Parallelism In Multiple Sequence Alignment

Qiong Bai

B. Sc., Simon Fraser University, 2003

Thesis Submitted In Partial Fulfillment Of

The Requirements For The Degree Of

Master Of Science

in

Mathematical, Computer, and Physical Sciences

(Computer Science)

The University Of Northern British Columbia

February 2006

Your file Votre référence
ISBN: 978-0-494-28349-3
Our file Notre référence
ISBN: 978-0-494-28349-3

Abstract

Aligning multiple sequences is a daunting task that can be extremely demanding on computer power and memory resources. By comparing the homologues sequences from different species of animals, one can draw inferences about the evolution of these species from their common ancestors. This thesis applied shared memory parallel processing techniques to the global alignment of multiple DNA and protein sequences, as well as gene orders. The multi-threaded method deployed for multiple DNA and protein sequence alignment was based on the divide-and-conquer technique, which firstly cut the sequences into sub-sequences, but it is computational expensive to find the optimal cut positions. Since speed performance is the focus of this thesis, we did not consider the optimal cut positions in our implementations. Meanwhile, an original and promising graph-based algorithm with parallel processing properties was introduced to simplify and speed up the alignment operations for multiple gene orders.

Contents

List of Figures

List of Tables

Publications from this Thesis

[1] "Multithreaded Multiple Sequence Alignments", Joanne Bai, Siamak Rezaei. The Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference, Shanghai, China, September 1-4, 2005.

[2] "Multiple Gene Order Alignmen", Siamak Rezaei, Joanne Bai. The Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference, Shanghai, China, September 1-4, 2005.

Acknowledgments

I would like to express my gratitude to all those who have given the support to complete this thesis.

I am greatly indebted to my supervisor, Dr. Siamak Rezaei, for kindly providing suggestions and encouragement which helped me in all the time of research and writing of this thesis. His comments have been of greatest help at all times.

My gratitude also goes to my supervision commettee members Dr. Liang Chen and Dr. Cecilia Alstrom-Rapaport, for their suggestions and encouragement that led to substantial improvements of this thesis.

I would also like to thank the external examiner Dr. Alan Wagner for reviewing this thesis and monitoring the work with his valuable suggestions.

I thank the Dean of Graduate Studies Dr. Robert Tait, the secretaries of the Dean of Graduate Studies Ms. Bethany Haffner and Ms. Trinity Posteraro for having always been so supportive.

I thank Pruthvi, Kouhyar, Abbie, Jeremy, and Sean for their nice company and suggestions during my education.

Lastly, but most importantly I am very grateful for the love and support of my parents Fan Bai and Yufen Chen, who have been always encouraging me to fulfill my dreams.

Chapter 1

Introduction

The focus of this thesis is the alignment of multiple biological sequences, including gene sequences (i.e. DNA and RNA), protein sequences, and genome sequences (a list of ordered gene blocks). In order to distinguish these two levels of sequences, DNA, RNA and protein sequences will be referred to as normal 'sequence', while the genome sequences will sometimes be referred to as 'gene orders' in this thesis.

1.1 Biological Background for Sequence Analysis

1.1.1 Biological Sequences

DNA (Deoxyribonucleic Acid) is a nucleic acid that carries the genetic information in the cell and is capable of self-replication and synthesis of RNA (Ribonucleic Acid). DNA consists of two long chains of nucleotides twisted into a double helix and joined by hydrogen bonds between the complementary bases A and T or C and G. The sequence of nucleotides determines individual hereditary characteristics. RNA is a very similar polymer. It is usually a single-stranded chain of alternating phosphate and ribose units with the bases A, U, G, C bonded to the ribose. The structure and base sequence of RNA are determinants of protein synthesis and the transmission of genetic information. Figure 1.1 shows a simple example of DNA sequences.

> A Single Strand DNA Sequence

ATTTCGTGCATATCTGACGTTAGGACCACGT

Figure 1.1: An example of single strand DNA sequences

The fundamental building blocks of life are proteins. Twenty different amino acids are commonly found in proteins, and each protein has a unique, genetically defined amino acid sequence. They serve as enzymes, structural elements, hormones, immunoglobulins, etc., and are involved in oxygen transport, muscle contraction, electron transport, and other activities throughout the body. One of the most important concepts in modern biology is that the functional properties of proteins is determined largely by the sequence of the 20 amino acids shown in Table 1.1.1 in the linear polypeptide chain. Thus, in theory, knowing the sequence of a protein (the order with which the amino acids occurred) one could infer its function.

What determines the order of amino acids in a protein? The central dogma of Molecular Biology states the relationships between genes and proteins. It describes that each gene in the DNA molecule carries the information needed to construct one protein, which, acting as an enzyme, controls one chemical reaction in the cell. The basic structure of any protein can be described by its sequence of amino acids, and the shapes those proteins fold up into make them different from one another. However, in this thesis, other than primary structure, the protein shapes, such as secondary structure, tertiary structure, and quaternary structure, will not be discussed. Figure 1.2 shows an example of a protein sequence.

> HAGSI hemoglobin alpha-A chain - bar-headed goose

VLSAADKTNVKGVFSKISGHAEEYGAETLERMFTAYPQTKTYFPHFDLQHGSAQIKAHGK
KVVAALVEAVNHIDDIAGALSKLSDLHAQKLRVDPVNFKFLGHCFLVVVAIHHPSALTAEV
HASLDKFLCAVGTVLTAKYR

Figure 1.2: A protein example: a goose hemoglobin protein known as "HAGSI"

	Name	Three-letter Code	One-letter Code
1	Alanine	Ala	A
2	Cysteine	Cys	C
3	Aspartic Acid	Asp	D
4	Glutamic Acid	Glu	E
5	Phenylalanine	Phe	F
6	Glycine	Gly	G
7	Histidine	His	H
8	Isoleucine	Ile	I
9	Lysine	Lys	K
10	Leucine	Leu	L
11	Methionine	Met	M
12	Asparagine	Asn	N
13	Proline	Pro	P
14	Glutamine	Gln	Q
15	Arginine	Arg	R
16	Serine	Ser	S
17	Threonine	Thr	T
18	Valine	Val	V
19	Tryptophan	Trp	W
20	Tyrosine	Tyr	Y

Table 1.1: The twenty amino acids found in proteins

1.1.2 Phylogenetic Trees

Biologists often use the idea of portraying graphically how extant species evolved to reconstruct evolutionary trees. "Phylogenetic tree" is a more technical name for "evolutionary tree". The sample tree shown in Figure 1.3 [1] portrays how certain warblers are related to one another. From this tree one can guess that at one time there was one general, warbler-like species, and this general species evolved into at least three different species. Each of these species further diversified into the extant genera known as Vermivora, Seirus, and Dendroica. These three genera, in turn, have eventually fragmented into numerous and various species recognizable today. For example, the genus Dendroica includes more than twenty-five species.

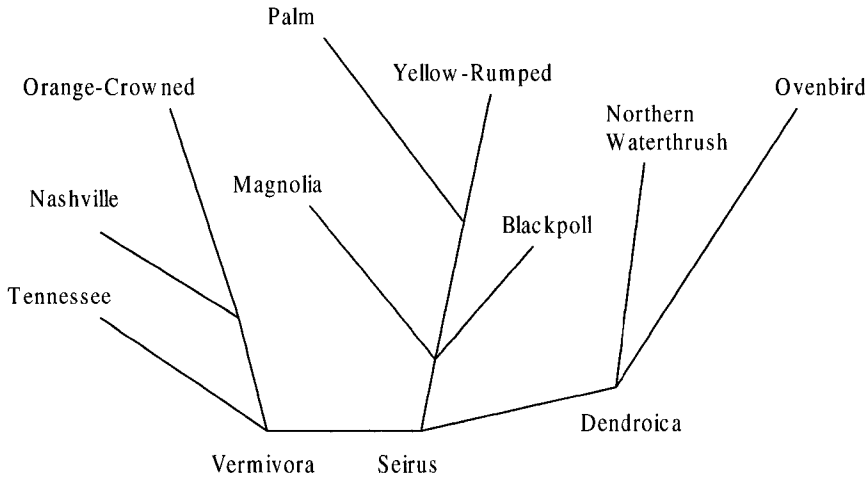


Figure 1.3: A phylogenetic tree for some warbler species

In the past, biologists working on phylogenetic trees had to find plain evidences or characteristics to content themselves. The more similar organisms looked, the more closely related they were assumed to be. That's often really the way it works, but also, this approach could lead to some pretty serious errors. Nowadays, new techniques have given us a more clear idea of what the real phylogenetic trees should look like. For instance, with a process known as DNA hybridization, scientists can actually

[1] adapted from http://www.backyardnature.net/evotrees.htm

determine how much genetic material that different species have in common. If a large portion of the genetic code (i.e. DNA) of two species or two groups of species is identical, then they are considered to be closely related. Otherwise, if only a small portion of them is identical, then their relationship is distant.

A phylogenetic tree is constructed with extant species on the leaves (terminal nodes), and the interior nodes representing hypothesized ancestors. Usually all interior nodes in such trees are binary. In the context of sequences, the nodes represent sequences, and the edges(branches) represent mutations, either explicitly or by a number indicating how many mutations have happened between the end nodes of an edge. In a tree constructed from protein sequences or DNA sequences, an interior node can in principle represent the same sequence as a leaf node.

Consider a set of DNA sequences 1, 2, 3, 4, 5.

Sequence 1: C A G G T A
Sequence 2: C A G A C A
Sequence 3: C G G A T G
Sequence 4: T G C G C T
Sequence 5: T G C G C A

The task is to construct a phylogenetic tree and find out which sequences are more closely related to each other. This is the same as finding the mutations that have occurred. A distance score will be used to reflect the number of mutations needed from one sequence to another. For example, transforming sequence 4 into 5 needs only 1 mutation, which occurs at the last position of sequence 4 and replaces T with A. Therefore, the distance score between sequence 4 and sequence 5 is 1. The smaller the distance score is, the closer the two sequences are related. Assume that if the distance score is no less than half of the average sequence length (no less than 3 in this case), then the two sequences won't belong to the same subset.

1. Looking at the first column, it seems that a mutation from C to T, or from T to C, has occurred in the past. In the mean time, check the distance scores in Figure 1.4 and divide the 5 sequences into two subsets: 1, 2, 3 and 4, 5

2. Looking at the second column in the larger subset 1, 2, 3 there could have been a mutation from A to G, or from G to A. Again, check the corresponding distance scores and divide it into 2 even smaller subsets: 1, 2 and 3

3. Repeating the same thing for the smaller subset 4, 5 and their distance score (which is 1) tells that they are closely related and may not be further divided.

Finally, we end up with a phylogenetic tree shown in Figure 1.4.

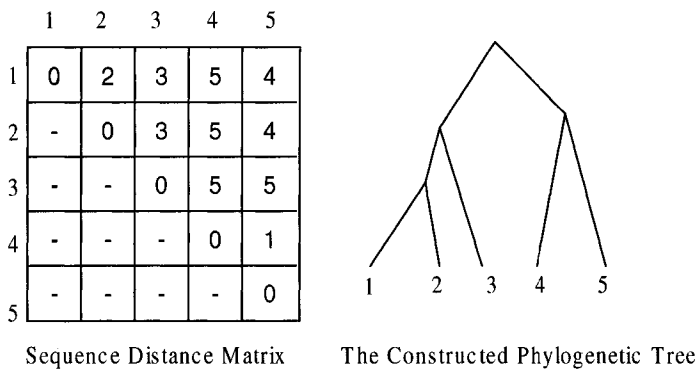


Figure 1.4: A phylogenetic tree built for five animals

1.1.3 Gene Orders

A gene order refers to the sequential location of genes on a chromosome. A chromosome is defined as a sequence of genes while a genome is defined as a set of chromosomes. Genes of an organism are arranged in a linear order in eukaryotes. This linear order of genes can be examined to understand the similarity between the genomes of

two species. A given genome may be transformed or evolved into the genome of a different organism by a sequence of elementary rearrangement events acting on the genes. Hannenhalli and Pevzner have shown how cabbage gene orders were transformed into turnip gene orders [30] by using an efficient algorithm that performs only three rearrangement operations.

In the early 1990s, it was found that there are groups of genes in mice that appear in the same order as they do in humans. These genes are likely to be present in the same order in a common ancestor of human and mice - the ancient mammalian genome. The human genome appears just the mouse genome cut into about 300 large genomic fragments, called syntency blocks, which have been pasted together in a different order. Both sequences are just two different shufflings of the ancient mammalian genome. For example, chromosome 2 in humans is built from fragments that are similar to mouse DNA residing on chromosomes 1, 2, 3, 5, 6, 7, 10, 11, 12, 14 and 17. Hence, a location of a gene in mice can often lead to clues about the location of a related gene in humans.

Every genome rearrangement results in a change of gene ordering, and a series of these rearrangements can significantly alter the genomic architecture of a species. The study of genome rearrangements involves solving the combinatorial puzzle of finding a series of rearrangements that transform one genome into another. Figure 1.5 presents a rearrangement scenario in which the mouse X chromosome is transformed into the human X chromosome. The elementary rearrangement event in this scenario is the flipping of a genomic segment, and it is called a reversal or an inversion. Details will be introduced in the next chapter.

1.2 Why Sequence Analysis

Sequence analysis is the process of making biological inferences from the known sequence of monomers in protein, DNA and RNA polymers. Currently, many worth-

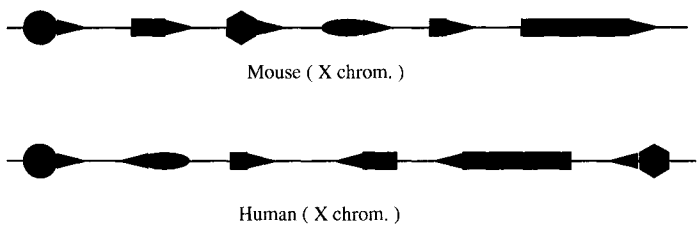


Figure 1.5: The transformation of the mouse gene order into the human gene order on the X chromosome: only the 6 longest syntency blocks are shown here

while things can be done with sequence analysis.

Regions of DNA that encode proteins are first transcribed into messenger RNA and then translated into protein. By examining the DNA sequence alone we can determine the sequence of amino acids that will appear in the final protein. In translation, codons of three nucleotides determine which amino acid will be added next in the growing protein chain. It is important then to decide which nucleotide to start translation, and when to stop, this is called an open reading frame. Every region of DNA has six possible reading frames, three in each of the two strands. The reading frame that is used determines which amino acids will be encoded by a gene. Typically only one reading frame is used in translating a gene in eukaryotes, and this is often the longest open reading frame. Once the open reading frame is known the DNA sequence can be translated into its corresponding amino acid sequence.

Sequence analysis helps to identify a protein's primary sequence according to the corresponding gene sequence. For example, if we need to find out where a protein is encoded in a DNA sequence, it is very useful to know what peptides would be encoded by all six reading frames.

Secondly, sequence analysis can be used to search databases for sequences similar to a new sequence. If someone has just determined a sequence of an interesting bit of DNA, one of the first questions he is likely to ask himself is "has anybody else seen anything like this?" Fortunately, there has been a very successful international effort to collect all the sequences people have determined in one place so that they can be searched.

Thirdly, sequence analysis performs a calculation of sequence alignments for evolutionary inferences and it aids in structural and functional analysis. Although it is not possible to completely predict the function or shape (structure) of a protein from a sequence, some useful inferences about structure and function can be drawn, by comparing the sequence of a protein of unknown structure and function to sequences of proteins with known structure and function. Second, the partial analysis done in the present will help reach the goal of structure or function prediction in the future. Third, by comparing the sequences of equivalent proteins from different species of animals (such equivalent proteins are called "homologues"), one can draw inferences about the evolution of these species from their common ancestors.

The third application of sequence analysis, sequence alignments, is going to be the topic of this thesis, and this includes multiple sequence alignments, and multiple gene order alignments.

Comparing sequences, structures, and sequences with structures is the most fundamental operation in biological sequence (i.e. DNA, RNA and protein) and structure analysis. When a comparison indicates a similarity between two proteins, it can immediately suggest relationships involving structure, function and the evolution of the two protein sequences from a common ancestor protein. When one of the proteins is well characterized in terms of structure and function, a close connection with a novel protein sequence may allow all the hard-earned biological data to be transferred to the new protein. The degree of certainty with which this transfer can be made depends on how similar the two sequences are. The two related protein sequences are said to be homologous, and the information are transferred by homology.

1.3 Why Parallelizing Multiple Alignments

Parallel computers are used primarily to speed up computations. A prallel algorithm can be significantly faster than the best possible sequential solution. There is a growing number of applications in sciences, engineering, business, and medicine requiring

computing speeds that cannot be delivered by conventional computer. These applications involve processing huge amount of data, or perform a large number of iterations, thus leading to inordinate running times. Parallel computation is the only approaches known today that would make these computation feasible.

Research of genome projects generates enormous amounts of information. Researchers want to access this information quickly and easily and also be able to transform this information into other useful information. Researchers also want to access cognate information, such as bibliographic or biological information associated with a given DNA sequence. Subsequently, there is a demand for increased computer power, both in speed and performance, and for enlarged memory capability, rapid networked communication, and improved database design.

Multiprocessing or parallel processing in general means the use of more than one processor or process in the computer handling of a given task. Multithreaded programming is one of the forms for parallel processing. So, what is a thread? Think of sewing needles as the CPUs (or Light Weighted Processes) and the threads in a program as the fiber. If you had two needles but only one thread, it would take longer to finish the job than if you split the thread into two and used both needles at the same time. Taking this analogy a little further, if one needle had to sew on a button (blocking I/O), the other needle could continue doing other useful work. If only one needle is used there would be some extra hours for the single needle to do other useful stuff. Moving to something more concrete, a thread is a sequence of instructions that can be executed in parallel with other threads. They are not entire processes, but rather lightweight threads of execution. Threads of a program are smaller portions of a process running concurrently (or in parallel).

Multiple sequence alignment is a demanding task to automatically generate an accurate alignment. An in-depth knowledge of evolutionary and structural relationships within a species family is often lacking or hard to use. General empirical models based on mathematically sound principles can be extremely demanding in CPU power and memory, and are difficult to apply. For some cases, statistical heuristics have been

developed to be able to cope with practical data set size. Therefore, in order to reduce the computation time and obtain useful data more efficiently it would be very ideal if some parallelism techniques could be applied for multiple sequence analysis.

Similarly, these complexity problems also occur in the computations for multiple gene order alignment. For instance, there are two different complexities for gene order alignment:

- For signed reversal and translocation distance, which marks the gene positively or negatively, the computation complexity is polynomial.
- For unsigned reversal distance, which only marks the gene positively, the complexity is much more than polynomial.

Besides, the existing algorithms for gene order alignment, including breakpoint analysis and reversal distance analysis, lack the parallelism properties for computational speedup.

1.4 Thesis Organization

The rest of the thesis will investigate the following issues:

- Chapter 2: discusses the related work that has been done for multiple sequence and genome (gene order) alignments, including literature reviews on alignment methods, algorithms, and applications.
- Chapter 3: presents a multithreaded implementation for multiple sequence alignments based on the mixed idea of Progressive alignment and Divide-and-conquer alignment. Depending on how the guide tree(s) would be applied, two different approaches are implemented for checking the improvements of alignment speed and sensitivity.
- Chapter 4: describes a new algorithm with parallelism properties for aligning two or more genome sequences, which is based on a precedence graph-based

consensus method and able to obtain the ancestor genome sequence. In this section, the algorithm formalization and the issues related to alignment result evaluation are included.

- Chapter 5: concludes the thesis and talks about the future work that is going to be involved.

Chapter 2

Algorithms and Related Work

In this chapter, algorithms and a broad range of the preceding work in multiple sequence and gene order alignment is described. It provides an overview of the earliest papers to the most recent development, including the approaches and applications for multiple alignment, and the work for parallelized multiple alignment.

Sequence alignments are either global or local. Global alignments find the best match over the total length of both sequences. In many cases, however, sequences share only a limited region of similarity. This may be a common domain or simply a short region of recognizable similarity. This case is dealt with by local alignment. Local alignment aims at identifying the best pair of regions, one from each sequence, such that the optimal alignment of these two regions is the best possible. In this thesis, only global alignment will be considered and discussed.

2.1 Pairwise Sequence Alignments

A residue is a single unit within a polymer, such as an amino acid within a polypeptide or protein. This term reflects the fact that sugars, nucleotides, and amino acids usually lose a few atoms (usually hydrogen and oxygen) when they are polymerised into a larger molecule. In making an alignment, a one-to-one correspondence is set up between the residues of the two sequences. This has the evolutionary implication

that at one time the paired residues were the same in an ancestral sequence and have diverged through the accumulation of point mutations in their DNA. Point mutation (or sometimes called substitution) is not the only process at work and extra residues may have been inserted or deleted giving rise to breaks or gaps in the alignment. These are referred to as insertions and deletions or, as indels.

For example, one possible alignment between the sequences

S = **A G C A C A C A**
T = **A C A C A C T A**

is A1:

A G C A C A C - A
A - C A C A C T A

Or A2:

A G - C A C A C A
A C A C A C T - A

A1 contains one insertion and one deletion. A2 needs one insertion, one deletion, and two replacements. If we assume that the cost for each of these one-bit operations (insertion, deletion, replacement) is 1, then the cost of A1 is 2, and the cost of A2 is 4. Therefore, A1 is a better alignment than A2 as it is cost effective.

2.1.1 Dynamic Programming Algorithms

There are different types of sequence comparison algorithms using dynamic programming and the corresponding parallel formulations. The first algorithm for comparing biological sequences using the dynamic programming techniques was introduced by Needleman and Wunsch [42] in 1970. The algorithm consists of two parts: the cal-

culation of the total score indicating the similarity between the two given sequence, and the identification of the alignments that leads the score.

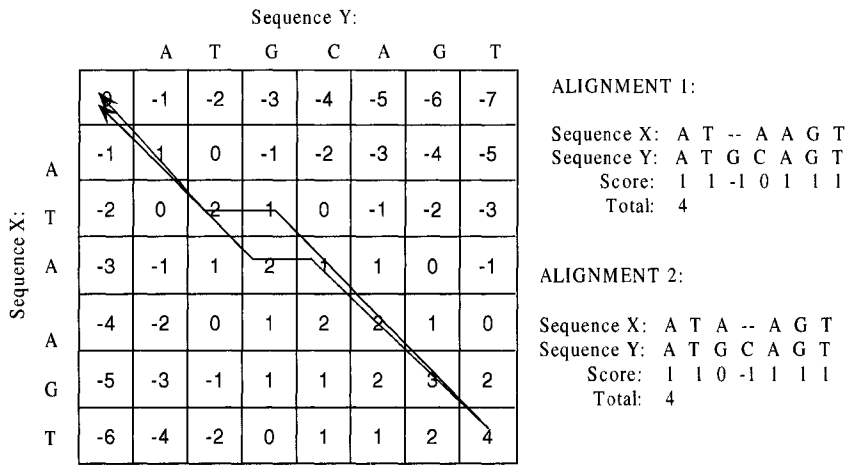


Figure 2.1: Alignment examples of Needleman-Wunsch algorithm

Considering the two sequences X and Y as shown in 2.1, a similarity matrix can be initialized with deceasing values (0, -1, -2, -3, -4, ...) along the first row and the first column to parallelize the consecutive gaps(insertions and deletions). The other elements of the matrix d[i,j] are calculated and filled by the maximum of the three values: d[i-1,j-1], d[i-1,j], and d[i,j-1]. In another words, the value of cell d[i,j] solely depends on the values of its 3 adjacent neighbors at the previous row, column and diagnose positions. The similarity matrix d[n, m] is built by applying the following recurrence equation:

$$d[i, j] = \max (d[i, j-1] + gp, d[i-1, j-1] + ss, d[i-1, j] + gp)$$

In this example, gp is -1. ss is 1 if the elements match or 0 otherwise. Since global alignment takes into account the entire sequences, the final score will always be found in the bottom right hand corner of the matrix. In our example, the final score 4 gives us a measure of how similar the two sequences are. Figure 2.1 shows the similarity matrix and the two possible alignments. The two arrows going up and left

represent two optimal paths. Tracing an optimal path backwards leads to an optimal alignment for two sequences.

The basic dynamic programming algorithm for making a global alignment is perhaps the most widely used and important algorithm in bioinformatics. Variations of it are used for local alignment and it can be extended to align more than two sequences (multiple alignment), such as Smith-Waterman's Algorithm, Fickett's Algorithm [42], and Wilbur-Lipman's algorithm [88].

Almost all alignment methods find the best alignment between two sequences under some scoring scheme. These scoring schemes can be as simple as '+1 for a match, -1 for a mismatch'. Indeed, many early sequence alignment algorithms were described in these terms. However, since a scoring scheme to give the biologically best score is wanted, the fact that biological molecules have evolutionary histories, three-dimensional folded structures, and other features which constrain their primary sequence evolution need to be taken into account. Therefore, in addition to the mechanics of alignment and comparison algorithms, the scoring system can be very complex.

2.2 Multiple Sequence Alignment

A multiple sequence alignment is an alignment of two or more sequences. It is a natural extension of two sequence alignment, called pairwise alignment. A simple example of multiple sequence alignment is shown in figure 2.2. Sequences may be multiply aligned to visualize the effect of evolution across homologues proteins or DNAs. Multiple alignment makes it possible to investigate a wide range of important biological phenomena like the following:

- Phylogenetic analysis
- Identification of conserved motifs and domains
- Structure prediction

```
Seq 1: V T S I T T C G S N I G N V K W Y L P G
Seq 2: V T S I T T C G S N I - - V N W Y L P G
Seq 3: V S T L L L C V G Y P - - V E W E G - -
```

Figure 2.2: A sample alignment of three sequences

Depending on the sequences compared and the goal or application, how the comparison is performed will be different. For these reasons, there have been many different kinds of algorithms and programs written to compare sequences. Considering the obvious properties of existing multiple alignment algorithms, it is convenient to classify them in three main categories [59]: exact, progressive and iterative.

2.2.1 Exact Algorithms

The simultaneous alignment of all the sequences is called dynamic programming alignment or exact algorithm. Exact algorithms are high quality heuristics that deliver an alignment usually very close to optimality. When solving the computational problem of multiple sequence alignment, a natural generalization is to expand the two sequence case. The basic algorithm for the global comparison of two sequences, is to assign a score. In the multiple sequence case, the issue of scoring becomes a little more complex.

With several sequences, scores are calculated by assigning a score to each column of the scoring matrix. The sum-of-pairs (or SP measure) is often used to calculate values for a sequence. The sum-of-pairs function is the sum of pairwise scores of all pairs of symbols in a given column. A k-dimensional array would be an obvious solution for dynamic programming for the alignment of k sequences, but a standard application of dynamic programming for multiple sequences takes exponential time. Even with time saving measures, a multiple sequence alignment of three sequences takes $O(n^3)$ time. As a result, only 3 or 4 sequences can be realistically used when implementing a global dynamic programming algorithm.

Carrillo and Lipman [51] recognized a lower bound on the cost of the optimal

multiple sequence alignment, and presented an elegant branch-and-bound approach. They described a technique for reducing the part of the graph (the dynamic programming matrix)that has to be examined. Only the paths that are contained in a certain "polytope" around the shortest path are explored by their algorithm.

The MSA program [52] implemented a slightly modified version of the Carrillo and Lipman's algorithm. Since the bounds of Carrillo and Lipman were not sufficiently tight for solving the "real world" multiple sequence alignment instances, Lipman et al. proposed heuristics to improve bounds. However, the number and length of sequences that can be aligned is limited because the number of computational steps and the amount of memory grow exponentially with the number of sequences to be analyzed.

2.2.2 Divide-and-Conquer Alignments

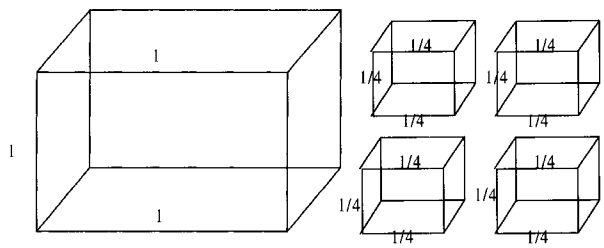
In Exact algorithm, the time and space complexity grows exponentially with the number of sequences (O (m^n) for equal sequence length m), and for practical solutions of such computationally expensive problems, generally, two approaches are used:

- One approach attempts to reduce the running time by using pruning techniques which still guarantee finding the highest-scoring alignment, but not reducing the worst-case complexity. Divide-and-Conquer is one example of these approaches.

- The other approach uses heuristics. This means that some 'rules of thumb' are used in the solution, and the best (or correct) solution is not necessarily found. For some of the heuristic methods it is possible to find an upper bound on the deviation of the result from the correct one.

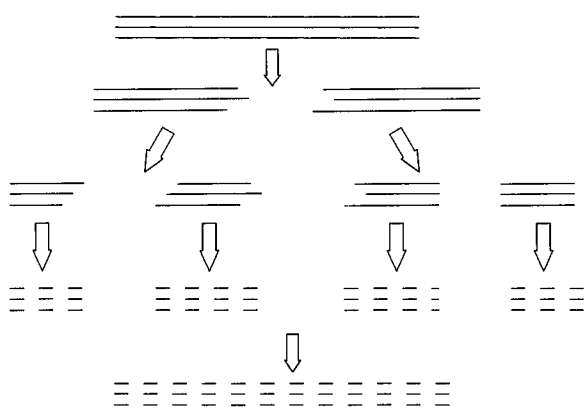
Divide and Conquer Alignment [79] [80] is one of the implementations based on the Carillo and Lipman algorithm to limit computations to a small area in the multi-dimensional search matrix. The idea of divide-and-conquer is straightforward: DCA guesses a point in the lattice on the optimal path, splits the sequences at that point, and recurses. On sequence fragments that are short enough, the exact algorithm

is invoked. Once all fragments are aligned, they are concatenated to yield an approximate alignment. By doing this, the search space for each sub-alignment can be significantly reduced (as shown in Figure 2.3). This method finds the split point at the middle way of the longest sequence, and then searches the multi-dimensional matrix for a set of coordinates with minimal score. DCA does not take direct advantage of heuristic procedures for finding the split points, such as using the presence of strong motifs in subsets of the input sequences. In molecular biology, motif refers to the conserved smallest group of atoms in a polymer that, when under the influence of a rotation-translation operator, will assemble the rest of the atoms in the chain.



(a) The original searching space: 1

(b) The searching space after dvide-and-conquer: (1/4 * 1/4 * 1/4) * 4 = 1/16



(c) The divide-and-conquer technique: The original 3 sequences are divided into 2 sets of sub-sequences. Keep dividing in each subset and aligning each individual subset. When all the sub-alignments are obtained, assembling them together to form the final global alignment.

Figure 2.3: Illustrating the search space with the divide-and-conquer technique

General Dynamic Programming (GDP) [27] is similar to DCA. The major difference is that DCA only commits to a set of split points without considering the genomics information hiding behind the sequences, whereas GDP uses local and global approximate alignments to generate a large number of plausible "anchor points" aimed at the optimal path through the lattice, and progressively computes the score moving from one anchor to the next. It has been claimed by Gracy and Sallantin [27] that DCP successfully aligned up to 18 sequences with results superior to ClustalW [82].

2.2.3 Progressive Alignment

Among the many strategies implemented for multiple sequence alignment, progressive alignment is the most successful and by far the most widely used. The idea is to establish an initial order (i.e. a guide tree) for joining the sequences and to follow this order to gradually build up the alignment. The strategies depend on a progressive assembly of the multiple alignments where sequences or alignments are added one by one so that never more than two sequences are simultaneously aligned using dynamic programming. This approach has the great advantage of speed and simplicity combined with reasonable sensitivity, even if it by nature is a heuristic method that does not guarantee any level of optimization.

The most prominent implementation for this approach is ClustalW [82], which, together with the window graphic user interface (GUI) version ClustalX [83], belongs to the Clustal family. Clustal [84] progressive sequence alignment includes following three steps, which are demonstrated in Figure 2.4:

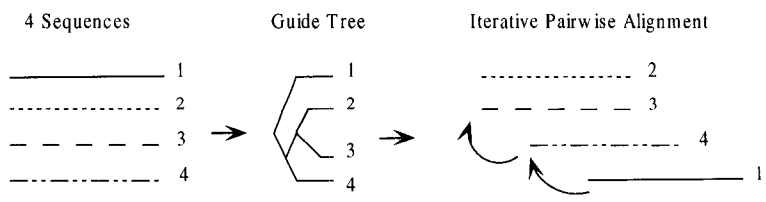


Figure 2.4: Demonstrating the 3-step progressive alignment by ClustalW

- Step 1: Pairwise Alignment
  - Aligns each sequence to each of the other sequences in the set, giving a similarity score for each comparison. Scores of each sequence pair are displayed in a similarity score matrix
  - The pairwise alignments are done by using dynamic programming (i.e. Needleman-Wunsch global alignment algorithm).
- Step 2: Guide Tree
  - The similarity matrix is transformed into a distance matrix, and is used by an algorithm (e.g. Neighbor-join algorithm [69] or UPGMA [54] [77])[1] to create a Guide Tree. The guide tree here is similar to the concepts of the phylogenetic tree introduced in Section 1.1.2, which specifies the relationships between sequences.
- Step 3: Progressive Alignment
  - Following the guide tree, the two most similar sequences are aligned, giving the consensus sequence. Consensus sequence is the sequence that reflects the most common choice of base or amino acid at each position of a series of related DNA, RNA or protein sequences. Areas of particularly good agreement often represent conserved functional domains.
  - The next sequence will be repeatedly added and aligned to the existing new consensus sequence, until the final alignment is achieved.

[1]UPGMA and Neighbor-join algorithms:

* UPGMA (Unweighted Pair Group Method using Arithmetic averages): It works by initially having all sequences in separate clusters and continuously joining these. The tree is constructed by considering all initial clusters as leaf nodes in the tree, and each time two clusters are joined, a node is added to the tree as the parent of the two chosen nodes. The clusters to be joined are chosen as those with minimal pairwise distance. The branch lengths are set corresponding to the distance between clusters, which is calculated as the average distance between pairs of sequences in each cluster.
* Neighbor Joining: The method works very much like UPGMA. The main difference is that instead of using pairwise distance, this method subtracts the distance to all other nodes from the pairwise distance.

ClustalW works well in alignment when sequences are closely matched to one another, or in another words, have many positions in common. However, in cases where the sequences are far apart from each other when pairwisely aligned, more errors appear, and these errors are propagated to the tree construction as well as the final results. Since the pairwise alignments are used in a greedy manner for progressive alignment, the alignments formed during the progression towards the final MSA cannot be changed any more. Thus, the difficulty with progressive alignment highly depends upon the initial pairwise sequence alignment. T-Coffee [60] is an alignment application that attempts to mitigate this shortcoming by using information from other global and local alignments to guide the progressive alignment.

T-Coffee is currently the most reliable MSA method available [49]. The basic philosophy is that instead of looking at pairs of sequences in isolation, this procedure allows information to be included from all other pairwise alignments. Thus the resulting guide tree makes the sequences to be aligned based on how well aligned they are with respect to the rest, so that the more confidently aligned sequences are matched up first and the least confidently aligned last. However, this method suffers from exaggerating the significance of shorter residue segments that share high percentage identity. As a result, although the method is beneficial for anchoring conserved domains in an alignment of closely related sequences, it has the opposite effect on outlier sequences that are usually misaligned due to the lack of a commonly conserved segment.

In most of the current top performing progressive alignment methods, such as Clustal series, DiAlign2 [58], POA [50], Praline [35], and T-Coffee [60], the dynamic programming (DP) strategy is adopted. The main difference between the available DP-based progressive methods is the way in which the information of aligned blocks of sequences is represented. While early methods used consensus sequences to represent alignment blocks, current methods mostly use a profile formalism to represent the information in a MSA [76].

2.2.4 Iterative Algorithms

Recent developments in multiple sequence alignment techniques have mainly focused on sensitive and optimal models to represent MSA information. A class of techniques that are able to revisit and optimize the MSA is that of iterative multiple alignment techniques. Pioneered by Hogeweg and Hesper (1984) [36], iterative techniques depend on algorithms able to produce an alignment and to refine it through a series of cycles or iterations until no more improvements can be made.

Iterative methods for MSA can be deterministic or stochastic, depending on the strategy used to improve the alignment. Deterministic iterative strategies are the simplest. They extract sequences one by one from a multiple alignment and realign them to the remaining sequences [29] [35]. This procedure is terminated when no more improvement can be made. Stochastic iterative methods include HMM training [33][6], simulated annealing [45], and evolutionary computations such as genetic algorithms [28] [90] and evolutionary programming [13]. The main advantage is to allow for a good conceptual separation between the optimization processes and objective functions (evaluation criteria). Objective function defines the aim of any optimization procedure.

PRRP [29], SAM [39], HMMER [23], SAGA [61], and MUSCLE [24] are some of the recent and less recent available iterative methods for multiple sequence alignment. Some of these methods are a mixture of progressive and iterative strategies.

2.2.5 Parallelized Multiple Sequence Alignment

Parallel processing, sometimes called concurrent processing contains two or more processors that work together to perform a task. Each process is a sequential program, namely, a sequence of statements that are executed one after another another. Whereas a sequential program has a single thread of control, a concurrent program has multiple threads of control.

The processes in a parallel program work together by communicating with each

other. Communication is programmed using shared variables or message passing. In shared memory model, when shared variables are used, one process writes into a variable that is read by another. In contrast, in message passing model, when message passing is used, one process sends a message that is receied by another. The focus of this thesis is on shared memory parallel algorithms.

Parallel algorithms for analyzing DNA and protein sequences are becoming increasingly important as sequence data continues to grow. While dynamic programming algorithms make large sequence alignment feasible, the quadratic time requirement still makes it a time-consuming process. A natural approach is to reduce the time requirement with the use of parallel computers.

Iyengar [5] examined the parallel characteristics of four sequence alignment algorithms. The four algorithms presented were the dynamic programming algorithm developed by Needleman, Wunsch, and Sellers (the NWS algorithm), Fickett's algorithm [42], a parallel algorithm using some of Fickett's ideas, and an algorithm which uses some of Wilbur and Lipman's ideas [88] for constructing alignments which are not always optimal. Iyengar found out that the NWS algorithm contains the most properties to be parallelized but also does more work than any of the other algorithms which were studied, and Fickett's algorithm contains the least parallelism properties.

A shared-memory multithreaded parallel version of the Needleman-Wunsch's Algorithm [42] using dynamic programming for pairwise alignment is presented by Martins [53], which handles the data dependencies very well and performs as many operations as possible independently. Another algorithm, Berger-Munson algorithm [8] was initially parallelized by Ishikawa et al. [40][41] on a parallel inference machine (PIM) using a parallel logic programming language KL1. Later, Yap et al. [89] extended and evaluated this approach on an Intel iPSC/860 parallel computer by applying speculative computation to the parallelization of the Berger-Munson algorithm, and achieved a higher speedup and a more scalable implementation.

Algorithms that both retain time optimality and reduce space requirement were first presented by Edmiston et al. [26] and further developed by Aluru et al on an IBM

SP-2 and a Pentium cluster [3]. Assume that n and m are the lengths of the sequences to be pairwisely aligned. Edmiston et. al. [26] discuss parallel algorithms for sequence and subsequence alignment that achieve linear speedup and can use up to min(m, n) processors. Lander et. al. [48] discusses an implementation on a shared memory parallel computer. These algorithms store the entire dynamic programming table. Huang [38] presented a parallel sequence alignment algorithm which increases the run-time to O $((m+n)^2/p)$, which is intended for a message-passing architecture with one-dimensional-array topology. Recently, Rajko et al. [64] claimed having developed the first space and time optimal parallel algorithm on an IBM xSeries cluster for the pairwise sequence alignment problem, which requires only O $((m+n)/p)$ space and O (mn/p) time, and is suitable for implementation on parallel computers.

A widely studied problem that is identical to a special case of the sequence alignment problem is string editing. Highly parallel algorithms for this problem have been developed for the hypercube models of computation [4], [65] using almost quadratic number of processors.

The parallel algorithms mentioned above for dynamic programming pairwise alignment can be extended to align more than two sequences. At present, several alignment tools for multiple sequences are parallelized and have become available online for free user access, such as pClustalW [14], Parallel ClustalW on 16 CPUs[55], DiAlign p [75], Praline [46], MUSCLE-p [25], and etc. The parallelism applied in these applications is not limited to the alignment algorithm itself, but also to the sequence processes.

2.3 Multiple Gene Order Alignment

2.3.1 Multiple Gene Order Alignments

Gene order alignment is another form of biological sequence alignment. If one gene can be thought as a word, gene order then can be presented as the order of words in the sentence. In multiple sequence alignment, the positions of residues are fixed, and a set of DNA or protein sequences are compared with each other, whereas in multiple

gene order alignment, a set of blocks of genes are compared in terms of their orders or positions.

The DNA of eukaryotes is subdivided into chromosomes. In prokaryotes, chromosomal DNA is circular, and the entire genome is carried on one chromosome. A chromosome has the self-replicating genetic structures of cells containing the cellular DNA that bears in its nucleotide sequence, the linear array of genes. Chromosome breakage and mistakes in repair, along with a number of other processes, give rise to changes in gene order. These have important consequences for the cell, the organism, the population, and for the evolution of species. Figure 2.5 shows the different X-chromosome gene orders for mice and humans. Gene order alignments will help to provide following information:

- Aligned gene orders allow us to speculate the closest common ancestor of the genomes.
- Gene order alignment is useful when faced with "missing link" problem where evolutionary intermediates are not known.
- Gene order alignment information could be used across multiple genomes to form a phylogenetic tree.

2.3.2 Representation of a genome

A unichromosomal genome can be considered as a sequence of n genes. Let's denote the genes by numbers 1, 2, ..., n, and represent the two signed orientations of gene i as i and -i. As such, a genome is represented as a signed permutation of the numbers 1, 2, ..., n. For example, a unichromosomal genome with n=5 genes is

5 -3 4 2 -1

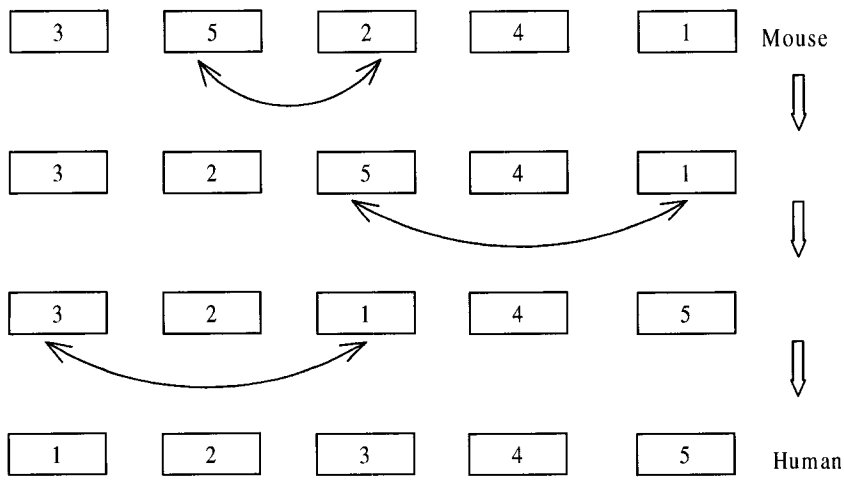


Figure 2.5: Mouse vs. Human: different X chromosome gene order

A multichromosomal genome consists of n genes spread over m chromosomes. It is represented as a signed permutation of 1, 2, ..., n, with delimiters ";" inserted between the chromosomes. For example, a genome with 12 genes spread over 3 chromosomes could be written as

7 -2 8 3 ;
5 9 -6 -1 12 ;
11 4 10 ;

The genome alignment being discussed here is about unichromosomal genome or one chromosome of the multichromosomal genome. Currently, neither repeated genes, nor the gaps arised from insertion or deletion, are considered.

2.3.3 Genome Rearrangement

The algorithmic study of comparative genomics tries to explain differences in gene orders in two or more genomes in terms of a limited number of rearrangement operations. For unichromosomal genomes, this requires the calculations of an edit distance between two linear orders on the same set of objects, representing the ordering of

homologous genes in two genomes. In the "signed" version of the problem, a plus or minus is associated with each gene, representing the direction of transcription.

Genome rearrangement operations considered here include reversal (also referred to as inversion), transposition, and translocation [70]. Some other operations also exist such as duplication, fusion and fission. Every study of genome rearrangement involves solving a combinatorial "puzzle" to find a shortest series of operations that transform one genome into another.

- **Reversal**: the inversion of any number of consecutive terms in the ordered set, which also reverses the polarity of each term within the scope of the inversion in the case of signed orders.
- **Transposition**: the movement of a piece of DNA around the chromosome (from one gene to another part of the genome), usually through the function of a transposable element. Transposition may or may not involve an inversion.
- **Translocation**: the rearrangement of a chromosome in which a segment is moved from one location to another, either within the same chromosome or to another chromosome. This is sometimes reciprocal, when one fragment is exchanged for another.

2.3.4 Rearrangement Distances

Literature often refers to the rearrangement events, reversal and translocation, as the genomic sorting problem. The key question is to find the minimum number of steps needed to transform genome A to genome B using reversal and translocation.

Breakpoint analysis

Breakpoint analysis [73] tries to minimize the *breakpoint distance* between two gene order sequences. A pair of elements in two permutations forms a *breakpoint* if they are consecutive in one but nonconsecutive in the other sequence, as illustrated in Figure 2.6. The breakpoint distance between two permutations is the number of breakpoints.

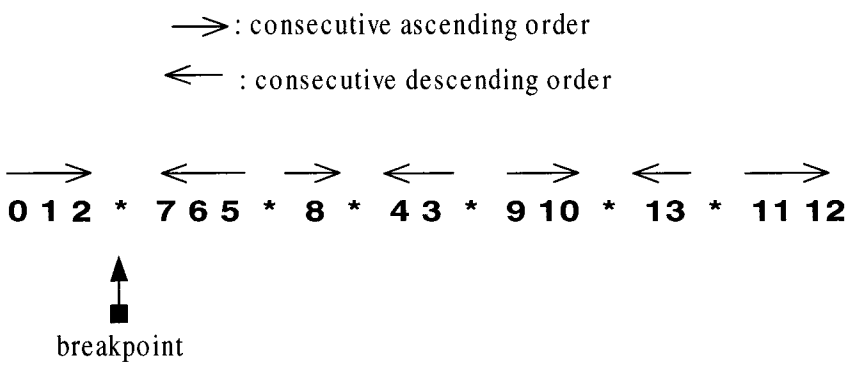


Figure 2.6: Examples for breakpoints: "->" means ascending order; "<-" means descending order; a single gene can be represented by either "->" or "<-"

Reversal Distance

One can also calculate the *reversal distance* between two genome permutations as well. A reversal in a signed permutation is an operation that takes an interval in a permutation, reverses the order of the numbers, and changes all their signs. For example, a reversal R(i, j) applied to the permutation:

P(1), ..., P(i-1), P(i), ..., P(j), P(j+1), ..., P(n)

and gives the following permutation:

P(1), ..., P(i-1), -P(j), ..., -P(i), P(j+1), ..., P(n)

By applying R(4, 8) to the following gene order:

1 2 3 4 5 6 7 8 9

One will obtain:

1 2 3 -8 -7 -6 -5 -4 9

The reversal distance for a pair of genomes can be computed in polynomial time, but the extension of pairwise alignment to multiple alignment using reversal distance has faced computability problems in the past, as a result multiple alignment using breakpoint distance has become a dominant technique for gene rearrangement.

GRAPPA[57] and MGR[9] are two applications designed for gene order analysis. Moret and his colleagues developed GRAPPA by improving the breakpoint analysis technique, while the MGR is based on reversal analysis.

2.3.5 Algorithms and Methods

Observations of gene duplication and repetitive sequences are much more common among eukaryotes than prokaryotes, while genome rearrangement can be readily observed between both closely-related and divergent organisms of all types. These additional evolutionary mechanisms distinguish the genome comparison and alignment task from traditional sequence alignment [15].

Comparing long sequences of genomes tends to be a very computationally expensive task, in terms of both time and memory. Traditional algorithms and methods for multiple sequence alignment do not scale to genomic size. If these algorithms were applied to genomic sequences of over a million bases, they would likely run out of memory or run for an unreasonable long time. In the past few years, there have been several attempts to solve these problems. Accordingly, several new methods and tools for genome comparison have been developed. However, most of them hold several techniques in common. Specifically, the existing tools can be grouped into one of two categories: *iterative pairwise alignment*, or *anchor-based multiple alignment*.

MLAGAN [12] is the most well-known tool based on iterative pairwise alignment, which will be introduced below. The majority of the remaining tools fall into anchor-based multiple alignment category, and almost all of these tools share very similar algorithmic concepts.

From Darling et. al.'s [15] point of view, an anchor-based alignment typically

proceeds in three steps. First, the aligner identifies a set of local alignments in regions of high similarity among the genomes. Next, a subset of the regions identified in the first step are selected as alignment anchors, based on whether the tool considers them as are part of the correct alignment. Finally, the alignment anchors are used to restrict the number of possible alignments considered when performing an ($O(n^2)$) gapped alignment using dynamic programming. Many tools assume that the genomes are collinear to complete a gapped alignment. Collinear means no significant inversion or rearrangement events took place since their divergence.

Pairwise Genome Alignment

Early research into genome alignment focused on scaling traditional pairwise alignment methods ($O(n^2)$) to handle much longer genome sequences. Pairwise genome alignment tools, such as MUMer [19], GLASS [7], and WABA [44] pioneered the use of anchoring to accelerate the alignment process.

MUMer, released in 1999, is the first software program capable of aligning whole genomes. It combines a suffix tree, longest increasing subsequence, and Smith-Waterman algorithms to align a pair of whole genomes.

LAGAN [10] is another pairwise whole genome alignment tool, which produces a global alignment of two genomes in three steps. First, it finds a set of local alignments using CHAOS algorithm [10]. Then, it chains an ordered subset of these local alignments to form a rough global alignment. Finally, it uses a bounded error dynamic programming algorithm to find the best alignment within a certain distance of the rough global alignment.

Multiple Genome Alignment

When the genomic DNA sequences of closely related organisms became available, there was an immediate need for reliable and automatic software to align three or more genomic sequences.

MGA ([37], 2002) was the first software tool for multiple genome alignment, which

is an anchor-based method that uses the same idea as the pairwise genome alignment tools (i.e. MUMer). The MGA algorithm works as follows:

- Find a set of exact matches in terms of the calculations of MultiMEMs (Maximal Multiple Exact Matches) [37]. A MultiMEM is an exact match that occurs in each sequence and cannot be extended in either direction without encountering a mismatch.
- The alignment is anchored at different subsets (or matches). Select an optimal chain of matches among all matches found.
- Close the gaps between chained matches using recursive calls.
- If the remaining gaps are short, hand them over to another sequence alignment tool (e.g., ClustalW); If the gaps remained are long, leave them open.

Compared with MUMer 2 [20], the extension of MUMer, which allows alignment of multiple genomes and translated protein sequences, MGA presents a significant improvement. The MUMer systems identify maximal unique matches and use them to anchor the alignments. MGA, however, completely discarded the uniqueness constraint, and searches for maximal multiple exact matches instead, which will theoretically increase the sensitivity of the alignments. Another improvement that MGA makes over the MUMer systems is that MGA needs less memory by constructing a virtual suffix tree.

Besides MGA and MUMer 2 mentioned above, some other anchor-based multiple genome alignment system are also available, such as EMAGEN [21], CHAINER [1][2], and M-GCAT [86].

Hohl et al. [37] also pointed out that an alignment of the genomes (gene orders) of several organisms makes sense only if the organisms are closely related. Otherwise, genome rearrangement should be taken into account. Shuffle-LAGAN [11] , a variant of the LAGAN alignment system, is the first genome comparison method that explicitly deals with genome rearrangements during the alignment process. Rather than

selecting a single collinear set of anchors, Shuffle-LAGAN selects anchors collinear in the first sequence with rearrangements permitted in the other sequence. However, Shuffle-LAGAN works only for pairwise comparison.

Mauve [16] presented an anchor-based alignment algorithm to address the presence of significant inversion and rearrangements in a set of genomes to be aligned. This algorithm is known as the first multiple alignment tool that considers genome rearrangement during the alignment process. Mauve's algorithm can be summarized as follows [16]:

- Find local alignment in terms of the calculations of MultiMEMs (Maximal Multiple Exact Matches) [37].
- Use the multiMEMs to build a phylogenetic guide tree.
- Select a subset of the multiMEMs to use as anchors. These anchors are partitioned into collinear groups called LCBs.
- Perform recursive anchoring to identifying additional alignment anchors within and outside each LCB.
- Perform a progressive alignment of each LCB using the guide tree.

Manual cure of a multiple genome alignment on actual genome sequence is too costly. Thus there is no 'gold standard' alignment to use when assessing the quality of calculated alignments. Darling et. al. [15] performed several experiments using simulated evolution environment to compare the accuracy to Multi-LAGAN, Shuffle-LAGAN, and MAVID. These experiments demonstrated that Mauve's algorithm clearly excels at aligning genomes with rearrangement.

As mentioned above, other than anchor-based, MLAGAN [12] is a tool based on iterative pairwise alignment. It performs best when aligning and analyzing the regions of homogeneity among distinct species, such as human and turkey. However, it was not designed to yield the same results when dealing with closely related multiple

genomes. MLAGAN performs a progressive alignment of the genomes using LAGAN, and takes a phylogenetic guide tree together with the sequences as input. Each time, by following the input guide tree, LAGAN will select two closest genome sequences and produce a pairwise alignment until all the sequences are aligned. MLAGAN then gives options to iteratively refine the resulted alignment until no significant improvement can be made.

2.4 Conclusion

During the past decades, considerable work for aligning multiple sequences has been done, but there is still room for improving alignment speed and sensitivity.

Multiple genome alignment is a large-scaled multiple sequence alignment. Traditional methods for sequence alignments cannot be applied directly for genome alignments. Since 1999, there have been several independent approaches to align multiple genomes. However, most of them hold several techniques in common and lack parallelism properties that would speed up the aligning process.

Chapter 3

Multithreaded Multiple Sequence Alignment

This chapter introduces a multithread approach for aligning multiple sequences. Depending on how the guide tree(s) would be applied for progressive alignment, two different software programs, a single-tree program and a multiple tree program, have been implemented. Their alignment results will be compared at the end to check the improvements of alignment speed and sensitivity. To our best knowledge, this is a novel approach to consider progressively aligning the same set of multiple sequences by building a different number of guide trees.

3.1 Complexity Analysis

According to the results of Wang and Jiang [87], the multiple sequence alignment problem is NP-complete for a class of scoring matrices used for reality biological applications.

From a computational point of view, there are several ways to address the lack of hard computing power for bioinformatics, especially in multiple sequence and gene order alignments. The first is by developing new, faster *heuristic* algorithms that reduce computational space for the most time-consuming tasks. The second is incor-

porating these algorithms into the ROM of a specialized chip. The third and most promising consideration, is parallel computing. In parallel processing, two or more microprocessors (or threads) can be used simultaneously to divide and conquer tasks that would overwhelm a single, sequential processor. However, parallel computing still requires new paradigms in order to harness the additional processing power for bioinformatics.

3.2 Evaluations and Comparisons

The most frequently used reference MSA database is BAliBASE (Benchmark Alignment database) [85]. It is the only reference database for multiple sequence alignment that has been specifically designed for MSA benchmarking. That's why BAliBASE is so appealing to MSA method developers. As a result, it has been used in many studies as the standard of truth for comparing the performance of new MSA methods with older ones. The current version of BAliBASE (version 2.0) [82] contains a total of 167 reference alignments placed in eight different categories, which are aimed at covering most of the problems alignment engines come up against:

1. Multiple sequence alignments containing equi-distant sequences of various conservation levels. Conserved gene is a gene that has remained essentially unchanged throughout evolution. Conservation of a gene indicates that it is unique and essential. There is not an extra copy of that gene with which evolution can tinker. Changes in the gene are likely to be lethal.

2. Alignments with a single orphan sequence.

3. Alignments comprising two distant groups of less than 25% sequence identity.

4. Alignments containing long insertions.

5. Alignments containing long deletions.

6. Sequence repeats.

7. Transmembrane sequences.

8. Domain permutation.

3.3 Research Problem

Progressive alignments (Section 2.2.3) use an approximation of a guide tree between the sequences as a guide tree that dictates the alignment order. The progressive strategy is appropriate for many alignment problems, but also suffers from greediness. Errors made in the first alignments during the progressive protocol cannot be corrected later as the remaining sequences are added in. Attempts to minimize such alignment errors have generally been targeted at global sequence weighting [81], where the contributions of individual sequences are weighted during the alignment process. However, such global sequence-weighting schemes carry the risk of propagating rather than reducing error when used in progressive multiple-alignment strategies [35].

Simultaneous alignments are high quality heuristics that deliver an alignment usually very close to optimality. Nonetheless, they remain an extremely CPU and memory-intensive approach, applicable only to about nine sequences of average length of 20 characters for the fastest implementation (DCA). From Figure 3.1 one can easily notice that the divide-and-conquer technique (Section 2.2.2) actually provides a perfect structure for parallel programming, and each sub problem can be computed independently. Another major advantage of using the divide-and-conquer technique is that extremely long sequences can be also acceptable by a multiple sequence alignment program as long as the sequences can be cut into small enough pieces.

Based on the characteristics of both progressive and divide-and-conquer alignments, long sequences will first be cut into several sets of sub-sequences, and each of these sub-sequences will be aligned progressively and independently by a light-weighted process, thread. In some cases, the sequences are extremely long and cannot be fed into a simultaneous alignment program even after they are divided into several shorter pieces. That's why sometimes progressive alignment will be still considered

for the sub-sequences alignments.

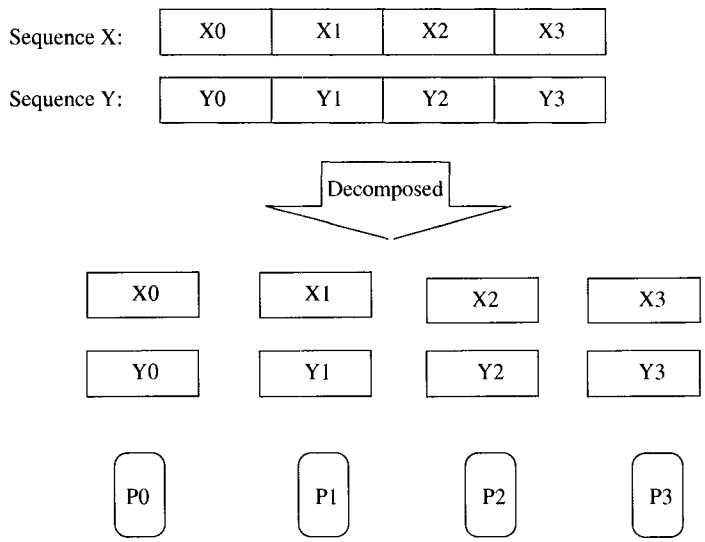


Figure 3.1: Divide-and-conquer technique

This multithreaded multiple sequence alignment approach actually combines the idea of divide-and-conquer alignment and progressive alignment. In order to check the alignment speed and sensitivity, two different alignment programs, depending on how the guide tree(s) would be applied, are developed for getting a better sense of which approach works better than the other.

3.4 Approaches and Simulations

The single-tree and multiple-tree alignment programs were implemented using shared memory Multithreaded Java Programming. There are two concepts for parallel processing, namely, processes and threads. A process is a single executable module that runs concurrently with other executable modules. For example, in a multi-tasking environment that supports processes, like Microsoft Windows, a word processor, an internet browser are separate processes and can run concurrently. Processes are separate executable, loadable modules as opposed to threads which are not loadable.

Multiple threads of execution may occur within a process. For example, from within a database application, a user may start both a spellin check and a time consuming sort. In the meantime, a thread can also be a task that runs concurrently with other tasks within a single executable file (e.g., within a single MS-DOS EXE file). Unlike processes, threads have access to common data through global variables.

Parallelism deployed in the program was based on the divide-and-conquer technique structure. On one hand it is clear that optimal cut positions exist; on the other hand it is clear that it is computational expensive to find them. Since the focus of this thesis is speed performance, only trivial idea are used now to determine the cut positions in the implementations, such as choosing the middle points recursively during each cut.

Since progressive alignment only performs global alignment and match sequences over their full lengths, problems with this approach can arise when highly dissimilar sequences are compared. Especially when there is a large difference in the lengths of the two sequences to be compared, global alignment routines become unwarranted. This is because highly similar internal regions may be overshadowed by dissimilar regions and the high gap penalties normally are required to achieve proper global matching. Moreover, many biological sequences are modular and show shuffled domains, and the repeats of internal sequence can also severely limit the applicability of global methods. Therefore, in the simulations, only long sequences with similar length and over 40% identical are tested for the single-tree and multiple-tree alignment programs.

3.4.1 Single-Tree Alignment

Single-tree alignment uses a uniform guide tree built for the full-length sequences at the beginning of the alignment. All the sub-alignment processes will follow this single tree when the sequences are cut into sub-sequences. Details of the single-tree alignment are presented in Figure 3.2.

The guide tree for single-tree alignment is built using UPGMA algorithm.

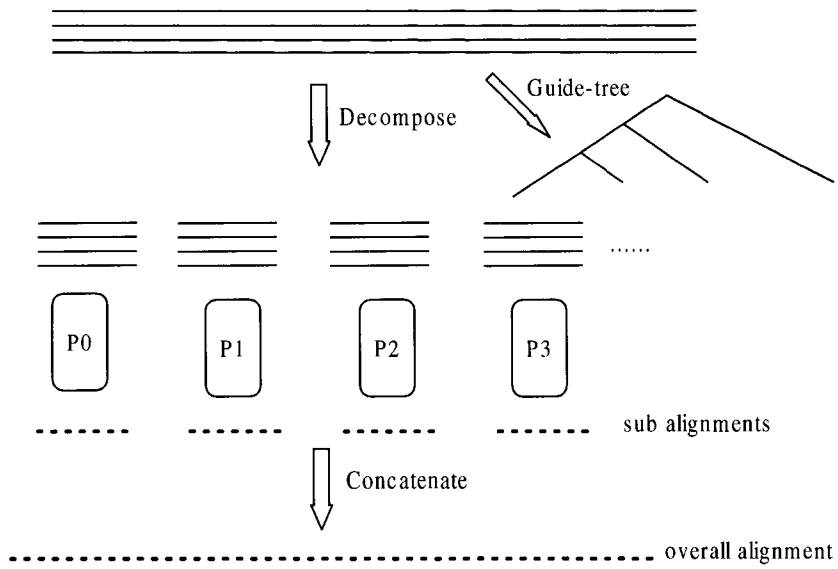


Figure 3.2: Details of single-tree alignment

3.4.2 Multiple-Tree Alignment

As opposed to single-tree alignment, multiple-tree alignment use different guide trees for each set of sub-sequences. Sequences will be cut into pieces first, then each set of the sub-sequences will build their own guide tree to guide their individual alignments. Figure 3.3 shows the details of multiple-tree alignment.

The guide tree for multiple-tree alignment is also built using UPGMA algorithm.

3.4.3 Speed Performance

One of the major advantages of multithreaded programming is program speedup with respect to time efficiency, because each thread processes a different piece of the same job simultaneously and independently. However, this is obviously the case for the multiple-tree alignment program, but not quite true for the single-tree alignment (Figure 3.4), which does not gain any speed improvement after some point. The reason is that in single-tree implementation, no matter how many threads are used, every time a single guide tree for the full length sequences are built at the start of alignment for all the sub-alignments done by different threads. It appears that

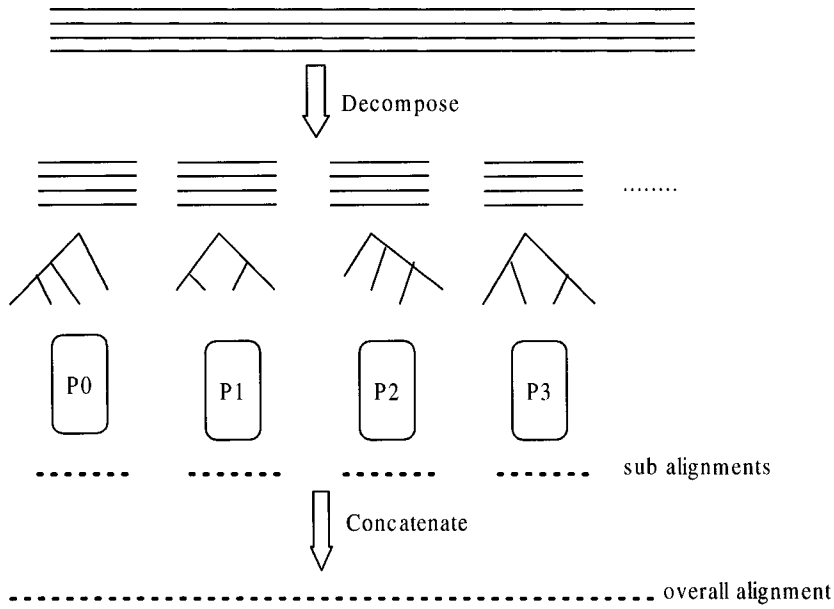


Figure 3.3: Details of multiple-tree alignment

building the uniformed alignment guide tree becomes the hot spot (the most time-consuming part) for all alignment procedures in the single-tree approach.

The figure also shows that the multiple-tree alignment program achieves seven times faster than the sequential program where one thread was used. We also note that the multithreaded multiple-tree algorithm is faster than the multithreaded single-tree algorithm. In other words, using the same number of threads, the multiple-tree approach performs faster and uses less memory.

3.4.4 Alignment Sensitivity

There are several publicly available databases which have benchmark alignments. A widely used one is BAliBase by Julie Thompson et al (Section 3.2). BAliBASE provides a module (BaliScore) that defines two scores. SP (Sum-of-Pair) score is the ratio of the number of correctly aligned pairs of positions in the test (predicted) alignment to the number of aligned pairs in the reference (structurally informed) alignment. TC (Total-Column) score is the ratio of the number of correctly aligned

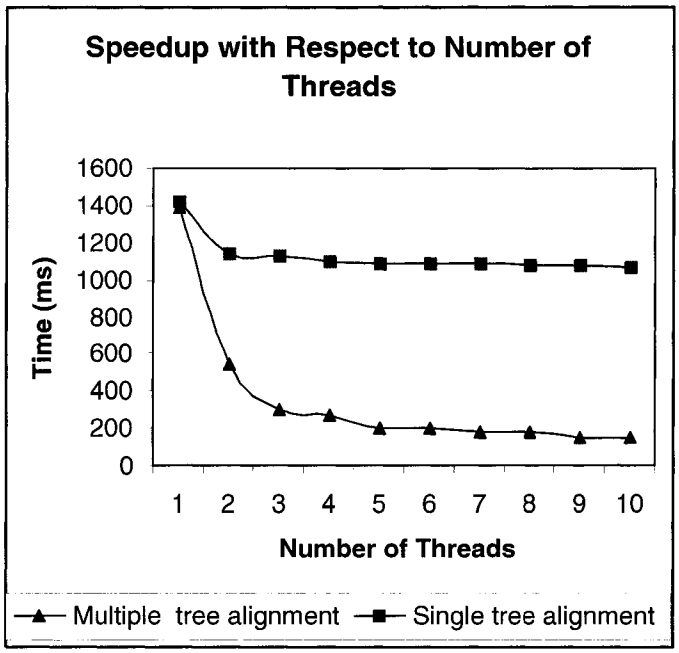


Figure 3.4: Speed improvement for single-tree and multiple-tree implementations in terms of different number of threads

columns in the test alignment to the number of aligned columns in the reference alignment. Both SP and TC scores range from 1.0 for perfect agreement to 0.0 for no agreement. The designers of BAliBASE recommend SP score as the best quality score for Refs1, 2 and 3, TC score as the best score for Refs4 and Refs5 [81].

Currently, the tests were done mainly based on Refs2, thus the following Figure 3.5 inflects the average SP scores calculated by BAliScore in terms of the number of threads used by the single-tree and multiple-tree alignment programs. It turns out that the quality of alignments drops down for both approaches when the number of threads increases, as unwanted gaps are inserted at the start or the end positions of the sub-alignments; This thus brings more gaps in the final full-length alignments and infects the values of SP scores.

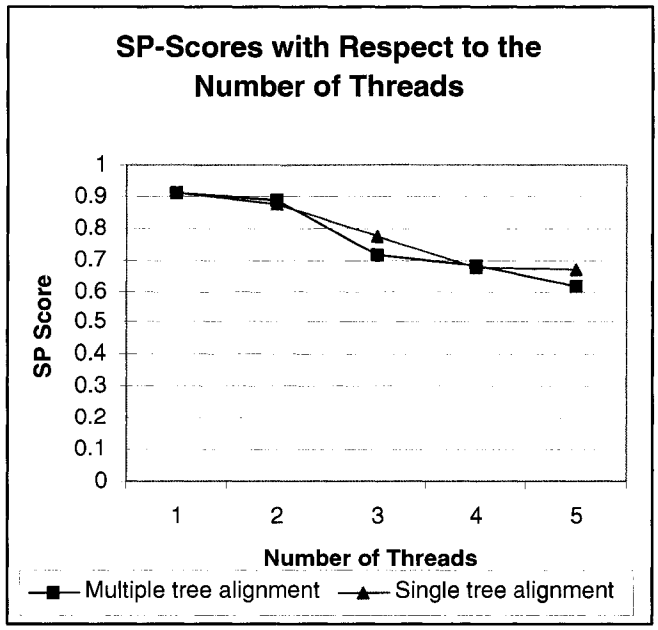


Figure 3.5: SP-scores for single-tree and multiple-tree implementations with respect to different number of threads

3.5 Sensitivity Improvements

3.5.1 Overlapping Alignment

This approach is considered to reduce the number of unwanted gaps introduced at the start or the end positions of the sub-alignments when multiple threads are used. In this approach, cut points are actually extended half length of the original sub sequence. Assume the length of original sub-sequence is t. The real length participating in the alignment is t + (t/2), where the (t/2) part overlaps the first half of its next neighbor. However, at the reassembling stage, those (t/2) overlapping parts will be ignored and only the first t characters in each subsequence will be extracted to build the final full length alignment.

3.5.2 Sliding Windows for Cut Points Calculations

When divide-and-conquer technique is applied for sequence alignment, finding an effective and efficient algorithm to calculate the cut positions for sequences is always a big issue. The basic idea of sliding window that we come up for cut position calculation is explained below.

Assume we have two sequences S1, S2, and N number of threads are going to be used for their alignment.

- Firstly, cut S1 evenly in terms of the number of threads, and mark the first cut position with 0. Then, mark the positions left to 0 with -1, -2, , -(S1/N)/2; and mark the positions right to 0 with +1, +2, , +(S1/N)/2
- Secondly, in S2 mark the same position with the same number as that in S1 accordingly
- For simplicity purpose, we assume that S1 has sliding window of size 2, and an optimal cut position for S2 will reside within 1 left or right shifts of position 0, the sliding windows for the first cut position of both sequences will look like below (Figure 3.6):

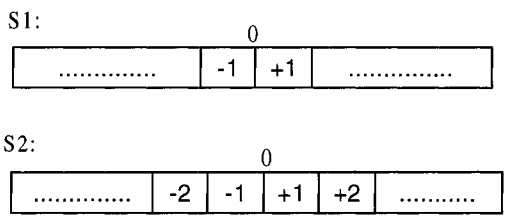


Figure 3.6: Illustrating the sliding window technique for improving alignment accuracy

The cut position of S2 will be computed in terms of the ***maximum*** score of the following alignments:

Score(0) = align [S1(-1,+1), S2(-1,+1)]

Score(-1) = align [S1(-1,+1), S2(-2,-1)]

Score(+1) = align [S1(-1,+1), S2(+1,+2)]

For example, if max = score(+1), then the first cut point of S2 will be in between cells +1 and +2.

Therefore, for general cases, if multiple sequences have similar length S and N number of threads will be used for the alignments, then the size of the sliding windows (thresholds) for all sequences will be less or equal than S/N, with half of the sliding cells being negatively marked and the other half positively marked. The maximum score calculated to get the cut position for next sequence upon the previous one will then be computed among score(0), score(-1), ..., score(-S/2N), score(+1), ..., ..., score(+S/2N), depending on the sliding window size of its previous sequence.

3.6 Conclusions

The same set of multiple sequences can be progressively aligned either by a single guide tree or multiple guide trees. The multiple-tree and single-tree alignment programs presented in this thesis are coming from the idea of combining the use of divide-and-conquer technique with the progressive alignment. Multiple-tree alignment seems having a better speedup performance than single tree alignment. The results also show that in terms of the number of threads used, using two threads has the best performance, which retains the similar accuracy as using one thread but accelerate the alignment by saving half of the time. However, neither the single tree program nor the multiple tree program shows very satisfying sensitivity results as the number of threads increases.

Chapter 4

Graph-based Consensus Multiple Gene Order Alignment

Multiple gene order alignment, sometimes called multiple genome alignment, is a larger-scaled multiple sequence alignment. In the last chapter, a multithreaded version for sequence alignment, to accelerate the alignment processing was introduced. This chapter will focus on the parallel processing techniques for multiple genome alignment. Previous approaches for gene-order analysis were largely based on breakpoint analysis or gene rearrangements, and lack parallelism properties. As a result, a novel algorithm for alignment of genome sequences based on a graph-based consensus method is described in this chapter. This algorithm can be applied to align two or more gene order sequences, obtain the ancestor gene order sequence, and be implemented in parallel, to further increase its computation speed.

4.1 Complexity Analysis

In multiple gene-order alignment, one edit operation consists of the inversion, or reversal, of any number of consecutive terms in the ordered set. In the case of signed orders, the operation also reverses the polarity of each term within the scope of the inversion. The calculation of the distance for unsigned genomes with inversions only

is NP-hard; for signed problem it is of polynomial complexity. For multi-chromosome genomes, another important edit operation is reciprocal translocation, representing the exchange of terminal fragments between two chromosomes. Some formulations of the distance problem for translocation are of polynomial complexity, and some are of NP-hardness. For the algorithm proposed here, the inversion operation is only considered for a single genome sequence.

4.2 Rationale for Graph Theory

Conceived by Euler, Cayley, and Hamilton, graph theory flourished in the twenties century to become a critical component of discrete mathematics. In the 1950s, Seymour Benzer applied graph theory to show that genes are linear [43]. When the Human Genome Project started, DNA sequencing was a routine but time-consuming and hard-to-automate procedure. In 1988 four groups of biologists independently and simultaneously suggested a different sequencing technique called *Sequencing by Hybridization*, abbreviated as SBH. SBH as a Hamiltonian Path Problem, and SBH as an Eulerian Path Problem lead to the graph algorithms for sequence reconstruction. Similarly, in this chapter, a graph algorithm for multiple gene order alignment will be introduced.

As mentioned at the beginning of this chapter, previous approaches to gene-order analysis were largely based on break-point analysis or gene rearrangements, and lack of parallelism properties. The algorithm that is going to be proposed here introduces a potentially simpler approach for multiple alignment of gene orders based on a notion of precedence. The algorithm can also be implemented in parallel to further increase its speed.

The original idea of this algorithm came from the mathematical and computational linguistics [71][67]. In this view, "gene order" can be compared with "canonical word order" for linguistic sequences [34]. The approach presented here is an extension of an earlier work for calculating the canonical word order based on word order and

precedence constraints [66].

4.3 The Precedence Graph-Based Consensus Algorithm

In the following an alternative algorithm is presented for aligning multiple genome sequences and constructing consensus sequences. Similarly to the other graph based approaches, a graph will be constructed over sequence fragments representing the data set. However, this is a different algorithm that does not explicitly try to calculate sequence alignments or overlaps. Instead, the graph will be traversed greedily, and a path will be tried to pursue through the graph so that each branch followed is as consistent with the previous one as possible. The goal of this alignment heuristic are to allow for more efficient gene order alignments that can accommodate for inversions, translocations, deletions, and insertions in the evolutionary process between ancestor and successor organisms. The precedence based gene order alignment algorithm presented here have a time complexity of $O(d * n^2))$ where d is the number of genomes being aligned and n is the number of gene blocks contained in the genomes.

4.3.1 The Minimum Spanning Tree

A weighted graph, also known as a network is a graph whose lines are weighted. The meaning of the weights depends on the application. For example, an airline might use a weighted graph to represent the routes between cities that it serves. In this example, the vertices represent the cities and the edges represent a route between two cities. The weight of the edge could represent the flight distance or the price of the flight between the two cities. The weight information, in this case, can be stored as the intersection value in an adjacency matrix. The representation of a simple network in adjacency matrix is shown in Figure 4.1.

A spanning tree is a tree that contains all of the vertices in the graph. The network

	A	B	C	D	E	F
A	0	6	3	0	0	0
B	6	0	2	5	0	0
C	3	2	0	3	4	0
D	0	5	3	0	2	3
E	0	0	4	2	0	5
F	0	0	0	3	5	0

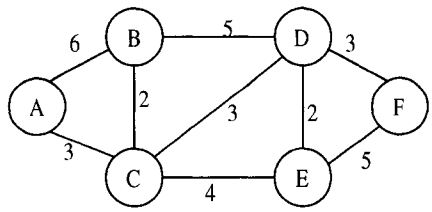


Figure 4.1: A simple network represented by a weighted graph and an adjacency matrix

shown in Figure 4.1 is also a **spanning tree** of the adjacency matrix.

Interesting algorithms, such as Kruskal's algorithm [47] and Prim's algorithm [63], derive the **minimum spanning tree** of a weighted graph such that the sum of its weights are guaranteed to be minimal. If the weights in the graph are unique, then there will be only one minimum spanning tree. Otherwise, there may be one or more minimum spanning trees. To create a minimum spanning tree in a strongly connected network, that is, in a weighted graph in which there is a path between any two vertices, the edges for the minimum spanning tree are chosen so that the following properties exist:

1. Every vertex is included.

2. The total edge weight of the spanning tree is the minimum possible that includes a path between any two vertices.

Figure 4.2 demonstrates how the spanning tree shown in Figure 4.1 is developed into a minimum spanning tree.

Similarly, in multiple gene order alignment, a precedence matrix is built for each

genome sequence. The intersection values interpret the relative positions of different genes that reside in the same genome, the details of which is described in the next section.

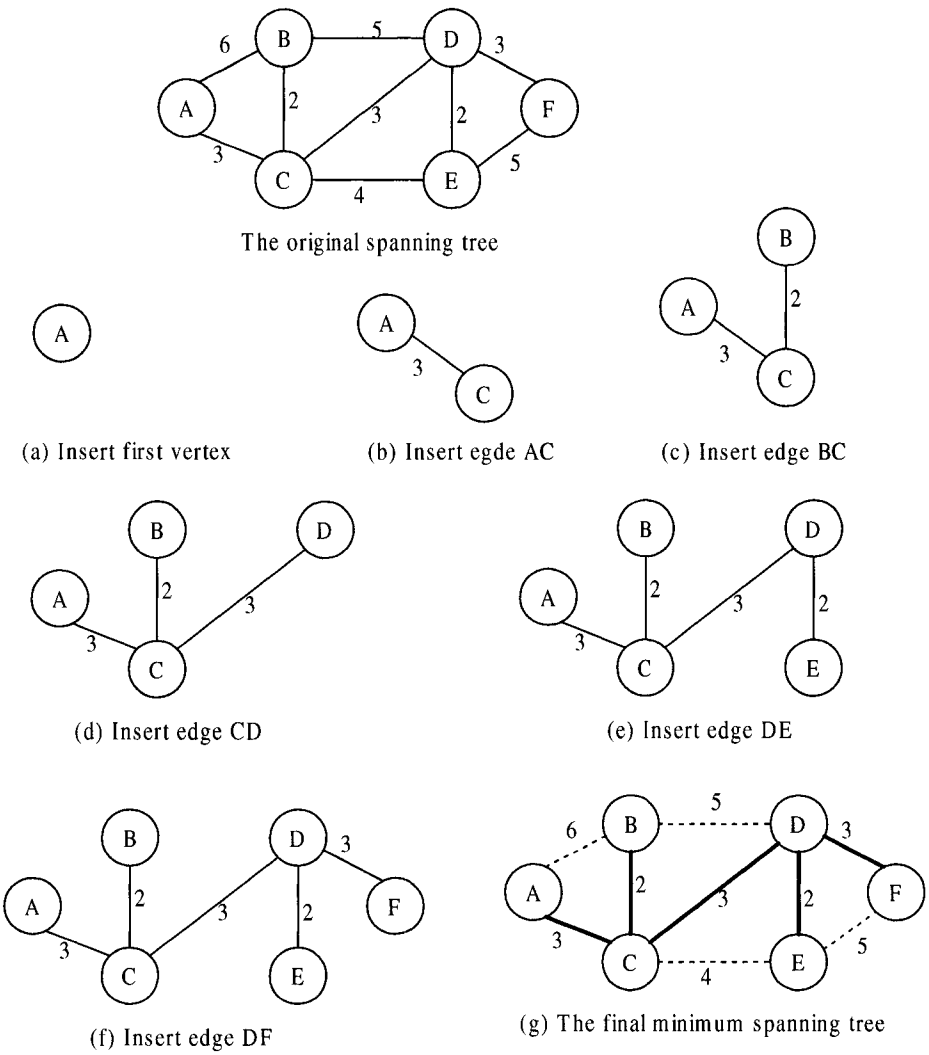


Figure 4.2: A simple network represented by a weighted graph and an adjacency matrix

4.3.2 An Example

This algorithm is based on *precedence distance* and contrasts with breakpoint or reversal distances. Consider the following three gene sequences:

a) 1 4 6 7 5 3 2
b) 1 4 6 5 7 2 3
c) 1 4 5 7 6 2 3

A gene order alignment of these sequences would like to be obtained. For this purpose a precedence matrix for each sequence is calculated. For the first sequence (a), its precedence matrix is:

	1	4	6	7	5	3	2
1	0	-1	-2	-3	-4	-5	-6
4	1	0	-1	-2	-3	-4	-5
6	2	1	0	-1	-2	-3	-4
7	3	2	1	0	-1	-2	-3
5	4	3	2	1	0	-1	-2
3	5	4	3	2	1	0	-1
2	6	5	4	3	2	1	0

In the matrix, each cell (i, j) has a corresponding value n, which specifies the distance between genes i and j in the same genome, and is distinguished by positive direction and negative direction. Positive n specifies gene i (the row list) recedes n positions from gene j (the column list); negative n specifies gene i precedes gene j by n positions; 0 simply shows gene i and gene j are the same.

The precedence matrix for the genome (b) is displayed as following:

	1	4	6	5	7	2	3
1	0	-1	-2	-3	-4	-5	-6
4	1	0	-1	-2	-3	-4	-5
6	2	1	0	-1	-2	-3	-4
5	3	2	1	0	-1	-2	-3
7	4	3	2	1	0	-1	-2
2	5	4	3	2	1	0	-1
3	6	5	4	3	2	1	0

Similarly, one can compute the precedence matrix for the third sequence (c):

	1	4	5	7	6	2	3
1	0	-1	-2	-3	-4	-5	-6
4	1	0	-1	-2	-3	-4	-5
5	2	1	0	-1	-2	-3	-4
7	3	2	1	0	-1	-2	-3
6	4	3	2	1	0	-1	-2
2	5	4	3	2	1	0	-1
3	6	5	4	3	2	1	0

Next, these three matrices are merged by adding corresponding values of (i, j) together to obtain the summarized matrix that results in the final alignment. For example, to calculate the value for cell (4,5) for the final matrix, which is in fact the distance between gene 4 and gene 5, the values of correspondent cell (4,5) in the three matrices are added up, i.e. (-3)+ (-2) + (-1). The value corresponding to cell (4,5) will then be -6.

The finally merged matrix is shown as following:

	1	2	3	4	5	6	7
1	0	-16	-17	-3	-9	-8	-10
2	16	0	-1	-13	7	8	6
3	17	1	0	14	8	9	7
4	3	-13	-14	0	-6	-5	-7
5	9	-7	-8	6	0	1	-1
6	8	-8	-9	5	-1	0	-2
7	10	-6	-7	7	1	2	0

Number 0 in the matrix does not necessarily represent two identical genes i and j now, as the addition operation is involved. Moreover, the resulted matrix is diagonally symmetric, thus only half of it needs to be calculated and the calculation for the other half is about copying the symmetric value and in the meantime, flipping the negative or positive sign accordingly. The merged matrix gives a precedence matrix from aligning the three sequences. If the aligned sequence can be reconstructed from this matrix, then the resulted multiple alignment for the three sequences will thus be obtained.

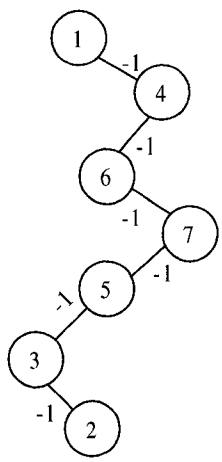


Figure 4.3: The minimum spanning tree built from the precedence matrix of sequence (a)

Taking a look at the precedence matrix corresponding to sequence (a), one can realize that traversing the vertices of **the minimum spanning tree** for this matrix will be 1,4,6,7,5,3,2 which represents exactly the original sequence (a) (Figure 4.3). The same thing happens for sequence (b) and (c) as well. In the ideal case, a directed graph is constructed, where each gene or gene block is a vertex, and edges represent distances between genes and gene blocks. A correct assembly of the original sequence is then constructed through this graph by pre-order traversing the minimum spanning tree of a precedence matrix of a genome sequence.

When coming to find out the consensus or ancestor of these three sequences, the same minimum tree traversal technique applies: calculating the minimum spanning tree for the final merged matrix, then the pre-order traversal of this tree will give the final aligned consensus sequence, or called the ancestor sequence for all (Figure 4.4).

The consensus sequence of sequence (a), (b), and (c) is: 1 4 6 5 7 2 3

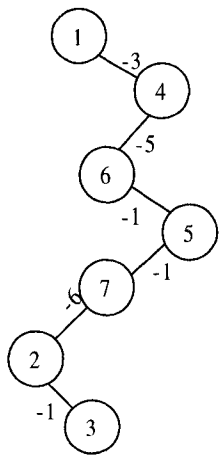


Figure 4.4: The minimum spanning tree built from the summarized final precedence matrix

4.3.3 Algorithm Formalization

Sequence alignment and gene order alignment consist of different operations. Sequence alignment is about aligning individual characters, and thus insertion and dele-

tion are involved. Gene order alignment cares more about gene order arrangement than words matching, and does not allow any change or modification applied to the current genes themselves. Another key difference between sequence comparison and gene order comparison is that in the former, gaps are introduced as required, whereas in the latter, the rearrangement of gene orders does not allow the insertion of gaps. Figure 4.5 shows the flow of the algorithm.

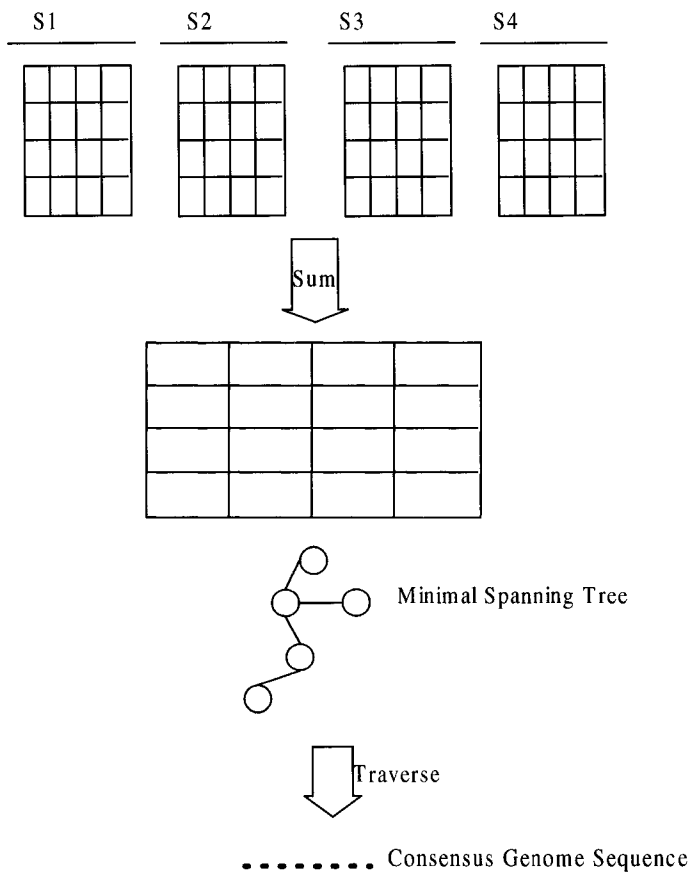


Figure 4.5: Illustrating the steps of the precedence graph-based consensus algorithm for genome sequence alignment

Based on a summarized precedence matrix for a set of genome sequences, a consensus genome sequence is constructed by building a minimum spanning tree through the matrix. The tree is traversed, trying to follow edges so that the vertices along the path are consistent with a subset of the genes at corresponding positions of the

sequences.

The algorithm takes as parameters all the genome sequences to be aligned. They will be used to build their own precedence matrix respectively. Assume there are n genome sequences, then n precedence matrix will be built accordingly for each of the sequences.

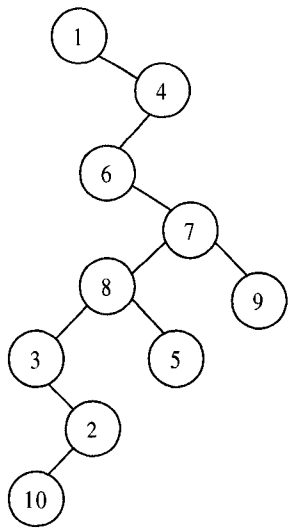
Building the final consensus matrix, the algorithm starts out by filling the first row and first column of the matrix with the union set of the genes appeared in all the sequences in sorted ascending order. The rest of the cells of the matrix will then be filled with the summarized value from the corresponding cells of the sequence matrices.

Then, the algorithm constructs a minimum spanning tree for the summarized matrix. Each time it picks the edge(s) in positive or negative direction with the possible minimum abstract value. By pre-order traversing the tree, a consensus sequence will be obtained, which is also the final alignment for all the sequences. Note that a slightly different final aligned sequence will be obtained depending on whether the tree is traversed in an increasing or decreasing order for each vertex of sub trees from right to left. Figure 4.6 shows an example for different final alignments obtained by different traversal orders.

Many algorithms and methods that have been widely used for current multiple genome sequence alignment. However, problems like repeated genes in the same sequence, different sequence containing different kinds of genes, and different sequence with different number of genes, are not considered in most of the current genome alignment software, such as GRIMM [92].

The graph-based consensus algorithm, a completely different approach from the existent anchor-based genome alignment algorithms and methods, can be used to align unsigned multiple genomes with some or all of the following properties:

- Genomes contain repeated genes



Traversing the above tree in decreasing pre-order, obtain:
1, 4, 6, 7, 9, 8, 5, 3, 2, 10

Instead, tracersing it in increasing pre-order, obtain:
1, 4, 6, 7, 8, 3, 2, 10, 5, 9

Figure 4.6: An example showing how to obtain different final alignments by traversing the minimum spanning tree in different orders

- Genomes are of different length
- Genomes consist of different kinds of genes

However, at this stage, the algorithm is at its early stage of evolution, and does not give any suggestions for aligning signed genome sequences. For example, the two orientations i and -i of gene i are treated as two completely different genes regardless of the fact that one is the reversal of the other. Besides, the algorithm now does not automatically change the sign of gene -i or i during the alignment. Gene -i and i will retain their original signs throughout the alignment process.

4.4 Parallelism of the Algorithm

The graph-based algorithm for multiple gene order alignment is straightforward, simple and easy to perform, and contains four steps:

1. Calculate an initial precedence matrix for each individual genome sequence.
2. Summarize the individual matrices and get the final matrix for consensus sequence.
3. Build a minimum spanning tree in terms of the final matrix.
4. Traverse the minimum spanning tree to get the consensus sequence.

The step that can be performed in parallel is the first step where the initial precedence matrix calculation for each genome sequence is independent of each other. This step would become the most time-consuming part of the algorithm if the number of genome sequences is fairly large and the computation is performed sequentially. In another words, simultaneously calculating the initial precedence matrices for all the sequences will tremendously reduce the computation time and increase the speed.

4.5 Evaluations and Comparisons

To the best of our knowledge, there is not yet a standard reference database used as benchmark for multiple gene order (or genome) alignment. But there are still ways available for evaluating the sensitivity or accuracy of an alignment generated by certain algorithms or software.

4.5.1 Evaluation without a Reference alignment

Without a correct alignment of the multiple genomes, an alignment calculated by any algorithm or method cannot be evaluated for accuracy. In fact, no manually generated multiple alignment benchmark data sets account for genome-scale evolutionary events

such as inversion, rearrangement, and horizontal transfer. Despite the lack of correct alignments, the alignment accuracy can be estimated by modeling evolution and aligning simulated data sets.

The inferential power yielded by evaluating alignment accuracy using simulated evolution is only as strong as the degree to which the simulation faithfully represents the actual evolutionary processes that governed the history of the genomes under study. In order to be able to evaluate a genome alignment, at lease a simplistic model of genome evolution needs to be constructed, which captures the major types, patterns, and frequencies of events in the history of the related genomes. Building such a genome evolution model involves many complex elements, such as a rooted phylogenetic tree, an ancestral sequence, evolved sequences, regions conserved throughout the simulated evolution. To effectively represent genome evolution, the simulation must include nucleotide substitutions and indels (insertion and deletion) in addition to genome-scale events such as horizontal transfer, inversion, and rearrangement.

4.5.2 Evaluation against a Reference alignment

Due to the complexity involved in evaluating genomes without a reference alignment, current evaluations are mostly done with a reference alignment based on what is available. There are two main schemes for comparing a proposed multiple genome alignment to a reference multiple genome alignment: the gene score and the sum-of-pairs score (different to the SP score mentioned in the previous section).

It is well known that pairwise alignment optimize residue exchange scores and gap penalties. Extending the pairwise sequence scores to get a single score for a multiple alignment would be an obvious way of scoring multiple alignments. This is referred to as the **Sum-of-Pairs** (SP) score for alignment, which is used widely by multiple sequence alignment. The sum-of-pairs (SP) is a common scoring scheme, where the score of each pair of sequences of the multiple alignment is added up to form the overall score. The higher the SP-score is, the better the multiple alignment is. For example, assume there are 4 sequences S1, S2, S3, S4. The sum-of-pairs score of the

alignment for all the 4 sequences is:

SP(S1, S2, S3, S4) = score(S1, S2) + score(S1, S3) +score(S1, S4) + score(S2, S3) + score (S2, S4) + score(S3, S4)

The gene score of a multiple genome alignment (MGA) is calculated by comparing the alignment genes of the proposed MGA with those in the corresponding reference MGA, and only those identical ones will be taken as correct ones. This is a more salient measure than the sum-of-pairs scores, where over all observed aligned gene pairs in a reference MGA, the fraction of those observed in the corresponding target MGA is compiled. Whereas, a single misaligned sequence can zero the gene score, the SP score only gradually goes down with more misaligned sequences. Note that the SP scoring system here involves two MGAs, and is therefore different than the previously mentioned SP scoring system for a single MGA without a reference.

As for the reference alignment, currently, there are quite a few multiple genome alignment software that is available online or free for downloading, such as Mauve [94], MGA [95], M-GCAT [96], GRIMM [92], CHAINER [91], and MALGEN(2.0) [93].

4.6 Validation

As mentioned above, the evaluation of multiple gene order (genome) alignment could be done with or without a reference alignment. Due to the complexity of evaluation without a reference alignment, most of the contemporary validations are done with a reference alignment. Since this algorithm is at its early stage of evolution and does not automatically change the signs of the genes, or in another words, does not consider the self-reversal of the genes, genes will retain their original signs throughout the alignment process. As a result, it would not be very ideal to evaluate the accuracy of the graph-based consensus algorithm using any reference alignments that are gener-

ated by various anchor-based algorithms that have taken the genes' self-reversal into consideration. Some heuristics and modifications need to be applied to this algorithm to automatically take genes' self-reversal into consideration, and thus make it more reasonable to be compared with any other reference alignments.

Fortunately, based on this work Pringle et al. [62] extended the graph-based approach proposed in this thesis and developed a more advanced precedence graph-based consensus approach and parallelized it utilizing the Message Passing Interface System. In order to accommodate for the self-reversal, insertion and deletion cases, a presence matrix has been added to our approach, which basically keeps track of the number of contributions that have been made to a particular gene pair distance. Pringle et al. also took weight distances into consideration by applying a weight function to the cells of precedence matrix based on the count and the distance sum of a particular gene pair, and penalizing precedence cells that do not have full participation from all gene orders being aligned.

In the evaluation of [62], two metrics were considered, the timing of the program and accuracy of the results it produces. The program was timed and the accuracy of a particular gene order arrangement was determine from the average percentage displacement between genes in the gene alignment and gene in the proper result. As well, the presence or absence of genes was considered between the final and alignment in the accuracy tests.

Pringle et al. shows that there is a linear relation between the timing of the utility and the number of genomes being considered by our proposed graph-based method. In the mean time, there is a polynomial relation between the size of the genomes and the program timing which is what was expected from the time complexity analysis.

Pringle et al. also showed that the accuracy of the results obtained by our proposed approach seemed to remain constant when switching between the original precedence system, the weighted system and the weighted penalty system. The accuracy of the results seems to indicate that our graph-based approach can handle deletions and inversions quite well.

4.7 Conclusion

Precedence gene order alignment is a promising approach to genome alignment that offers a below exponential solutions to the problem. The research in multiple sequence analysis can be compared to the developments in the study of canonical word order in linguistics [34] [66]. The multiple gene order analysis can be compared with a level of analysis in linguistics where the focus is on diachronic change in structure and syntax rather than lexicon. The precedence graph-based consensus algorithm for calculating a multiple alignment for gene orders is a linguistically motivated approach in the area of multiple genome alignment, which is completely different from the existing anchor-based alignment algorithms and methods. The core of the algorithm is that pre-order traversing the minimum spanning tree of a precedence matrix of a genome sequence constructs the original sequence. The advantage of this method is partly efficiency, since it computes the final alignment by computing a minimum spanning tree from a two-dimensional matrix and finding a path through the minimum spanning tree, which is computable in n^2 time, while computing the final alignment from the anchored-based approaches for the unsigned genome, the complexity is more than polynomial. However, This original method of precedence calculation is valid in the cases of translocation and reversal mutations between gene orders and a common ancestor but in the case of deletions or insertions it can cause a false minimal edge in the precedence matrix which could have ill effects on the result of the alignment.

Chapter 5

Discussion and Future Directions

This thesis focused on improving the computational speed of the multiple biological sequence alignment, including aligning characters (for DNA, RNA and protein sequences) and aligning gene orders (for genome sequences) in computational molecular biology. Some issues related to multiple alignment accuracy and evaluation were also introduced and discussed.

Sequence alignment deals with comparing different DNA or different protein sequences. This is done by writing one on top of the other padding them with spaces ("indel", for insertion or deletion) to achieve identical length. In DNA, the criterion to distinguish among the many possibilities of this arrangement is the number of unequal letters ending up on top of each other minus the number of spaces that were introduced. For protein sequence comparison the pairs of matched letters are weighted and the adjacent spaces are summarized into blocks which receive a penalty.

Alignments may be performed on a pairwise basis, across multiple sequences or they can involve the alignment of a sequence to a previously aligned set of sequences (sometimes called a profile). The information exhibited by a multiple sequence alignment allows the deduction of putative structural and functional features. Through a reliable multiple sequence alignment of a set of homologous sequences, the evolutionary pathway, corresponding to mutations as well as insertions and deletions of sequence fragments, can often be traced under the model of divergent evolution.

Based on the mixed idea of Progressive alignment and Divide-and-conquer alignment, two different multithreaded multiple sequence alignment programs, depending on how the guide tree(s) would be applied, were implemented for checking the improvements of alignment speed and sensitivity. The single-tree alignment built a uniform guide tree for the full-length sequences at the beginning, which were used by all the sub-alignments as the guide tree. In the multiple-tree alignment, sequences were firstly cut into pieces and these sub-sequences built their own guide trees to guide their individual alignments. Multiple-tree alignment seemed having a better speedup performance than the single tree alignment, but neither of them, at this stage, showed ideal sensitivity results as the number of threads increases. Therefore, some heuristic methods for fixing the cut points were suggested for future improvement, such as overlapping alignment and sliding window alignment.

Gene order alignment is a larger-scale sequence alignment. The key difference between gene order analysis and sequence analysis is that sequence analysis is at the gene level (i.e. individual character mutations), and gene order analysis is at the chromosome or genome level (i.e. gene order mutations). High-throughput DNA sequencing technology has enabled researchers to rapidly determine the genome sequences of a wide variety of organisms, laying the foundation for comparative genomics [15]. Gene order analysis, sometimes called Genome sequence analysis, help scientists make sense of diverse elements in the genome, understanding how they are organized within the genome of each species, and characterizing the changes in genome organization during evolution. Comparative genomics plays an important role in making inferences and gathering information specific to the evolution of a species or genetics diseases. It provides insights to important tasks such as identifying regions of homogeneity or regions of genetic anomalies. The research in multiple gene rearrangement has been the basis for development of algorithms for multiple gene order alignment [22].

There are similar complexity problems with multiple gene order alignment (or referred as multiple genome alignment) algorithms. Besides, the existing anchor-based gene order alignment algorithms and methods lack the parallelism properties for

computational speedup. As a result, a precedence graph-based consensus algorithm was formalized for aligning two or more genome sequences, which is very original, and was able to efficiently obtain the ancestor genome sequence for multiple genome sequences. The algorithm argues for a potentially simpler approach for multiple gene order alignment based on a notion of precedence derived from the canonical word order studies in linguistics. This algorithm also possesses the parallelism properties to be implemented in parallel for further speed improvement. However, at the current stage, this algorithm does not consider the self-reversal of genes during the alignment process, and thus some improvement and extension of this work needs to be done in the future.

5.1 Contribution

In this thesis, two levels of parallelism are considered for multiple biological sequence alignments to accelerate the alignment processes: one is at the sequence level (aligning the characters in a DNA, RNA or protein sequence), the other one is at the gene order level (aligning the gene orders for a genome sequence). To the best of our knowledge, it is a novel algorithm to progressively align the same set of multiple sequences in parallel by building different number of phylogenetic guide trees. In the meantime, the precedence based gene order alignment approach is a promising first step in non-exponential gene order approaches. Its timing was shown to have a cubic time complexity and the approach overall was shown to have a high to moderate accuracy range. Other than anchor-based algorithms and methods, it is the first time that a graph algorithm was developed for multiple genome (gene order) alignment. Briefly, the contribution can be summarized as the following:

- Sequence level: multithreaded techniques together with different tree-building techniques was deployed specifically for the multiple alignment of extremely long sequences. It turns out that the multiple-tree alignment program achieves seven times faster than the sequential program where one thread was used.

- Gene order level: opposed to the existing complex anchor-based algorithms and methods for handling gene order alignments that lack parallelism properties and cannot handle gene insertion, deletion, and duplications, a simpler and novice precedence graph-based consensus algorithm that can be parallelized is formalized to speed up the alignment process, which is also able to handle gene insertion, deletion, and duplications.

5.2 Future Directions

5.2.1 Multiple Sequence Alignment

The multiple-tree and single-tree alignment implementations described in this thesis presented ideal efficiency improvement for multiple sequence alignment. The optimal multi-threaded approach is with two threads, where the alignment sensitivity is retained and the speed is increased by saving half of the alignment time. However, neither of the multiple-tree implementation nor the single-tree implementation shows very satisfying sensitivity results as the number of threads increases. Therefore, improving their computation efficiency and in the meantime maintaining their sensitivity performance is still an issue. The Following three ways could be considered and investigated in the future.

1. Firstly, some more effective calculations should be found and performed to decide the cut points.

2. Secondly, weights could be considered and given to the sequences: down weighting the sequences that are very similar to other ones in the data set and up weighting the most divergent sequences. The weights could be calculated directly from the branch lengths in the initial guide tree and the guide tree guides all the subsequent alignments.

3. Thirdly, affine gap penalties and varying substitution matrices may be applied dynamically in the progressive alignment.

5.2.2 Multiple Gene Order Alignment

The original method proposed in this thesis for determining precedence simply uses the sum of relative positions of gene pairs across all gene orders to determine that gene orders precedence value in the final precedence matrix. Though Pringle et al. [62] has made a considerate contribution to it, there is still a lot of room for improving its performance, such as:

1. Modifying the algorithm and making it be able to consider the self-reversal of genes during alignment process. Some heuristics may be applied.

2. Clarifying the ways for minimum spanning tree traversal: whether increasing pre-order traversal or decreasing pre-order traversal gives a better result.

3. Evaluating its alignment results against the reference alignments generated by other approaches.

4. Developing a parallelized version of this algorithm for arbitrary number of genome sequences and making it publicly available.

5. Currently this metric does not handle insertions that well so future implementations may better utilize the presence matrix to increase its overall accuracy in the case of insertion mutations.

Bibliography

[1] M. I. Abouelhoda, and E. Ohlebusch. A Local Chaining Algorithm and its Applications in Comparative Genomics. Proceedings of the 3rd Workshop on Algorithms in Bioinformatics, 2812:1-16, 2003.

[2] M. I. Abouelhoda, and E. Ohlebusch. CHAINER: Software for Comparing Genomes. In 12th International Conference on Intelligent Systems for Molecular Biology/3rd European Conference on Computational Biology.

[3] S. Aluru , N. Futamura and K. Mehrotra. Parallel Biological Sequence Comparison Using Prefix Computations. J. Parallel and Distributed Computing, 63(3):264-272, 2003.

[4] A. Apostolico , M. J. Atallah , L. L. Larmore and S. MacFaddin. Efficient Parallel Algorithms for String Editing and Related Problems. SIAM J. Computing, 19(5):968-988, 1990.

[5] Arun K. Iyengar, Parallel Characteristics of Sequence Alignment Algorithms, In the Proceedings of the ACM/IEEE conference on Supercomputing, Pp. 304-313, Reno, United States,1989.

[6] P. Baldi, Y. Chauvin, T. Hunkapiller, and M. A. McClure. Hidden Markov models of biological primary sequence information. Proc. Nat. Acad. Sci. 91:1059-1063, 1994.

[7] S. Batzoglou, L. Pachter, J. P. Mesirov, B. Berger and E. S. Lander. Human and Mouse Gene Structure: Comparative Analysis and Application to Exon Prediction. Genome Research, 10:950-958, 2000.

[8] M. P. Berger and P.J. Munson. A Novel Randomized Iteration Strategy for Aligning Multiple Protein Sequences. Computer Applications in the Biosciences, 7:479-484, 1991.

[9] G. Bourque and P. A. Pevzner, Genome-Scale Evolution : Reconstructing Gene Orders in the Ancestral Species, Genome Res. 12(1):26-36, Jan 2002.

[10] M. Brudno, and B. Morgenstern. Fast and sensitive alignmetn of large genomic sequences. Lecture notes in Computer Science. 138-147, 2002.

[11] M. Brudno, S. Malde, A. Poliakov, C. B. Do, and et. al. Global alignment: finding rearrangements during alignment. Bioinformatics, 19(1):154-162, 2003.

[12] M. Brudno, C. B. Do, and G. Cooper. Langan and nulti-lagan: Efficient tools for large-scale multiple alignment of genomic DNA. Genome Res. 13:721-731, 2003.

[13] L. Cai, D. Juedes and E. Liakhovitch. Evolutionary Computation Techniques for mutiple sequence alignment. Congress on evolutionary computation 2000, 829-835, 2000.

[14] J. Cheetham, F. Dehne, S. Pitre, A. Rau-Chaplin, and P. J. Taillon. Parallel CLUSTAL W For PC Clusters. International Conference on Computational Sciences and Its Applications, 2003.

[15] A. E. Darling, B. Mau, M. Craven, and N. T. Perna. Multiple alignment of rearranged genomes. Proceedings of the 2004 IEEE Computational Systems Bioinformatics Comference (CSB 2004), 748-749, 2004.

[16] A. E. Darling, B. Mau, F. R. Blattner, and N. T. Perna. Mauve: multiple alignment of conserved genomic sequence with rearrangements. Genome Research, 14:1394-1403, 2004.

[17] David Sankoff. Historical linguistics as a stochastic process. PhD thesis, McGill University, 1969

[18] David Sankoff, Mathematical developments in lexicostatistical theory. In T. A. Sebeok (Ed.), Current trends in linguistics 11: Diachronic, areal and typological linguistics, 93-112. The Hague: Mouton.

[19] A. L. Delcher, S. Kasif, R. D. Fleischmann, J. Peterson, O. White and S. L. Salzberg. Alignment of Whole Genomes. Nucleic Acids Research 27:2369-2376, 1999.

[20] A. L. Delcher, A. Phillippy, J. Carlton and S. L. Salzberg. Fast Algorithms for largescale genome alignment and comparison. Nucleic Acids Research 30:2478-2483, 2002.

[21] J. S. Deogun, J. Yang, and F. Ma. EMEGEN: an efficient approach to multiple whole genome alignment. The 2nd Asia Pacific Bioinformatics Conference (APBC2004), Dunedin, New Zealand. Conferences in Research and Practice in Information Technology, 29:113-121, 2004.

[22] T. Dobzhansky and A. Sturtevant, Inversions in the chromosomes of Drosophila pseudoobscura, Genetics 23:28-64, 1938.

[23] S. R. Eddy. Multiple alignment using hidden Markov models. Third international conference on intelligent systems for molecular biology (ISMB). Cambridge England: Menlo Park CA: AAAI Press (1995).

[24] R. C. Edgar. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. BMC Bioinformatics, 5: 113, 2004.

[25] R. C. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Research 32(5):1792-97, 2004.

[26] E.W. Edmiston, N.G. Core, J.H. Saltz and R.M. Smith, Parallel processing of biological sequence comparison algorithms. International Journal of Parallel Programming, 17(3), Pp. 259-275, 1988.

[27] Jenome Gracy and Jean Sallantin. Multiple sequence alignment using anchor points through generalized dynamic programming. Proceedings of Genome Informatics Workshop 1994. Universal Academic Press, Tokyo, 1994.

[28] R. R. Gonzalez. Multiple protein sequence comparison by genetic algorithms, SPIE-98, 1999.

[29] O. Gotoh. Significant improvement in accuracy of multiple protein sequence alignments by iterative refinements as assessed by reference to structured alignments. J. Mol. Biol., 264:823-838, 1996.

[30] S. Hannenhalli and P. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). In Proc. 27th Annual ACM Symposium on the Theory of Computing, (a):178-189, 1995.

[31] S. Hannenhalli and P. Pevzner. Transforming men into mice (applications of comparative physical maps in molecular evolution). The proceedings of the 7th annual AVM-SIAM symposium on discrete algorithms, Philadelphia: SIAM, 304-313, 1996.

[32] S. Hannenhalli and P. Pevzner. To cut or not to cut (polynomial algorithm for sorting signed permutations by reversals). In Proc. 27th Annual ACM Symposium on the Theory of Computing, (a):178-189, 1995.

[33] D. Haussler, A. Krogh, I. S. Mian, and K. Sander. Protein modeling using hidden markov models: analysis of globins. Proceedings for the 26th Hawaii International Conference on Systems Sciences. Wailea HI U.S.A.: Los Alamitos CA: IEEE Computer Society Press, 1993.

[34] J. Hawkins. A Performance Theory of Order and Constituency, Cambridge University Press, 1994.

[35] J. Heringa. Two strategies for sequence comparison: profile-preprocessed and secondary structure-induced multiple alignment. Comput. Chem., 23(3-4):341-64, 1999.

[36] P. Hogeweg, B. Hesper. The alignment of sets of sequences and the construction of phyletic trees: an integrated method. J Mol Evol., 20(2):175-86, 1984.

[37] M. Hohl, S. Kurtz, and E. Ohlebusch. Efficient Multiple Genome Alignment. Proceedings of the Tenth International Conference on Intelligent Systems for Molecular Biology, Bioinformatics, 18(S1):S312-S320, 2002.

[38] X. Huang, A space-efficient parallel sequence comparison algorithm for a message passing multiprocessor, International Journal of Parallel Programming, 18(3), Pp 223-239, 1989.

[39] R. Hughey, A. Krogh. Hidden Markov models for sequence analysis: extension and analysis of the basic method. Computer Applications in Biological Science, 12:95-107, 1996.

[40] M. Ishikawa, M. Hoshida, M. Hirosawa, T. Toya, O. Kentaro, and K. Nitta. Protein Sequence Analysis Program: Multiple Sequence Alignment by Parallel Iterative Aligner. Demonstrations Int'l Conf. Fifth Generation Computer Systems, Tokyo, 57-62, 1992.

[41] M. Ishikawa, M. Hoshida, M. Hirosawa, T. Toya, O. Kentaro, and K. Nitta. Protein Sequence Analysis Program by Parallel Inference Machine. Proc. Int'l Conf. Fifth Generation Computer Systems, Tokyo, 294-299, 1992.

[42] James W. Fickett, Fast Optimal Alignment, Nucl. Acids Res. 12(1):175-179, 1984.

[43] N. C. Jones and P. Pevzner, An Introduction to Bioinformatics Algorithms, MIT Press, 2004.

[44] W. J. Kent and A. M. Zahler. Conservation, Regulation, Synteny, and Introns in a Large-scale C. briggsae-C. elegans Genomic Alignment. Genome Research 10:1115-1125, 2000.

[45] J. Kim, S. Pramanik and M. J. Chung. Multiple sequence alignment using simulated annealing. Comp. Applic. Biosci., 10:419-426, 1994

[46] J. Kleinjung, N. Douglas, J. Heringa. Parallelized multiple alignment. Bioinformatics, 18:1270-1271, 2002.

[47] J. B. Kruskal. On the shortest spanning subtree and the traveling salesman problem. In: Proceedings of the American Mathematical Society, 7:4850, 1956.

[48] E. Lander, J.P. Mesirov and W. Taylor, Protein sequence comparison on a data parallel computer, In the Proceedings of the International Conference on Parallel Processing , Pp.257-263, 1998.

[49] T. Lassmann, E. L. Sonnhammer. Quality assessment of multiple alignment programs. FEBS Lett., 529(1):126-30, 2002.

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

[51] H. Carrillo and D. J. Lipman. The multiple sequence alignment problem in biology. SIAM J. Appl. Math., 48(5):1073-1082, 1988.

[52] D. Lipman, S. Altschul, and J. Kececioglu. A tool for multiple sequence alignment. In Proc. Natl. Acad. Sci. U.S.A., 86:4412-4415, 1989.

[53] W. S. Martins, et al. A Multithreaded Parallel Implementation of a Dynamic Programming Algorithm for Sequence Comparison. Pacific Symposium on Biocomputing, 6:311-322, 2001.

[54] C. Michener and R. Sokal. A quantitative approach to a problem in classification. Evolution, 11:130-162, 1957.

[55] D. Mikhailov, H. Cofer, and R. Gomperts. Performance Optimization of Clustal W: Parallel Clustal W, HT Clustal and MULTICLUSTAL: SGI, 2001.

[56] M. Monwar, and S. Rezaei. Parallelized Multiple Biological Sequence Alignment with MPI: The Divide-and-Conquer Approach. To appear in the proceedings of the IMECS 2006 Conference, Hong Kong, June, 2006.

[57] B. M.E. Moret, D.A. Bader, T. Warnow, S.K. Wyman, and M. Yan. GRAPPA: a high-performance computational tool for phylogeny reconstruction from gene-order data. Botany 2001, Albuquerque, NM, August 12-16, 2001.

[58] B. Morgenstern. DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. Bioinformatics, 15(3):211-8, 1999.

[59] C. Notredame. Recent progress in multiple sequence alignment: a survey. Pharmacogenomics, 3:131-144, 2002

[60] C. Notredame, D. Higgins, J. Heringa. T-Coffee: A novel method for multiple sequence alignments. Journal of Molecular Biology, 302:205-217, 2000

[61] C. Notredame, D. Higgins. SAGA: sequence alignment by genetic algorithm. Nucleic Acids Res., 24:1515-1524, 1996.

[62] R. Pringle, and S. Rezaei. Precedence Based Multiple Gene Order Alignment. To appear in UNBC working papers in Bioinformatics, 2006.

[63] R. C. Prim. Shortest connection networks and some generalisations. In: Bell System Technical Journal, 36:1389-1401, 1957.

[64] S. Rajko and S. Aluru. Space and Time Optimal Parallel Sequence Alignments. IEEE transactions on parallel and distributed systems, 15(12):1070-1081, 2004.

[65] S. Ranka and S. Sahni. String Editing on an SIMD Hypercube Multicomputer. J. Parallel and Distributed Computing, 9:411-418, 1990.

[66] S. Rezaei. Fuzzy Word Order Constraints. Constraints vs. Preferences workshop, Poznan, Poland, 1999.

[67] S. Rezaei. Linguistic and Computational Analysis of Word Order and Scrambling in Persian, PhD thesis, Edinburgh University, Feb. 1999.

[68] S. Rezaei, and M. Monwar. Divide and Conquer Algorithm for ClustalW-MPI. To appear in the proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering, Ottawa, May, 2006.

[69] N. Saitou, and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. Molecular Biology Evol, 4(4):406-425, 1987.

[70] D. Sankoff and M. Blanchette. Multiple genome rearrangement and breakpoint phylogeny. Journal of Computational Biology, (5)555-570, 1998.

[71] D. Sankoff. Historical linguistics as a stochastic process. PhD thesis, McGill University, 1969.

[72] D. Sankoff. Mathematical developments in lexicostatistical theory. In T. A. Sebeok (Ed.), Current trends in linguistics 11: Diachronic, areal and typological linguistics, 93-112. The Hague: Mouton.

[73] D. Sankoff and M. Blanchette, Multiple genome rearrangement, RECOMB, 1998.

[74] D. Sankoff, Genome rearrangement with gene families. Bioinformatics, 15:909-917, 1999.

[75] M. Schmollinger, K. Nieselt, M. Kaufmann, and B. Morgenstern. Dialign P: fast pair-wise and multiple sequence alignment using parallel processors. BMC Bioinformatics, 5:128-135, 2004.

[76] V. A. Simossis, J. Kleinjung and J. Heringa. Homology-extended sequence alignment. Nucleic Acids Research, 33(3):816-824, 2005.

[77] P. Sneath and R. Sokal. Numerical Taxonomy. Freeman, San Francisco, 1973.

[78] Stjepan Rajko, Space and Time Optimal Parallel Sequence Alignments, In the Proceedings of the International Conference on Parallel Processing (ICPP'03), 2003.

[79] Jens Stoye. Multiple sequence alignment with the divide-and-conquer method. Gene, 211:GC45-GC56, 1998.

[80] J. Stoye, S. W. Perry, and A. Dress. Improving the divide-and-conquer approach to sum-of-pairs multiple sequence alignment. Appl. Math. Lett., 10(2):67-73, 1997.

[81] J. Thompson, F. Plewniak, et al. BAliBASE: A Comprehensive comparison of multiple alignment programs. Nucleic Acids Research, 27(13):2682-2690, 1999.

[82] J. D. Thompson., D. G. Higgins, and T. J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. Nucleic Acids Research, 22:4673-4680, 1994.

[83] J. D. Thompson, T. J. Gibson, F. Plewniak, F. Jeanmougin, and D. G. Higgins. The ClustalX windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. Nucleic Acids Research, 24:4876-4882, 1997.

[84] D. G. Higgins, J. D. Thompson and T. J. Gibson. Using CLUSTAL for multiple sequence alignments. Methods Enzimol., 266:237-244, 1996.

[85] J. Thompson, F. Plewniak, et al. BAliBASE (version 2.0): A benchmark alignment database, including enhancements for repeats, transmembrane sequences and circular permutations. http://www-igbmc.u-strasbg.fr/BioInfo/BAliBASE2/, 2003.

[86] T. Treangen, and X. Messeguer. M-GCAT: multiple genome comparison and alignment tool. The 5th Annual Spanish Bioinformatics Conference(JBI 2004), 30-33, 2004.

[87] L. Wang, and T. Jiang. On the complexity of multiple sequence alignment. J. Comp. Biol. 1(4):337-348, 1994.

[88] W. Wilbur and D. Lipman, Rapid similarity searches of nucleic acid and protein data banks, Proc. Nat. Acad. Sci. USA, 726-730, 1983.

[89] T. K. Yap, O. Frieder, and R. L. Martino. Parallel computation in biological sequence analysis. IEEE transactions on parallel and distributed systems, 9(3):283-293, 1998.

[90] C. Zhang and A. K. Womg. A genetic algorithm for multiple molecular sequence alignment, compt. Appl. Biosci., 13:561-581, 1997

[91] CHAINER: Software for Comparative Genomics. http://theorie.informatik.uni-ulm.de/Personen/mibrahim/chainer.html

[92] GRIMM: Genome Rearrangement Algorithms. http://www.google.ca/firefox?client=firefox-ar&ls=org.mozilla:en-US:official

[93] MALGEN: Multiple ALignment of GENomes. http://alggen.lsi.upc.es/recerca/align/intro-align.html

[94] Mauve: Multiple Genome Alignment. http://gel.ahabs.wisc.edu/mauve/

[95] MGA: Multiple Genome Aligner. http://bibiserv.techfak.uni-bielefeld.de/mga/

[96] M-GCAT: Multiple Genome Comparison and Alignment Tool. http://alggen.lsi.upc.es/recerca/align/mgcat/intro-mgcat.html