5.2-(1) and 3-(2)). To evaluate COM-

BAT, we have tested the appearance of every matching pair of intervals found by COMBAT in the BLASTZ result (transformed by the same J used by COMBAT). Consider a pair of matching J-interval (a, b) in COMBAT result as a true positive case if there exists a pair of matching regions (x, y) (a and x in genome A, b and y in genome B) in BLASTZ result and one of the following conditions is satisfied: 1) a is contained in x and b is contained in y; 2) x is contained in a and y is contained in b; 3) the starting positions of a/b is within J bases of those of x/y, respectively; 4) the ending positions of a/b is within J bases of those of x/y, respectively. In Figure 5.2-(1), 95% of the 625 partner interval pairs found by COMBAT are true positives. In the other direction, out of 8,389 matching regions in the BLASTZ result, 7% are confirmed by COMBAT. In Figure 5.2-(4), there are 84% true positives out of 1235 PIPs, and they cover 11% of the BLASTZ result. In Figure 5.2-(5), there are 85% true positives out of 971 PIPs, and they cover 9% of the BLASTZ result. This high specificity indicates a promising wide use of COMBAT. The low coverage is not surprising because only highly similar protein-coding regions are expected to be found.

5.3 Human Genome and Cow Assembly Comparison

We have also applied COMBAT using mer generation scheme II and III to the most recent Human Assembly (hg18, Mar. 2006) and Cow Assembly (bosTau2, Mar. 2005), both downloaded from the UCSC Genome Bioinformatics Site. In this experiment, we consider chromosome I from these genomes. Call the human chromosome I *Hchr1* ($\sim$ 252Mb), and the cow chromosome I *Cchr1* ($\sim$

105Mb). We use $J = 1000$ in these experiments. The corresponding alignments are shown in Figure 5.3, with the X-axis showing the J-indices of matches along the *Hchr1* sequence, and the Y-axis showing those along the *Cchr1* sequence. Red dots represent J-indices of matching J-intervals found by COMBAT; Blue dots represent J-indices of matching regions found by BLASTZ. Figure 5.3-(1), (2) and (3) show the experimental results using mer generation scheme II, and Figure 5.3-(4) and (5) describe the experimental results using scheme III.

To evaluate the performance of COMBAT, again we have tested the appearance of every matching J-intervals found by COMBAT in the BLASTZ result (transformed by the same J used by COMBAT). Specificity and sensitivity values are shown in Figure 5.3-(6). Specificity varies from 92% to 81% and sensitivity varies from 33% to 12%. Generally speaking, using mer generation scheme III brings more noise to alignment results but gains better coverage; using mer generation scheme II produces reliable results with low coverage.

5.4 Implementation and Speed

The computational core of the COMBAT algorithm was implemented as a C++ program and all experiments were performed on NYU Bioinformatics Group's cluster of Pentium IV machines with 3 GB memory running RedHat Linux 7.3.

To compare 0.25 Gb of human sequence against 0.24 Gb of cow sequence ($\sim$ 1/10 of total genomes) and produce the one-to-one mapping list of highly similar regions, it took 23 CPU hours under the configuration shown in Figure 5.2-(1), and took 2 CPU hours under the configuration shown in Figure 5.2-(4).

To compare 0.25 Gb of human sequence against 0.15 Gb of cow sequence and produce the one-to-one mapping list of highly similar regions, it took 1.6

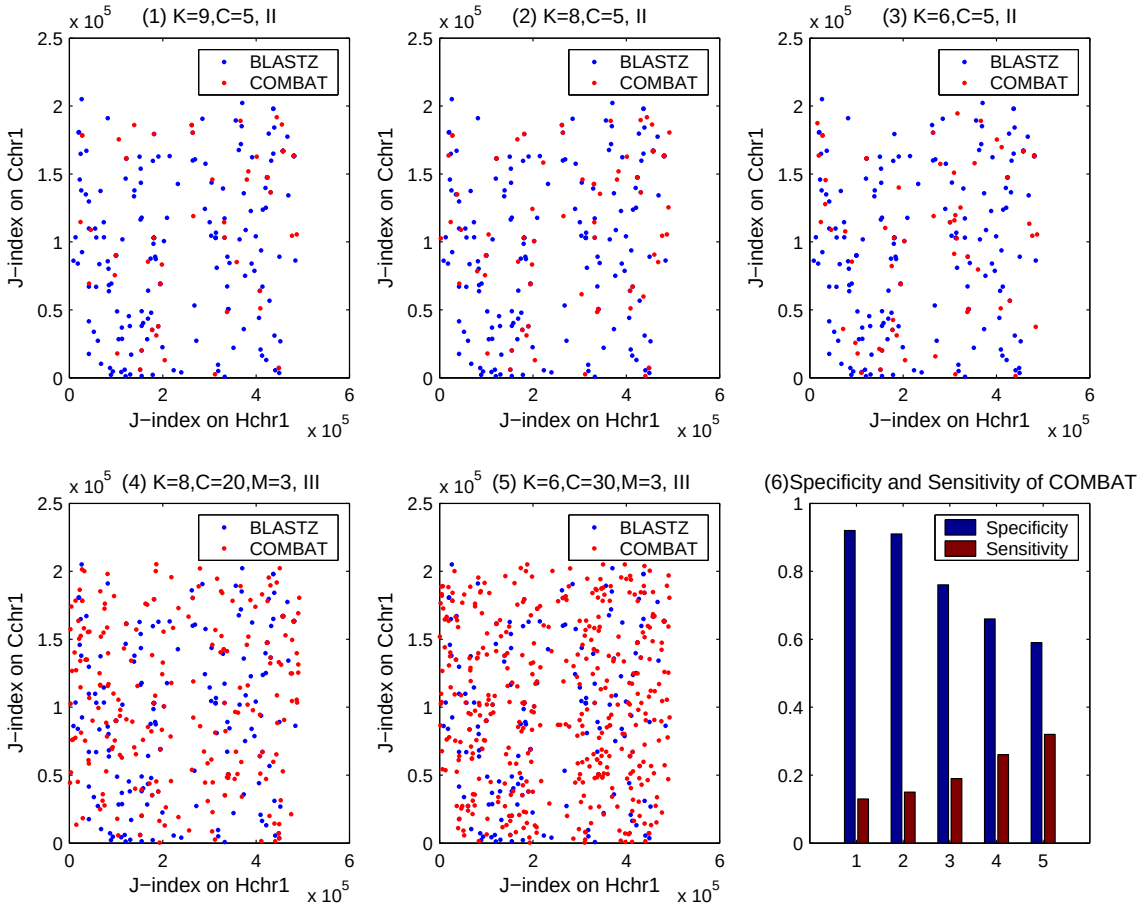


Figure 5.3: (1)-(5): Alignment maps between *Hchr1* and *Cchr1*. $T = 5$ for all maps. (6): Specificity and sensitivity measurement of COMBAT from the experiments (1)-(5).

CPU hours to obtain the result shown in Figure 5.3-(1), and took 28 CPU hours to obtain the result shown in Figure 5.3-(5). Other experiment running times vary between these two extreme ones.

This speed should be compared with the published report of BLASTZ taking 481 days of CPU time to align 2.8 Gb of human sequence against 2.5 Gb of mouse sequence on a cluster of 1024 833-Mhz Pentium III [48].

5.5 Error Estimation

In this section we discuss how to estimate the lower bound and the upper bound of the threshold parameter T used in COMBAT. Consider two random J-intervals a in genome A and b in genome B (each of length J over an alphabet of 20 amino acids and 1 stop codon). For the sake of simplicity, we consider these intervals in one orientation only. Let P_k denote the probability that there is a common K-mer at any position. Assuming that letters occur at any given position with equal probability and statistical independence, we get $P_k = 1/(21)^K$. Let the positive-valued random variable w denote the number of common K-mers in a and b. We can show that w follows a Poisson distribution with parameter $\lambda_w = J^2 P_k$. The expectation of a new random variable $\binom{w}{i}$ can be estimated by considering all possible $\binom{J}{i}$ subsets of K-mers from a and counting the probability of each such subset having exact matches with i K-mers in b.

$$
\begin{aligned}
E\left[\binom{w}{i}\right] &= \binom{J}{i}(JP_k)((J-1)P_k)\cdots((J-i+1)P_k) \\
&\approx \frac{J^{(i)}}{i!}\left(\frac{J}{21^K}\right)^i \approx \frac{\left(J^2/21^K\right)^i}{i!}
\end{aligned}
$$

Using Brun's sieve, the probability that two randomly selected J-intervals from genome A and genome B have exactly m K-mers in common is:

$$Pr[w = m] = e^{-(J^2/21^K)} \frac{\left(J^2/21^K\right)^m}{m!}$$

Using parameters of this Poisson distribution, we can choose a lower threshold such that two random J-intervals are unlikely (with probability $> 1 - \epsilon$) to have more than θ_w K-mers in common. Using Chebychev's inequality, we see that a conservative choice would be:

$$\theta_w = \mu_w + \frac{\sigma_w}{\sqrt{\epsilon}}, \qquad \text{where} \quad \mu_w = \frac{J^2}{21^K},\ \sigma_w = \frac{J}{21^{K/2}}$$

As argued earlier, by using the one-tailed Chebychev bound, we have:

$$Pr(w > \theta_w) = Pr(w - \mu_w > \frac{\sigma_w}{\sqrt{\epsilon}}) < \epsilon$$

By choosing a very small value of ϵ (for example, $\epsilon \approx O(1/G)$, where G is the genome size), we could make the probability of false positive adequately small.

In the other direction, let s be a desired similarity value, in the sense that COMBAT must almost always find pairs a and b, whenever they have a similarity value of s or higher. The number of observed K-mers shared by a and b can be viewed as a random variable v: $B(|a \cap b|, s)$ which has a Binomial distribution with mean $\mu = |a \cap b|s$ and variance $\sigma^2 = |a \cap b|s(1 - s)$. Using the Chernoff bound, we can choose an upper threshold of $|a \cap b|s/2 > Js/4$ to guarantee a probability of success larger than $(1 - \epsilon)$, if J is sufficiently large, i.e., $Js > 16\ln(1/\epsilon)$. Assuming $\epsilon = 1/G$, and $16\ln(G)/s < J \ll G$, we will need to satisfy the following inequality:

$$\frac{J^2}{21^K} + J\sqrt{\frac{G}{21^K}} < \theta < Js/4$$

$s = 0.8$	
$G = 10^9$	$G = 10^6$
$J = 1000, K = 8$ $162 < \theta < 200$	$J = 1000, K = 6$ $108 < \theta < 200$
$s = 0.6$	
$G = 10^9$	$G = 10^6$
$J = 1000, K = 9$ $35 < \theta < 150$	$J = 1000, K = 6$ $108 < \theta < 150$

Table 5.5: Exemplary choices of parameters given G and s when $\epsilon = 1/G$

or

$$\frac{J}{21^K} + \sqrt{\frac{G}{21^K}} < \theta' < s/4$$

Since G and s are determined by the genomes, we need only to choose K and J. Table 5.5 shows some exemplary choices of parameters. The variable θ has the same meaning as the T parameter in Table 4.1. Since ϵ is extremely small here, the suggested range of θ is very conservative. Note that since our estimations are rather conservative, we found that, in practice, COMBAT performs quite well even for suboptimal choices of parameters.

Chapter 6

CAPO: Comparative Analysis and Phylogeny with Optical-Maps

Given DNA sequences of various taxa, a standard technique in evolutionary analysis is to first perform a multiple sequence alignment (on DNA sequences or protein sequences). From the resultant distance matrix a phylogenetic tree can be built describing the relationship of the various taxa with respect to each other. These distance-based methods compress sequence information into a single number and the two sequences with shortest distance are considered as most closely related taxa. However, the high cost of sequencing techniques and the biological diversity among the genomes, together makes it impossible to study phylogeny using detailed sequences of many strains of large-number of related species. The low cost and high speed of the Optical Mapping technique provide an elegant solution to this dilemma, provided that one can devise suitable tool to infer phylogeny from optical mapping data instead of sequence data. CAPO

(Comparative Analysis and Phylogeny with Optical-Maps) is a novel tool that combines the Stable Marriage (SM) algorithm and a distance-based method (either the UPGMA or the NJ method) to infer phylogeny among multiple strains or genomes.

Standard methods for constructing phylogenetic trees are reviewed below, followed by a review of an existing statistical method developed by OpGen to infer phylogeny using optical-map comparison. Then the CAPO method is explained in section 6.3. CAPO experiments and discussion are given in the end.

6.1 Review of Evolutionary Analysis

A phylogenetic tree represents the evolutionary history of a family of organisms. Constructing phylogenetic trees is a crucial step for biologists in finding out how all the extant species are related to one another in terms of common ancestors. Numerous computer tools have been developed to construct such trees. The tools proposed for construction of phylogenetic trees can be classified into two groups: the phenetic methods (distance matrix method, Michener and Sokal, 1957) and the cladistic methods (maximum parsimony and maximum likelihood, Hennig 1966). Popular programs of constructing phylogenetic trees include PHYLIP[1] (phylogenetic inference package by J. Felsenstein) and PAUP[2] (phylogenetic analysis using parsimony from Sinauer Assoc.).

Phenetic methods use various measures of overall similarity for grouping. They can use any number or type of characters, but the data used must be

[1]Available at evolution.genetics.washington.edu/phylip.html

[2]Available at paup.csit.fsu.edu.

converted into numerical values. The organisms are compared to each other for all of the characters and then the similarities are calculated. After this, the organisms are clustered based on these similarities. Such methods place a greater emphasis on the relationships among data sets than on the paths they have taken to arrive at their current states. They do not necessarily reflect evolutionary relations. The cladistic method is based on the idea that members of a group share a common evolutionary history and are more closely related to members of the same group than to any other organisms. This method differs from phenetics in that it does not give equal weight to all characters. Cladistic approaches focus more on evolutionary pathways than on relationships. Figure 6.1 shows how to select an appropriate method to infer phylogeny given single-gene sequences.

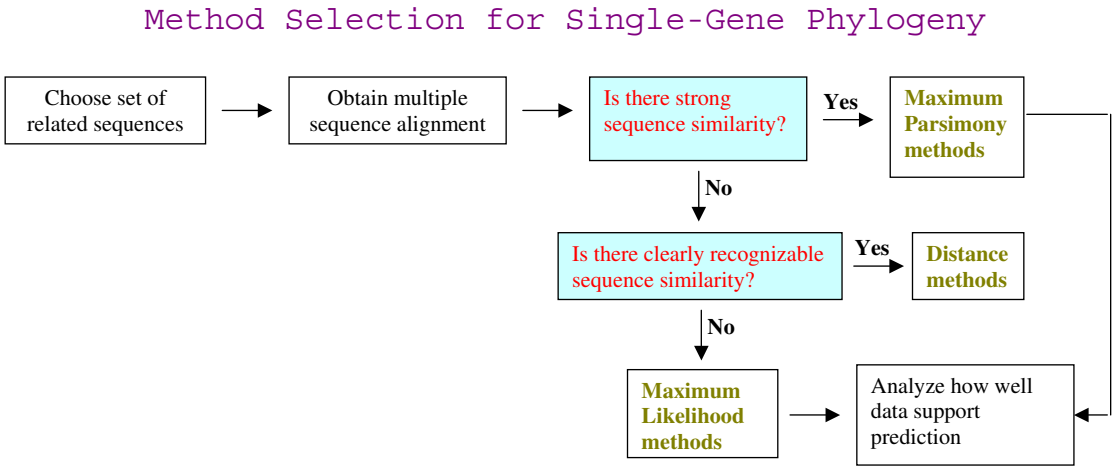


Figure 6.1: Procedure of selecting an appropriate method to infer phylogeny given single-gene sequences.

In the following paragraphs, we review five well-known algorithms: Un-

weighted Pair Group Method using Arithmetic Average [50], Neighbor Joining [46], Fitch Margoliash [20], Maximum Parsimony [17, 19], and Maximum Likelihood [18].

6.1.1 Unweighted Pair Group Method with Arithmetic Mean (UPGMA)

UPGMA [50] is a sequential clustering algorithm. It works by first constructing a distance matrix, then iteratively amalgamating two Operational Taxonomy Units (OTUs) at each stage. This amalgamated pair is represented by a new internal node in the tree. Whenever two nodes are merged into a new node, UPGMA recalculates the distances between the new node and other nodes. This process is repeated until all OTUs are grouped into a single cluster. UPGMA produces a rooted tree. It is suitable for constructing phylogenetic trees for taxa with the relatively constant rate of evolution.

UPGMA is simple and fast. But it has several disadvantages:

- It implicitly assumes the existence of an ultrametric tree: the total branch lengths from the root to any leaf are all equal. In other words, it assumes that there is a "molecular clock", which ticks at a constant pace, and that all the observed species are at an equal number of ticks from the root, which is often not the case.

- It assumes the additive property.

6.1.2 Neighbor Joining (NJ)

NJ [46] is a heuristic greedy algorithm. It begins with distance matrix and a star-like tree. At each stage two closest neighbors are joined into a new node, which becomes the root of the new tree. The branch lengths from the two nodes to the new node are calculated. The two nodes are replaced by the new node in the distance matrix, thus reducing the number of OTUs by 1. NJ then updates the distance matrix and repeats the node merging process until all OTUs left are joined into a root node. Unlike UPGMA, which chooses the neighbors with minimum distance, NJ chooses the neighbors that minimize the sum of branch lengths at each stage.

NJ has following advantages:

- It is fast and well suited for data sets of substantial size and also for the postprocessing step of bootstrap analysis.
- It is especially suitable when the rate of evolution of the separate lineages under consideration varies.

Unfortunately, NJ suffers from several disadvantages:

- It depends heavily on the evolutionary model applied.
- It assumes the additive property.

Both UPGMA and NJ algorithms use distance matrices to reflect evolutionary relationships and compress sequence information into a single number, so they cannot reflect the changes of character states of sequences. UPGMA and NJ are both relatively fast, so they are suitable for analyzing large data sets, and can handle species that are not very strongly similar. In general, NJ gives better results than UPGMA.

6.1.3 Fitch Margoliash (FM)

FM [20] assumes that the expected error is proportional to the square root of the observed distances. It compares the two most closely related taxa to the average of all the other taxa. It then moves through the tree sequentially to calculate the distances between decreasingly related taxa until all the distances are found.

Its advantage is following:

- It does not assume a constant rate of evolution and therefore can produce varied branch lengths from a common ancestor.

Its main disadvantage is:

- It requires longer execution time than UPGMA and NJ.

6.1.4 Maximum Parsimony (MP)

Maximum parsimony trees [17, 19] are built upon this principle that simple hypotheses are more preferable than complicated ones. Consequently construction of trees using this method requires using the smallest number of evolutionary changes among the OTUs in order to explain the phylogeny of the species under study. This method compares different parsimonious trees and chooses the tree that has the least number of evolutionary steps (substitutions of nucleotides in the context of DNA sequence). MP is a character-based Maximum Parsimony algorithm. It starts with a multiple alignment and construct all possible phylogenetic trees for the species of interest. It scores each of these topologies and chooses a tree with the fewest 'evolutionary changes' as the final tree, where an evolutionary change is a transformation step from one character state to

another. Character states can be DNA bases, the loss or gain of a restricted site, or the absence or presence of morphological features.

Its advantages are:

- It allows the use of all known evolutionary information in tree building.
- It produces numerous unrooted, "most parsimonious trees".

Its disadvantages are:

- It requires long computation time, although faster than Maximum likelihood.
- It yields little information about branch length.
- It usually performs well with closely related sequences, but often performs badly with very distantly related sequences.

6.1.5 Maximum Likelihood (ML)

The maximum likelihood method evaluates the topologies of different trees and chooses the best one based on a specified model. This model is based on the evolutionary process that can account for the conversion of one sequence into another. It evaluates a hypothesis about evolutionary history in terms of the probability that the proposed model and the hypothesized history would give rise to the observed data set. The topology parameter is branch length. MP starts with a multiple alignment and lists all possible topologies of each data partition (column). It then calculates probability of all possible topologies for each data partition and combines data partitions. It identifies tree with the highest overall probability at all partitions as most likely phylogeny.

It enjoys many advantages:

- It is more accurate than other methods. It is often used to test an existing tree.
- All the sequence information is used.
- Sampling errors have least effect on the method.

It suffers from the following disadvantages:

- It is extremely slow, and thus, impractical for analyzing large data set.

6.2 The Statistical Method for Optical Map Comparison Used by OpGen

To calculate the pairwise map similarity value, OpGen Inc. uses the SOMA map aligner (see appendix A) to find all the local alignments between the two strains above a certain score threshold. Given two optical-maps $mapA$ and $mapB$, the percentage similarity is found by taking $(alignedL_A + alignedL_B)/(L_A + L_B)$, where $alignedL_A$ is the length of aligned restriction fragments of $mapA$, and L_A is the total length of restriction fragments of $mapA$.

After the percentage similarity values are computed, they are fed into a statistical package available in the language "R" and a clustering method selected from the following list: the nearest neighbor, furthest neighbor, or UPGMA. According to OpGen (personal communication) this method performs reasonably well for small data sets (less than 10) in which there is a lot of map similarity variations. As the data set sizes grow, OpGen often found that the clustering method generates groups that are no longer biologically meaningful.

We have performed pairwise alignment between *Escherichia coli O*157 : *H*7 *str. Sakai* and *Escherichia coli O*157 : *H*7 *EDL*933 using SOMA map aligner with its default settings, as shown in Figure 6.4. This pairwise alignment takes $\sim$ 15 min. So in practice the efficiency of SOMA also limits the usage of OpGen's method of inferring phylogeny.

6.3 Problem Formulation and the CAPO Methodology

Instead of using sequence alignment results, we use optical map comparison for pairwise distance measures. The advantages of this approach against traditional distance measures based on sequence alignments are:

- Most of the traditional distance measures for phylogenetics have been developed for comparing genes or proteins, not genomes. They assume linear alignments (e.g., clustalw) where the only differences between sequences are point mutations. They do not handle indels very well and they can not handle translocations or local inversions at all – these regions end up being scored as mismatches when the distance measure is computed. So whole segments of DNA may not contribute to the score, if they are identical but out-of-order. Our method takes whole genome optical map into consideration. Genome evolution events including insertion, deletion, inversion, and translocation all contribute to corresponding map similarity measures.

- Multiple sequence alignment methods can only be used for small number of short genomes because of the complexity and efficiency of those methods.

Since optical map data is much more 'condensed' than sequence data, our approach is very fast and can be easily applied to analyzing large data set.

- There are many different strains in the same species. The genome content could be very different. We cannot sequence every single strain unless DNA sequencing gets really inexpensive and efficient. Our approach is not limited by the availability of sequence data.

The problem of comparing optical maps can be formulated mathematically as follows: an optical map can be viewed as an ordered sequence of "restriction sites", or equivalently, "restriction fragment lengths". A vector of decimal numbers, $H_k = (h_1, h_2, ..., h_m)$, can beis used to represent a single map k, where h_i with index $1 \leq i \leq m$ is the length of the i-th restriction fragment. The size of an optical map k is then defined as $s_k = \sum h_i,\ h_i \in H_k$. The input to CAPO is an $N \times M$ matrix $O = (o_{ij})$, where each row corresponds to an optical map of a strain or a genome. Each column corresponds to a position in that map. N is the total number of maps, and M is the number of restriction fragments in the longest map in that input. Because sequences of different strains or genomes vary in length, the final optical maps usually do not have the same number of restriction fragments. We force them to have M fragments by appending zeros to the end of shorter map vectors. Notice that all the restriction maps in the input must be digested by the same set of restriction endonucleases to make the map comparison meaningful in genome evolution study.

The CAPO algorithm is based on pairwise optical map comparison and bipartite graph matching, combined with standard distance methods of phylogeny tree construction. It consists of two major phases. First, pairwise optical map

comparison is performed to generate a pairwise similarity matrix $S = (s_{ij})$, where s_{ij} is the map similarity between the i-th and j-th map in the input matrix O. S is used as input to the second phase of CAPO, which infers phylogeny among input strains or genomes. Output is in the Phylip format used by many phylogenetic analysis packages. This consists of a series of nested parentheses describing the branching order with the sequence names (the branch lengths are ignored in the current version of CAPO). Users can display the phylogeny tree using the NJPLOT program distributed with the ClustalX package[3]. The two algorithms implemented in CAPO are detailed in the following sections.

6.3.1 Heuristic Algorithm for Pairwise Optical Map Comparison

This algorithm, used in CAPO's phase one, is a heuristic mer-based algorithm for pairwise optical map comparison. A 'mer' (or more elaborately "restriction-fragment-mer") in an optical map is an ordered sequence of restriction fragment lengths. A 'k-mer' is a mer with k fragment lengths. Mathematically, a k-mer comprises k decimal numbers, and their positions reflect the sequence order of the corresponding restriction fragments. After choosing a mer size k, we generate all k-mers in an optical map for both forward and backward orientations. Each k-mer is indexed by its position in the optical map. To compare two optical maps i and j, we examine all common k-mers between them as follows: we count the number of common k-mers as c_{ij}, and compute the pairwise map similarity s_{ij}, $s_{ij} = c_{ij}/(s_i + s_j)$, where s_i and s_j are the sizes of the two optical

[3]The latest version of the ClustalX program is available at ftp://ftp-igbmc.u-strasbg.fr/pub/ClustalX/.

maps. $s_{ij} = 0$ if $i = j$. The obtained pairwise similarity matrix S is used as input to the next phase inferring phylogeny.

Common mers are searched in a manner allowing for sizing errors. For example, given two k-mers, $k_1 = (f_1,\ f_2,\ ...,\ f_k)$ in map 1 and $k_2 = (g_1,\ g_2,\ ...,\ g_k)$ in map 2, we consider k_1 and k_2 as a pair of common k-mers if and only if the following condition is satisfied:

$$\frac{F_i \cap G_i}{F_i \cup G_i} \geq \rho,\ for\ all\ 1 \leq i \leq k.$$

where F_i is interval $(f_i - \sigma_{fi},\ f_i + \sigma_{fi})$, σ_{fi} is the standard deviation for fragment f_i; G_i is interval $(g_i - \sigma_{gi},\ g_i + \sigma_{gi})$, σ_{gi} is the standard deviation for fragment g_i; ρ is a cutoff determining the least overlap degree between two common intervals. The standard deviation of a restriction fragment collection is estimated via observations of experiment data. Details are given in section 6.3.3.

6.3.2 Stable Matching Algorithm for Inferring Phylogeny

Given a matrix of distances among a set of taxa, both the UPGMA and NJ methods are widely used in phylogenetic analysis to show how similar or dissimilar they are. The UPGMA method assumes equal rates of evolution, so that branch tips come out equal. The NJ method allows for unequal rates of evolution, so that branch lengths are proportional to amount of change. CAPO combines the standard stable marriage (SM) algorithm for bipartite graph matching problem with either the UPGMA or the NJ method for inferring phylogeny. The SM problem is introduced in chapter 3.

Usually a phylogenetic tree is constructed in stepwise manner. Every time two most similar sequences are clustered together, they are combined into a new node, representing their least common ancestor. The clustering process

continues until there is only one node left. Therefore, given n taxa, traditional distance-based methods need $O(n)$ iterations to construct a phylogenetic tree. In normal cases, our method is capable of constructing a phylogenetic tree in $log(n)$ iterations, though its worst-case number of iterations is comparable to traditional distance-based methods. It works as follows:

Initialization: Define T to be the set of leaf nodes, one for each given optical map. If the UPGMA method is used, we set the distance matrix $D = (d_{ij}) = (s_{ij})$, where s_{ij} is the map similarity obtained from phase one. If the NJ method is used, we compute $u_i = \sum_{j=1}^{n} s_{ij}/(n-2)$ for each node i in T, where n is the total number of nodes in T. The distance matrix D is recomputed to be $D = (d_{ij}) = (s_{ij} - u_i - u_j)$.

Iteration:

1. **Build a bipartite graph**: we partition D along diagonal line into two parts: the upper triangular part UT and the lower triangular part LT. Pairs in UT form the left column in the bipartite graph, and pairs in LT form the right column. Each node i has a preference list of nodes, ranked by d_{ij}.

2. **Apply the stable marriage algorithm and produce a set X of stable pairs** [52]. Such a 'stable pair' is a pair of nodes connected by the stable marriage algorithm and is clustered into a new internal node if this pair passes the following test in the cleaning step.

3. **Clean the set** X: sort stable pairs in decreasing order of d_{ij} and keep only the first m pairs in X that are disjoint[4] with each other.

[4]Two pairs (a, b) and (c, d) are disjoint with each other if and only if no two nodes in different pairs are the same.

4. **Connect nodes and update the distance matrix D in a loop until X is empty**. In each loop execute the following operations: i) extract the first pair (i, j) in X; ii) join them with a new internal node v_{ij}[5]; iii) compute the distances between node v_{ij} and the remaining nodes k[6]; iv) delete d_{ij} in D and add the new distances to D; and finally, v) connect nodes i and j in T with v_{ij}.

Termination: When only two nodes i and j remain unconnected in T, connect them to the root node of the tree T.

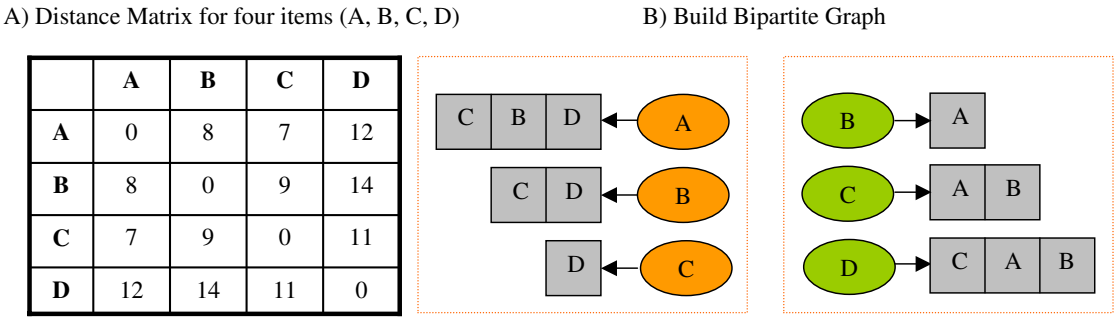


Figure 6.2: An example of building a bipartite graph given a distance matrix. A) A distance matrix M of four items (A, B, C, D). B) The corresponding bipartite graph.

An example of building a bipartite graph given a distance matrix is shown

[5]The node v_{ij} has its cluster size $n_{ij} = n_i + n_j$ (initially, $n_i = 1$).

[6]If the UPGMA method is used, we use

$$d_{(ij),k} = (\frac{n_i}{n_i + n_j})d_{ik} + (\frac{n_j}{n_i + n_j})d_{jk}.$$

If the NJ method is used, we use

$$d_{(ij),k} = \frac{d_{ik} + d_{jk} - d_{ij}}{2}.$$

in Figure 6.2. Each node has a preference list(grey boxes) ordered by distances. Left panel contains pairs in the upper triangular part of M; right panel contains pairs in the lower triangular part of M. For example, the first row in the left panel means "item A prefers to pair with C, B, D, in the decreasing order of preferences".

We can assume that each iteration reduces the number of nodes by a factor of k, where k is a constant. On average this algorithm terminates in $O(logn)$ iterations (n is the number of optical maps in the analyzed data set). In each iteration, building a bipartite graph is the most expensive step, which involves sorting each preference list and it takes $O(m^2logm)$ (m is the number of nodes left in current iteration). After i-th iteration m becomes $\frac{n}{k^i}$. So the complexity of this algorithm is as follows:

$$\begin{aligned}
T &\approx \sum_{i=0}^{logn}(\frac{n}{k^i})^2 log(\frac{n}{k^i}) \\
&= \sum_{i=0}^{logn}(\frac{n^2}{k^{2i}})(logn - ilogk) \\
&\leq \sum_{i=0}^{logn}\frac{n^2 logn}{k^{2i}} \\
&= n^2 logn \sum_{i=0}^{logn}\frac{1}{k^{2i}} \\
&= n^2 logn(\frac{1-(\frac{1}{k^2})^{logn}}{1-\frac{1}{k^2}}) \\
&= n^2 logn(O(1)) \\
&= O(n^2 logn)
\end{aligned}$$

Comparatively, a standard distance-based method for building a phylogenetic tree terminates in n iterations. In each iteration, the major time consuming step is to look for the smallest distance in a $m \times m$ distance matrix (m is the number of clusters left in current iteration), which takes $O(m^2)$. So the

complexity of a standard distance-based method is as follows:

$$T' \approx (\sum_{m=n}^{1} m^2) = O(n^3)$$

Therefore, in normal cases our stable marriage algorithm for inferring phylogeny is more efficient than a standard distance-based method. Although in worst case scenario our algorithm takes $O(n^3 logn)$, but this case should be very rare.

6.3.3 Correction of Sizing Errors

Optical maps of different strains of the same species would vary due to single nucleotide differences (SNPs), small insertions and deletions (RFLPs) as well as many genomic rearrangement events that leave their footprints on restriction site patterns. Further variations are introduced by the experimental process. These can be due to: sizing errors, partial digestion, short missing restriction fragments, false cuts, ambiguities in the orientation, optical chimerisms, and so on [3, 40]. We can classify these error factors introduced by the experimental process into three types — sizing errors, digestion errors, and orientation errors.

The sizing error statistics is estimated from observations of experiments done by OpGen, Inc. These observations (including fragment lengths and standard deviations) are what are reported in the output from the GENTIG [2] software that OpGen used to produces optical maps. A first-degree polynomial fit for the three pairs of variables: $L \sim stddev(L)$, $sqrt(L) \sim stddev(L)$, and $1/sqrt(L) \sim stddev(L)/L$ is shown in Figure 6.3, where linear correlation coefficient is referred as *cc*. No apparent linear relation is observed between any pair of them since none of these pairs have linear correlation coefficient close

enough to one (> 0.95). These results indicate that it may not be appropriate to estimate standard deviations using any of these 'linear relations' (results not shown). Therefore data interpolation is used instead to estimate standard deviations $stddev(L)$ for a restriction fragment whose length is L. This data interpolation step is performed in the following way: given a fragment length L, find L_l and L_r from the error plot shown in Figure 6.3-(a) where L_l and L_r are the closest left neighbor and right neighbor of L, respectively ($L_l < L < L_r$); compute $stddev(L)$ using $stddev(L) = (stddev(L_l) + stddev(L_r))/2$.

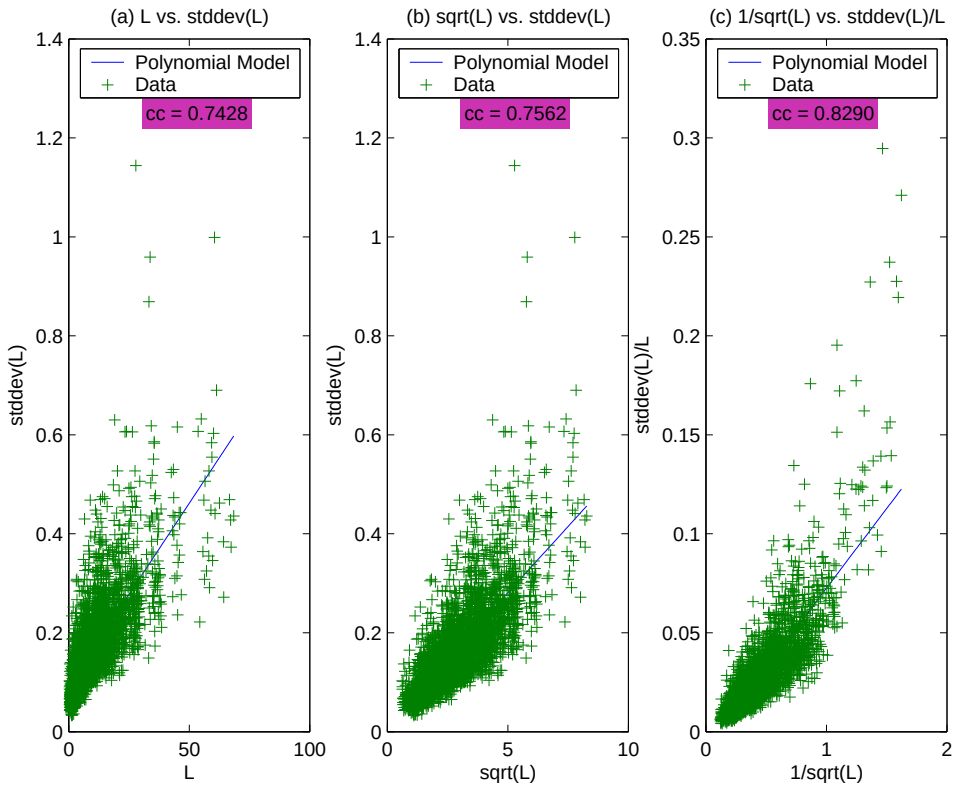


Figure 6.3: First-degree polynomial fit for restriction fragment sizing error. (a) L vs. $stddev(L)$, $cc = 0.7428$; (b) $\sqrt{L}$ vs. $stddev(L)$, $cc = 0.7562$; (c) $1/\sqrt{L}$ vs. $stddev(L)/L$, $cc = 0.8290$.

6.4 Phylogenetic Tree Comparison Measure

A tree comparison measure is a measure of the similarity between two trees, say $T1$ and $T2$. There are two basic kinds: The first one counts the minimum number of operations required to transform $T1$ into $T2$ using some transforming method (e.g. the nearest neighbor interchange (NNI) metric [55]); The second one represents the two trees as sets of simpler structures (such as clusters or quartets) and then uses various measures of similarity between sets (e.g. Estabrook et al's quartet measures [16]). These two categories of tree comparison measure are not mutually exclusive (e.g. the partition metric [44]).

The partition metric [44] is used to compare phylogenetic trees in testing CAPO. In this method, the difference between two trees (referred as dT score) is defined as the number of edges for which there is no equivalent edge on the other tree. A dT score between two binary unrooted trees with n leaves is in the interval 0 (for isomorphic trees) and $2(n-3)$ (if all leaves are placed differently).

The reasons of choosing this method are:

- It treats trees as sets of clusters, which is how most biologists interpret trees.
- The partition metric is easy and fast to compute.
- It is widely implemented (in PAUP 3.0 and COMPONENT 1.5).

This method also has a disadvantageous feature: Two trees differing solely in the position of one taxon can be maximally different [44].

6.5 Material

6.5.1 Data set I

These data were obtained from collaborators at OpGen, Inc. The data contained eleven optical maps constructed commercially by OpGen[7] for varying *E. coli* strains. Information describing this data set is listed in Table 6.1. All the organisms in data set I are *E. coli* bacteria, and are identified by their individual strain names. Sequence data are not available for most but four of these *E. coli* strains, including $Escherichia\ coli\ CFT073$, $Escherichia\ coli\ K12$, $Escherichia\ coli\ O157:H7\ str.\ Sakai$, and $Escherichia\ coli\ O157:H7\ EDL933$.

OpGen uses the following procedure to produce this data: i) purified chromosomal DNA is deposited onto an optical mapping surface using a microfluidic device; ii) the DNA is encased in a thin layer of acrylamide and incubated with the restriction enzyme BamHI (it cleaves only at sites containing the 6 bp long sequence GGATCC) in a humidified chamber at $37^\circ\ C$ for $60 \sim 120$ mins; iii) the digested DNA is labeled with fluorescent YOYO-1 and the individual molecules are imaged with fluorescence microscopy; iv) digital images are collected by an automated image-acquisition system and image files are processed to create single-molecule optical maps; v) individual molecule restriction maps are overlapped by using GENTIG (GENomic conTIG) map-assembly software. Briefly, GENTIG [2, 40] works by comparing single-molecule restriction maps and estimating the probability that these two molecules arose from overlapping genomic locations given a description of the likelihood of possible experimental errors resulting from incomplete digestion, spurious cuts, and sizing errors.

[7]Website of OpGen Inc. is http://www.opgen.com/

Through repeated overlapping of molecules, the assembler reconstructs the ordered restriction map of the genome. This technique has been previously applied to map other bacterial genomes [33, 35, 36].

Species	Genome Refseq ID	Length(no. of nt.)
*Escherichia coli CFT*073	NC_004431	5,231,428
*Escherichia coli K*12	NC_000913	4,639,675
*Escherichia coli O*157 : *H*7 *str. Sakai*	NC_002695	5,498,450
*Escherichia coli O*157 : *H*7 *EDL*933	NC_002655	5,528,445
*EC*1231	NA	NA
400	NA	NA
536	NA	NA
*AB*1	NA	NA
*DEC*5*A*	NA	NA
503	NA	NA
886	NA	NA

Table 6.1: Data Set I: 11 *Escherichia Coli* Strains

OpGen Inc. developed an interface for viewing optical-maps, called MapViewer. MapViewer allows users to visualize optical-maps, to move maps around, pull up sequence information when available, and change the orientation of the maps. Figure 6.4 shows the optical maps for data set I using MapViewer. A pairwise alignment between *Escherichia coli O*157 : *H*7 *str. Sakai* and

*Escherichia coli O*157 : *H*7 *EDL*933 is shown. Regions that match exactly once are colored green, and regions that match to more than one locations are colored red.

6.5.2 Data set II

28 genomic sequences of Enterobacteriaceae taxa are downloaded from the NCBI database, and then cleaved "in silico" with the restriction enzyme BamHI. Their optical maps were constructed using the *SilicoMap* software provided by OpGen[8]. Information describing this data set is listed in Table 6.2. Figure 6.5 shows the optical maps for data set I using MapViewer.

6.6 CAPO Experiments and Discussion

Experimental results are provided in this section using CAPO on both real optical mapping data of 11 *E. coli* strains and simulated optical mapping data of 28 entire genomes of *Enterobacteriaceae* taxa. All of the tests were ran on a 2.4-GHz Pentium IV machine with 3GB of RAM.

6.6.1 Parameter Optimization

Users have choices for two parameters in CAPO: k (mersize) and ρ (cutoff value involved in determining whether two restriction fragment lengths are 'equal' considering sizing errors). The effect of parameter settings in CAPO is tested in the following experiments using the two data sets: $k = 2$, $\rho = 0.9$ (see

[8]The SilicoMap tool is built upon the BioPerl [51] toolkit which is able to perform an in silico restriction digest, after which, it is straightforward to find the lengths of each of the resulting fragments and create the map.

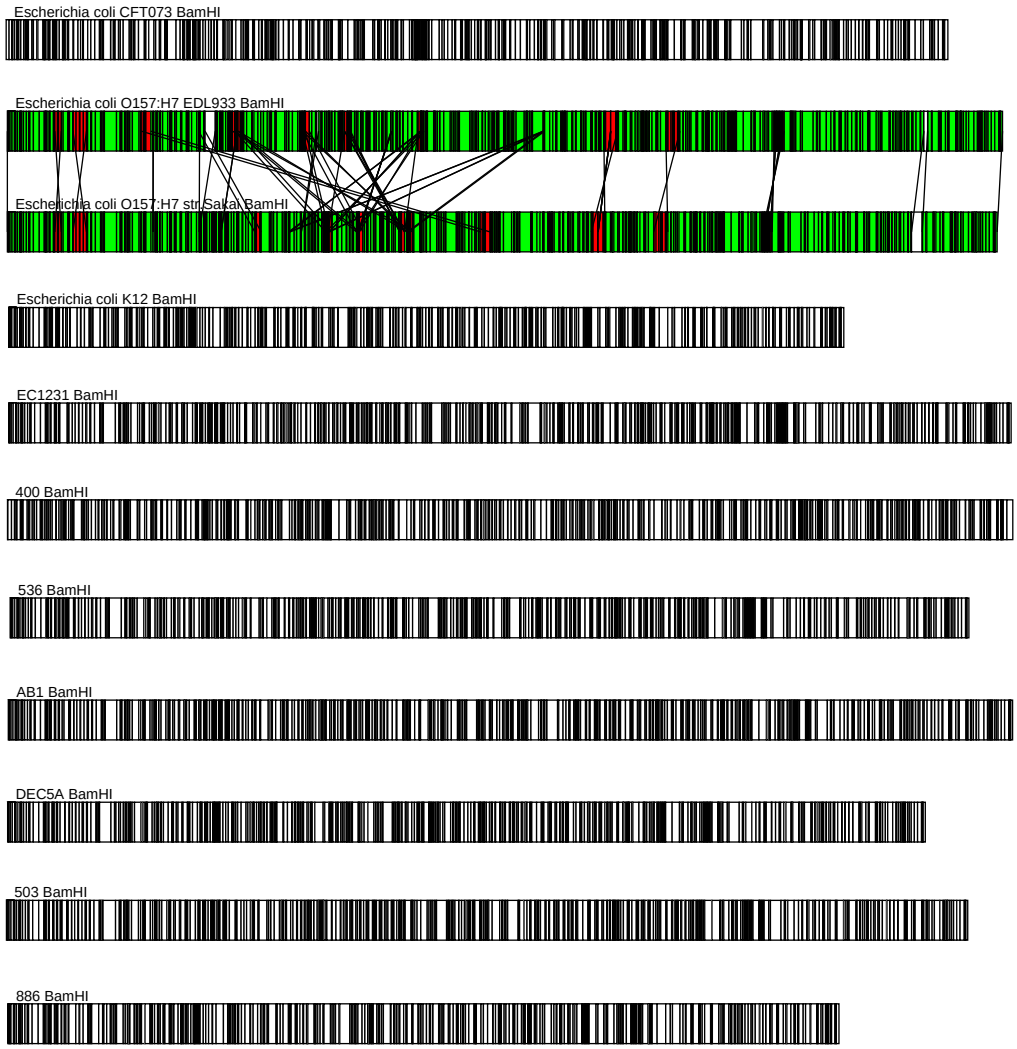


Figure 6.4: View maps of Data set I using MapViewer. A pairwise alignment between *Escherichia coli O*157 : *H*7 *str. Sakai* and *Escherichia coli O*157 : *H*7 *EDL*933 is shown.

Species	Refseq ID	Length
Buchnera aphidicola str. APS(Acyrthosiphonpisum)	NC_002528	640,681
Buchnera aphidicola str. Sg(Schizaphisgraminum)	NC_004061	641,454
Buchnera aphidicola str. Bp(Baizongiapistaciae)	NC_004545	615,980
Candidatus Blochmannia floridanus	NC_005061	705,557
Candidatus Blochmannia pennsylvanicus str. BPEN	NC_007292	791,654
*Erwinia carotovora subsp. atroseptica SCRI*1043	NC_004547	5,064,019
*Escherichia coli CFT*073	NC_004431	5,231,428
*Escherichia coli K*12	NC_000913	4,639,675
*Escherichia coli O*157 : *H*7 *str. Sakai*	NC_002695	5,498,450
*Escherichia coli O*157 : *H*7 *EDL*933	NC_002655	5,528,445
*Escherichia coli UTI*89	NC_007946	5,065,741
*Escherichia coli W*3110 *DNA*	AC_000091	4,646,332
*Photorhabdus luminescens subsp. laumondii TTO*1	NC_005126	5,688,987
*Salmonella typhimurium LT*2	NC_003197	4,857,432
*Salmonella enterica subsp. enterica serovar Typhi Ty*2	NC_004631	4,791,961
*Salmonella enterica subsp. enterica serovar Typhi str. CT*18	NC_003198	4,809,037
Salmonella enterica subsp. enterica serovar Paratyphi A str. ATCC 9150	NC_006511	4,585,229
Salmonella enterica subsp. enterica serovar Choleraesuis str. SC − *B*67	NC_006905	4,755,700
Shigella flexneri 2*a str.* 301	NC_004337	4,607,203
*Shigella boydii Sb*227	NC_007613	4,519,823
*Shigella sonnei Ss*046	NC_007384	4,825,265
*Shigella dysenteriae Sd*197	NC_007606	4,369,232
Sodalis glossinidius str. $'morsitans'$	NC_007712	4,171,146
Wigglesworthia glossinidia endosymbiont of Glossinabrevipalpis	NC_004344	697,724
*Yersinia pestis CO*92	NC_003143	4,653,728
Yersinia pestis biovar Medievalis str. 91001	NC_005810	4,595,065
Yersinia pestis KIM	NC_004088	4,600,755
Yersinia pseudotuberculosis IP 32953	NC_006155	4,744,671

Table 6.2: Data Set II: 28 *Enterobacteriaceae* Taxa

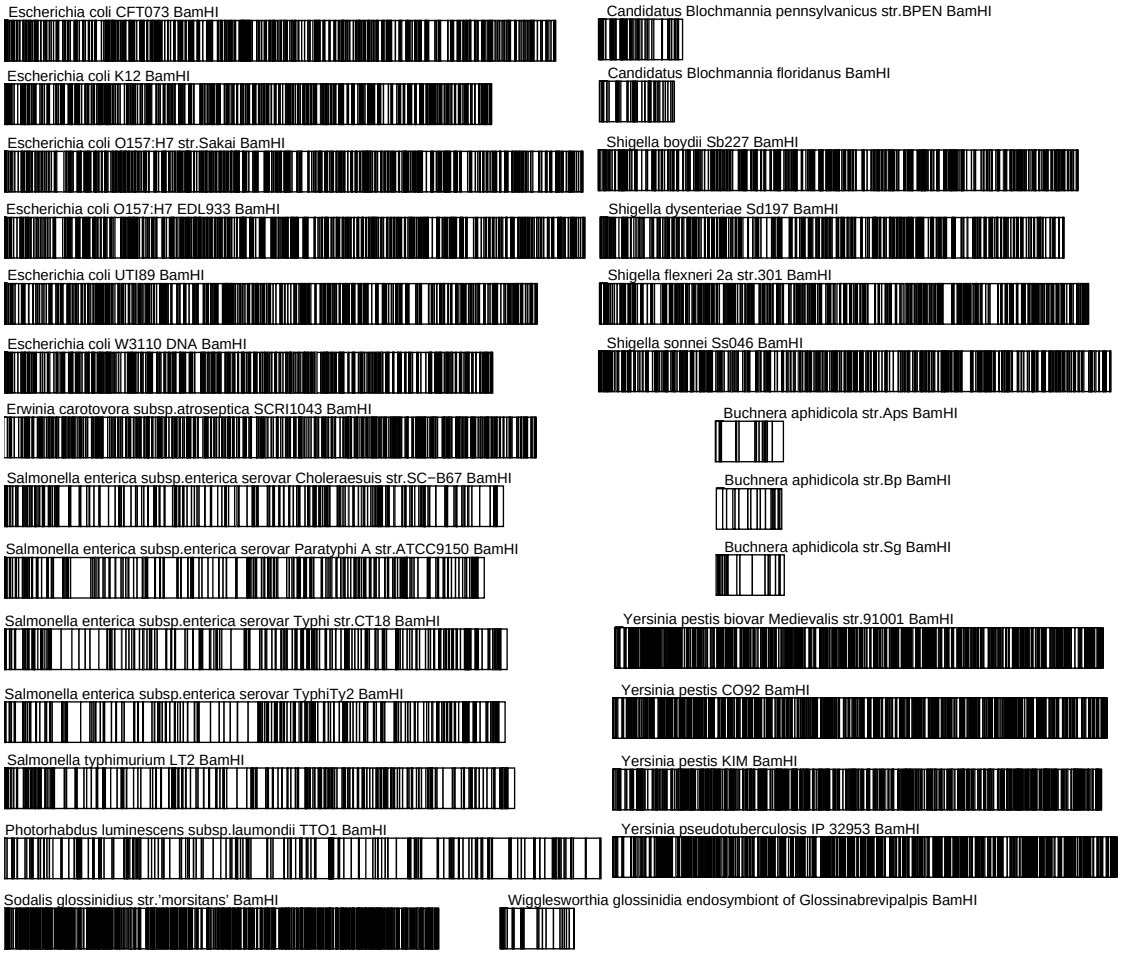


Figure 6.5: View maps in Data set II using MapViewer

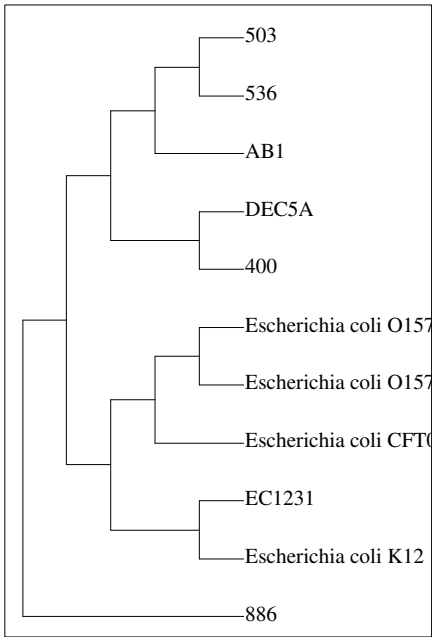
Parameter Setting a	$k = 2$, $\rho = 0.9$	multi-merge mode
Parameter Setting b	$k = 3$, $\rho = 0.8$	multi-merge mode
Parameter Setting c	$k = 4$, $\rho = 0.7$	multi-merge mode
Parameter Setting d	$k = 3$, $\rho = 0.8$	single-merge mode

Table 6.3: Experimented parameter settings

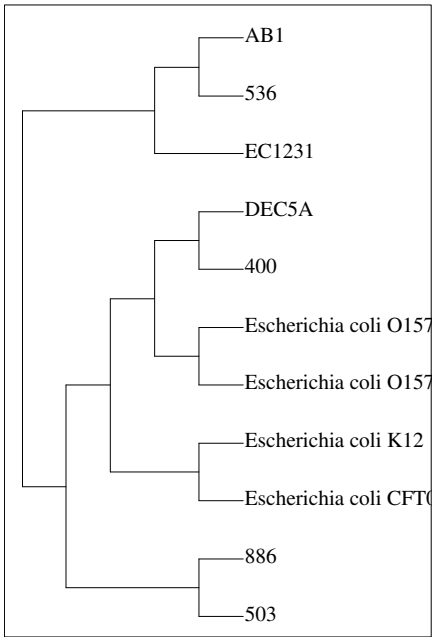
Figure 6.6); $k = 3$, $\rho = 0.8$ (see Figure 6.7); $k = 4$, $\rho = 0.7$ (see Figure 6.8) [9] . To tolerate sizing errors more flexibly it is reasonable to use smaller cutoff value of ρ if a larger mersize is chosen.

Shown in Figure 6.6-Figure 6.8, the 'best' results (whose phylogenetic trees are most biologically meaningful) are produced using $k = 3$, $\rho = 0.8$. Table 6.4 lists the dT measure for the two data sets using either SM-UPGMA or SM-NJ with different parameter settings (see Table 6.3). For data set I the possible maximum value of dT score is 16; For data set II the possible maximum value of dT score is 50. It is shown in Table 6.4 that setting b produces the most 'reliable' trees (Trees produced using setting b differ less to trees using setting a and c; trees produced using setting a differ more to trees using setting c.). Based on these two observations we choose setting b ($k = 3$, $\rho = 0.8$) as the default parameter setting.

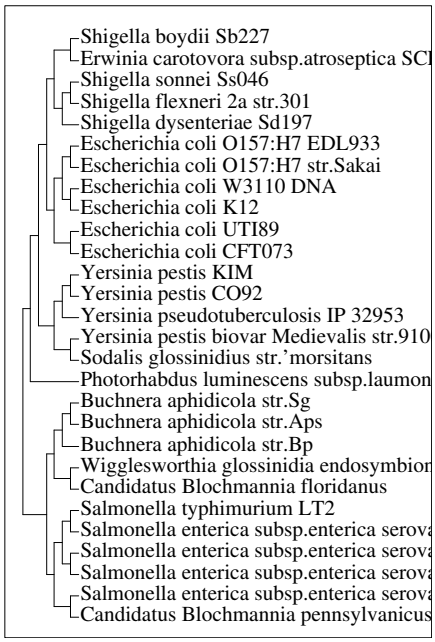
[9] A phylogenetic tree produced by CAPO using the stable marriage algorithm combined with the UPGMA method is referred as 'SM-UPGMA tree'. A phylogenetic tree produced by CAPO using the stable marriage algorithm combined with the NJ method is referred as 'SM-NJ tree'.



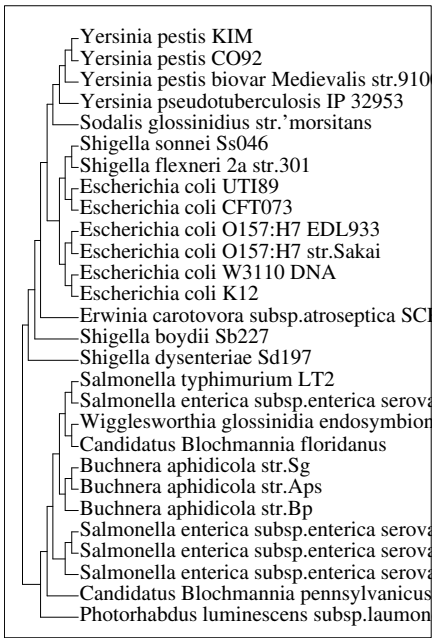
A1: SM-UPGMA tree for data set I



A2: SM-NJ tree for data set I

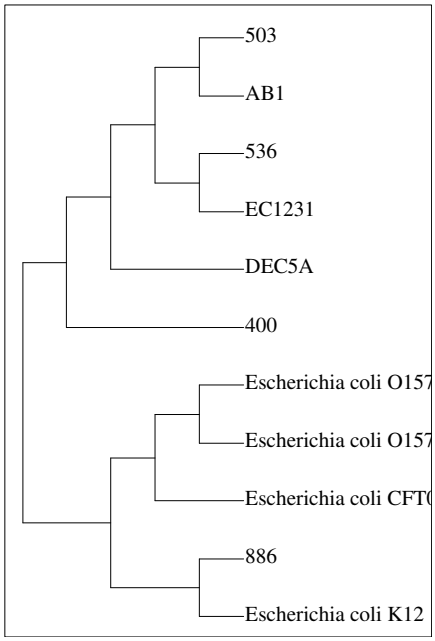


B1: SM-UPGMA tree for data set II

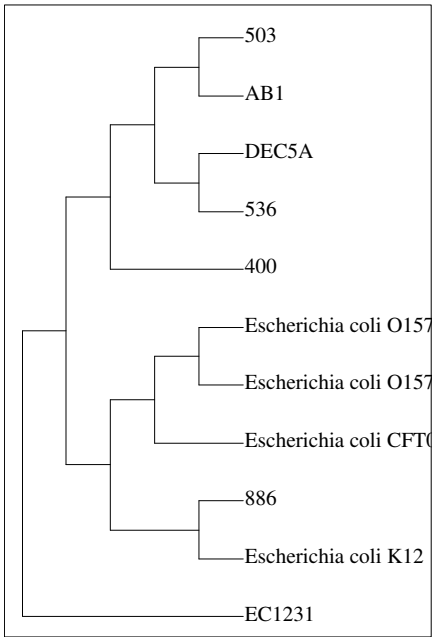


B2: SM-NJ tree for data set II

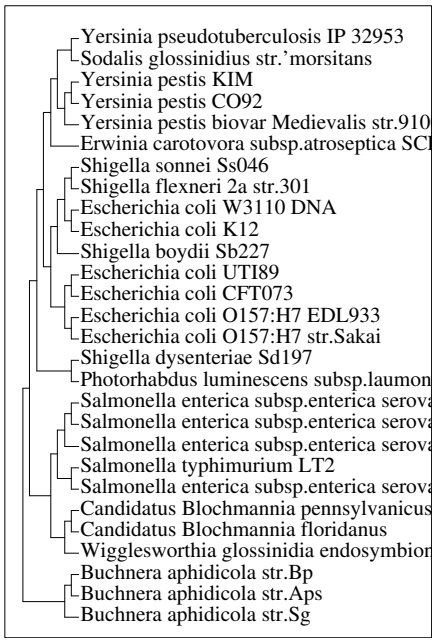
Figure 6.6: Phylogenetic tree for data set I and II ($k = 2$, $\rho = 0.9$)



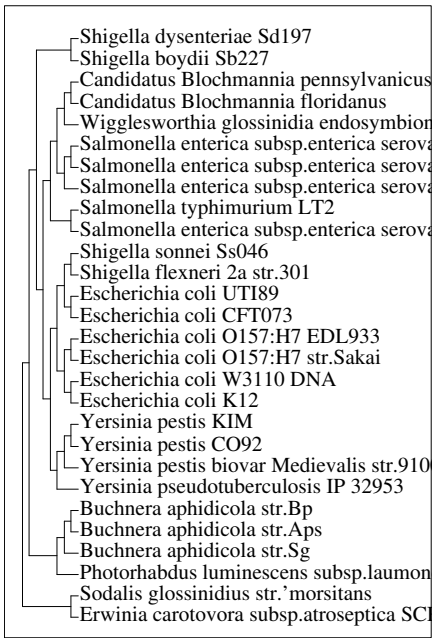
A1: SM-UPGMA tree for data set I



A2: SM-NJ tree for data set I

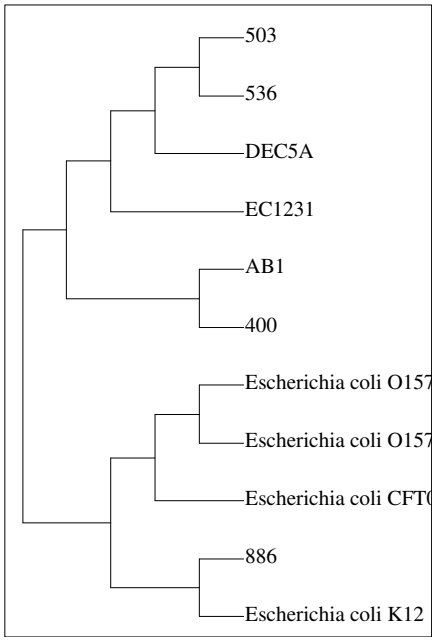


B1: SM-UPGMA tree for data set II

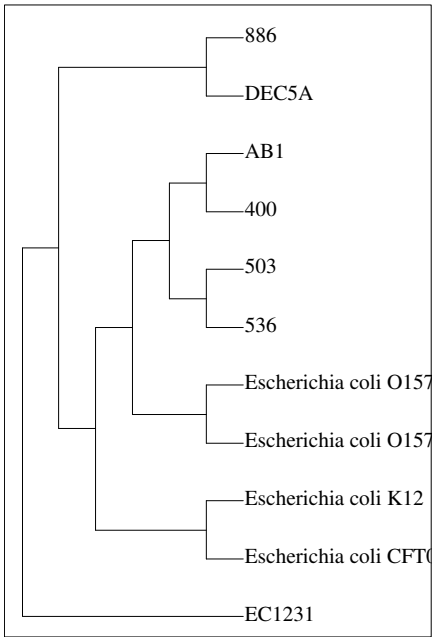


B2: SM-NJ tree for data set II

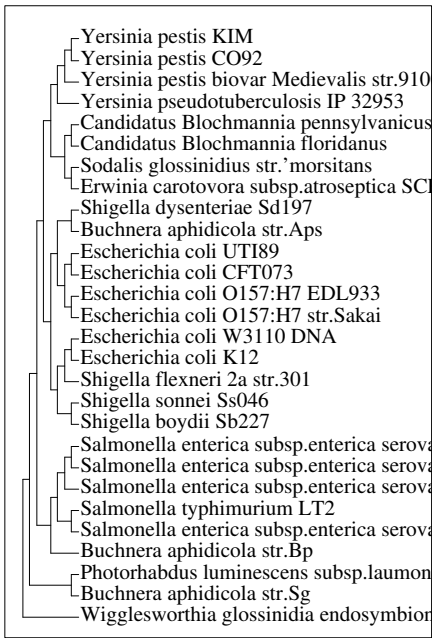
Figure 6.7: Phylogenetic tree for data set I and II ($k = 3$, $\rho = 0.8$)



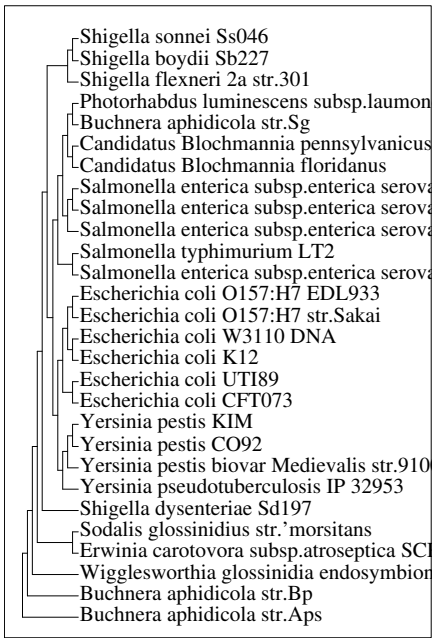
A1: SM-UPGMA tree for data set I



A2: SM-NJ tree for data set I



B1: SM-UPGMA tree for data set II



B2: SM-NJ tree for data set II

Figure 6.8: Phylogenetic tree for data set I and II ($k = 4$, $\rho = 0.7$)

Data Set I			
Method	Settings		dT score
SM-UPGMA	a	b	14
	b	c	8
	a	c	12
	b	d	6
SM-NJ	a	b	16
	b	c	14
	a	c	14
	b	d	12
Data Set II			
Method	Settings		dT score
SM-UPGMA	a	b	32
	b	c	28
	a	c	40
	b	d	20
SM-NJ	a	b	24
	b	c	14
	a	c	32
	b	d	28

Table 6.4: Tree comparison measure by the partition metric using different parameter settings

6.6.2 Phylogenetic Tree Evaluation

Since there are no 'true' phylogenetic trees available for comparison with our results, we evaluate the quality of these trees based on optical map alignments, the taxonomy information given by the NCBI database, and tree topology overlap between the two distance methods measured by the partition metric [44]. For example, using the SOMA map aligner developed by OpGen, we find that the map of *Escherichia coli K*12 is very similar to that of 886, and these two strains are clustered closely by CAPO with default setting (see Figure 6.7-A1,A2). CAPO also assigns the rest of three known *E. coli* strains close evolutionary distances. Using data set II, we observe that CAPO clusters biologically closely related taxa together (the *Buchnera aphidicola* strains, the *Candidatus Blochmannia* strains, the *E. coli* strains, the *Salmonella* strains, etc.), as would be desired. Lastly, phylogenetic trees produced by CAPO for the same data set using different distance methods share substantial tree topology overlap, shown in Table 6.5 (Using default setting b, the dT score for data set I is 8, and the dT score for data set II is 24) [10].

6.6.3 Impact of Single-Merge Mode and Multi-Merge Mode

One might wonder whether there is any effect on the phylogenetic tree topology by merging more than two clusters at one iteration. To address this concern, we generated phylogenetic trees for both data sets using 'single-merge mode' (merge exactly two clusters at one iteration), as shown in Figure 6.9. Compared with trees produced in 'multi-merge mode' (merge multiple pairs of disjoint clusters

[10]Setting a, b, and c have the same meaning as in Talbe 6.4.

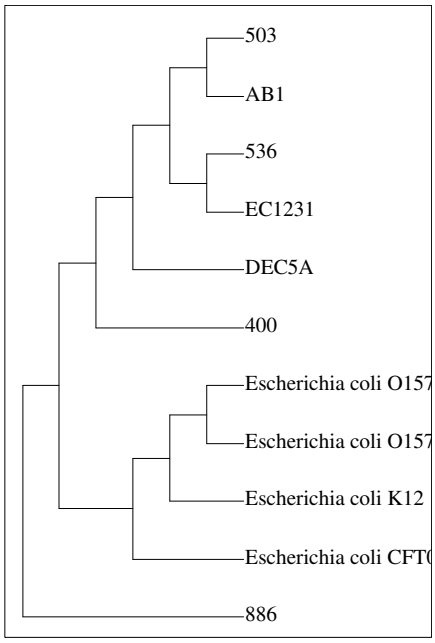
Data Set I			
Methods		Setting	dT score
SM-UPGMA	SM-NJ	a	14
		b	8
		c	12
Data Set II			
Methods		Setting	dT score
SM-UPGMA	SM-NJ	a	24
		b	24
		c	26

Table 6.5: Tree comparison measure by the partition metric for the same data set using different distance methods

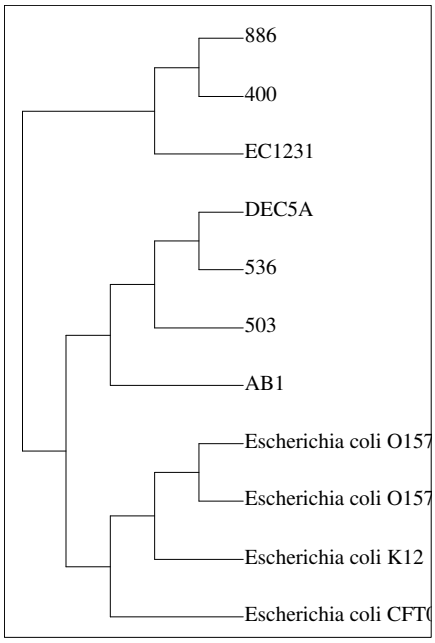
found by the stable marriage procedure at one iteration), as shown in Figure 6.7, we do see some tree topology changes, especially between Figure 6.7-A2 and Figure 6.9-A2. But most corresponding trees share substantial tree topology overlap, as shown by the partition measure listed in Table 6.4.

6.6.4 Experiments with OpGen's Definition of Pairwise Map Similarity

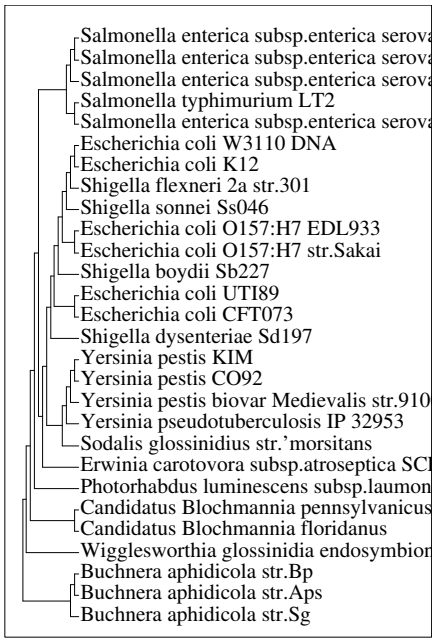
OpGen's experiments are shown in Figure 6.10. Apparently we define pairwise map similarity differently from what OpGen does. So we also experimented OpGen's definition of pairwise map similarity in CAPO, as shown in Figure 6.11. We observe a $> 90\%$ tree overlap between trees produced by using OpGen's method and trees produced by CAPO with the single-merge mode. More di-



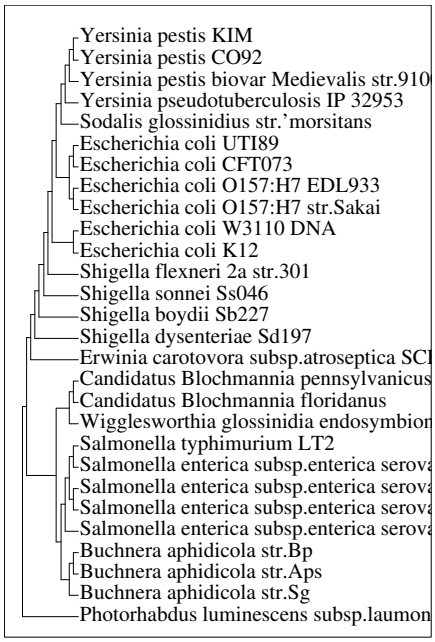
A1: SM-UPGMA tree for data set I



A2: SM-NJ tree for data set I

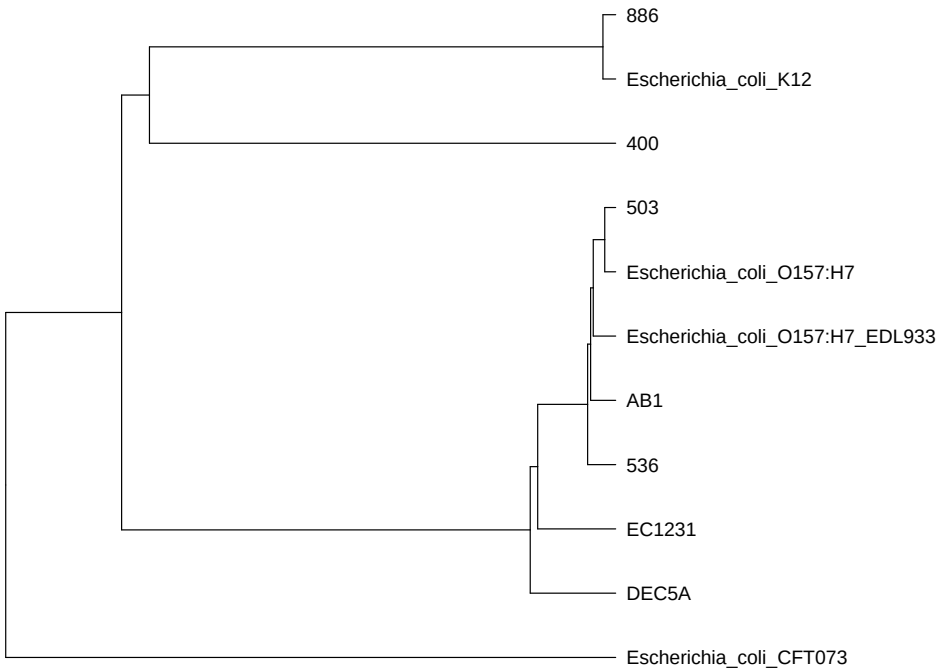


B1: SM-UPGMA tree for data set II

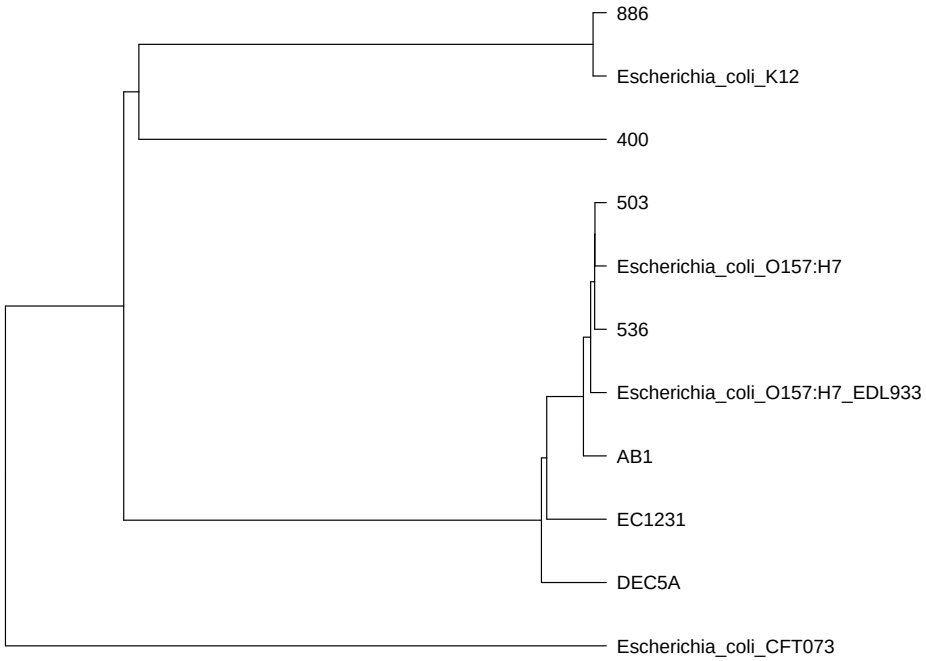


B2: SM-NJ tree for data set II

Figure 6.9: Phylogenetic trees constructed by CAPO for data set I and II using default setting and single merge mode.



A: UPGMA tree



B: Nearest Neighbor Tree

Figure 6.10: Phylogenetic trees generated by OpGen for data set I

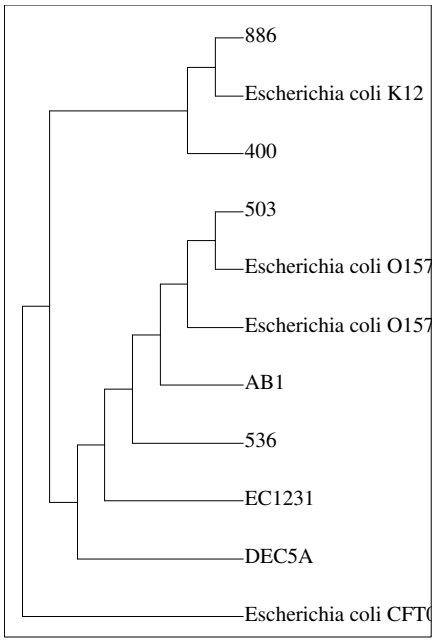
vergency appears between trees using OpGen's method and trees using CAPO with the multi-merge mode. This divergency is due to the impact of single-merge mode and multi-merge mode which need to be studied further in the future.

6.6.5 Cluster Sizes

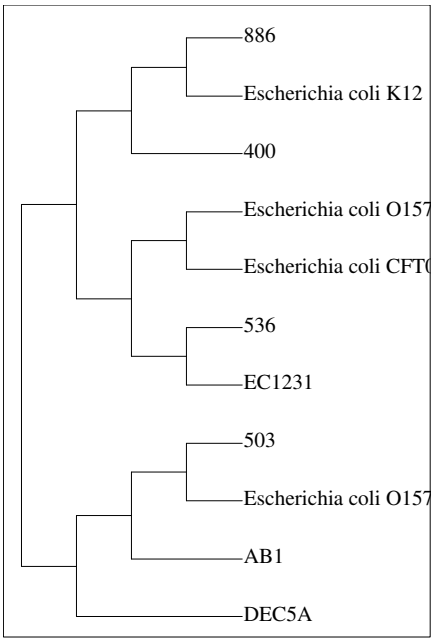
CAPO constructs phylogenetic trees in much fewer iterations than standard distance methods. For data set I, CAPO SM-UPGMA trees and SM-NJ trees are constructed in 5 and 6 iterations, respectively. For data set II, CAPO SM-UPGMA trees and SM-NJ trees are constructed in 8 and 9 iterations, respectively. Number of remaining clusters in each iteration is shown in Figure 6.12.

6.6.6 Implementation and Speed

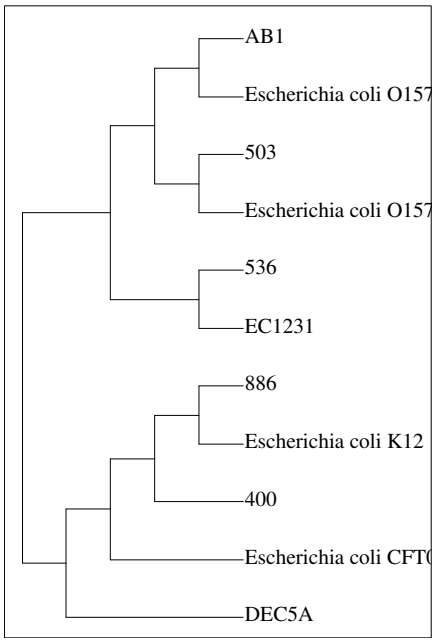
The core module of CAPO is implemented in C++ and all experiments were performed on a Pentium IV PC with 3 GB memory. Experiments for data set I and II took $\sim$ 4 sec. and $\sim$ 18 sec., respectively. The speed of CAPO indicates its potential for wide usage in analyzing large phylogenomic data set.



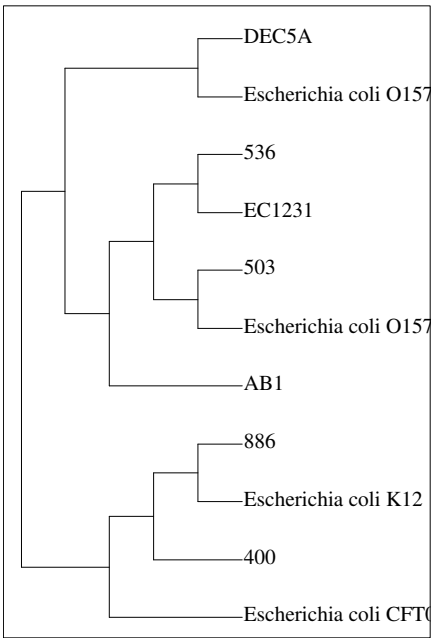
A1: UPGMA tree with single-merge mode



A2: UPGMA tree with mulit-merge mode



B1: NJ tree with single-merge mode



B2: NJ tree with mulit-merge mode

Figure 6.11: Phylogenetic trees generated by CAPO for data set I using OpGen's definition of pairwise map similarity

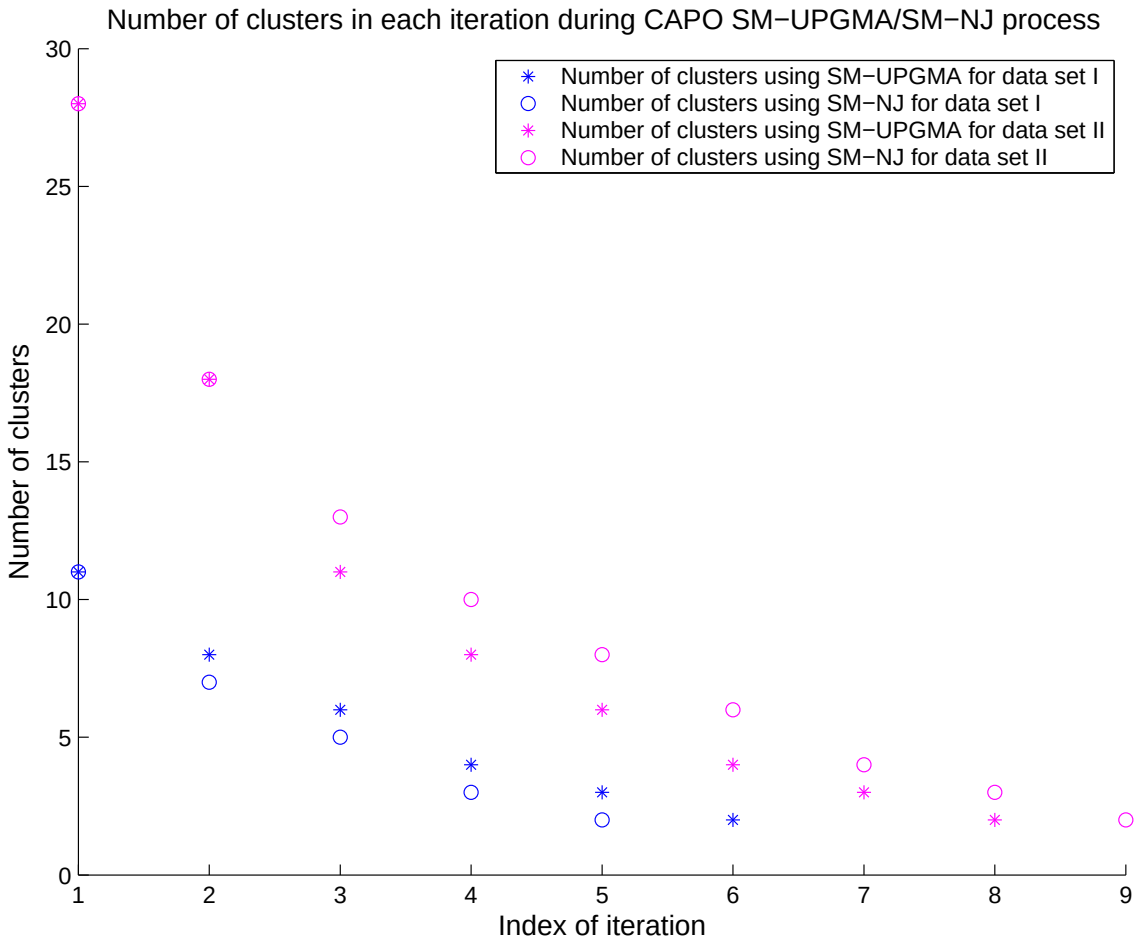


Figure 6.12: Number of clusters in the iterations of the experiments of data set I and II using CAPO SM-UPGMA/SM-NJ.

Chapter 7

Conclusions and Future Work

This dissertation presents two efficient comparative analysis tools for use in comparative analysis based on sequence and on optical-map data, respectively. These two tools handle different biological problems, but share substantial similarity in their algorithm designs.

7.1 The COMBAT Tool for Pairwise Genome Comparison

7.1.1 Summary

The annotation of whole-genome sequences to indicate their functional elements is clearly one of the most important and difficult challenges facing the biosciences community. The strategy of using cross-species DNA comparisons for identifying functionally important sequences is a powerful approach. To get adequate speed when performing whole genome comparison many high-speed alignment programs incorporate a fast search stage that uses a heuristic to identify regions

likely to be homologous. Providing a way of indexing sequences is a key to an efficient search stage. COMBAT indexes both genomic sequences. By using an index of intervals instead of genomic positions, we have been able to reduce by J-fold the size of the index for a vertebrate genome, and make it practical to run whole genome comparison on a single CPU machine. We have shown that COMBAT is capable of rapidly finding matching regions across vertebrate species working in translated mode. Detailed alignments can then be retrieved by using standard alignment algorithms [Smith-Waterman,1970; Needleman-Wunsch,1981]. So, the complex large-scale genome comparison problem is simplified by using COMBAT. We also solve the problem of finding a one-to-one mapping in a multiple mapping list by using the stable marriage algorithm.

7.1.2 Future Work

The following issues related to COMBAT remains to be explored in the near future:

- Try other mer generation methods (some are suggested in chapter 4-4.1.3) in a manner reflecting different genome similarity level.
- Compare COMBAT to other global alignment methods to quantify COMBAT's ability to detect translocated conserved elements.
- Use nucleotide sequences and other databases (including database of Expressed Sequence Tags, databases of regulatory sites, databases of *RNAi*, etc.) than protein databases to extend COMBAT's ability to identify conserved non-coding regions.

- Evaluate COMBAT using simulated sequence data for artificial species with known evolutionary rates.
- Explore the use of COMBAT for multiple genome comparison.

7.2 The CAPO Tool for Comparative Analysis and Phylogeny with Optical-maps

7.2.1 Summary

Optical-maps provide a window through which one may explore the machinery of genome evolution. CAPO tool is a novel, inexpensive and potentially rapid method for inferring phylogeny based on pairwise optical map comparison and bipartite graph matching. CAPO combines the stable matching algorithm with either Unweighted Pair Group Method with Arithmetic Averaging (UPGMA) or the Neighbor-Joining method (NJ) for constructing phylogenetic trees. This new algorithm constructs phylogenetic trees in logarithmical steps in average cases. Using optical maps constructed in silico and in vivo, we have demonstrated many desirable properties of CAPO, and in particular, shown that the UPGMA-flavored trees and the NJ-flavored trees it produces share substantial overlapping tree topology and are biologically meaningful.

7.2.2 Future Work

The following issues related to CAPO needs to be explored in the future:

- Compute branch lengths in CAPO, and find some reasonable way of solving negative branch length problem.

- Further study the impact of the single-merge mode and multi-merge mode on CAPO's performance.
- Further study the performance of CAPO using various tree comparison methods.
- Study the bias of the CAPO method by using simulated optical-map data for artificial species with known evolutionary rates.

Appendix: The SOMA Tool for Aligning Optical Maps

SOMA is a program for aligning optical maps and was developed at OpGen Inc. Map alignments between two different strains are generated using a dynamic programming algorithm. This method finds the optimal alignment of two restriction maps according to a scoring model that allows for fragment sizing errors, false and missing cuts, and missing small fragments. For a given alignment, the score is proportional to the log of the length of the alignment, penalized by the differences between the two maps. More details are given in the following description of the error model it uses:

1) Error Modeling

Optical maps may be corrupted by the following two primary kinds of errors: sizing error and missing fragment error. Sizing errors can be modeled as follows:

$$stddev(L) = C * sqrt(L)$$

with $C \sim 0.05 - 0.1$ and L in Kbp. For example, given a fragment of length $L = 10$ Kbp and assuming $C = 0.1$, the standard deviation of the measurement error is 320 bp or a relative standard deviation of about 3.2%.

Another significant kind of error is due to limited measurement resolution: cut sites that are "too close" together are often missed. That is, an in silico map may have two cut sites that are close, and so appear as one cut site in the optical consensus map. This process is modeled using a simple exponential probability:

$$P(missing\ a\ cut\ of\ length\ d) = exp(-ln(2) * d/m)$$

where d is the distance from the missing cut site to the nearest measured cut site and m is the distance at which half of the cut sites are missing. In the case of an optical consensus map, m is about 1 Kbp. Therefore, there is a 71% chance of missing a cut that is 500 bp away from a detected cut and a 25% chance of missing a cut that is 2 Kbp away from a detected cut.

2) Matching Method

SOMA uses a restriction matching algorithm that is similar to Smith-Waterman. The scoring function is of the form:

$$s = size(L1, L2) - cut(m1, m2)$$

where $L1$ is the length of map 1, $L2$ is the length of map 2, $m1$ is the number of mis-matched cuts on map 1, and $m2$ is the number of mis-matched cuts on map 2. The size function is a linear function in $|L1 - L2|$:

$$size(L1, L2) = 1 - |L1 - L2|/budget$$

where *budget* is the "error budget", or how much relative error we will allow before we say two fragments do not match (i.e., have a negative score). The error budget is calculated based on the allowable discrepancy, measured as the

number of standard deviations, treated as "matching". So, if one allows two standard deviations of error for a match, then the error budget will be about 6.4% for a 10 Kbp fragment. The cut penalty function assigns a fixed cut penalty (currently 1.5) to the number of mis-aligned cuts. However, the mis-aligned cuts are weighted according to the missing fragment model described above. That is, if it is "far" from the nearest matching cut site, then that mis-matched cut is assigned a weight of one, $1 - P(d)$. The weight goes to zero as the distance to the nearest cut site goes to zero, $1 - P(d) \to 0$ as $d \to 0$.

Bibliography

[1] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman. Gapped blast and psi-blast—a new generation of protein database search programs. *Nucleic Acids Res.*, 25:3389–3402, 1997.

[2] T. Anantharaman, B. Mishra, and D. Schwartz. Genomics via optical mapping iii: Contiging genomic dna and variations.

[3] T. Anantharaman, B. Mishra, and D. Schwartz. Genomics via optical mapping ii: Ordered restriction maps. *Journal of Computational Biology*, 4(2):91–118, 1997.

[4] T. Anantharaman, V. Mysore, and B. Mishra. Fast and cheap genome wide haplotype construction via optical mapping. volume 10, pages 385–396. Pacific Symposium on Biocomputing, 2005.

[5] C. Aston, B. Mishra, and D. Schwartz. Optical mapping and its potential for large-scale sequencing projects. *Trends in Biotechnology*, 17:297–302, 1999.

[6] S. Batzoglou, L. Pachter, J. Mesirov, B. Berger, and E. Lander. Human and mouse gene structure: Comparative analysis and application to exon prediction. *Genome Res.*, 10:950–958, 2000.

[7] E. Birney and R. Durbin. Using genewise in the drosophila annotation experiment. *Genome Res.*, 10:547–548, 2000.

[8] E. Birney and et al. Ensembl. *Nucleic Acids Res.*, 32:468–470, 2004.

[9] N. Bray, I. Dubchak, and L. Pachter. Avid: A global alignment program. *Genome Res.*, 13:97–102, 2003.

[10] M. Brudno and B. Morgenstern. Fast and sensitive alignment of large genomic sequences. In *Proc. of the IEEE Computer Society Bioinformatics Conference*, pages 138–150, 2002.

[11] C. Burge and S. Karlin. Prediction of complete gene structures in human genomic dna. *J.Mol. Bio.*, 268:78–94, 1997.

[12] W. Cai, J. Jing, B. Irvin, L. Ohler, E. Rose, H. Shizuya, U. Kim, M. Simon, T. Anantharaman, B. Mishra, and D. Schwartz. High-resolution restriction maps of bacterial artificial chromosomes constructed by optical mapping. *Proc. Natl. Acad. Sci. U.S.A.*, 95:3390–3395, 1998.

[13] A. Delcher, S. Kasif, R. Fleischmann, J. Peterson, O. White, and S. Salzberg. Alignment of whole genomes. *Nucleic Acids Res.*, 27:2369–2376, 1999.

[14] A. Delcher, A. Phillippy, J. Carlton, and S. Salzberg. Fast algorithms for large-scale genmoe alignment and comparison. *Nucleic Acids Res.*, 30(11):2478–2483, 2002.

[15] J. Deogun, J. Yang, and F. Ma. Emagen: An efficient approach to multiple whole genome alignment. In *the 2nd Asia Pacific Bioinformatics Conference (APBC2004)*, volume 29, Dunedin, New Zealand, 2004.

[16] G. Estabrook, F. McMorris, and C. Meacham. Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units. *Syst. Zool.*, 34:193200, 1985.

[17] J. Felsenstein. Alternative methods of phylogenetic inference and their interrelationship. *Systematic Zoology*, 28:49–62, 1979.

[18] J. Felsenstein. Evolutionary trees from dna sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17:368–376, 1981.

[19] J. Felsenstein. A likelihood approach to character weighting and what it tells us about parsimony and compatibility. *Biological Journal of Linnean Society*, 16:183–196, 1981.

[20] W. Fitch and E. Margoliash. The construction of phylogenetic trees - a generally applicable method utilizing estimates of the mutation distance obtained from cytochrome c sequences. *Science*, 155:279–284, 1967.

[21] K. Frazer, L. Elnitski, D. Church, I. Dubchak, and R. Hardison. Cross-species sequence comparisons: A review of methods and available resources. *Genome Res.*, 13:1–12, 2003.

[22] D. Gale and L. Shapley. College admissions and the stability of marriage. *Am. Math. Monthly*, 60(1):9–15, 1962.

[23] M. Gelfand, A. Mironov, and P. Pevzner. Gene recognition via spliced sequence alignment. volume 93, pages 9061–9066, 1996.

[24] A. Goldberg, S. Plotkin, D. Shmoys, and E. Tardos. Using interiorpoint methods for fast parallel algorithms for bipartite matchings and related problems. *SIAM Journal on Computing*, 21(1):140–150, 1992.

[25] D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology.* Cambridge University Press, New York, 1997.

[26] S. Henikoff and J. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl Acad. Sci. USA*, 89:10915–10919, 1992.

[27] M. Hohl and E. Ohlebusch. Efficient multiple genome alignment. In *Proceedings of the 10th Intervational Conference on Intelligent Systems for Molecular Biology*, pages 312–320, 2002.

[28] K. Iwama, D. Manlove, S. Miyazaki, and Y. Morita. Stable marriage with incomplete lists and ties. In *Proc. ICALP '99*, pages 443–452. 1999.

[29] W. James Kent. Blat-the blast-like alignment tool. *Genome Res.*, 12:656–664, 2002.

[30] J. Jing, Z. Lai, C. Aston, J. Lin, D. Carucci, M. Gardner, B. Mishra, T. Anantharaman, H. Tettelin, L. Cummings, S. Hoffman, J. Venter, and D. Schwartz. Optical mapping of plasmodium falciparum chromosome 2. *Genome Res.*, 9:175–181, 1999.

[31] W. Kent and A. Zahler. Conservation, regulation, synteny, and introns in a large-scale c. briggsae - c. elegans genomic alignment. *Genome Res.*, 10:1115–1125, 2000.

[32] A. Krogh. Using database matches with for hmmgene for automated gene detection in drosophila. *Genome Res.*, 11:817–832, 2000.

[33] Z. Lai, J. Jing, C. Aston, V. Clarke, J. Apodaca, E. Dimalanta, D. Carucci, M. Gardner, B. Mishra, and et al. A shotgun optical map of the entire plasmodium falciparum genome. *Nat. Genet.*, 23:309–313, 1999.

[34] I. Lee, D. Westaway, A. Smit, K. Wang, J. Seto, L. Chen, C. Acharya, M. Ankener, D. Baskin, C. Cooper, and et al. Complete genomic sequence and analysis of the prion protein gene region from three mammalian species. *Genome Res.*, 8:1022–1037, 1998.

[35] A. Lim, E. Dimalanta, K. Potamousis, G. Yen, J. Apodoca, C. Tao, J. Lin, R. Qi, J. Shiadas, and et al. Shotgun optical maps of the whole escherichia coli o157 :h7 genome. *Genome Res.*, 11:1584–1593, 2001.

[36] J. Lin, R. Qi, C. Aston, J. Jing, T. Anantharaman, B. Mishra, O. White, M. Daly, K. W. Minton, J. Venter, and D. Schwartz. Whole-genome shotgun optical mapping of deinococcus radiodurans. *SCIENCE*, 285:1558–1562, 1999.

[37] B. M., C. Do, G. Cooper, M. Kim, and E. Davydov. Lagan and multi-lagan: Efficient tools for large-scale multiple alignment of genomic dna. *Genome Res.*, 13:721–731, 2003.

[38] E. McCreight. A space-economical suffix tree construction algorithm. *J. ACM.*, 23:262–272, 1976.

[39] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proc. 18th Intl. Conf. on Data Engineering (ICDE)*, San Jose CA, 2002.

[40] B. Mishra. Optical mapping. *Encyclopedia of the Human Genome, Nature Publishing Group, Macmillan Publishers Limited, London, UK*, 4:448–453, 2003.

[41] B. Morgenstern. Dialign 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, 15(3):211–218, 1999.

[42] B. Morgenstern, O. Rinner, S. AbdeddaÏm, D. Haase, K. Mayer, A. Dress, and H. Mewes. Exon discovery by genomic sequence alignment. *Bioinformatics*, 18(6):777–787, 2002.

[43] C. Notredame, D. Higgins, and J. Heringa. T-coffee: A novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.*, 302:205–217, 2000.

[44] D. Penny and M. Hendy. The use of tree comparison metrics. *Syst. Zool.*, 34:75–82, 1985.

[45] H. S. and H. J.G. Performance evaluation of amino acid substitution matrices. *Proteins*, 17(1):49–61, 1993.

[46] N. SAITOU and M. NEI. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4:406–425, 1987.

[47] S. Schwartz, L. Elnitski, M. Li, M. Weirauch, and et al. Multipipmaker and supporting tools: alignments and analysis of multiple genomic dna sequences. *Nucleic Acids Research*, 31(13):3518–3524, 2003.

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

[49] S. Schwartz, Z. Zhang, K. Frazer, A. Smit, C. Riemer, J. Bouck, R. Gibbs, R. Hardison, and W. Miller. Pipmaker-a web server for aligning two genomic dna sequences. *Genome Res.*, 10:577–586, 2000.

[50] P. Sneath and R. Sokal. *The principles and practice of numerical classification.* Numerical Taxonomy, W. H. Freeman, San Francisco, 1973.

[51] J. Stajich, D. Block, K. Boulez, S. Brenner, S. Chervitz, C. Dagdigian, and et al. The bioperl toolkit: Perl modules for the life sciences. *Genome Res.*, 12(10):1611–1618, 2002.

[52] B. Sun, J. Schwartz, O. Gill, and B. Mishra. Combat: Search rapidly for highly similar protein-coding sequences using bipartite graph matching. In *Computational Science - ICCS 2006: 6th International Conf.*, pages 654–661, Reading, UK., 2006.

[53] W. Taylor. Protein structure comparison using bipartite graph matching and its application to protein structure classification. *Mol. Cell Proteomics*, 1(4):334–339, 2002.

[54] J. Thompson, D. Higgins, and T. Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680, 1994.

[55] M. S. Waterman and T. F. Smith. On the similarity of dendrograms. *J. Theoret. Biol.*, 73(4):789–800, 1978.