Contents

1

Comparisons of Long Genomic Sequences: Algorithms and Applications

Michael Brudno
University of Toronto

Inna Dubchak
Lawrence Berkeley National Laboratory

1.1 Introduction

Comparing genomic sequences across related species is a fruitful source of biological insight, because functional elements such as exons tend to exhibit significant sequence similarity due to purifying selection, whereas regions that are not functional tend to be less conserved. The first step in comparing genomic sequences is to *align* them – that is, to map the letters of one sequence to those of the others. There are several categories of alignments: *local alignments* identify local similarities between regions of each sequence, *global alignments* find a mapping between all the letters of the sequences. Alignments can be either *pairwise*, between two sequences, or *multiple* that compare several sequences. The main challenge in developing algorithms for genomic alignment is that these must be fast enough to deal with megabase

0-8493-8597-0/01/$0.00+$1.50

long sequences and gigabase long genomes, but also accurately map individual base pairs. While generating alignments is difficult computationally, visualization of alignments also presents challenges, such as how to enable users to interact with the data and the processing programs in the context of enormous datasets. Visualization frameworks should be easy to understand by a biologist and provide insight into the mutations that a particular region has undergone. Finally, alignments are useful only if they help shed light on the important functional elements in the genomic sequence. In this chapter, after a detailed discussion of algorithms used to construct genomic alignments and methods to visualize them we give a short overview of several algorithms that use an alignment to improve predictions of transcription factor binding sites.

1.2 Local Alignment

Local alignment is the basic problem of finding similar fragments in two sequences, regardless of the order and location of these similarities. Consequently local alignments allow one to identify rearrangements between two sequences, and are suitable for aligning draft sequences. The original local alignment algorithm is the Smith-Waterman [62] dynamic programming approach. This algorithm, however, runs in time proportional to the product of the lengths of the sequences. As this is impractical for comparing two long genomic intervals there has been extensive work since the mid 1980s on development of fast approaches for local alignment of genomic sequences. While the details of all of these approaches are different, most share some overriding paradigms. In particular, almost all algorithms start with seed generation – the location of short, exact or nearly exact matches between two sequences. Because this can be accomplished quickly by indexing one of the two sequences in an appropriate data structure, such as a lookup table or some variant of a suffix tree, these seeds help to reduce the search area of the local alignment algorithm to just the regions that are likely to be similar. Once generated, nearby seeds may be joined together: the presence of several seeds close to each other is a stronger evidence of homology than a single seed. Finally the individual seeds (or groups of seeds) are extended to find regions that did not match exactly but are still conserved. These three steps form the basis of most local alignment algorithms for DNA sequences; the individual algorithms differ in how they solve each of the steps.

1.2.1 Seed generation

Perhaps the simplest way to generate seeds between two sequences is a straight lookup table technique: all k-long words (k-mers) of one sequence (the database) are indexed in a table, and the k-mers of the other sequence (the query) are used to retrieve from the lookup table the locations at which the particular k-mer of the query is present in the database sequence. This approach was used in the two first (and perhaps the best known) local aligners for long sequences, FASTA [53] and BLAST [1, 2]. An alternative approach for seed generation is to use a suffix tree or one of its variants (such as Aho-Corasick automaton) to search for the seeds. This approach suffers from higher constant overhead, in terms of both the memory requirements and the running time, but allows for the use of longer k-mer matches as it takes advantage of sparseness of longer seeds: there are 4^k possible DNA words of length k, and only a small fraction of them may be present in a particular sequence. This makes suffix tree approaches preferable when one is searching for longer seeds, while direct lookup tables are preferable for shorter ones. Instead of searching for k-long words that match exactly between the two sequences it is possible to use degenerate words as seeds, that is, words where a certain number of the letters are allowed not to match. These seeds allow for a higher sensitivity than exact matching k-mer seeds, but are more computationally intensive to generate. These seeds can be found in one of several ways, all of which, however, lead to an exponential running time increase in the number of degeneracies: it is possible to create indices where all of the possible degenerate positions are absent (i.e. index all possible 7 bases within an 8-mer). This leads to each word being represented only once in an index, but the number of indices grows as $\binom{k}{n}$ where n is the number of degeneracies allowed. Alternatively, one can index each word at all of its degenerate locations. This way there is only one index, but each word is indexed in many places, resulting in a blowup in the amount of memory required for the index. A popular alternative to both of these techniques is known as a "spaced" seed, which was initially implemented in the PatternHunter program [46]. These seeds are similar to degenerate seeds, in that they allow certain positions to not match, but these positions are pre-specified. For example a 110101 seed requires 4 (1st, 2nd, 4th and 6th) out of 6 positions to match. The other two positions may not match. This pattern is referred to as a (4,6) spaced seed. Because the degenerate positions are known ahead of time it is possible to use just a single index to look up all seeds. PatternHunter seeds have been shown to be preferable to regular fixed-length k-mers because two adjacent k-mers will no longer share $k-1$ positions, reducing the correlation between adjacent words. If in the case of exact matching seeds it is sufficient to introduce a mutation every $k-1$ positions in order to prevent the algorithm from finding a seed, this is not sufficient in the case of spaced seed. In fact, for the optimal (9,15) seed 111001010011011 it is necessary to have a mutation at least every 7 base pairs to prevent a single seed from being found.

PatternHunter seeds are preferable to exact matching k-mers when the stretch of similarity between the two sequences is long; their efficacy drops off when one is comparing sequences with very short ($<$ 30 bases) stretches of homology. Another way to generate seeds is to use k-mers extended to the first mismatch, which are called max-mers or maximal extended matches (MEMs) of a certain minimal length. Max-mers can be found in strictly linear time in the sequence length by using a suffix tree (see chapter 5). While max-mers do not offer a sensitivity improvement over k-mers (wherever there is a k-mer there is also a max-mer and vice-versa) they have the advantages of returning only a single seed for every stretch of exact matches, no matter how long. When one is comparing very close sequences, such as two primates or two strains of bacteria, this will lead to a reduced number of seeds that need to be analyzed. However, when one is comparing more distant sequences (such as humans and mice) most of the matches tend to be short, and MEMs do not offer a significant reduction in the number of seeds, while the extra overhead of finding MEMs makes k-mers or PatternHunter seeds preferable. Additional theoretical work on seeds has included vector seeds [11] and random projections [16], but these have not been used in any alignment program.

1.2.2 Joining of neighboring seeds

While one seed can be a good indicator of homology, several smaller seeds located "near" each other can be an even better indicator. This idea has been used by many programs, starting with the first heuristic local aligner FASTA, that required m seeds of length k within a certain window to start a local alignment. It has also been common practice in BLAST and several other programs to search for two nearby seeds (spaced by a few base pairs) before starting an extension. These approaches are commonly implemented by creating a lookup table or hash table for all of the diagonals of the dynamic programming matrix, numbering all of the diagonals from 1 to X+Y for two sequences of length X and Y. Once a new seed is found, its diagonal is looked up the table. If there is already a seed

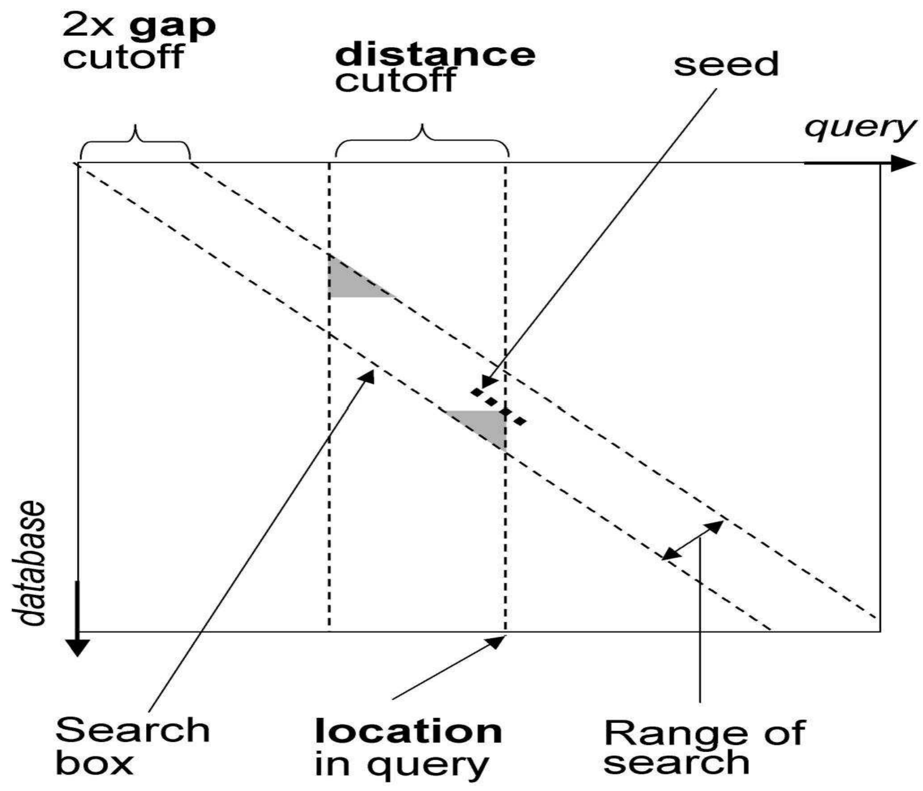


FIGURE 1.1: The CHAOS algorithm: The seed shown can be chained to any seed which lies inside the *search box*. All seeds located less then *distance bp* from the *current location* are stored in a skip list, in which we do a range query for seeds located within a *gap* cutoff from the diagonal on which the current seed is located. The seeds located in the grey areas are not available for chaining to make the algorithm independent of sequence order

linked to the diagonal, and the seed is close enough to the new one, the two seeds are joined together.

An alternative approach is used in the CHAOS program [12]. While similar to FASTA, it allows for groups of seeds to be on nearby diagonals. Each seed, once found, is stored in a skip list (a probabilistically balanced binary tree), indexed by the difference of its indices in the two sequences (diagonal number). For each new seed a range query is done in the skip list, searching for previously stored seeds which have a diagonal number within the permitted gap penalty from the diagonal at which the current seed is. The distance criterion is enforced by removing from the skip list any seed that is too far away from the position at which the new seeds are generated. This finds the possible previous seeds with which the current one can be chained. The highest scoring chain is picked, and it can then be further extended by future seeds (see Figure 1.1).

1.2.3 Ungapped and gapped extension

Once the seeds or groups of seeds are found, it is common practice to do some form of extension of the seeds in order to find the boundary of the homology. For this the most common approach is to do gap-free extensions which were first introduced in the original BLAST program. Here one keeps on adding one letter from each sequence to the already existing alignment, keeping track of the total score. The score goes up if the two letters match, and goes down otherwise. Once the score has fallen significantly lower than the maximum the extension is stopped and the alignment with the maximum score is returned. This process is commonly referred to as BLAST, or ungapped extension and is used in all of the early versions of the BLAST algorithm, as well as the CHAOS program. The alternative to this approach is a Smith-Waterman extension, where one is allowed to introduce gaps around the seeds. This process is much more time consuming, and is most useful when comparing distant genomic sequences. It is used in the BLASTZ alignment program [56], where gapped extensions are only triggered for ungapped alignments meeting a particular scoring threshold.

1.3 Global alignment

Global alignments find the correspondence between two strings end-to-end by building a monotonically increasing map between the letters of each sequence. This produces a more accurate alignment at the base-pair level when the conserved features (for example genes) are in the same order in the compared species. The original global alignment algorithm is Needleman-Wunsch [51], which requires time proportional to the product of the lengths of the aligned sequences. This algorithm is too inefficient for comparing megabase long genomic sequences. Faster and more accurate methods have been developed recently: DIALIGN [50, 12], MUMmer [22, 23], GLASS [4], WABA [38], AVID [9] and LAGAN [14]. All these methods rely on an anchoring approach. It is worth noting that the anchors in global alignment serve the same purpose as the seeds in local alignment – to reduce the inherently quadratic search space. The overall approach followed by all of the tools mentioned above (as well as several others) can be summarized as 1) generate the fragments (local stretches of similarity) between the sequences, 2) resolve the set of fragments into the highest scoring consistent set using the sparse dynamic programming approach or some alternative, and 3) run a thorough global alignment algorithm either between the anchors, or in some region around them.

1.3.1 Finding potential anchors

The next chapter (chaining) shows how to construct the highest scoring consistent chains from fragments. When these algorithms are applied in the context of global alignment, the immediate question is how these fragments should be found, and what is a meaningful fragment. Perhaps the most straightforward method is to use k-mers as these fragments, as was done in the GLASS alignment program. Because individual k-mers are not a reliable guide to homology they were supplemented with a short extension by running the Needleman-Wunsch algorithm in a 12×12 window around each k-mer. Simultaneously the authors of MUMmer program suggested the use of maximum unique matches (MUMs) as the fragments used for the chaining algorithm. MUMs are maximum exact matching strings between two sequences that appear exactly once in each of the sequences. The fact that the MUM is a unique word in each of the sequences, reduces the probability of a false positive match. The disadvantage of this approach is its inability to find anchors between divergent genomes, where the maximal exact matches are usually too short to be unique.

These two ideas were combined in the AVID program, which searches for maximal exact (not necessarily unique) matches, and does 12×12 Needleman-Wunsch windows around each match to verify its quality.

The most recent approach for generating the fragments has been the use of full local alignments. The first implementation of this idea was in the DIALIGN/CHAOS combination [12], where CHAOS local alignments were used to narrow down the search space of the DIALIGN program. CHAOS was subsequently used in a similar way in the LAGAN program, while BLAST local alignments were used in the ORCA program (Arenillas and Wasserman, unpublished).

Other approaches have leveraged biological knowledge in the process of anchor selection, with two prominent examples being the WABA [38] and CONREAL [5] programs, which search for anchors that are likely protein coding regions and transcription factor binding sites, respectively. In the case of WABA, the anchors between two sequences are “wobblemers”, or k-mers where every third position is allowed not to match. Wobblemers are equivalent to PatternHunter seeds of the form 110110110...110. Allowing the third base pair to mutate more accurately reflects the conservation pattern of protein coding sequences as mutations in the third base pair of a codon are likely to be “silent”: they will not change the amino acid being coded. This allows the wobblemers to be an effective method for anchoring protein regions. The CONREAL program, on the other hand, searches for potential transcription factor binding sites using the TRANSFAC suite of programs [47]. Two potential binding sites for the same transcription factor in the two sequences are used as the potential anchor points. While both WABA and CONREAL have been shown to be successful for their main purpose, that is alignment of protein coding and promoter regions, respectively, they are not general purpose alignment tools. Using a model of conservation rooted in a biological phenomenon usually does not work as well at modeling a different type of conservation: WABA will not work well for promoters and CONREAL will not align gene coding regions accurately.

1.3.2 Building a consistent set of anchors

The simplest way to find a consistent set of local alignment hits so that they can be used as anchor points is to use a greedy approach, where the strongest match (one with highest score using some scoring function) is accepted first, and every subsequent match is included if it does not conflict with any of the previous ones. This, however, leads to the problem of incorrect anchor points when several slightly weaker similarities may be ignored because of one strong one. Still, several alignment programs, including OWEN [54], use this approach.

Eppstein and colleagues [26] showed that it was possible to find the highest scoring consistent set of fragments in time $O(nlogn)$ via a sparse dynamic programming chaining procedure. For practical reasons it is not uncommon to do several rounds of anchoring: at the first step just the most confident fragments are fixed, a second pass fixes the less confident ones, etc. This approach, commonly known as hierarchical anchoring is useful because it allows one to search for lower levels of conservation: using a small seed size while looking at long sequences would lead to a very large number of seeds that are difficult to process. Because fewer anchors are found in each pass, hierarchical anchoring allows for an accurate analysis of each potential anchor.

1.3.3 Filling in the gaps

Once the set of anchors is fixed it is possible to reduce the search space of the final, slow global alignment algorithm to just the areas consistent with the anchors; usually Needleman-

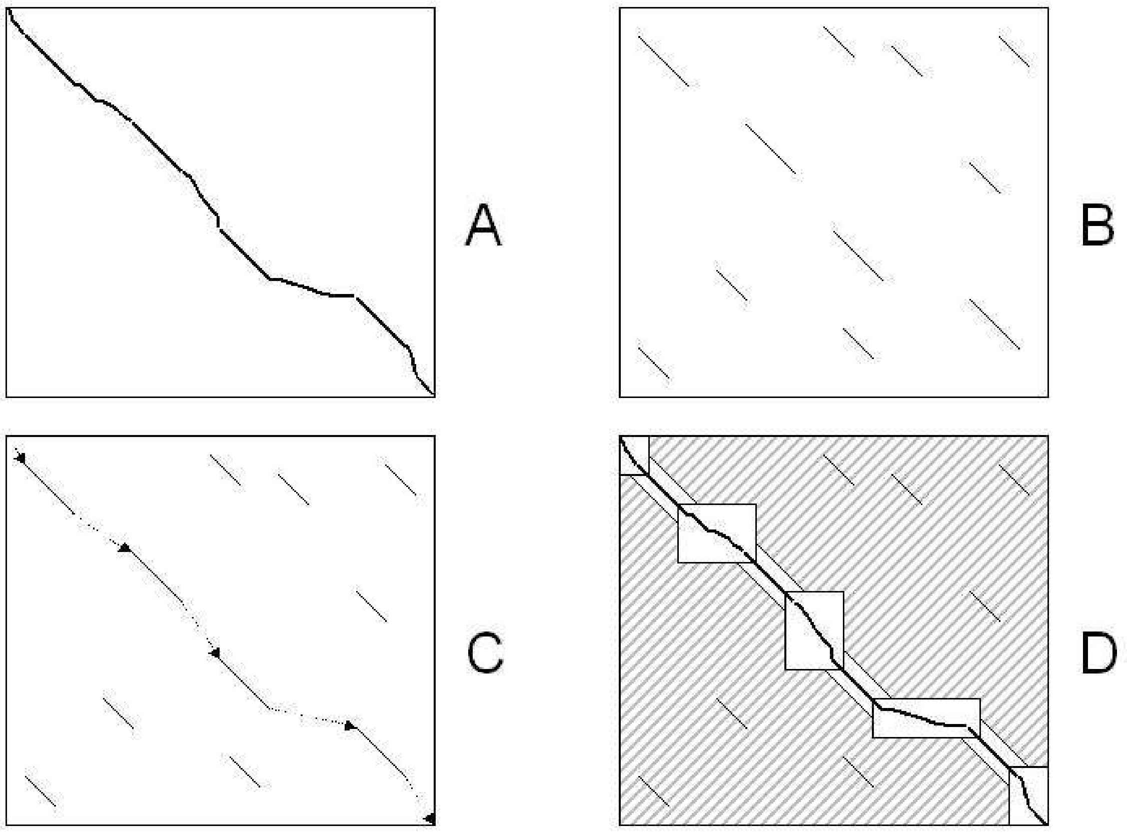


FIGURE 1.2: General scheme for most global alignment algorithms: (a) optimal map between the sequences (target) (b) potential anchors are found (c) highest scoring increasing subset of them is located (d) dynamic programming is done in the limited area. (From [14]).

Wunsch is used, though the CHAOS/DIALIGN combination uses the DIALIGN program. Both k-mer and max-mer matches can be used as "hard" anchors: because these consist only of matches with no gaps, the optimal alignment has to go exactly over them. By comparison, local alignments are not reliable hard anchors, as a local alignment may have small errors. For this reason LAGAN implemented a more flexible anchoring scheme, where the anchor is not fixed, but rather noted, and the global alignment is required to go near the anchor, though not necessarily exactly over it (Figure 1.2).

1.4 Multiple Global Alignment

Similarity across large evolutionary distances can reveal conserved, and likely important biological features [31, 35, 65]. More recently it has been shown that it is unnecessary to compare distant species: it suffices to compare many close ones, and through the process of phylogenetic shadowing it is possible to separate the conserved regions from the neutral ones [7]. Comparative analysis, however, depends on *multiple alignments*. Multiple alignments have been shown to be more powerful than pairwise ones both in that they give a higher resolution of conserved regions [17] and a higher overall accuracy alignment [14]. Multiple alignments also allow for estimates of local rates of evolution that give quantitative measures of the strength of evolutionary constraints and the importance of functional elements [61, 63, 19]. Multiple alignments are considerably more difficult to compute than pairwise alignments: the running time scales as the product of the lengths of all the sequences. Formally, the problem is NP-complete under several formulations [68, 8]. For this reason heuristic approaches are usually applied. Below we will first discuss the problem of scoring a multiple alignment, and then talk about two possible ways of obtaining a multiple alignment, the first via multi-way anchoring and the second via progressive alignment.

1.4.1 Scoring a multiple alignment

Perhaps the most basic, and at the same time the most non-trivial issue with multiple alignment is how to score it. The most common method used is the sum-of-pairs scoring, where the score for a particular column is set to just the sum of all the pairwise substitution and gap events. Alternatively it is possible to use a consensus model: for every column one finds the most likely character and penalizes divergence from the character. Another approach is to measure entropy in the column (see [25] for a summary). It is also possible to combine these approaches: LAGAN, for example, uses sum-of-pairs scoring for matches and mismatches and consensus for gaps. The problem of accurately scoring a multiple alignment remains very much an open one. Some innovative solutions have been published recently, such as T-COFFEE [52], and a full probabilistic framework for modeling multiple sequence alignments was implemented in HANDEL [34].

1.4.2 MGA and DIALIGN alignment algorithms

In order to generate global alignments of long genomic sequences it is necessary to generate anchor points between several sequences. For this only one truly multiple method has been suggested: multi-MEMs, that is multiple exact matches between all of the sequences being aligned. Because it is possible to find the highest scoring consistent chain in an arbitrary number of sequences in sub-quadratic time (see next chapter) these multi-MEMS can be used to reduce the search space for a multiple alignment algorithm. In between the anchor points it is necessary to run a sensitive multiple sequence alignment method to align the individual base pairs. This approach was first implemented in the MGA [33] program. The drawback to the MGA approach is that the requirement of each anchor being present in all sequences is too strict when one is comparing distant sequences. Because it is hard to find enough anchor points, the running time of the procedure becomes prohibitive. Consequently the MGA aligner, while being the first true global multiple alignment method for long genomic sequences is suitable only for comparing very close homologs, such as different strains of a bacterium.

Another approach for construction of multiple alignments from pairwise ones was implemented in the DIALIGN program [50]. The DIALIGN alignment consists of segments of ungapped homology (diagonals). To create a multiple alignment the set of pairwise diagonals is sorted according to the weights of the diagonals in a greedy way. Diagonals are incorporated one by one into the multiple alignment starting with the diagonal of maximum weight, provided they are not contradictory with the diagonals already incorporated. Diagonals contradictory with the growing set of consistent diagonals are rejected. The result of this selection process is a consistent set of diagonals – i.e., a multiple alignment.

1.4.3 Progressive alignment

The most common approach used for multiple alignment of several sequences is a progressive strategy, which constructs a multiple alignment by successive applications of a pairwise alignment algorithm. The best known system based on progressive alignment is perhaps CLUSTALW [66]. Some other systems include MULTALIGN [3], MULTAL [64], and PRRP [29]. The basic idea behind this approach is that pairwise alignment techniques can be generalized from alignment of two sequence to alignment of two profiles, that is sequences where each position consists of some fraction of A's, C's, T's and G's rather than an individual letter. Because an alignment can be thought of as a profile (the gap can, for instance, be treated as a fifth character) it is possible to generate a multiple alignment via a bottom-up traversal of the phylogenetic tree, where after generating the alignments corresponding to a node's left and right children, one aligns these to get an alignment for the node itself.

Anchoring an alignment of two profiles

The progressive approach described in the previous paragraph can be used to build an alignment of two profiles in time proportional to the product of their lengths. This, however, is too slow for long sequences, and anchoring approaches have been used to reduce the running time of this problem in a very similar manner to the pairwise anchoring problem described above. The MLAGAN multiple alignment program uses all pairwise local alignments between the sequences to generate the set of anchor points. For example, given the sequences X and Y, the alignment between them X/Y and a third sequence Z, the anchors between X/Y and Z are computed as follows: First, all anchors in the rough global maps between X and Z, and between Y and Z, are mapped to their coordinates in the X/Y alignment and become anchors between X/Y and Z, with score equal to their original score. Second, each anchor between X and Z that overlaps an anchor between Y and Z, is reweighed with score equal to $(s1 + s2)\ I/U$, where $s1$, $s2$ are the scores of the (X, Z) and (Y, Z) anchors, respectively, I is the length of intersection, and U is the length of union of the anchors (summed in X/Y and Z). The rough global map between X/Y and Z is the highest scoring consistent chain of these anchors. An alternative method was suggested by the authors of the MAVID program [10]: they use phylogenetic methods to predict the likely ancestor of two sequences, and use the sequence of the ancestor as the representative of the sequences in the progressive steps.

1.5 Alignment of Sequences with Rearrangements

One common way in which the genome evolves is through rearrangement (shuffling) of blocks of DNA. Some of the most common rearrangement events are inversions (a block of DNA changes direction, but not location in the genome), translocations (a piece of DNA moves to a new location in the genome), and duplications (two copies of a block of DNA appear where there was one previously). Of the two main methods for alignment, global alignment, which shows how one sequence can be transformed into another using a combination of the simple edits, and local alignment, which identifies local similarities between regions of sequences, neither handles rearrangement events satisfactorily. Global alignment algorithms do not handle these events at all: the map between the two sequences that a global alignment algorithm creates must be monotonically increasing. While local alignment methods are able to identify homology in the presence of rearrangements between two sequences, they do not suggest how the two sequences could have evolved from their common ancestor.

Also, in the case where both sequences have n paralogs (copies) of a particular gene or feature, local aligners return n^2 local alignments between all of the pairs, whereas a simple global alignment more clearly reflects the evolutionary process.

Varre and colleagues proposed a distance metric between two DNA sequences that models various rearrangement events [67]. Their algorithm, called Tnt1, builds a second sequence from an initially empty string using insertions and copying of blocks from the first sequence. The distance between the two strings is defined to be the Kolmogorov complexity of the program that builds the second sequence. This algorithm has several shortcomings, the most notable being its inability to handle simple edit operations. It was also too slow to have practical applications to genomic sequences. The first programs for alignment of long genomic sequences with rearrangements were Shuffle-LAGAN [13] for pairwise sequences and Mauve [21] for multiple sequences. These are now discussed in turn.

1.5.1 Pairwise alignment with Shuffle-LAGAN

The Shuffle-LAGAN (S-LAGAN algorithm) was built on the LAGAN global alignment framework but allowed for rearrangements using a novel chaining technique. The S-LAGAN algorithm consists of three distinct stages. During the first stage the local alignments between the two sequences are found using the CHAOS tool. Second, the maximal scoring subset of the local alignments under certain gap penalties is picked to form a *1-monotonic conservation map.* It is the structure of this map that makes S-LAGAN different from standard anchored global aligners. Finally, the local alignments in the conservation map that can be part of a common global alignment are joined into *maximal consistent subsegments*, which are aligned using the LAGAN global aligner.

Building the 1-Monotonic Conservation Map

Most tools for rapid global alignment start with a set of local alignments, which they resolve into a "rough global map" – the set of anchors described in section 3.2. The rough global map must be non-decreasing in both sequences. In order to allow S-LAGAN to catch rearrangements, this assumption is relaxed to allow the map to be non-decreasing in only one sequence, without putting any restrictions on the second sequence. This is called a 1-monotonic conservation map.

To build this map we first sort all of the local alignments based on their coordinates in the base genome. For every next alignment we chain it to the previous one that gives the highest overall score subject to the affine chaining penalties. The penalty enforced depends on whether the previous alignment is on the same or different strand than the previous one, and whether it is before or after it in the coordinates of the second sequence (roughly speaking the four cases correspond to regular gap (same strand, after), inversion (different strand, after), translocation (same strand, before), and inverted translocation (different strand, before). By using the Eppstein-Galil sparse dynamic programming algorithm we can reduce the running time of this chaining procedure to $O(nlogn)$ from $O(n^2)$. The resulting highest scoring chain is 1-monotonic (strictly increasing in the base genome, but without any restrictions on the second genome order). The 1-monotonic chain can capture all rearrangement events besides duplications in the second genome.

Aligning Consistent Subsegments

Two local alignments are considered to be consistent if they can both be a part of a global alignment. Once we have a 1-monotonic conservation map it is straight-forward to generate the maximal consistent subsegments of the map by simply sorting all of the local alignments in the 1-monotonic map by their coordinates in the 1-monotonic sequence, taking the first alignment to be the start of a consistent subsegment, and adding additional local alignments while they are all consistent. As soon as an alignment is found to be inconsistent with the current subsegment, we start a new subsegment. Every consistent subsegment is extended to the nearest adjacent local alignment, so as to include areas of homology that did not fall into the local alignment, and are aligned using LAGAN. The overlap between adjacent consistent subsegments is resolved by doing a linear passes through the alignments, and finding the optimal breakpoint at which to end the first alignment and start the second one.

1.5.2 Multiple alignment with Mauve

The general problem of aligning multiple genomes that have undergone recombination events such as translocation, inversion, and duplication-loss remains an open problem. One early method that has shown promise has been implemented in the Mauve genome alignment package [21]. Like the other methods described in this chapter, Mauve uses a seeding procedure to generate candidate anchors, chains these anchors, and finally computes a progressive multiple alignment between anchors. Unlike previous methods, Mauve does not build a single consistent set of anchors, but rather builds one consistent set of anchors for every collinear segment of the genome sequences. Each consistent set of anchors is referred to as a Locally Collinear Block (LCB). LCBs are bounded on either side by a breakpoint: a change in LCB order or orientation among a pair of sequences.

The original Mauve algorithm used a seed-and-extend technique to generate multi-MUMs which were used as candidate anchors. Because multi-MUMs must be exact, unique matches occurring in every genome the original anchoring method had limited sensitivity. Current releases of Mauve use a spaced seed pattern to match multiple sequences simultaneously. The inexact matching method substantially improves anchoring sensitivity. Given a set of potential anchors that match in all of the genomes being aligned, Mauve uses a greedy breakpoint elimination method to filter out matches due to paralogy and random similarity. The greedy breakpoint elimination method repeats three core steps. (1) Use breakpoint analysis [45] to identify breakpoints in the anchor order, yielding LCBs (2) Calculate the weight o f each LCB as the sum of its constituent anchor lengths (3) Identify the lowest weight LCB. If it has weight $<$ MinWeight then delete its anchors and return to step 1; otherwise end.

In the original Mauve paper, the genome alignment algorithm was applied to a group of nine closely related Enterobacteria, identifying numerous genome rearrangements and sites of differential gene content. Currently over 300 bacterial genomes have been finished and many more are nearly complete. As genome sequencing continues, we expect that automated methods for aligning multiple genomes with rearrangements will be tantamount to understanding the evolutionary forces shaping gene and genome function.

1.6 Whole genome alignment

The availability of the whole-genome human, mouse, and later rat assemblies presented for the first time the challenge of building multiple alignments of several large genomes. The problem of aligning genomes is more difficult than that of aligning sequences not only because of the size of the problem – a mammalian genome has about three billion basepairs – but also because of necessity of find the orthologous blocks, matching areas between the genomes, in which to apply alignment algorithms. Finding these blocks between two species computationally is a non-trivial task. Local alignment tools find a lot of high scoring matching segments, in particular the orthologous segments, but in addition they identify many paralogous relationships, or even false positive alignments resulting from simple sequence repeats and other artifacts [18]. The initial approaches for whole genome comparison developed for human and mouse genomes were based either on local alignment [57, 46, 6], or on a local/global technique, where stretches of one genome are mapped onto the others by a local aligner, and then the homology is confirmed and refined by a global one [20].

1.6.1 Local alignment on a whole genome scale

Perhaps the most straightforward approach to aligning two whole genomes is to do all-by-all local alignment. This is a challenging task due to the large amount of computation required, and is also difficult due to the problem of setting a threshold for individual local alignments: if the threshold is set too low many false positive local alignments could be found. If it is set too high true positive alignments are missed. Additionally classical local alignment methods do not consider whether a particular local alignment falls into a larger syntenic region. This leads to difficulties with local alignments that are either the result of repeats that were not masked, or with paralogous copies. Nevertheless local alignments were initially used for whole genome comparisons as they were better able to consider the rearrangements that are present between two large mammalian genomes. The initial local alignmets of the human and mouse genomes was done with BLASTZ [57] and PatternHunter [46]. For the human/mouse/rat three way alignments MULTIZ, a multiple sequence version of BLASTZ was used [6].

1.6.2 Local-global tandem approach

A computational strategy for finding and aligning orthologous regions that combines advantages of both local and global alignment techniques was first applied to the comparative analysis of the mouse and human genomes [20] and then expanded to the human, mouse and rat genomes [15]. In this technique the mouse genome is split up into contigs of 250 Kbp. The potential human homologs for each contig are found using the BLAT aligner [36]. The human sequence is then extended around the BLAT anchor, and aligned to the mouse contig using a global aligner (Figure 1.3). If the BLAT hits fall on both strands, then the aligner is called both with the original mouse contig and a reverse-complemented copy, making it possible to catch some inversions. This procedure has been expanded to three way alignment for comparing the human, mouse and rat genomes: First, the mouse and rat genomes are aligned using the BLAT program for approximate mapping followed by global alignment of selected regions. This step results in a set of mouse-rat *multi-contigs* (global alignments of rat contigs and mouse genomic sequence) as well as the remaining unaligned sequences. Second, the multi-contigs are aligned to human using the union of all available BLAT local alignments from mouse to human and from rat to human; mouse or rat sequences that could not be aligned to the other rodent are also aligned to human. The local/global tandem approach combines advantages of local and global alignment schemes in order to obtain both specificity (with respect to identifying only orthologous alignments) and sensitivity (in terms of coverage of genomic features of interest). At the same time the method is highly dependent on the parameters used at the local alignment stage, and it has difficulty aligning sequences that have undergone micro-rearrangements: small changes in the order of conserved elements due to inversions, duplications, or translocations of very short (sometimes only tens of base pairs) pieces of DNA.

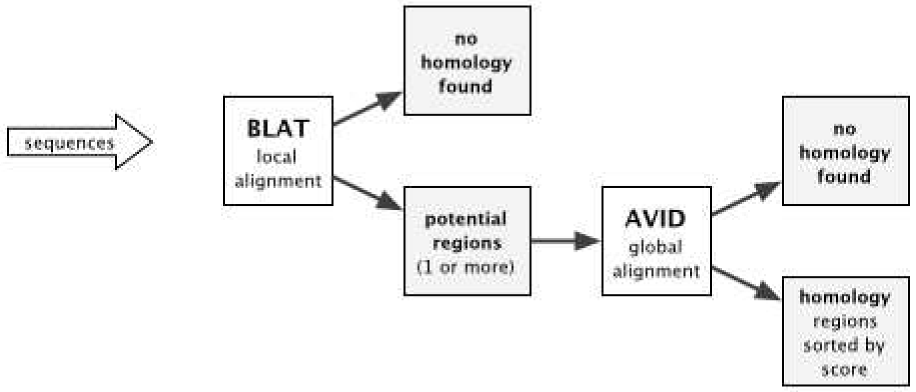


FIGURE 1.3: General computational scheme of tandem local/global genome alignment from (from [20]). The scheme used to aligns individual contigs, supercontigs, or long fragments of assemblies

1.7 Visualization

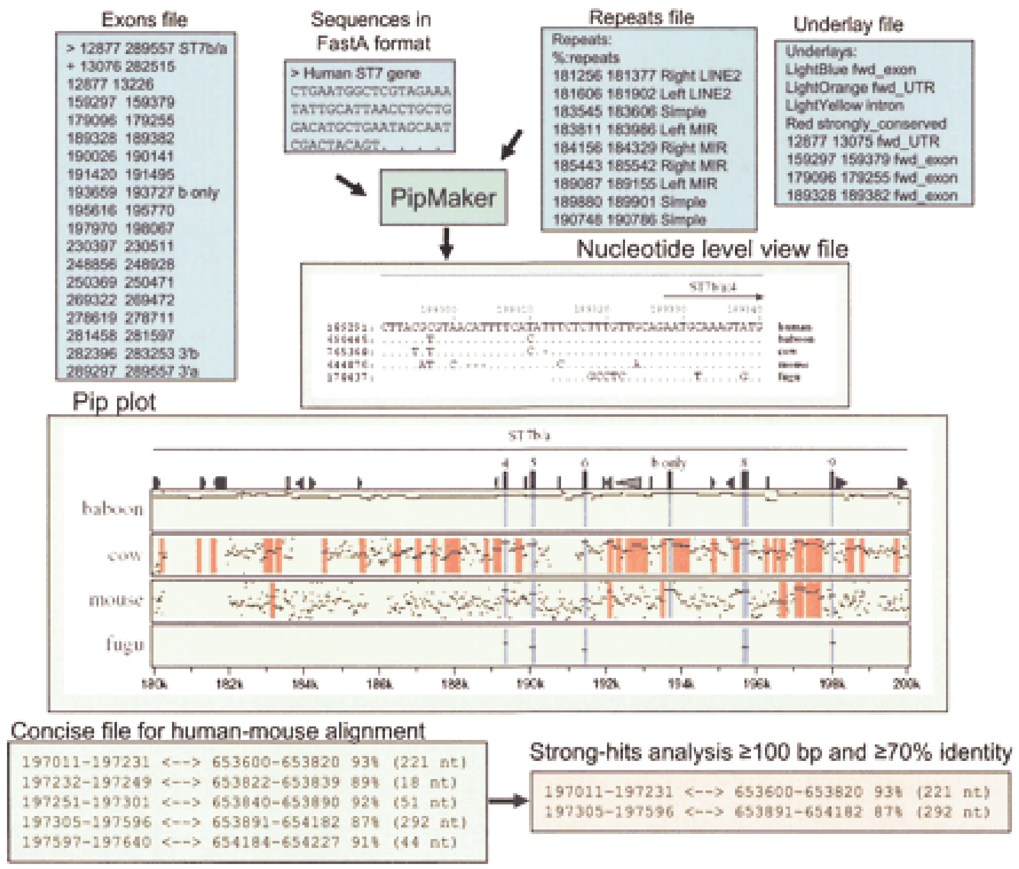
After obtaining alignments of two or more genomic sequences the next step is to analyze the level of overall homology, distribution of highly conserved elements and other comparative features. Visualization of results is a critical component of a comparative sequence analysis since manual examination of alignment on the scale of megabase long genomic regions is not efficient. Alignment-browsing systems should identify regions that exhibit properties suggestive of a particular biological function, for example well-conserved segments within an alignment, or matching the consensus sequence for a specific transcription factor binding site [49].

1.7.1 Visualization of pairwise alignments

There are several publicly available visualization tools for long pairwise DNA alignments. PIPMaker [57] represents the level of conservation in ungapped regions of BLASTZ local alignment as horizontal dashes called percent identity plots or pips. VISTA [28, 24, 48] displays comparative data in the form of a curve, where conservation is calculated in a sliding window of a gapped global alignment.

PipMaker and the companion server MultiPipMaker (`http://bio.cse.psu.edu`, [55]) visualize BLASTZ [56] local alignments. PipMaker visualizes the local alignments in two different formats: Percent identity plots (pips) and a dot plots. Pips present a compact, understandable display of local alignments on long genomic regions [32]. The program plots the position (in the base sequence) and percent identity of each gap-free segment of the alignments. The top horizontal axis is automatically marked with the positions of repeats and exons. The positions of CpG islands are also computed and displayed along the horizontal axis.

VISTA system is fundamentally based on global alignments, and its plot is generated by moving a user-specified window over the entire alignment and calculating the percent identity over the window at each base pair. The X-axis represents the base sequence; the Y-axis represents the percent identity. If the user supplies an annotation file, genes and


Exons file
Sequences in FastA format
Repeats file
Underlay file
PipMaker
Nucleotide level view file
Pip plot
ST7b/a
b only
baboon
cow
mouse
fugu
180k
182k
184k
186k
188k
190k
192k
194k
196k
198k
200k
Concise file for human-mouse alignment
Strong-hits analysis ≥100 bp and ≥70% identity

```
> 12877 289557 ST7b/a
+ 13076 282515
12877 13226
159297  159379
179096  179255
189328  189382
190026  190141
191420  191495
193659  193727 b only
195616  195770
197970  198067
230397  230511
248856  248928
250369  250471
269322  269472
278619  278711
281458  281597
282396  283253 3'b
289297  289557 3'a
```

```
> Human ST7 gene
CTGAATGGCTCGTAGAAA
TATTGCATTAACCTGCTG
GACATGCTGAATAGCAAT
CGACTACAGT. . . . .
```

```
Repeats:
%:repeats
181256 181377 Right LINE2
181606 181902 Left LINE2
183545 183606 Simple
183811 183986 Left MIR
184156 184329 Right MIR
185443 185542 Right MIR
189087 189155 Left MIR
189880 189901 Simple
190748 190786 Simple
```

```
Underlays:
LightBlue fwd_exon
LightOrange fwd_UTR
LightYellow intron
Red strongly_conserved
12877 13075 fwd_UTR
159297 159379 fwd_exon
179096 179255 fwd_exon
189328 189382 fwd_exon
```

```
197011-197231 <--> 653600-653820 93% (221 nt)
197232-197249 <--> 653822-653839 89% (18 nt)
197251-197301 <--> 653840-653890 92% (51 nt)
197305-197596 <--> 653891-654182 87% (292 nt)
197597-197640 <--> 654184-654227 91% (44 nt)
```

```
197011-197231 <--> 653600-653820 93% (221 nt)
197305-197596 <--> 653891-654182 87% (292 nt)
```

FIGURE 1.4: adapted from Frazer et al., 2003. [27] PipMaker: input and output files. The Pip plot shown is a subregion of the human ST7 interval compared with the orthologous baboon, cow, mouse, or fugu sequences. Each panel represents a pairwise comparison between human sequence and that of the indicated species

exons are marked above the plot. Conserved segments with percent identity X and length Y are defined to be regions in which every contiguous sub-segment of length Y was at least X% identical to its paired sequence. These segments are merged to define the conserved regions. Conserved regions calculated by using user-submitted cutoffs are highlighted under the curve, with different colors indicating a conserved non-coding regions, exons and UTRs. Figures 1.4 and 1.5 from the review of Frazer and coauthors ([27]) show the PipMaker and VISTA input and output.

Currently the selection of a particular visualization tool is mostly defined by the type of alignment used for the analysis. It is important to note, that as alignment algorithms become more sophisticated, it is becoming harder to distinguish between local and global alignment tools. For example, a chaining option for BLASTZ [56] allows for the extraction of global alignments from BLASTZ local alignments, and similarly Shuffle-LAGAN [13] is a hybrid *glocal* aligner that explicitly deals with rearrangements between sequences. Thus visualization methods also will have to become universal.

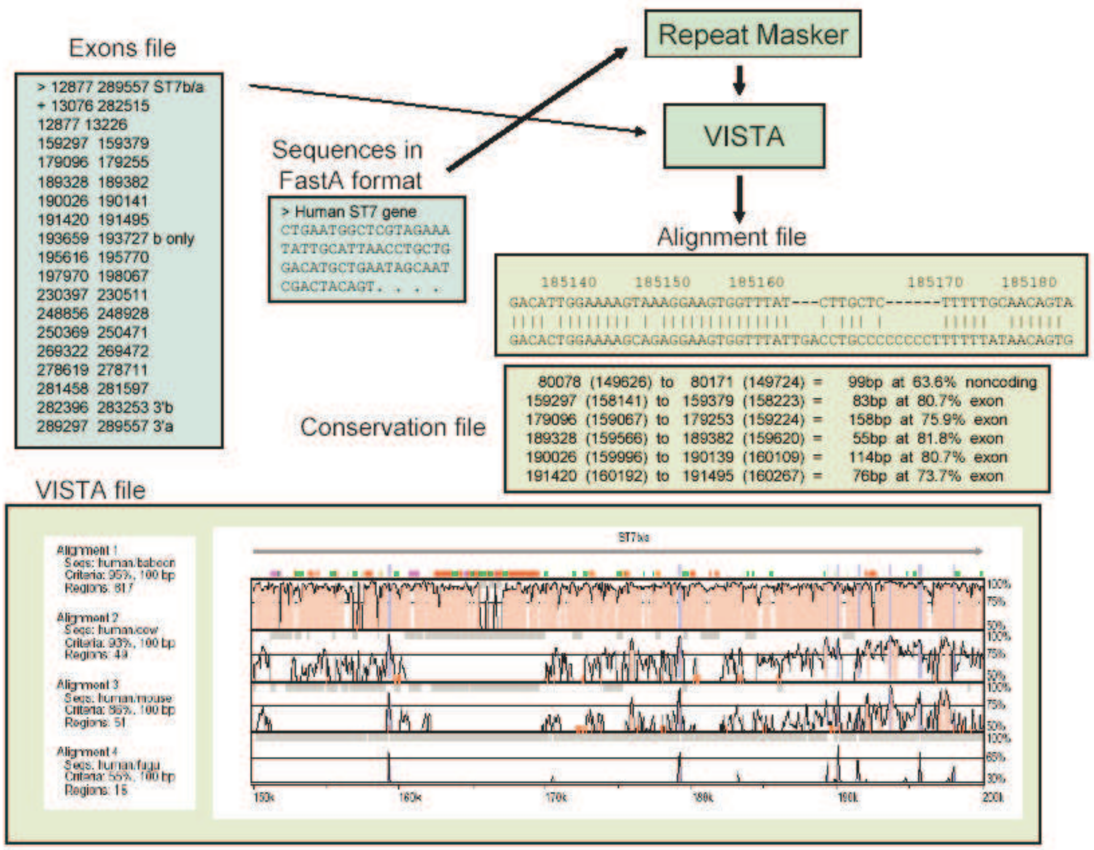


FIGURE 1.5: adapted from Frazer et al., 2003 [27]. VISTA: input and output files. The VISTA plot shown here is also a subregion of the human ST7 interval compared with the orthologous baboon, cow, mouse, or fugu sequences. Conserved sequences represented as peaks [noncoding (red) and coding (blue)] are shown relative to their positions in the human genome (horizontal axes), and their percent identities (50%-100%) are indicated on the vertical axes.

1.7.2 Visualization of multiple alignments

Both the VISTA and PipMaker approaches, described above, support visualization of a multiple alignment by projecting the alignment to a particular base sequence and in effect visualizing pairwise alignments between this base sequence and any number of homologous genomic intervals. This approach, however, only shows a part of the multiple alignment; it will be missing any similarity between fragments of two sequences other that the base that is not present in the base genome. For example, if there is a multiple alignment of human, mouse and rat with human used as the base, areas of conservation between mouse and rat that are not present in human will not be displayed. Full visualization of a multiple alignment is a difficult and largely unsolved problem, and currently is an area of active research.

The first tool to support visualization of multiple alignments was SynPlot [30]. The SynPlot graphical output includes a similarity profile of the long-range alignment together with a diagrammatic representation of both loci. Unlike PIPMaker and VISTA, SynPlot uses an alignment as a base coordinate, so the positions of all features in the individual sequences are mapped to the alignment coordinates. Feature files generated during annotation contain the positions of exons and repeat elements and can be directly imported into the graphical output. Therefore, the SynPlot output conveys comparative gene structure, repeat patterns (plus any other user-defined patterns), and relative sequence homology in a single linear plot. Its main drawback is that the single plot does not allow the user to distinguish the source of the similarity within the multiple alignment: a strongly conserved region in 3 of 5 species that is absent in the other two would look very similar to a weakly conserved region in all 5 of them. A recently developed program from the VISTA family, Phylo-VISTA (short for Phylogenetic VISTA, Shah et al., 2004)[58], uses the phylogenetic relationship as a guide to display and analyze the level of conservation across internal tree nodes. Using the entire multiple alignment, not a reference sequence, as a base in the x-axis allows for additional capabilities in visualization, such as presentation of comparative data together with available annotations for all sequences and computation of a measure of similarity for any node of the tree. The phylogenetic relationship among species is important for building and analyzing multiple alignments, thus visualizing sequence alignment data while taking phylogenetic trees into account is important to make the results easy to interpret.

1.7.3 Whole-genome visualization of alignments

The algorithmic challenge of whole genome alignment has been accompanied by user interface challenges, such as how to visualize information related to enormous datasets and how to enable users to interact with the data and the processing programs.

The principle of selecting a whole genome as a base sequence is utilized on the whole-genome scale in the UCSC genome browser (`http://genome.ucsc.edu` [39]) and the VISTA browser (`http://pipeline.lbl.gov` [20, 15]). These tools provide complementary information for a number of genomes including human, mouse, rat, drosophila. The UCSC browser represents annotations as a series of horizontal "tracks" over the genome sequence. Each track displays a particular type of annotation, such as Genscan gene predictions, mRNA alignments, interspersed repeats, and others. There are two types of tracks to display comparative data such as alignments and various statistical measures of alignments. The first is a curve, the other is a conventional UCSC genome browser block-based display. The curve track called "Conservation" shows a measure of evolutionary similarity in multiple species based on the phylogenetic Hidden Markov Model (phylo-HMM) [59, 60] and MULTIZ alignments [6] of human, chimpanzee, mouse, rat, and chicken whole-genome assemblies. Unlike the "Conservation" track, other comparative genomics tracks show particular fragments of alignments as boxes. Thus, "Chained Blastz" shows genomic alignment of different assemblies to the base sequence. Another track, "Alignment Net" [37] shows the best chain for every part of the base genome. There are some other comparative block tracks, such as tight subset of best alignments, differential view of the human/chimp alignment, and others.

The VISTA Browser is a Java applet for interactively visualizing results of alignment of entire genomes in the VISTA format on the scale of whole chromosomes along with annotations [15]. The user may select any genome as the reference or base, and display the level of conservation between this reference and the sequences of another species in a particular interval. The browser has a number of options, such as zoom, extraction of a region to be displayed, user-defined parameters for conservation level, and options for selecting sequence elements to study. A VISTA display is also implemented as a custom track linked to the

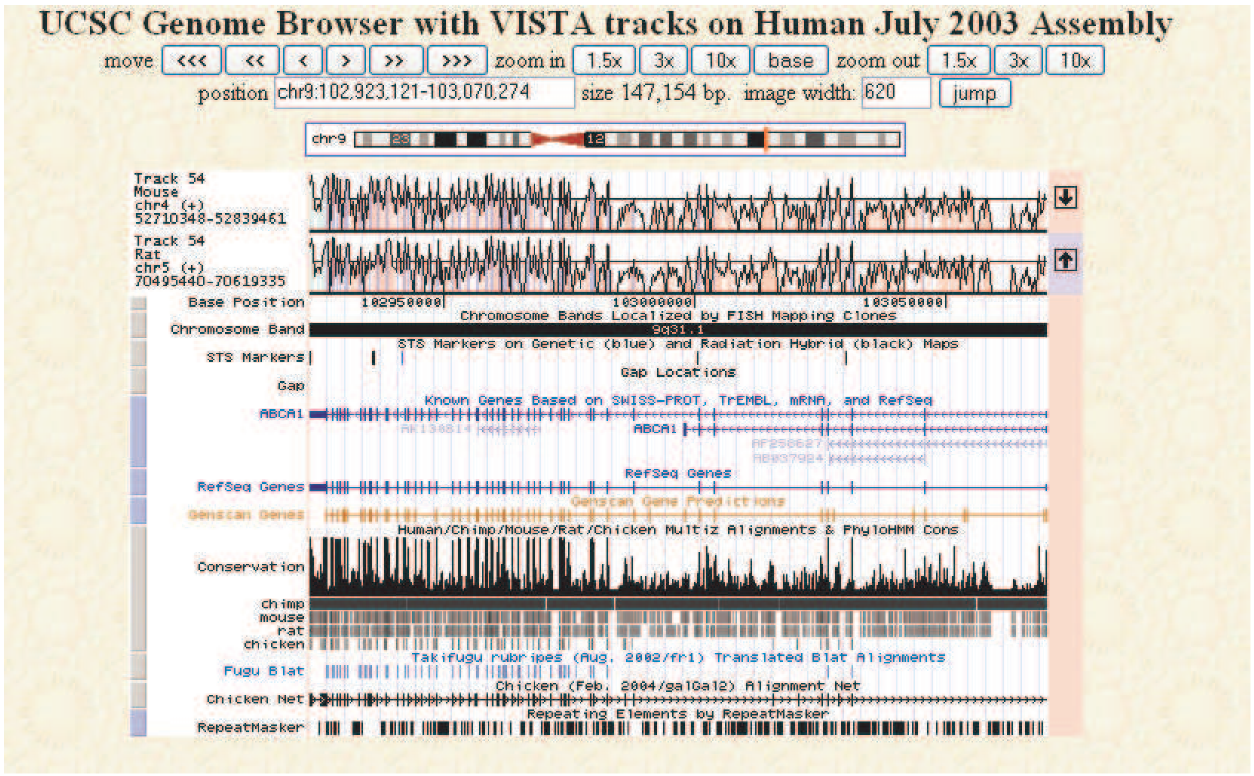

UCSC Genome Browser with VISTA tracks on Human July 2003 Assembly
move <<< << < > >> >>> zoom in 1.5x 3x 10x base zoom out 1.5x 3x 10x
position chr9:102,923,121-103,070,274 size 147,154 bp. image width: 620 jump
chr9
Track 54 Mouse chr4 (+) 52710348-52839461
Track 54 Rat chr5 (+) 70495440-70619335
Base Position 102950000 103000000 103050000
Chromosome Bands Localized by FISH Mapping Clones
Chromosome Band 9q31.1
STS Markers on Genetic (blue) and Radiation Hybrid (black) Maps
STS Markers
Gap Locations
Gap
Known Genes Based on SWISS-PROT, TrEMBL, mRNA, and RefSeq
ABCA1
AK130814
ABCA1
AF258627
AB037924
RefSeq Genes
RefSeq Genes
Genscan Gene Predictions
Genscan Genes
Human/Chimp/Mouse/Rat/Chicken Multiz Alignments & PhyloHMM Cons
Conservation
chimp
mouse
rat
chicken
Takifugu rubripes (Aug. 2002/fr1) Translated Blat Alignments
Fugu Blat
Chicken (Feb. 2004/galGal2) Alignment Net
Chicken Net
Repeating Elements by RepeatMasker
RepeatMasker

FIGURE 1.6: UCSC browser [36] with custom-built VISTA tracks showing conservation between the human chr. 9 interval aligned with orthologous mouse and rat sequences.

UCSC Browser. Figure 1.6 shows the UCSC Browser display of conservation with added VISTA conservation tracks and a control panel as a custom module accessible through the Berkeley Genome Browser.

1.8 Applications of alignments

Identifying transcriptional and other regulatory elements represents a significant challenge in annotating the genomes of higher vertebrates because these elements are usually very short (5 to 20 bp in length) and have low information content. Adding comparative sequence analysis methods allowed for improving and refining signal searches. These methods help filter computational predictions to reduce noise of false positive predictions at the price of some decrease in sensitivity. Here we briefly describe two approaches. One of them uses functional site prediction information together with comparative data to refine and improve predictions, while the other utilizes comparative and co-expression data.

Clues for identifying sequences involved in the complex regulatory networks of eukaryotic genes are provided by the presence of transcription factor binding sites (TFBS) motifs, the clustering of such binding site motifs, and the conservation of these sites between species. rVISTA [43, 44] takes advantage of all these established strategies to enhance the detection of functional transcriptional regulatory sequences controlling gene expression through its ability to identify evolutionarily conserved and clustered TFBSs. Although the identification of conserved TFBSs on a small genomic interval can be achieved by phylogenetic footprinting [32, 40], a strength of the rVISTA algorithm is its ability to efficiently analyze large genomic intervals and potentially whole genomes. The clustering modules and the user-defined customization of visualized sites make this a further useful tool for the investigation of TFBSs. To take advantage of combining sequence motif recognition and multiple sequence alignment of orthologous regions in an unbiased manner, rVISTA analysis proceeds in four major steps: (1) identification of transcription factor binding sites (TFBS) matches in the individual sequences (the program uses available position weight matrices in the TRANSFAC database and independently locates all TFBS matches in each sequence), (2) identification of globally aligned noncoding TFBSs, (3) calculation of local conservation extending upstream and downstream from each orthologous TFBS, and (4) visualization of individual or clustered noncoding TFBSs. Available sequence annotations are used to identify aligned TFBS matches in noncoding genomic intervals.

Another tool, Consite [41] uses the same principle of combining predictions of TFBS and sequence conservation information. This program used a good quality binding profiles collection of TFBS developed by the authors and provided an efficient graphical web application to visualize results of the analysis. It is also worth mentioning here that CONREAL algorithm, described earlier, is both an alignment algorithm and a tool for finding TFBS. By using potential TFBSs as anchor points CONREAL attempts to more accurately align these. This leads to a decrease in the false positive rate, achieved in the same way as in the rVISTA program.

A novel approach for finding transcription factor binding sites in a genome is to use both conservation information within a species and information about coregulation of genes in the same genome. While the latter allows for ab initio prediction of TFBSs for which no known motifs exist, the former leads to a reduction in the false positive rate. Two programs that use this idea are PhyloCon and CompareProspector.

PhyloCon (Phylogenetic Consensus) [69] is based on the Consensus algorithm previously established by the same group and takes into account both conservation among orthologous genes and co-regulation of genes within a species. This algorithm first aligns conserved regions of orthologous sequences into multiple sequence alignments, or profiles, and then compares profiles representing co-regulated genes. Motifs emerge as common regions in these profiles, and a greedy approach is used to search for common subprofiles in the sequences. PhyloCon thus integrates knowledge of co-regulation of genes in a single species and sequence conservation across multiple species to improve performance of motif finding.

Similar ideas were used in CompareProspector [42]. This program takes as input the upstream sequences of a group of genes that are known or predicted to be coregulated, as well as local sequence conservation calculated based on alignments with orthologous sequences. CompareProspector uses a Gibbs sampling approach to search for motifs in the input sequences, biasing the search toward conserved regions by integrating sequence conservation into the posterior probability in the sampling process. CompareProspector was tested on two data sets from humans using human-mouse comparisons and two data sets from *Caenorhabditis elegans* using *C. elegans* – *C. briggsae* comparisons, and demonstrated the power of comparative genomics-based biased sampling in eukaryotic regulatory element identification.

1.9 Conclusion

In this chapter we have tried to illustrate the standard methodologies that have been used for creation and visualization of alignments of genomic sequence. We would like to emphasize that Comparative Genomics is a vibrant, growing field, and that this chapter represents a snapshot in time. We are bound to have missed covering some programs and algorithms, for example because they appeared too late for us to include them (most of this chapter was written in the summer of 2004). Some parts, especially individual programs and particular visualization techniques may become dated very quickly, however we are hopeful that the readers will find our synopsis of the underlying methodologies helpful.

1.10 Acknowledgments

The section on Mauve was contributed by Aaron Darling. We would like to thank Katya Permiakova for reading an early draft of the chapter and making many useful comments. We would like to thank our co-authors on many of the papers on which this chapter is partially based, espcially Serafim Batzoglou, Greg Cooper, Chuong (Tom) Do, Burkhard Morgenstern, Alex Poliakov, Arend Sidow, and many others. Their ideas have contributed to our understanding of this area and helped define the general directions of this chapter. We are also greatful to the biologists in the Genomics Division of Lawrence Berkeley National Laboratory, especially Edward Rubin and Kelly Frazer (now at Perlegen Sciences) who introduced us to many challenging problems in comparative genomics.

References

[1] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[2] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res.*, 25(17):3389–3402, September 1997.

[3] G. J. Barton and M. J. Sternberg. A strategy for the rapid multiple alignment of protein sequences. Confidence levels from tertiary structure comparisons. *J Mol Biol*, 198(2):327–337, Nov 1987.

[4] 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 Res*, 10(7):950–958, Jul 2000.

[5] E. Berezikov, V. Guryev, R. H. A. Plasterk, and E. Cuppen. CONREAL: conserved regulatory elements anchored alignment algorithm for identification of transcription factor binding sites by phylogenetic footprinting. *Genome Res*, 14(1):170–178, Jan 2004.

[6] M. Blanchette and *et al.* Aligning multiple genomic sequences with the threaded blockset aligner. *Genome Res.*, 14(4):708–715, 2004.

[7] D. Boffelli, J. McAuliffe, D. Ovcharenko, K. D. Lewis, I. Ovcharenko, L. Pachter, and E. M. Rubin. Phylogenetic shadowing of primate sequences to find functional regions of the human genome. *Science*, 299(5611):1391–1394, Feb 2003.

[8] P. Bonizzoni and G. D. Vedova. The complexity of multiple sequence alignment with sp-score that is a metric. *Theor. Comput. Sci.*, 259(1-2):63–79, 2001.

[9] N. Bray, Dubchak I., and Pachter L. Avid: A global alignment program. *Genome Res*, 13(1):97–102, 2003.

[10] N. Bray and L. Pachter. Mavid: constrained ancestral alignment of multiple sequences. *Genome Res.*, 14(4):693–699, 2004.

[11] B. Brejova, D. Brown, and T. Vinar. Vector seeds: an extension to spaced seeds allows substantial improvements in sensitivity and specificity. In G. Benson and R. Page, editors, *Algorithms and Bioinformatics: 3rd International Workshop (WABI)*, volume 2812 of *Lecture Notes in Bioinformatics*, pages 39–54, Budapest, Hungary, September 2003. Springer.

[12] M. Brudno, M. Chapman, B. Gottgens, S. Batzoglou, and B. Morgenstern. Fast and sensitive multiple alignment of large genomic sequences. *BMC Bioinformatics*, 4(1):66, 2003.

[13] M. Brudno and *et al.* Glocal alignment: finding rearrangements during alignment. *Bioinformatics*, 19 Suppl 1:54–62, 2003. Evaluation Studies.

[14] M. Brudno and *et al.* Lagan and multi-lagan: efficient tools for large-scale multiple alignment of genomic dna. *Genome Res*, 13:721–731, 2003.

[15] M. Brudno and *et al.* Automated whole-genome multiple alignment of rat, mouse, and human. *Genome Res*, 14(4):685–692, 2004.

[16] J. Buhler. Provably sensitive indexing strategies for biosequence similarity search. *Journal of Computational Biology*, 10(3/4):399–417, 2003.

[17] M. A. Chapman, I. J. Donaldson, J. Gilbert, D. Grafham, J. Rogers, A. R. Green, and B. Gottgens. Analysis of multiple genomic sequence alignments: a web resource, online tools, and lessons learned from analysis of mammalian SCL loci. *Genome Res*, 14(2):313–318, Feb 2004.

[18] R. Chen, J.B. Bouck, G.M. Weinstock, and R.A. Gibbs. Comparing vertebrate whole-genome shotgun reads to the human genome. *Genome Res*, 11:1807–1816, 2001.

[19] G. M. Cooper, M. Brudno, E. D. Green, S. Batzoglou, and A. Sidow. Quantitative estimates of sequence divergence for comparative analyses of mammalian genomes. *Genome Res*, 13(5):813–820, May 2003.

[20] O. Couronne, A. Poliakov, N. Bray, T. Ishkhanov, D. Ryaboy, E. Rubin, L. Pachter, and I. Dubchak. Strategies and tools for whole-genome alignments. *Genome Res*, 13(1):73–80, Jan 2003.

[21] A.C.E. Darling and *et al.* Mauve: multiple alignment of conserved genomic sequence with rearrangements. *Genome Res*, 14(7):1394–1403, Jul 2004.

[22] A. L. Delcher, S. Kasif, R. D. Fleischmann, J. Peterson, O. White, and S. L. Salzberg. Alignment of whole genomes. *Nucleic Acids Research*, 27(11):2369–2376, 1999.

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

[24] I. Dubchak, M. Brudno, L.S. Pachter, G.G. Loots, C. Mayor, E.M. Rubin, and K.A. Frazer. Active conservation of noncoding sequences revealed by 3-way species comparisons. *Genome Research*, 10:1304–1306, 2000.

[25] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids.* Cambridge Univ. Press, 2000. Durbin.

[26] D. Eppstein, Z. Galil, R. Giancarlo, and G. F. Italiano. Sparse dynamic programming I: linear cost functions. *J. ACM*, 39(3):519–545, July 1992.

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

[28] K.A. Frazer, L. Pachter, A. Poliakov, E.M. Rubin, and I. Dubchak. Vista: computational tools for comparative genomics. *Nucleic Acids Res.*, 32:W273–9, July 2004. Web Server issue.

[29] O. Gotoh. Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. *J Mol Biol*, 264(4):823–838, Dec 1996.

[30] B. Gottgens and *et al.* Long-range comparison of human and mouse scl loci: localized regions of sensitivity to restriction endonucleases correspond precisely with peaks of conserved noncoding sequences. *Genome Res.*, 11:87–97, 2001.

[31] B. Gottgens and *et al.* Transcriptional regulation of the stem cell leukemia gene (SCL)–comparative analysis of five vertebrate SCL loci. *Genome Res*, 12(5):749–759, May 2002. Letter.

[32] R. C. Hardison, J. Oeltjen, and W. Miller. Long human-mouse sequence alignments reveal novel regulatory elements: a reason to sequence the mouse genome. *Genome Res*, 7(10):959–966, Oct 1997.

[33] M. Hohl, S. Kurtz, and E. Ohlebusch. Efficient multiple genome alignment. *Bioinformatics*, 18 Suppl 1:312–320, 2002. Evaluation Studies.

[34] I. Holmes and W. J. Bruno. Evolutionary HMMs: a Bayesian approach to multiple alignment. *Bioinformatics*, 17(9):803–820, Sep 2001.

[35] M. Kellis, N. Patterson, M. Endrizzi, B. Birren, and E. S. Lander. Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature*, 423(6937):241–254, May 2003.

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

[37] W. J. Kent, R. Baertsch, A. Hinrichs, W. Miller, and D. Haussler. Evolution's cauldron: duplication, deletion, and rearrangement in the mouse and human genomes. *Proc Natl Acad Sci U S A*, 100(20):11484–11489, Sep 2003.

[38] W. J. Kent and A. M. Zahler. Conservation, regulation, synteny, and introns in a large-scale C. briggsae-C. elegans genomic alignment. *Genome Res*, 10(8):1115–1125, Aug 2000.

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

[40] W. Krivan and W.W. Wasserman. A predictive model for regulatory sequences directing liver-specific transcription. *Genome Res*, 11(9):1559–1566, Sep 2001.

[41] B. Lenhard, A. Sandelin, L. Mendoza, P. Engstrom, N. Jareborg, and W.W. Wasserman. Identification of conserved regulatory elements by comparative genome analysis. *Journal of Biology*, 2(2):13, May 2003. Epub.

[42] Y. Liu, X.S. Liu, L. Wei, R.B. Altman, and S. Batzoglou. Eukaryotic regulatory element conservation analysis and identification using comparative genomics. *Genome Res.*, 14(3):451–458, 2004.

[43] G. Loots, I. Ovcharenko, L. Pachter, I. Dubchak, and E. Rubin. rvista for comparative sequence-based discovery of functional transcription factor binding sites. *Genome. Res.*, 12:832–839, 2002.

[44] G.G. Loots and I. Ovcharenko. rvista 2.0: evolutionary analysis of transcription factor binding sites. *Nucleic Acids Res.*, 32:W217–21, 2004. Web Server issue.

[45] Blanchette M. and *et al.* Breakpoint Phylogenies. *Genome Inform Ser Workshop Genome Inform*, 8:25–34, 1997. JOURNAL ARTICLE.

[46] B. Ma, J. Tromp, and M. Li. PatternHunter: faster and more sensitive homology search. *Bioinformatics*, 18(3):440–445, 2002.

[47] V. Matys and *et al.* TRANSFAC: transcriptional regulation, from patterns to profiles.

Nucleic Acids Res, 31(1):374–378, Jan 2003.

[48] C. Mayor and *et al.* Vista: visualizing global dna sequence alignments of arbitrary length. *Bioinformatics*, 16:1046–1047, 2000.

[49] W. Miller. Comparison of genomic dna sequences: solved and unsolved problems. *Bioinformatics*, 17:391–397, 2001.

[50] B. Morgenstern, K. Frech, A. Dress, and T. Werner. Dialign: finding local similarities by multiple sequence alignment. *Bioinformatics*, 14(3):290–294, 1998.

[51] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48(3):443–453, Mar 1970.

[52] C. Notredame, D. G. Higgins, and J. Heringa. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *J Mol Biol*, 302(1):205–217, Sep 2000.

[53] W. R. Pearson. Rapid and sensitive sequence comparison with FASTP and FASTA. *Methods Enzymol*, 183:63–98, 1990.

[54] M.A. Roytberg, A.Y. Ogurtsov, S.A. Shabalina, and A.S. Kondrashov. A hierarchical approach to aligning collinear regions of genomes. *Bioinformatics*, 18(12):1673–1680, 2002.

[55] S. Schwartz and *et al.* Pipmaker-a web server for aligning two genomic dna sequences. *Genome Res.*, 10(4):577–586, 2000.

[56] S. Schwartz and *et al.* Human-mouse alignments with blastz. *Genome Res.*, 13(1):103–107, April 2003.

[57] S. Schwartz and *et al.* Nisc comparative sequencing program: Multipipmaker and supporting tools: alignments and analysis of multiple genomic dna sequences. *Nucleic Acids Res.*, 31:3518–3524, 2003.

[58] N. Shah, O. Couronne, L. A. Pennacchio, M. Brudno, S. Batzoglou, E. W. Bethel, E. M. Rubin, B. Hamann, and I. Dubchak. Phylo-VISTA: interactive visualization of multiple DNA sequence alignments. *Bioinformatics*, 20(5):636–643, Mar 2004. Evaluation Studies.

[59] A. Siepel and D. Haussler. Combining phylogenetic and hidden markov models in biosequence analysis. *In Proceedings of the Seventh Annual International Conference on Computational Molecular Biology (RECOMB 2003)*, pages 277–286, 2003.

[60] A. Siepel and D. Haussler. *Statistical Methods in Molecular Evolution*, chapter Phylogenetic hidden Markov models. Springer, 2004. in press.

[61] A. L. Simon, E. A. Stone, and A. Sidow. Inference of functional regions in proteins by quantification of evolutionary constraints. *Proc Natl Acad Sci U S A*, 99(5):2912–2917, Mar 2002.

[62] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J Mol Biol*, 147(1):195–197, Mar 1981.

[63] K. Sumiyama, C. B. Kim, and F. H. Ruddle. An efficient cis-element discovery method using multiple sequence comparisons based on evolutionary relationships. *Genomics*, 71(2):260–262, Jan 2001.

[64] W. R. Taylor. A flexible method to align large numbers of biological sequences. *J Mol Evol*, 28(1-2):161–169, Dec 1988.

[65] J. W. Thomas and *et al.* Comparative analyses of multi-species sequences from targeted genomic regions. *Nature*, 424(6950):788–793, Aug 2003.

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

[67] J.S. Varre and *et al.* Transformation distances: a family of dissimilarity measures

based on movements of segments. *Bioinformatics*, 15(3):194–202, Mar 1999.

[68] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348, 1994.

[69] T. Wang and G.D. Stormo. Combining phylogenetic data with co-regulated genes to identify regulatory motifs. *Bioinformatics*, 19(18):2369–2380, December 2003.