Interactive visualization in large scale comparative genomics

Willems, C.M.E.

Award date:
2007

TECHNISCHE UNIVERSITEIT EINDHOVEN
Department of Mathematics and Computer Science

Interactive visualization in large scale comparative genomics

by
C.M.E. Willems

Supervisors:

dr.ir. H.M.M. v.d. Wetering (TU/e)
dr.ir. M.W.E.J. Fiers (WUR)

Eindhoven, November 4, 2007

Abstract

The genetic material of all organisms (DNA) is structurally and functionally equivalent. If two organisms share similar properties (phenotypes), these properties often derive from a common ancestor and are encoded similarly in the DNA. Comparative genomics is a molecular biological discipline searching for such similarities between DNA sequences as an indication of common ancestry and hence, a hint of similar function. The number of similarities between DNA sequences may be *enormous* and *chaotically distributed*. Besides, on *large scale*, individual similarities might be negligible small, but can be part of a much larger similarity.

This Master project tries to create visual order in a set of similarities between DNA sequences on a large (chromosome) scale. The result is an overview of dense locations of similarities, which current tools have problems to visualize. The overview of similarities can interactively be manipulated using the interactive DNAVis2 genome browser.

The development and research process is documented in this report containing requirements, solutions, design decisions, evaluation and necessary background information.

Acknowledgments

This Master project could not exist without support of Plant Research International (PRI) at Wageningen University and Research Centre (WUR). Especially I want to thank dr.ir. Mark Fiers for his inspirational attitude to bioinformatics. Furthermore, I would like to thank dr.ing. Stefan de Folter, drs. Paul Keizer, ir. Erwin Datema and ir. Carole Koning-Boucoiran for evaluating an early prototype of DNAVis2.
From the Mathematics and Computer Science department at Eindhoven University of Technology (TU/e), I would like to thank ir. Danny Holten for supporting me to apply his concept of hierarchical edge bundles. I would like to thank prof.dr.ir. Jack van Wijk and dr. Erik de Vink for participation in the assessment committee. Last, but certainly not least I would like to thank dr.ir. Huub van de Wetering for his excellent mentorship.

Niels Willems

Table of Contents

Chapter 1

Introduction

Why am I unique? Why do I look like my parents? Why is it likely to die because of the same disease as an ancestor? Why am I a human being and not a tomato? Why have tomatoes been red for ages and why are they not blue nowadays? Why are tomatoes red and some other fruits not? Why?

For more than two centuries people investigate heredity and variation of generations of organisms. Research started by attentive people like Darwin and Mendel who noticed patterns in types of organisms and characteristics between successive generations. Discovery of Deoxyribonucleic Acid (DNA) introduced a consistent model explaining the observations of Darwin and Mendel. Determining the order of the building blocks of DNA - called sequencing - has improved tremendously in the last decades, which results in enormous amounts of data. All this data needs to be analyzed to understand its part in relation to the structure and function of an organism.

Availability of computing power has provided a way of processing genetic material, but still analysis of DNA is manual labour. A biologist[1] writes an hypothesis, searches for relevant data, computes characteristics from the data, analyzes the characteristics and concludes whether the hypothesis holds. Due to a lot of noisy characteristics and the sheer size of the data, tools are needed to help the biologist to draw a conclusion. *Information visualization* is one of the possible techniques. By drawing characteristics instead of printing them as a textual listing, it is possible to use human pattern recognition capabilities more efficiently to extract *valuable information*.

Information visualization is not the core business of biologists or even bioinformaticians. Therefore, the Applied Bioinformatics cluster of PRI at WUR and the Visualization expertise group of the Mathematics and Computer Science department at TU/e have established a cooperation in 2003. Peeters performed a Master project [1][2] resulting in a tool called DNAVis, which visualizes annotated DNA sequences. Visualization experts do not have a rich level of biological knowledge to evaluate their own work. On the other hand, bioinformaticians lack visualization knowledge. Therefore, gaps have to be bridged [2] in a multidisciplinary cooperation like this.

[1]For now a biologist is assumed to be male.

[2]It is highly recommend to have a quick look at Peeters's Master's thesis before reading this report.

This Master project continues development of DNAVis by introducing a comparative genomic visualization. Because of participation of WUR in the International Tomato Sequencing Project [3], in the long run genetic data will become available of multiple species of tomato and potato. Investigation of these closely related organisms is interesting, because of the ability to determine the common parts of genetic material. What is essential genetic material for any kind of tomato? Why is a tomato not a potato, even if they differ genetically only a little? Why is a cherry tomato small and a beefsteak tomato large, even if they differ genetically only a little? Why is one tomato species resistent to a disease and are others not?

This report explains a solution to aid biologists answering questions in the field of comparative genomics. Style, structure and terminology of this report is based on Eindhoven's way of software engineering [4, 5]. The inception phase consist of chapter 2 and chapter 4. Chapter 2 explains the biological context of the system. State of art of visual comparative genomic tools are described in chapter 4. In chapter 3 the problem is specified and requirements are defined. Design of the system is described in 5. In chapter 6 requirements are verified and a conclusion is drawn in chapter 7. Finally, chapter 8 describes what happened after [1] and future work is defined.

Chapter 2

Domain analysis

This chapter guides the reader in the field of bioinformatics, in particular comparative genomics: the science of comparing DNA sequences.
The first section introduces DNA annotations, which is a general genomic concept. Annotations are a key element in genome browsers like DNAVis [1]. In this project the successor of DNAVis, called DNAVis2, is extended with comparative visualization.
The remainder of this chapter introduces a biological context [6, 7, 8, 9] related to problems solved in this project. This chapter does not provide a proper biology course, but state relevant domain knowledge which allows non-biologists to read this thesis. Basic terminology of genomics described in appendix A is assumed as prior knowledge.

2.1 Semantics of DNA

In the last ten years, the amount of available DNA sequence data has made an explosive growth. Sequence data are obtained from a process called sequencing, that distills a list of individual bases from the DNA of an organism and stores these in a digital file. Sequencing technologies are still improving, by retrieving more DNA sequence in a decreasing period of time.

Like one character of a plain text file, a single nucleotide has no meaning at all. When reading a sufficient large part of a text file, the intention of the text becomes clear. To be able to understand a DNA sequence, a large number of bases has to be considered. In contrast to a plain text file, the language in which DNA is 'written' is not complete yet. Moreover, sequence data may be incomplete, contain errors and parts may express nothing useful. To cope with these problems and allow further research DNA annotations are introduced.

2.1.1 DNA annotation

A DNA annotation [10] is a typed label, which marks a part of the sequence with the semantics of that subsequence. As an example, an annotation of type 'gene' may express that bases between 1000 and 2500 are responsible for the color of the eyes of the organism. Annotations may overlap and can have hierarchical relations. The example annotation may consist of two separate 'exon'-typed annotations from 1000 to 1500 and 2250 to 2500 stating that the head and tail of the gene encode the color of the eyes. Typically, hierarchical relations exist between 'gene' and 'exon' annotations, but there are many more hierarchical relations between other types of annotations.

Figure 2.1 displays[1] how sequence and annotation data are related to each. Assignment of annotations is done by performing experiments, applying statistics and heuristic methods and comparing sequences. Experiments are performed using real organisms, which may result in an assignment of a function to a subsequence. Heuristic methods predict using the structure of a sequence where an annotation may occur. Comparative tools assign annotations to sequences based on known annotations. If a sequence contains a subsequence which is sufficient similar to an annotated subsequence then the annotation may be applicable to the non-annotated subsequence. In section 2.3 will become clear why this is correct. Comparative tools also assign annotations to sequences indicating that a subsequence is similar to another subsequence of this or another sequence. Like in [1], these special annotations are called similarity annotations.

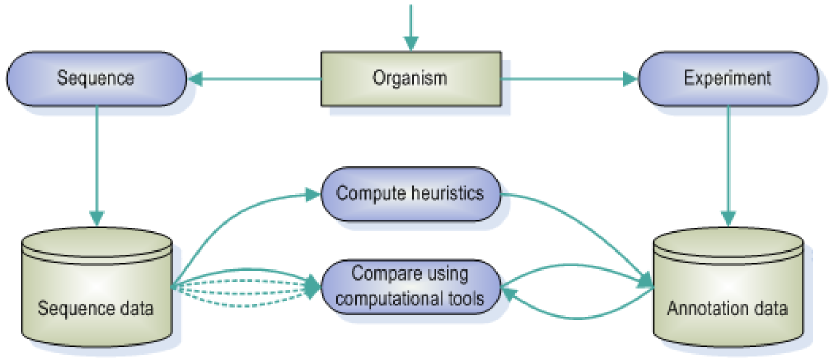


Figure 2.1. The data flow of sequence and annotation data.

In the last couple of years, the amount of annotation has made a substantial growth. Fortunately, this is not a problem, since annotation data is highly structured, in contrast to sequence data. By means of biological criteria (such as type and location), annotations can be reduced to a much smaller set, which is useful for a specific research purpose.

2.1.2 Visualization of annotated DNA

Visualization may assist a biologist to gain insights in sequence data by means of graphical representations of annotations (glyphs) rendered along a sequence. Such a tool is called a genome browser. This subsection briefly describes main concepts of the genome browser DNAVis2, which is the point of departure of this project from an implementation point of view.

[1] These diagrams occur now and then in this report and are combined to one overview diagram in appendix B, with explanation of the semantics of the symbols.

When creating DNAVis2 [1], existing genome browsers lacked of responsiveness, because of web-based implementations. Due to graphics hardware, a real-time interactive application is born. The DNAVis2 application consists of two visualizations called Linear view and Matrix view.

2.1.2.1 Linear view

One Linear view displays annotation data sets along one DNA sequence. Figure 2.2 displays three Linear views on top of each other. A Linear view consist of various horizontal bars, which are displayed on a perspective wall. Perspective walls aside the front wall enable a detail+context view on the sequence. By means of zooming, the level of detail is enlarged or reduced on the perspective wall. The three Linear views in figure 2.2 have a decreasing level of detail from top to bottom. Panning allows browsing in horizontal direction through the data set.

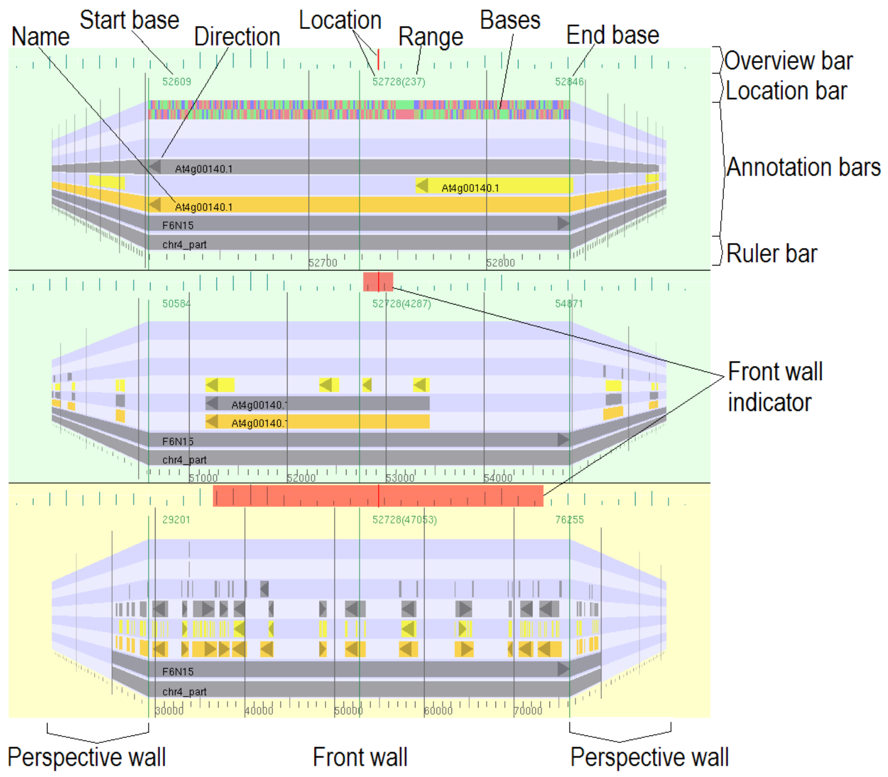


Figure 2.2. Three linked Linear views of a part of chromosome 4 of *Arabidopsis thaliana* at various levels of detail: from detail level on top to overview level at the bottom.

The topmost bar is the overview bar, which displays the density of annotations of the whole data set as a histogram. A red front wall indicator displays which part of the data is visible on the front wall.
Below the overview bar the location bar is positioned. The location bar displays indices of the start, center and end base of the front wall. Between brackets the range of the front wall is displayed.
Next, a number of annotation bars provide space to display glyphs. By default, the glyph is a rectangle which is colored according a color map. If the level of detail is high enough, a

triangle indicates the direction of the annotation and the name is displayed in the glyph. Recent developments (see subsection 8.1.2) allow developers to define new glyphs in a plug-in based architecture. A glyph is displayed at the location on the sequence as defined in the annotation. Equal types of annotations are located at one or more annotation bars and an annotation bar may be shared with other types. One annotation bar contains individual bases with their complements, when zoomed in at detail level.
The lowest bar is the ruler bar which provides an indication of the scale and position of the annotation bars.

2.1.2.2 Linking

One type of visualization may be opened in separate windows of DNAVis2. Such a window is called a view. Two views which respond to zooming or panning actions can be linked. When two or more views are linked, zooming and panning actions are copied by other views. For example, if two Linear views are linked and the user pans one Linear view 100 bases to the left, then the other Linear view simultaneously pans 100 bases to the left.
It is possible to synchronize two views on the same location or on the same level of detail. Like in figure 2.2, synchronization of location allows creation of a context+detail environment. Detailed browsing is possible in the topmost view and the others create a context, which prevent the user to get lost in the data set.

2.1.2.3 Matrix view

The Matrix view combines two Linear views with a Dotplot [11] which allows comparison of at most two sequences. In figure 2.3 an example of a Dotplot is drawn. Creation of a Dotplot is simple: draw two sequences along the adjacent sides of a table. For all cells, draw a dot if both row and column contain the same base of the sequences. An almost diagonal line indicates similar subsequences. A top-left to bottom-right line indicates a 'match' which occurs when both subsequences are similar when reading them in the same direction. A bottom-left to top-right line indicates an 'inversion' which occurs when both subsequences are similar when reading them in different directions. Subsection 2.3.2 explains the reason of this distinction.

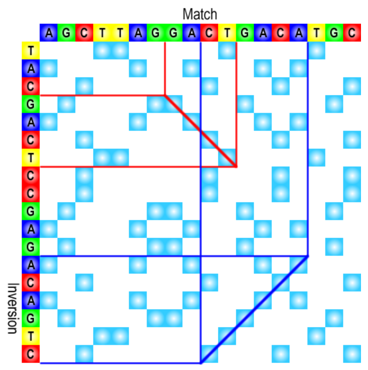

Match
A G C T T A G G A C T G A C A T G C
T A C G A C T C C G A G A C A G T C
Inversion

Figure 2.3. An example of a small Dotplot

A Dotplot of complete sequences is too large to detect similar subsequences, since the diagonal lines are too small to catch the eye. On the other side, the Dotplot may be too noisy. Both problems are solved in the Matrix view (figure 2.4).
To see small similar regions, the Matrix view allows various levels of detail, like the Linear view. The Dotplot is only visible between the subsequences on the front wall of the Linear view, which are located on the adjacent sides of the Dotplot. The two Linear views and Dotplot of one Matrix view are always linked synchronously . Hence, the location and level of detail of the Linear views and Dotplot match, which allows the user to determine the position of the Dotplot by means of the Linear views. The Linear views of the Matrix view can be linked separately to other views in DNAVis2, to create an extra context, for example.
To be able to abstract from noise, similarity annotations are drawn transparently as red rectangle on top of the Dotplot. These similarities are computed by comparative tools and have the same dimensions as the matching subsequences.

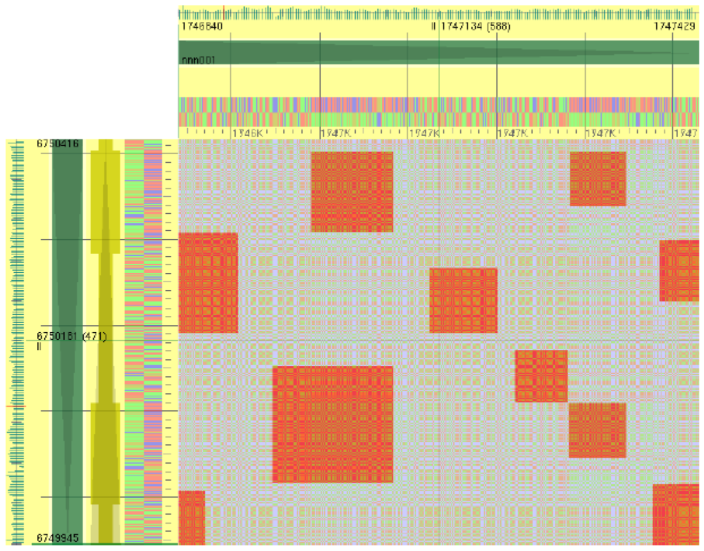


Figure 2.4. A Matrix view in DNAVis

2.1.2.4 Summary

The existing genome browser DNAVis2 is the point of departure of this project. It allows browsing of annotations and comparison of at most two DNA sequences. Multiple views may be linked, which allows various consistent views of one or more data sets.

2.2 The roots of comparative genomics: genetics

Genetics is the science of variation and heredity of organisms. Classic theory of variation of organism - evolution theory - is described in 2.2.1. In subsection 2.2.2 Mendel's heredity theory is described. Integration of Darwin's evolution theory and Mendel's heredity theory is mentioned in subsection 2.2.3.

2.2.1 Variation

Variation is a macro-scale phenomenon of genetics. It describes why organisms are different and how they change over many generations.
The main contribution to variation theory is attributed to Charles Darwin (1809-1882). His evolution theory suggests a common descent of all organisms. Furthermore, it describes mechanisms of changing characteristics over a long period in time. The idea of a common descent is born after the Beagle voyage (1837-1838) at the Galápagos Islands. At first, Darwin thought all Galápagos finches were counted as same species, but after a while he realized that each island had its own distinct species. He extended this theory for all organisms to conclude that all organisms have a common ancestor (figure 2.5). Darwin's idea of a common descent rejects earlier theories which assume that species are created separately.

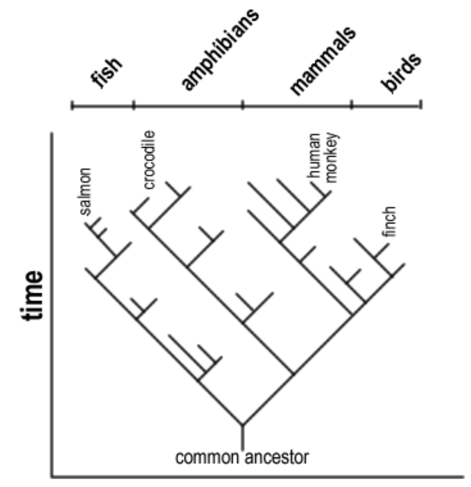


Figure 2.5. Evolution as a branching tree (after [7])

A main statement in Darwin's evolution theory is not only that species can change, but also why they fit in their environment. Natural selection is known as "survival of the fittest", but because of a lingual ambiguity it may be better to define it as "survival of the most fitting". Only those who fit in their environment will survive.

Natural selection requires four factors of an organism to operate:

i. *Reproduction*: Organisms create new generations by themselves.
ii. *Heredity*: Organisms look like their parents.
iii. *Variation of characteristics*: Individuals do not look like others.
iv. *Variation of fitness*: A variable probability that an individual will survive to reproduce itself.

All these conditions are necessary in natural selection. If there is no reproduction, then there will be no new generation, hence evolution ends after the first generation. Absence of heredity would cause random characteristics of new generations, so there is no predictability. If there is no variation in characteristics of individuals, then everybody is equal. Knowing conditions i and ii this results in equal new generations, which is not the case. Finally, if there is no variation in fitness then every individual would have the same chance to survive and reproduce itself. Again this is not the case in real-life. For example [7], an adult Atlantic cod lays over five million eggs from which only two can survive in the dangerous ocean (although many eggs might just be unfortunate).

The implausible notion of heredity in Darwin's evolution theory is the blending theory. A characteristic of an offspring has the average of the same characteristic of its parents. For example, a tall and a small parent create a medium length offspring. However, in general this is not the case, because over a long period of time this would converge to equal-length persons. In the next subsection another theory of heredity is described.

2.2.2 Heredity

Heredity is a micro-scale phenomenon of genetics. It argues how characteristics of an organism are passed on to next generations. Heredity theory answers why I look like my parents and sister and why I do not look like my neighbors[2].

A monk called George Mendel (1823-1884) lived in Brno, Czech Republic. As high school teacher of natural science, he is considered the main pioneer in the field of heredity theory. Using pea plants, he discovered a discrete technique which answers questions about inheritance of parental characteristics of individual organisms. Until this discovery it was only possible to answer such a question with 'probably' or 'unlikely'. But Mendel was capable to quantify chances of inheritance of a parental characteristic.

Mendel's theory of heredity is based on hereditary units, which nowadays are known as genes. One or more genes are responsible for a specific characteristic of an organism. Genes are located at chromosomes of organisms. Higher organisms (such as human beings) have multiple copies of a chromosome. A chromosome of a pair is inherited from one parent each (figure 2.6). Both chromosomes of a pair are structurally equal, they both have a gene at the same location (locus). However, the genes may differ. Two genes at a locus of a pair of chromosomes are called the genotype. A genotype is one of all combinations of alleles which can occur at a locus.

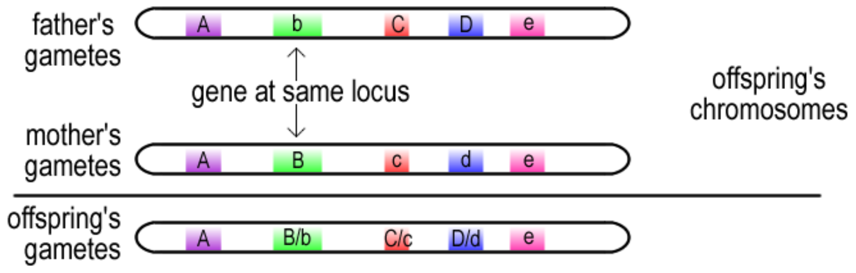


Figure 2.6. A pair of chromosomes of a higher organism and its gametes

Consider figure 2.6, which displays an example of a pair of chromosomes. Each chromosome consists of five genes which are drawn as shaded rectangles. In this example only two types of alleles per gene are taken in to account. Alleles can be dominant (uppercase character) or

[2]Unless I have family ties with them.

recessive (lowercase character). The actual behavior of the organism (phenotype) is determined by the combination of the two alleles. The behavior of the recessive allele is only expressed if both alleles are recessive, otherwise the behavior of the dominant allele is expressed.

An organism receives one chromosome of both of its parents, which have two chromosomes themselves. How is this single chromosome related to the pair of chromosomes of a parent? The single chromosome is called the gametes. The result of a gene in the gametes is just flipping a fair coin with an allele of both chromosomes on each side. In figure 2.6 the result is computed for this organism. If the organism has allele 'A' and 'A' at a locus, then the gametes has also allele 'A' at that locus. If the organism contains allele 'B' and 'b', then the gametes can be either 'B' or 'b', displayed as 'B/b'.

Because genotypes have no order (aA=Aa), some genotypes occur more frequent than others. In figure 2.7 the Mendelian ratio is displayed if both parents have a dominant and recessive gene. Both parents have a 50% chance of transferring a dominant gene in its gametes. Combining both parent's gametes results in a $\frac{1}{4}$AA: $\frac{1}{2}$Aa: $\frac{1}{4}$aa distribution, because genotype 'Aa' is equal to genotype 'aA'. If phenotypes are taken into account, the distribution is even more dramatic: $\frac{3}{4}$A:$\frac{1}{4}$a, since 'A' is dominant over 'a'. Note that in terms of phenotypes, it is possible that if both parents propagate a dominant allele, their offspring may propagate a recessive allele.

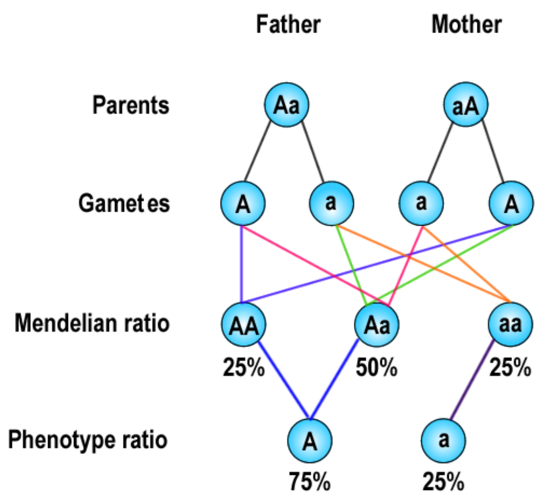


Figure 2.7. Distribution of alleles

The theory of heredity is more expressive than this simple example. For example, it is possible to take more distinct alleles and non-uniform distributions into account.

2.2.3 Synthetic theory of evolution

Until 1918 Darwin's and Mendel's theories were accepted separately. The lack of well-defined heredity in Darwin's evolution theory motivated researchers to try to join both theories. Ronald Aylmer Fisher wrote the most important article [12] which demonstrates that Darwin's theory is derivable from Mendel's principles. After publication of this article, the opposite derivation is shown independently by R.A. Fisher, J.B.S. Haldane and S. Wright. These results are known as the synthetic theory of evolution.

2.3 Modern comparative genomics

In 1953 Watson and Crick [13] proposed a double helix model containing genetic information in a uniform way for all organisms. This milestone completed a research period which resulted in a well-defined model of DNA. Even more than fifty years after this discovery, DNA it is still a hot research topic. And at least for the next couple of decades it still will be.
Subsection 2.3.1 indicates why DNA sequences are compared. The origin of differences in DNA sequences are described in subsection 2.3.2. Finally, in subsection 2.3.3 computational comparison tools are explained.

2.3.1 Comparative genomics in a nutshell

In comparative genomics, sequences of DNA are compared with each other. By comparing multiple sequences, similar subsequences can be found. Such a subsequence is called an Homologous Region (HR). Searching for HRs relies on Darwin's idea of a common descent.

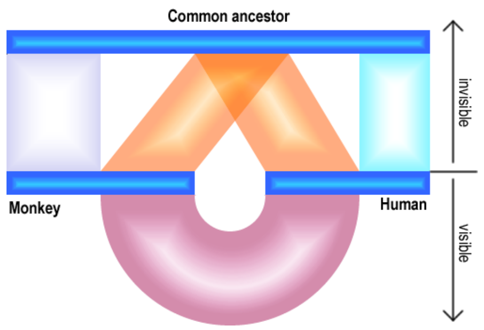


Figure 2.8. Search for evolutionary evidence

In figure 2.8 a sketch is given which explains why DNA sequences of different organisms are compared. The parallelograms and circular shape connect almost equal subsequences. If both a monkey and a human inherit subsequences through common descent, then it is possible that both inherit some of the same subsequences. If both organisms inherit the same subsequences, then it is likely that these subsequences provide similar functionality, for example by creating a similar protein. The common ancestor is unknown, hence the HRs between the organisms and the common ancestor are hypothetical: By comparing organisms, only HRs are visible as drawn between the monkey and the human being.

DNA sequences can also be compared because of insights in ancient (genome) duplications. If chromosomes of the same organism are compared, then it is possible to detect how subsequences are duplicated (figure 2.9). While replicating, sequences are subject to its environment (e.g. temperature, radiation), which result in erroneous copies. These copies are called mutations,

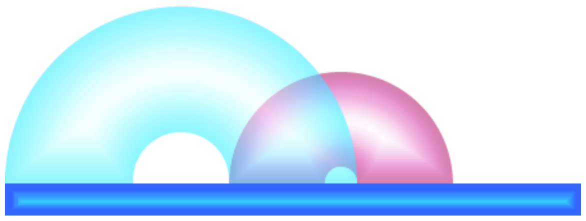


Figure 2.9. Search for duplications

which are discussed in subsection 2.3.2. If DNA would rely on only one subsequence per function, then the function would not exist for long, since mutations occur frequently. Note that not all genes are duplicated, these are subject to evolution.

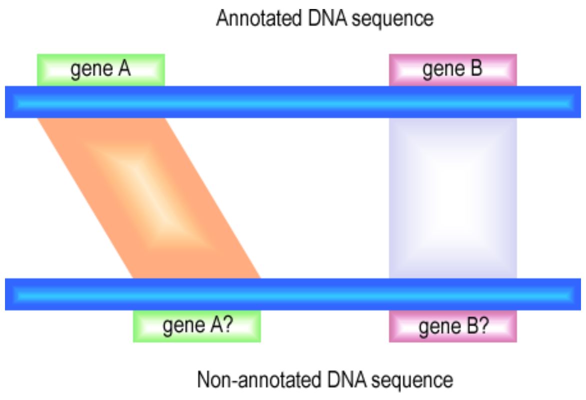


Figure 2.10. Annotation prediction

The last mentioned function of comparing of DNA sequences is prediction of annotations. With an annotated sequence, information can be transferred to a novel, related, sequence. Based on the assumption that a conserved subsequence expresses a conserved function it is possible to transfer an annotation from one sequence to another based on an HR. An HR matching an annotation on the well-annotated sequence is likely to be an annotation on the non-annotated sequence. This is correct because of the function-preservation assumption based on Darwin's common descent theory.

2.3.2 Mutations

Processes in a cell are not perfect, partly due to environmental influences, and hence DNA accumulates mutations. Investigation of these mutations is important, as they can give insight in, for example, diseases.

Mutations in DNA replication, for example, occur in various types. The main types (figure 2.11) of mutations are:

- *Substitution*, one or more bases are replaced by others.
- *Insertion*, one or more base pairs are added at a location of the DNA sequence.

- *Deletion*, one or more base pairs are removed from a location of the DNA sequence.
- *Rearrangement*, bases are located in another way then before the rearrangement.

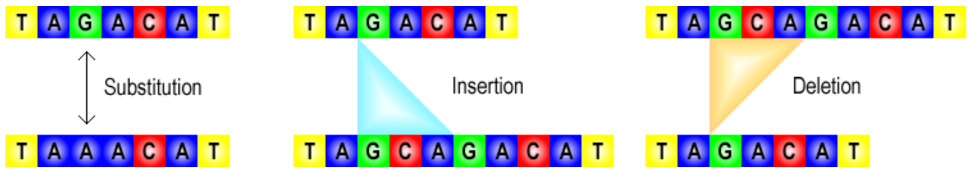


Figure 2.11. Mutations

2.3.2.1 Substitution, insertion and deletion

Substitution, insertion and deletion mutations have similar results. If only one base is substituted by another base, it could be harmless. In table 2.1 the genetic code is displayed: Each codon (a triplet of bases) of a coding sequence corresponds with the production of an amino acid. A list of codons, starting with a start codon (AUG) and ending with a stop codon (UAA, UAG or UGA) produces a protein, which is a chain of amino acids. Each protein expresses a function of the cell.

Different codons can produce the same amino acid. For example, if the codon starts with CU then the third base does not matter, always Leucine will be produced. Hence, a mutation on the third base does not always influence the production of the amino acid. The expressed function is conserved.
On the other side a mutation can be radical and even fatal. If the original codon was UGU and the third base is replaced by an A, then the amino acid is a stop codon (UGA). This results in the production of a shorter protein. It is likely that this protein does not express the same function as the intended protein.

		2nd base				
		U	C	A	G	3rd base
1st base	U	UUU Phenylalanine	UCU Serine	UAU Tyrosine	UGU Cysteine	U
		UUC Phenylalanine	UCC Serine	UAC Tyrosine	UGC Cysteine	C
		UUA Leucine	UUA Leucine	UAA Ochre, *Stop*	UGA Opal, *Stop*	A
		UUG Leucine	UCG Serine	UAG Amber, *Stop*	UGG Tryptophan	G
	C	CUU Leucine	CCU Proline	CAU Histidine	CGU Arginine	U
		CUC Leucine	CCC Proline	CAC Histidine	CGC Arginine	C
		CUA Leucine	CCA Proline	CAA Glutamine	CGA Arginine	A
		CUG Leucine	CCG Proline	CAG Glutamine	CGG Arginine	G
	A	AUU Isoleucine	ACU Threonine	AAU Asparagine	AGU Serine	U
		AUC Isoleucine	ACC Threonine	AAC Asparagine	AGC Serine	C
		AUA Isoleucine	ACA Threonine	AAA Lysine	AGA Arginine	A
		AUG Methionine, *Start*	ACG Threonine	AAG Lysine	AGG Arginine	G
	G	GUU Valine	GCU Alanine	GAU Aspartic acid	GGU Glycine	U
		GUC Valine	GCC Alanine	GAC Aspartic acid	GGC Glycine	C
		GUA Valine	GCA Alanine	GAA Glutamic acid	GGA Glycine	A
		GUG Valine	GCG Alanine	GAG Glutamic acid	GGG Glycine	G

Table 2.1. The genetic code. Each codon encodes an amino acid.

If one or two bases (in modulo 3) are inserted or deleted, then the production of the protein is mixed up, due to a rearrangement of codons. These mutations are called frameshift mutations.

All codons after insertion or deletion are shifted by one or two bases. This results in a mess, since all amino acids are changed. However, if a multiple of three is inserted or deleted a whole amino acid molecule is removed or added. The remainder of the list of codons is the same.

2.3.2.2 Rearrangements

Between chromosomes, rearrangement take place. This mechanism is known as recombination. When a subsequence exchanges, the original sequences are broken in three parts (call them head, center and tail per sequence) to allow the center parts to change location. After an exchange, parts of one sequence are joined together. The six parts may be joined in various combinations, called rearrangements.

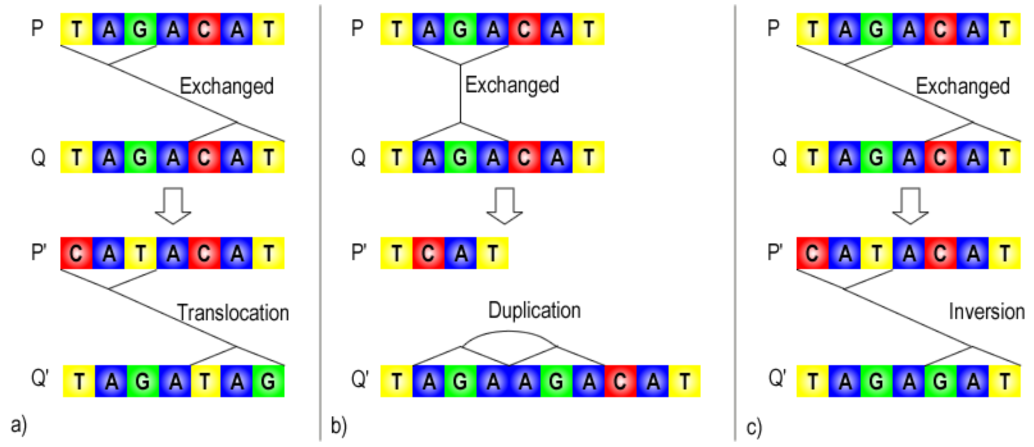


Figure 2.12. Translocation, duplication and inversion

Figure 2.12 contains three rearrangements: translocation, duplication and inversion. Each rearrangement is applied to sequence P and Q. Above the arrow the initial state of the sequence is displayed and below the final state is displayed after the rearrangement.
Translocation takes place if the center parts are exchanged as expected: the center parts of the sequences are switched and are joined in both foreign sequences. In figure 2.12a 'TAG' of sequence P is exchanged with 'CAT' of sequence Q. After recombination, the subsequence 'TAG' is located at Q' and 'CAT' occurs in P'.
Duplication takes place if two equal center parts are joined in the same sequence. In figure 2.12b, 'AGA' of P would be exchanged with 'AGA' of Q, but only one subsequence is moved to the other sequence. Hence, one sequence contains 'AGA' twice.
An inversion occurs when a center part is joined in the opposite direction. In figure 2.12c 'TAG' of sequence P is exchanged with 'CAT' of sequence Q like in 2.12a. This time 'TAG' is joined in the reverse direction in Q.

The effect of a rearrangement depends on the length and location of the exchanged subsequence. If a subsequence is long enough to contain full genes, then the functions of genes may be preserved, which has no dramatic effect. However, if only small subsequences or subsequences of unequal length are exchanged, then results might be as fatal as frameshift mutations from the previous subsection. If a rearrangement takes place within a coding sequence then the function of this coding sequence is likely to be lost.

2.3.3 Sequence comparison tools

DNA research heavily depends on information processing. There are number of reasons for that:

- Data structures of DNA are simple and allow computations.
- Computations using DNA structures are complex because of fuzzy, inexact results.
- Availability of high-performance computational resources.
- Availability of various mathematical-founded theories: e.g. logic, linguistics, pattern-matching and cryptology.
- Due to complexity, manual research would take too much time.

In comparative genomics, HRs and mutations between two or more sequences can be computed using a wide range of tools. Given two or more sequences, a tool generates a list of positions of almost common subsequences. Each tool provides results with different characteristics in computation time and quality. In the next subsections methods are discussed which compute similarities between two sequences. To compare multiple sequences, it is possible to compare each pair of sequences using pairwise methods or use tools which allow comparison of multiple sequences. Comparison tools for multiple sequences are discussed in the final subsection of this chapter.

2.3.3.1 Dynamic programming approach

A Dynamic Programming (DP) method computes an alignment between two sequences. An alignment is a set of positions of void bases (gaps) inserted in both sequences. A gap occurs zero or more times between every two non-void bases. The quality of an alignment is determined by a score. This score is the sum of the pairwise comparison of bases of the aligned sequences. A match of two bases results in a bonus and a mismatch results in penalty.

Alignments are differentiated in global and local alignments. A global alignment algorithm (e.g. Needleman-Wunsch [14]) computes a solution of an alignment with the optimal score. A local alignment algorithm (e.g. Smith-Waterman [15]) optimizes the score with respect to maximal cohesion of highly similar regions. This means that if two sequences have a similar region, then the alignment respects this similar region, instead of breaking it down to force a better score.

Both DP algorithms compute an optimal alignment, however computation of whole genomes is expensive, it takes quadratic time proportional to the length of the compared sequences. It is suggested [16] not to align sequences larger than 10,000 base pairs with DP algorithms, whereas full genomes are multiples of millions of base pairs.
Another disadvantage of the DP approach is absence of ability to find rearrangements, like inversions.

2.3.3.2 Anchoring approach

Other methods than DP are needed to be able to compare full genomes. MUMmer [17] is an example of an anchoring alignment approach, which is based on the concept of a Maximal Unique Match (MUM). MUMmer can deal with some rearrangements like inversions. Another advantage of MUMmer is the ability to find matches with a certain degree of fuzzyness.

Whole genome alignment is achieved by MUMmer using the following steps:

i. Compute potential MUMs between the compared sequences.
ii. Reduce the set of MUMs such that subsequences related to MUMs are not overlapping.
iii. Align non-MUM regions using the Smith-Waterman algorithm.

A subsequence S is called a MUM if it meets the following properties:

- *Maximal*, S is not a subsequence of another MUM.
- *Unique*, S occurs only once in both compared sequences.

Let N be the length of the longest sequence of the pair of sequences which are compared. MUMmer computes using a suffix tree MUMs in $\mathcal{O}(N)$ time and $\mathcal{O}(N)$ space. Reduction of overlapping MUMs is done in $\mathcal{O}(N \log N)$ time by computing the Longest Increasing Subsequence. Finally, the alignment is completed by filling the gaps between the MUMs using classical alignment techniques in $\mathcal{O}(k^2)$ time, where k is the length of the gap. Theoretically, MUMmer runs in $\mathcal{O}(N^2)$ if no MUMs are found, but this is not common. In practice, MUMmer runs in almost linear time for evolutionary close related sequences.

2.3.3.3 Seeding approach

A seeding approach like Basic Local Alignment Search Tool (BLAST) [18] is a fifty times faster approximation of the Smith-Waterman algorithm, but that is not the most wondering of this approach. The input of a BLAST search is one sequence which is pairwise compared to a database of well-known subsequences. By Darwin's common descent assumption, this method provides an indication of known knowledge for a fresh non-annotated sequence.

2.3.3.4 Multiple sequence approach

Until now, pairwise comparison approaches are described. To be able to compare more than two sequences it is possible to compare each pair, like in a cartesian product. However, this has a lack of precision. In figure 2.13 two pairwise comparisons are displayed as being a comparison of three sequences. HR α is similar to HR γ, but they do not match, because α is 4 bases wide and γ is 6 bases wide. It is obvious that the topmost sequence is similar to the bottom sequence. Could this be more sophisticated?

Yes, it is possible. A method is proposed [16] which is highly similar to MUMmer, but with attention to not matching HRs. If a MUM is redefined over multiple sequences, these mismatches will not occur:

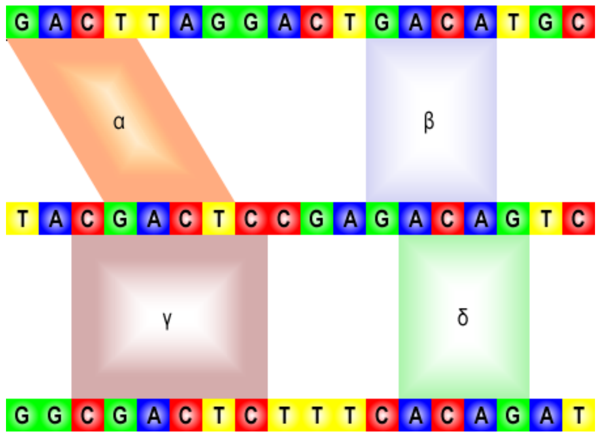

G A C T T A G G A C T G A C A T G C
α
β
T A C G A C T C C G A G A C A G T C
γ
δ
G G C G A C T C T T T C A C A G A T

Figure 2.13. Pairwise multiple comparison

- *Maximal*, S is not a subsequence of another MUM.
- *Unique*, S occurs only once in **all** compared sequences.

In figure 2.14 the new definition is applied, which results in an alignment which is less noisy. With the new definition α fits γ and vice versa. By manual inspection it is easier to understand what is going on. Notice that the similarity of the bottom two sequences is higher than displayed, since the sequences are equal to those in figure 2.13.

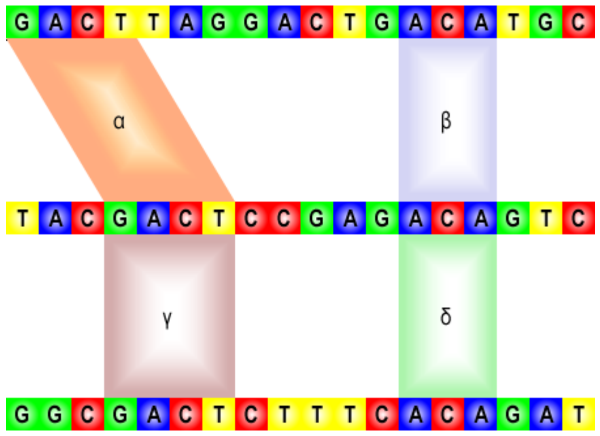

G A C T T A G G A C T G A C A T G C
α
β
T A C G A C T C C G A G A C A G T C
γ
δ
G G C G A C T C T T T C A C A G A T

Figure 2.14. Multiple comparison

2.3.4 Summary

Organisms are genetically not fully disjunct. Similar parts of DNA sequences imply shared functionality in cells, by Darwin's common descent idea. On the other hand, duplication is a way to offer resistance against evolution. Comparative computational tools aid the biologist to detect these shared functionality.

Chapter 3

Problem description

In this chapter the research problem is extracted from the biological context as described in the previous chapter.
In the first section, biological structures are defined to specify problems and explain solutions. An information visualization problem [19] consists of two problems: a domain specific problem and a graphical representation problem. Discovery of similarities of tomato DNA sequences is our domain specific problem, which can be solved by interpreting a multiple sequence alignment as described in section 3.2 with the definitions of section 3.1. Our graphical representation problem is described in section 3.3. Requirements are defined in section 3.4.

3.1 Definitions

We define alphabets $\Sigma_{seq} = \{A,C,G,T\}$ and $\Sigma_{al} = \Sigma_{seq} \cup \{\text{-}\}$, where a letter represents a base and '-' is a gap symbol. A sequence is an element of Σ^*_{seq}, denoted as S, S_0, S_1, etcetera. A gapped sequence is an element of Σ^*_{al}, denoted as S', S'_0, S'_1, etcetera. A sequence S of N symbols is represented by $s_0 s_1 \ldots s_{N-1}$. Furthermore, $S[i] = s_i$ and subsequence $S[i,j] = s_i s_{i+1} \ldots s_j$. A subsequence is denoted as σ, τ, etcetera. The empty sequence is represented by ϵ. For symbol $a \in \Sigma$ and (sub)sequence $\sigma \in \Sigma^+$ where $\sigma = S[l,r]$ the following functions are defined:

$$C(\sigma) = \frac{l+r}{2} \tag{3.1}$$

$$W(\sigma) = |l-r|+1 \tag{3.2}$$

$$\pi_H(\epsilon) = \epsilon \tag{3.3}$$

$$\pi_H(a +\!\!+ \sigma) = \begin{cases} a +\!\!+ \pi_H(\sigma), & \text{if } a \in \Sigma \setminus H; \\ \pi_H(\sigma), & \text{if } a \in H. \end{cases} \tag{3.4}$$

In (3.1) the geometric center C, in (3.2) the width W, and both in (3.3) and (3.4) the projection π_H of a subsequence are defined, where H is the set of removed symbols in Σ.
For example, if S = ACTGGTA, then S[2] = T, S[1,3] = CTG, $C(\text{S}[1,3]) = 2$, $W(\text{S}[1,3]) = 3$ and $\pi_{\{\text{A}\}}(\text{S})$ = CTGGT.

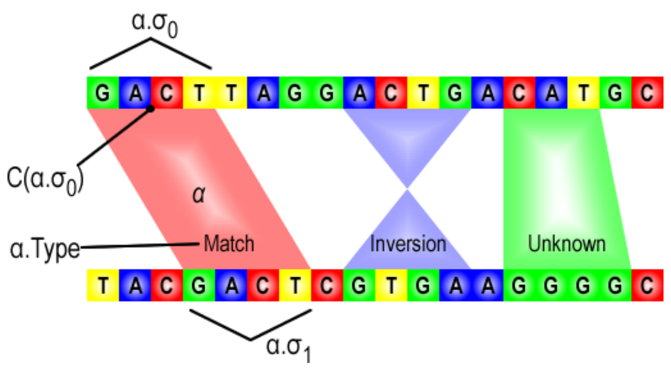

α.σ0
G A C T T A G G A C T G A C A T G C
C(α.σ0)
α
α.Type
Match
Inversion
Unknown
T A C G A C T C G T G A A G G G G C
α.σ1

Figure 3.1. Graphical representations of Homologous Regions

An HR α of sequences S_0 and S_1 is characterized by a 4-tuple:

$$\alpha = (S_0[l_0, r_0], S_1[l_1, r_1], \text{Score}, \text{Type}) \tag{3.5}$$

An HR is denoted by α, β, etcetera. Individual attributes of an HR α are accessible using a dot notation (.), e.g. the Score of α is α.Score. We use a short-hand notation σ_i for subsequences: $S_i[l_i, r_i] = \sigma_i$. In figure 3.1 the attributes of an HR α are displayed: $\alpha.\sigma_i$ are subsequences on S_i, which are the sequences of the pairwise alignment. The width of an HR α is defined as:

$$W_{HR}(\alpha) = \frac{W(\alpha.\sigma_0) + W(\alpha.\sigma_1)}{2} \tag{3.6}$$

The Score of α is an indication of similarity of $\alpha.\sigma_0$ and $\alpha.\sigma_1$. We assume the Score is a positive floating point percentage. In practice, the meaning of Score differs between computational comparative tools.
An example of a score is the percentage of matching bases in a global alignment of two sequences. A global alignment (S'_0, S'_1) of sequences S_0 and S_1 are two gapped sequences, which are constructed in such a way that the number of equal bases on each position is as large as possible. The number of matching bases in subsequences $\sigma, \tau \in \Sigma^*_{al}$ is defined as:

$$\text{Match}(\sigma, \tau) = \begin{cases} (\#i : 0 \leq i < W(\sigma) \wedge \sigma[i], \tau[i] \in \Sigma_{seq} : \sigma[i] = \tau[i]), & \text{if } W(\sigma) = W(\tau); \\ 0, & \text{otherwise.} \end{cases}$$

Definition *A pair* $(\sigma, \tau) \in \Sigma^*_{al} \times \Sigma^*_{al}$ *with:*

i. $W(\sigma) = W(\tau)$

ii. $\pi_{\{\text{-}\}}(\sigma) = S_0 \wedge \pi_{\{\text{-}\}}(\tau) = S_1$

iii. $(\forall i : 0 \leq i < W(\sigma) : \sigma[i] \neq \text{-} \vee \tau[i] \neq \text{-})$

that maximizes Match(σ, τ),
is called a global alignment $(S'_0, S'_1) \in \Sigma^*_{al} \times \Sigma^*_{al}$ *of sequences* $S_0, S_1 \in \Sigma^*_{seq}$.
□

Note that an alignment is not unique and other Match functions can be used. The score of a global alignment (S'_0, S'_1) of sequence S_0 and S_1 is defined as:

$$\text{Score}(S'_0, S'_1) = \frac{\text{Match}(S'_0, S'_1)}{W(S'_0)} \cdot 100\% \tag{3.7}$$

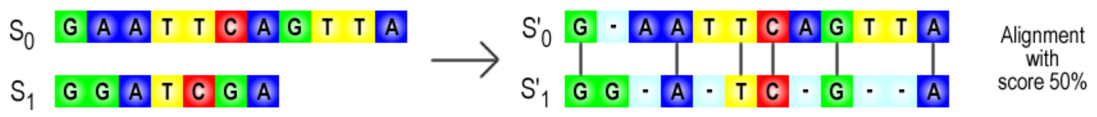


Figure 3.2. An example score of the global alignment of two sequences [20]

An example of a global alignment is displayed in figure 3.2: S'_0 and S'_1 have the same length (i), the gap symbol projection of the solutions results in the original sequences (ii), each index contains at least one base in one of both sequences (iii) and the number of matching bases is optimal. The score of this global alignment is 50%, since 6 bases match out of 12.

The Type attribute of an HR can be either Match, Inversion or Unknown[1]. An HR is a Match if the subsequences are similar - high score - when reading them in the same direction. An Inversion occurs if subsequences are similar when reading them in different directions. In section 5.1, the semantics of an Unknown typed HR will become clear.

3.2 The multiple sequence alignment problem

Given two or more DNA sequences, the Multiple Sequence Alignment (MSA) problem identifies similar subsequences, which occur in given DNA sequences. A pair of similar subsequences is also known as an HR. At least two flavors of HRs are common in literature. The first interpretation insists that a similar subsequence occurs in all sequences. In the second interpretation, a similar subsequence only occurs in at least two sequences. In this project the latter interpretation is investigated. Consequently, sequences only need to be compared pairwise.

A set of HRs is identified by tools like Mummer [17] or a tool of the Blast family [18], resulting in a set of HRs. Because of efficiency reasons, this set of HRs is determined using a set of constraints such as: a minimal width of a subsequence or when the algorithm should stop to extend a subsequence if the score is too low.

Because the set of HRs is large - thousands - and *may* not propagate an obvious pattern, visualization can aid the biologist to interpret them.

3.3 Visualization challenge

Plant Research International wants to be prepared for analysis of upcoming sequence data of various species of tomatoes, as a result of the International Tomato Sequencing Project [3]. Because interactive analysis of multiple sequences is not available in current tools yet (see chapter 4), the interactive genome browser DNAVis2 is extended. Unfortunately, sequence data of various tomato species is not available yet, so we have to search for alternatives.

[1]From now in this report, matches are colored red, inversions are tinted blue and unknowns are rendered in green. These colors are configurable in the application.

A naive way of visualization displays HRs as straight lines (see figure 3.3). On the left, a set of HRs is displayed: one chromosome of a dog, a monkey, a mouse and a human. On the right of figure 3.3, a set of HRs of seven chromosomes of the worm *Caenorhabditis elegans* is displayed. These sets of HRs are both computed using MUMmer, but not with the same parameters. The left alignment is much tighten (more perfect matching) than the right one. From these two alignments we conclude that a pattern of sets of HRs can have various characteristics of complexity. The left rendering can easily be simplified, whereas the right alignment does not express an obvious pattern.

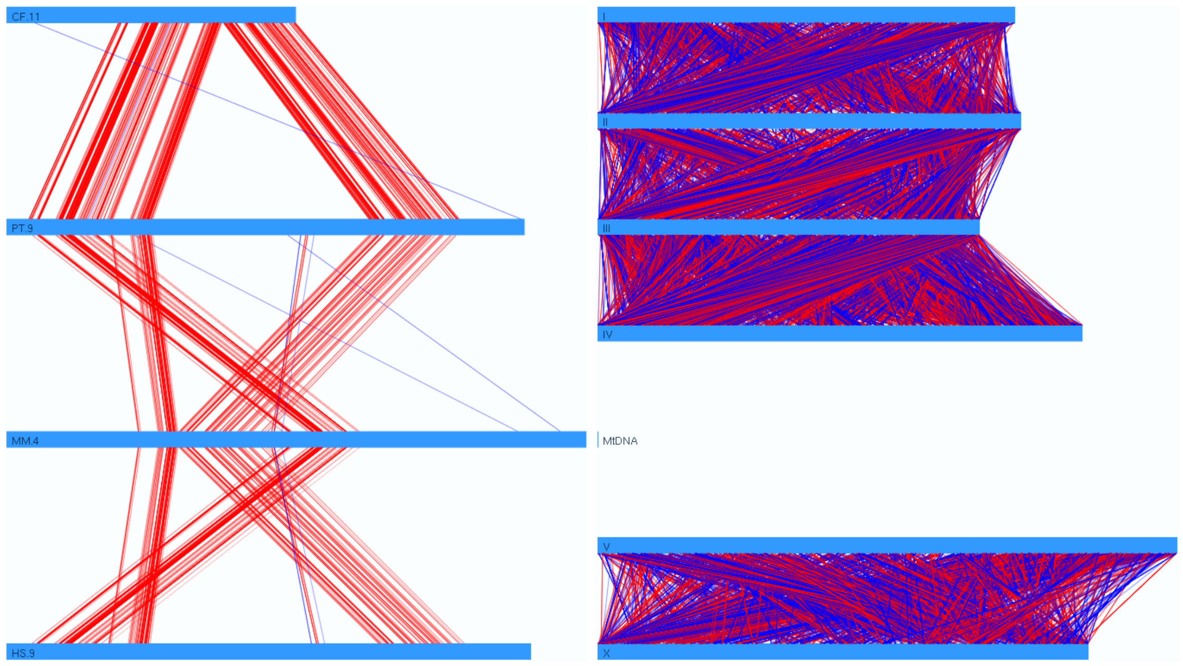


Figure 3.3. Rendering of homologous regions as lines between sequences. On the left, from top to bottom, a chromosome of a dog, monkey, mouse and human are aligned. On the right, all seven chromosomes (I, II, III, IV, MtDNA, V, X) of worm *Caenorhabditis elegans* are displayed.

The complexity of alignments of sequences of tomato sequences is unknown yet. We expect the complexity is not as simple as the left and not as disordered as the right alignment of figure 3.3. Between these complexity boundaries we aim at the following visualization challenge:

The visualization challenge of this project is to emphasize dense locations of HRs in full sequence scale MSAs, which do not express an obvious pattern, when visualized as straight lines.

Inspiration for this research is found in Nature [21], where a set of HRs is computed of the *Arabidopsis thaliana* genome using MUMmer and a BLAST tool. The result is displayed in the left part of figure 3.4, where each band between two subsequences indicates a dense location of HRs.
A data set of HRs of *Arabidopsis thaliana* is found at The Institute for Genomic Research (TIGR) [22], which is naively visualized in the right alignment of figure 3.4. The complexity of this alignment is between the extrema of complexity of figure 3.3, because there seems to be a pattern, but not as simple as the left alignment of figure 3.3. For example, a red area of matches at the end between the second and third sequence catches the eye. Note that the left

part of figure 3.4 displays all dense locations between each pair of sequences, whereas the right part displays only HRs between consecutive sequences.

The TIGR data set represents a good alternative for tomato species: We expect that upcoming sequences of tomato species have a lot of HRs, but may be not as cluttered as the TIGR data set. Hence, we will solve a more difficult problem. An advantage of the TIGR data set is the possibility of verification, since the left part of figure 3.4 indicates dense locations of HRs.

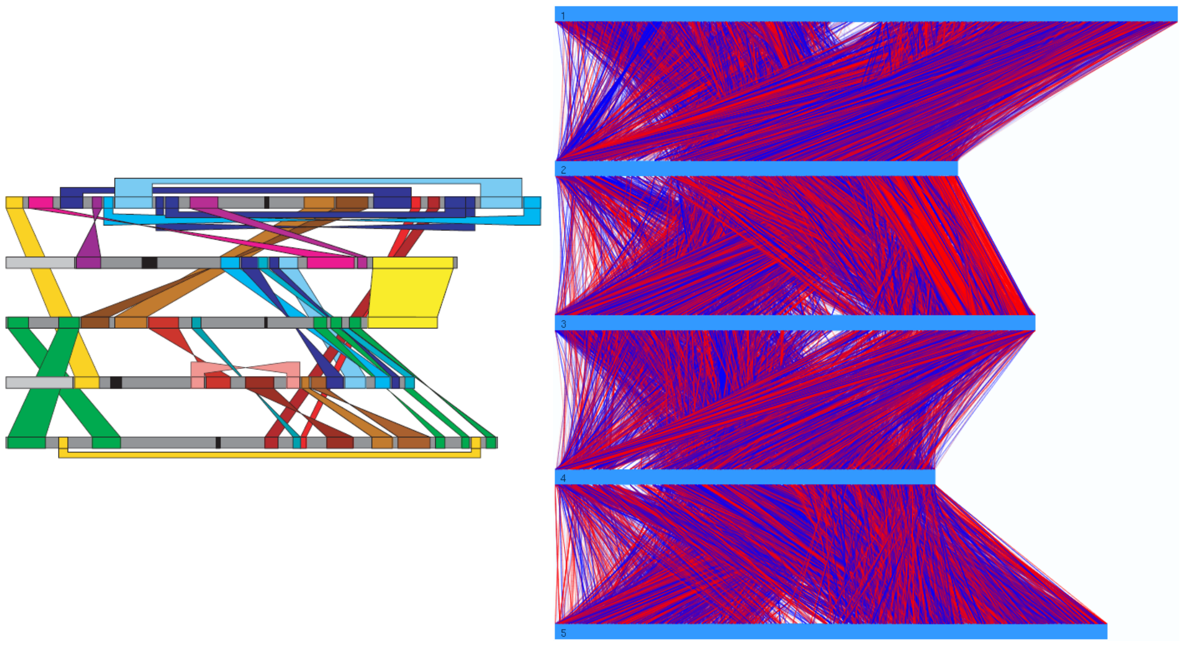


Figure 3.4. Left: an overview of dense locations of homologous regions between the five chromosomes of *Arabidopsis thaliana* [21]. Right: homologous regions [22] of *Arabidopsis thaliana* rendered as straight lines.

3.4 Requirements

This section contains propositions which should hold for the MSA visualization to be made. These propositions - called requirements - are carefully defined in accordance with a group of potential users. In the validation phase described in chapter 6, the system is verified using these requirements. A brief overview of the requirements is listed in appendix C, which is useful when reading further chapters.
The requirements are divided in two parts. The new visualization needs to fit in the existing DNAVis2 framework, as described in subsection 3.4.1. Subsection 3.4.2 states domain specific requirements describing what is required to be an MSA visualization.

3.4.1 DNAVis2 preservation

The new MSA visualization $\mathcal{M}$ is part of the DNAVis2 application (see section 8.1). Therefore $\mathcal{M}$ has to comply with a number of constraints to preserve unique features of DNAVis2. Requirements as stated in this subsection are based on [1].

3.4.1.1 Data sets

Input of DNAVis2 consist of sequence and annotation data sets. By adding $\mathcal{M}$ to DNAVis2 this should remain. This seems to be self-evident, but the input data of $\mathcal{M}$ is structurally different from common annotations, since not one, but two subsequences are labeled with one annotation encoding an HR.

In general, data sets of HRs are generated using computational comparative tools. A separate script converts raw data to an annotation data set, which is loaded in DNAVis2. The number of HRs varies extremely per compared genome. We have encountered up to 200,000 HRs between one pair of sequences, but the most common number of HRs is about 10,000. This results in 250,000 HRs for a full genome of five sequences, which we take as an upperbound of the size of the data set, due to limited resources. Ideally, this results in the following requirement:

Requirement I (Data size) *The user should be able to visualize data sets of arbitrary size.*

In biology research, two types of biologists exist: On one hand, theoretical biologist interpret data sets, on the other hand, bioinformaticians are interested in composition and computation of data sets. Consequently, some web-based genome browsers, like GBrowse at FlyBase [23], contain only a fixed number of sequences prepared by bioinformaticians and intended for theoretical biologists. Due to limited public resources, web-based genome browsers allow uploading of a few small data sets. We want that users are not limited to a fixed number of data sets:

Requirement II (Own data sets) *The user should be able to use its own data set.*

3.4.1.2 Levels of detail

DNAVis2 is known for its real-time continous zooming capabilities from base level to overview level. Zooming is needed, because a biologist can have hypotheses at various levels of the genome. To recognize low level patterns using annotations, the following requirement should hold:

Requirement III (Bases) *The user should be able to view individual bases of a DNA sequence.*

Apart from details, the user should always be able to retrieve patterns from a higher level up to full sequence scale:

Requirement IV (Overview) *The user should always have an overview of a full data set.*

3.4.1.3 Interaction

In comparison with existing web-based annotation browsers, DNAVis allows for responsive interaction. Graphics hardware allows rapid updates of a visualization, resulting in smooth animations. To influence these animations in a useful way, actions of the user are needed. These actions include - but are not limited to - panning, zooming and selection:

Requirement V (Navigation) *The user must be able to navigate through the data set in an interactive and continuous way.*

Interaction is not limited to one view because of connectivity of various views, called linking (see subsection 5.4.3). In DNAVis2 it is allowed to link different views, therefore $\mathcal{M}$ has to be able to cope with zooming and panning:

Requirement VI (Link preservation) *The user must be able to link $\mathcal{M}$ to other* DNAVis2 *views.*

3.4.2 Domain requirements

This subsection contains requirements related to visualization of HRs. These requirements describe what is needed to get insights in the MSA problem.

3.4.2.1 Data sets

The data to be visualized is a set of HRs. These HRs indicate locations of similarity. Biologists are curious to discover how these locations are distributed:

Requirement VII (Homologous regions) *The user should be able to view locations on sequences which are similar to each other.*

HRs are more complex than ordinary annotations from the Linear and Matrix view in DNAVis2. Therefore some explicit assumptions have to be made. In [1] it is not stressed that annotations are - in most cases - results of different computational tools in a homogenous format. The following assumption states explicitly that DNAVis2 is a visualization tool and not a computation tool:

Assumption I (Pre-computation) *The user should pre-compute data sets of homologous regions.*

Annotations should be delivered in the General Feature Format (GFF) data format (appendix E) in DNAVis2. Because of the ability to express hierarchical relations $\mathcal{M}$ will only visualize GFF3 data sets. Data sets in GFF2 are not guaranteed to be visualized properly in $\mathcal{M}$:

Assumption II (Data format) *The user should deliver data sets of homologous regions in GFF3 format.*

In contrast to the Matrix view, multiple sequence alignments display similarities of an arbitrary number of sequences. Because of a finite capacity of screen space, we limit ourselves in practice to five sequences to satisfy the following requirement:

Requirement VIII (Number of sequences) *No assumption should be made for the number of sequences in the visualization.*

3.4.2.2 Interests

It is unlikely that all HRs are interesting while revealing a hypothesis. Therefore mechanisms should be available to specify and emphasize *homologous regions of interest*.

Biologist may be interested in HRs because of one or more known subsequences. A subsequence implies interests for a certain interval on the sequence. If an interval is defined, then all HRs in this interval are of interest to the user. By linking an interval to a Linear view, annotations may provide a better indication whether the region is really similar to another:

Requirement IX (Region of interest) *The user should be able to specify one or more intervals on sequences which are more interesting than others.*

HRs may be of interest because of other reasons than a location as stated in requirement IX. Other properties of HRs which may indicate interest are:

i. A score which exceeds a threshold value
ii. Length of one of the two intervals of an HR
iii. The type of the HR
iv. On which sequence(s) it is located
v. Overlap with an annotation of interest

The user should be able to express his interest by means of a set of these criteria:

Requirement X (Filtering) *The user should be able to specify - using criteria i to v of subsection 3.4.2.2 - a subset of homologous regions which are of interest.*

Obviously, HRs of interest should be displayed more catching than non-interesting. This requirement implicitly states existence of a context of HRs of interest, since an HR can only appear catchy if others do not:

Requirement XI (Emphasizing) *Homologous regions of interest should be emphasized by $\mathcal{M}$.*

3.4.2.3 Manipulation

It may occur that a visualization does not express the information as intended by the biologist. However, the biologist is convinced that the information is in the data. Therefore, techniques should be available which allow the biologist to add knowledge, which influences the visualization.

Highly conserved sequences, such as chromosomes of one organism, tend to become a mess when drawing similarities between sequences as straight lines (see the right part of figure 3.3). A technique is needed to provide insights by uncluttering of the HRs:

Requirement XII (Uncluttering) *The user should be able to unclutter homologous regions of highly conserved sequences.*

HRs which are closely located, may indicate a larger combined HR. Abstraction provides a way to reduce the number of individual HRs, which decreases noise. The abstraction mechanism provides an opportunity to get a better overview:

Requirement XIII (Abstraction) *The user should be able to join homologous regions to get a better overview.*

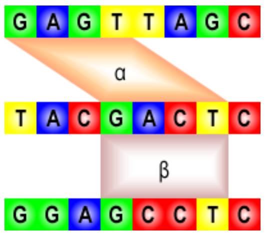


Figure 3.5. Derivation of HRs

If a set of HRs of interest is defined, relations with other pairwise alignments are interesting. If an HR of interest overlaps with an HR which is not of interest, the latter one can become of interest because it is similar. Figure 3.5 displays the transitive relation of similarity: if HR α is of interest then β may also be of interest, because it expresses a similar subsequence:

Requirement XIV (Derivation) *The user should be able to propagate homologous regions of interest in one pairwise alignment to other pairwise alignments by means of overlap of homologous regions.*

Chapter 4

State of the art in visual comparative genomic tools

This chapter contains a feasibility study of a multiple sequence alignment visualization as described in the previous chapter. By verification of our requirements to existing similar methods, gaps in a comparison matrix indicate feasibility of the new visualization.

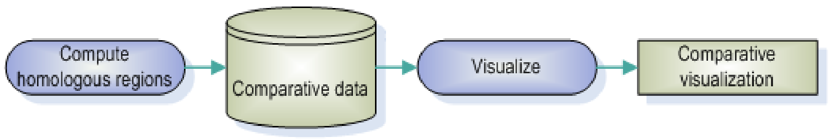


Figure 4.1. A part of the data flow of existing comparative visualization methods

The majority of comparative visualization tools do not use human perception to recognize HRs, such as the Dotplot (figure 2.3). In figure 4.1 the data flow is displayed of comparative visualization tools: Computational tools like Blast and MUMmer (subsection 2.3.3) compute a set of HRs. This set is stored in a tool-specific format or in an annotation data format such as GFF (appendix E.2). Using the set of HRs a comparative visualization should provide insights in the distribution of HRs. Dense locations of HRs indicate interesting positions, because of high similarity. A biologist may also be interested in specific locations on a sequence because of an annotation, to search for similar occurrences.

Selected methods in this chapter are either promising concepts or real-world tools, used in industry and research. Each description of a tool contains a table that lists results of verification of the requirements as enumerated in appendix C. Some requirements may not be verifiable for concepts because of implementation issues (such as input and interaction).
The following methods are discussed in this chapter: Circos [24], GenomePixelizer [25], Artemis Comparison Tool [26], Multiple Genome Comparison and Alignment Tool [16], Biological Arc Diagrams [27], MAtDB's Redundancy Viewer [28], Vista [29] and SynBrowse [30].

4.1 Circos

Comparative genomics is one of the various applications of the generic Circos [24] application. Circos (figure 4.2) displays multiple sequences on the border of a circle, which encloses curves representing HRs. Using criteria, a color can be assigned to an HR. In figure 4.2 regions between mouse and human are black, regions between mouse and rat are blue and regions between rat and human are orange.

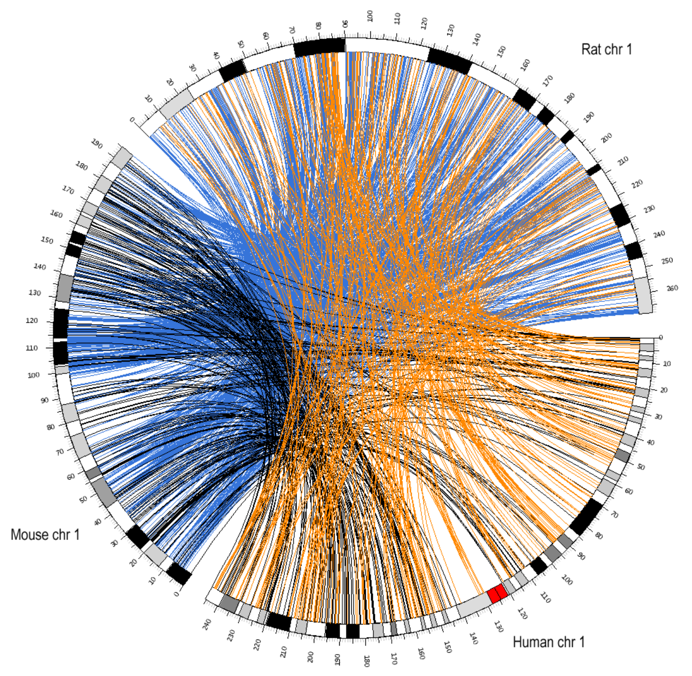


Figure 4.2. Comparison of the first chromosome of human, mouse and rat using Circos [24]

To use Circos an own web-server has to be set up, because Circos is a web-based application which generates static images without interaction. Using color maps and filtering it is somehow possible to emphasize regions of interest. Unfortunately, uncluttering is not solved in this tool. This would improve the application tremendously.

	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	XIII	XIV
Circos	✓	✓	✗	✓	✗	✗	✓	✓	✗	∽	∽	✗	✗	✗

4.2 GenomePixelizer

GenomePixelizer [25] allows browsing through a data set containing both HRs and annotations. This tool is not user friendly due to various input text files and a disturbing graphical user interface.

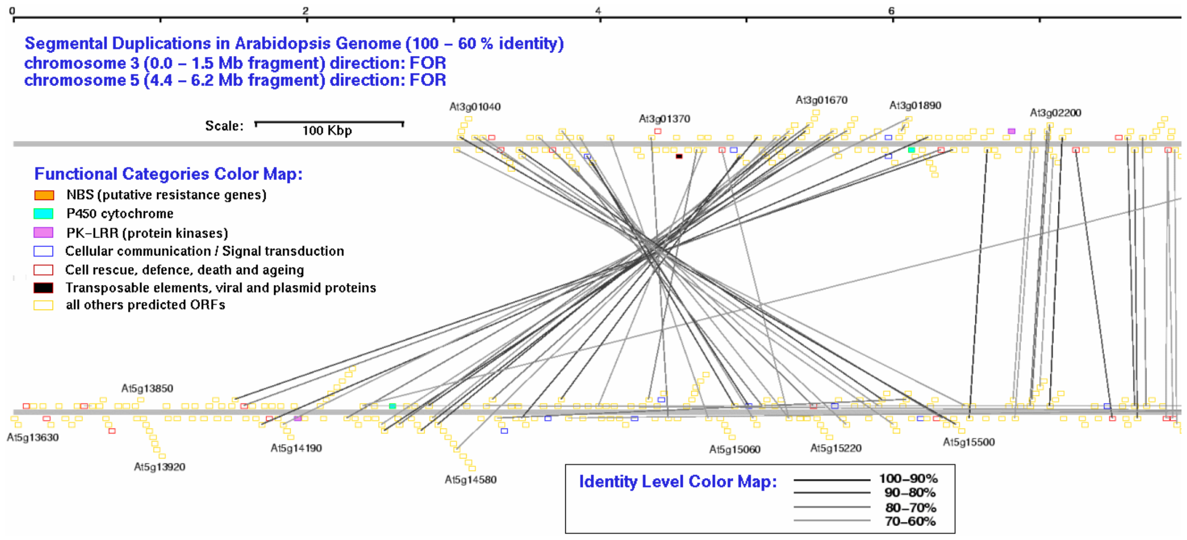

0
2
4
6
Segmental Duplications in Arabidopsis Genome (100 – 60 % identity)
chromosome 3 (0.0 – 1.5 Mb fragment) direction: FOR
chromosome 5 (4.4 – 6.2 Mb fragment) direction: FOR
Scale:
100 Kbp
At3g01040
At3g01370
At3g01670
At3g01890
At3g02200
Functional Categories Color Map:
NBS (putative resistance genes)
P450 cytochrome
PK–LRR (protein kinases)
Cellular communication / Signal transduction
Cell rescue, defence, death and ageing
Transposable elements, viral and plasmid proteins
all others predicted ORFs
At5g13850
At5g13630
At5g13920
At5g14190
At5g14580
At5g15060
At5g15220
At5g15500
Identity Level Color Map:
100–90%
90–80%
80–70%
70–60%

Figure 4.3. Homologous regions and annotations of a part of chromosome III and V of the *Arabidopsis thaliana* genome in GenomePixelizer [25]

In figure 4.3, sequences are displayed as horizontal lines on top of each other. If annotations are provided to the tool, then these are stacked on top and below the sequence. Homology is defined by similarity of annotations. Remarkable is the feature of emphasizing manually defined groups of annotations by means of a color map. For example, in figure 4.3 annotations are grouped by functional categories.

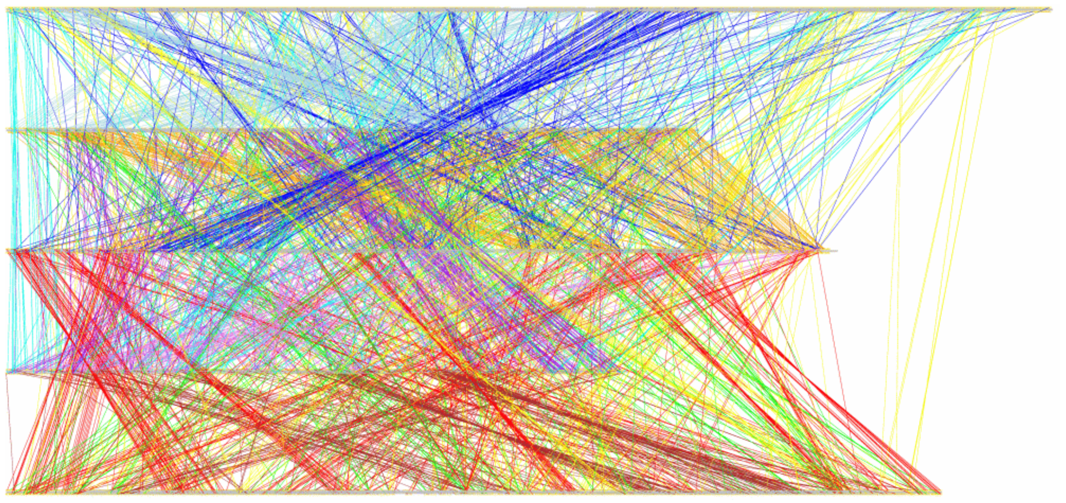


Figure 4.4. Homologous regions with at least 80% identity of the *Arabidopsis thaliana* genome in GenomePixelizer [25]

Figure 4.4 is generated with GenomePixelizer and displays an overview of the *Arabidopsis thaliana* genome. This overview allows recognition of dense locations of HRs, but contains a lot of noise. GenomePixelizer does not support mouse interaction. It is possible to argue that grouping of annotations is a kind of abstraction, but grouping is not performed by the tool.

	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	XIII	XIV
GenomePixelizer	✓	✓	✗	✓	✗	✗	✓	✓	✗	∽	✗	✗	∽	∽

4.3 Artemis Comparison Tool

The Artemis Comparison Tool (ACT) [26] is an interactive tool for comparison of annotated sequences (figure 4.5). ACT is not limited to browsing, because creation and editing of annotations are allowed. A strong point of ACT is connectivity to web-services, which perform small computations, like a BLAST search or an alignment for a couple of thousand bases. Another unique feature (with respect to this set of selected tools) is the ability to add various graphs on top of a sequence which calculate heuristics of a sequence, like GC-content.

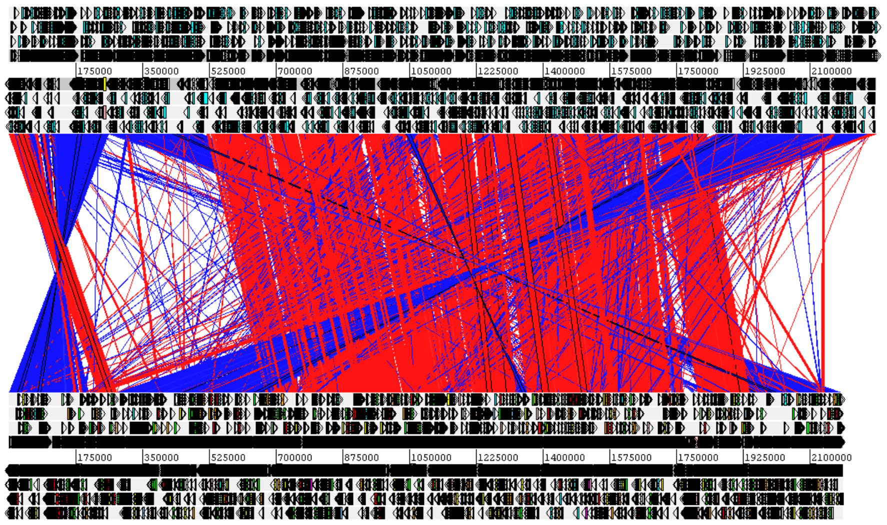


Figure 4.5. A comparison of bacteria *neisseria meningitidis MC58* on top and *neisseria meningitidis Z2491* (both from ACT's example web-page [26]) on the bottom in Artemis Comparison Tool. The arrows indicate annotations, the red and blue rectangles indicate matches and inversions respectively.

Levels of detail are highly flexible, since both individual bases and overview are observable, however not continuously. Continuous panning and responsive selection fulfill the Navigation requirement partially.

	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	XIII	XIV
ACT	✓	✓	✓	✓	∽	✗	✓	✓	∽	✓	∽	✗	✗	✗

4.4 SynBrowse

In [30] a synteny browser is proposed called SynBrowse (figure 4.6). Synteny is a special case of searching for homology, where an order-preserving set of HRs is computed, which suggests

evolutionary relationships. This means that matching intervals have the same order on both sequences. From a visualization perspective synteny is delightful, because by definition crossings of HRs are reduced to a minimum. However, the data contains crossing HRs, hence for non-evolutionary purposes, SynBrowse deceives the user by over-simplification.
SynBrowse is an extension of the web-based Generic Genome Browser [31], which is popular in the bioinformatics community.

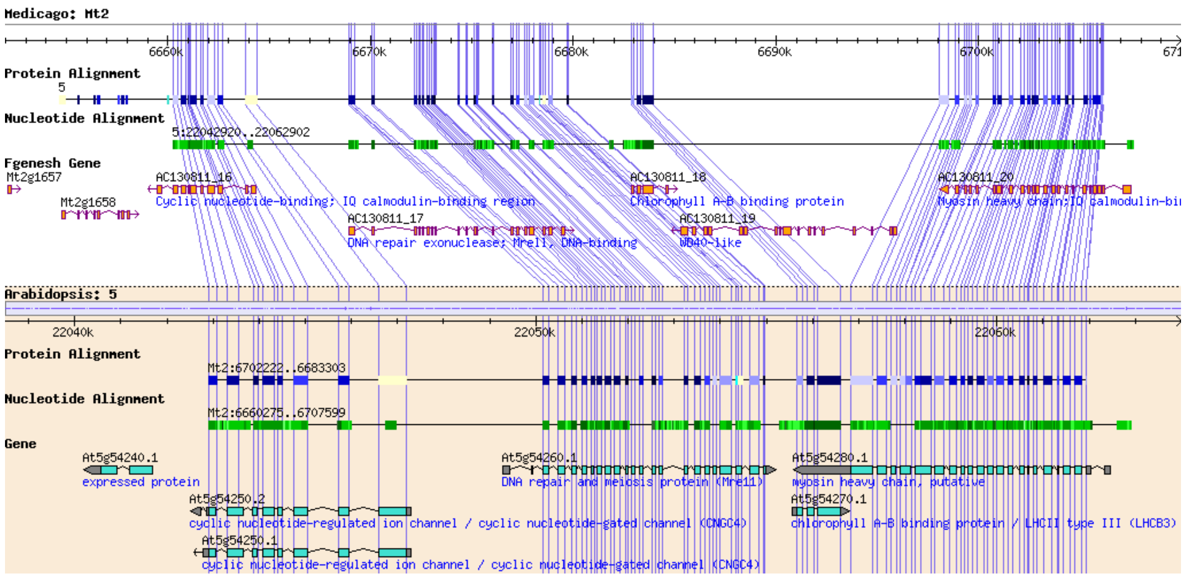

Medicago: Mt2
6660k
6670k
6680k
6690k
6700k
Protein Alignment
Nucleotide Alignment
5:22042920..22062902
Fgenesh Gene
Mt2g1657
Mt2g1658
AC130811_16
Cyclic nucleotide-binding; IQ calmodulin-binding region
AC130811_17
DNA repair exonuclease; Mre11, DNA-binding
AC130811_18
Chlorophyll A-B binding protein
AC130811_19
WD40-like
AC130811_20
Myosin heavy chain; IQ calmodulin-bi
Arabidopsis: 5
22040k
22050k
22060k
Protein Alignment
Mt2:6702222..6683303
Nucleotide Alignment
Mt2:6660275..6707599
Gene
At5g54240.1
expressed protein
At5g54250.2
cyclic nucleotide-regulated ion channel / cyclic nucleotide-gated channel (CNGC4)
At5g54250.1
cyclic nucleotide-regulated ion channel / cyclic nucleotide-gated channel (CNGC4)
At5g54260.1
DNA repair and meiosis protein (Mre11)
At5g54280.1
myosin heavy chain, putative
At5g54270.1
chlorophyll A-B binding protein / LHCII type III (LHCB3)

Figure 4.6. A detailed comparison of parts of chromosome 2 of *medicago truncatula* on top and chromosome 5 of *Arabidopsis thaliana* on the bottom in SynBrowse.

Both individual bases and an overview are possible to generate. The application is web-based, resulting in bad responsiveness of interactions (zooming and panning using textboxes). Because of synteny, unclutting is for free. SynBrowse allows various kinds of alignments at different levels, therefore the Abstraction requirement is partially fulfilled.

	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	XIII	XIV
SynBrowse	✓	✓	✓	✓	✗	✗	✓	✓	✗	✗	✗	∽	∽	✗

4.5 Multiple Genome Comparison and Alignment Tool

In the Multiple Genome Comparison and Alignment Tool (M-GCAT) [16] both alignment algorithm and interactive visualization are combined in one tool (figure 4.7). M-GCAT is a proper MSA tool, meaning that M-GCAT aligns multiple sequences instead of combining multiple pairwise alignments as explained in subsection 2.3.3.4. Alignment is done using multi-MUMs (see subsection 2.3.3.4) resulting in fast results. Experiments demonstrate scalability to at least ninety entrobacterial sequences.

The visualization tool produces a pretty good overview and selection of a multi-MUM allows visibility of individual bases. The smooth overview of HRs using M-GCAT may deceive the user,

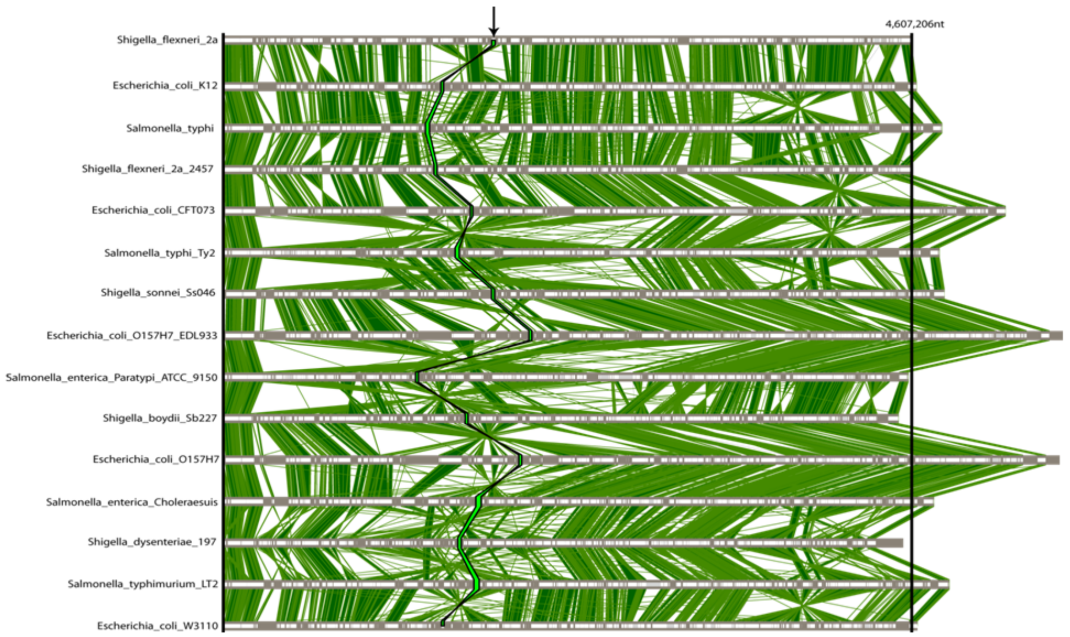


Figure 4.7. A comparison of fifteen entrobacterial sequences in M-GCAT [16]. One multi-MUM is highlighted under the arrow.

because cluttered HRs are hidden. Navigation is far from continuous, but includes panning, zooming and selection. By selecting one multi-MUM the derivation requirement is for free.

	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	XIII	XIV
M-GCAT	✓	✓	✓	✓	✗	✗	✓	✓	✗	∽	✗	✗	∽	✓

4.6 Biological Arc Diagrams

The concept of Biological Arc Diagram (BARD) [27] is an extension of Wattenberg's Arc Diagrams [32]. In an arc diagram, the sequence under investigation is drawn as a horizontal line. Straight sides of half a ring are positioned at similar subsequences. BARDs (figure 4.8) can cope not only with exact matches, but also with fuzzy matches and reverse strand matches. The latter matches are those where one side in reverse order matches the other's complementary bases.

BARDs can be used as an MSA tool, by putting all sequences under investigation next to each other, with whitespace between them. However, it seems to become a mess on full sequence scale, because of the number of crossing arcs. Another disadvantage of BARDs is the lack of emphasis of the right noticeable results: Arcs with a large radius catch more the eye than small wide arcs, but the latter are more important since these represent a larger match.
Individual bases cannot be determined, but BARD results in a pretty good overview. Interaction, interests and manipulation are not applicable, since the application generates static images.

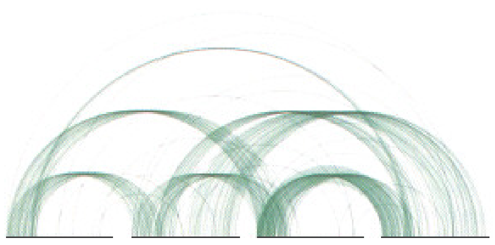


Figure 4.8. A biological arc diagram [27] of four subsequences of the *Cryptococcus neoformans* fungus

	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	XIII	XIV
BARD	.	.	✗	✓	.	.	∽	✓	✗	✗	✗	✗	✗	✗

4.7 MAtDB's Redundancy Viewer

The Redundancy Viewer of the MIPS Arabidopsis thaliana Database (MAtDB) [28] is a visualization tool which is worth to mention. As a result of an image of duplicated segments in the analysis of the *Arabidopsis thaliana* genome [21], the Redundancy Viewer provides interactive analysis of HRs. This tool is completely dedicated to the *Arabidopsis thaliana* genome.
The tools starts with the left side of figure 4.9, then an HR can be selected for further inspection, as depicted in the right side of figure 4.9. In the detailed view, the two sequences involved in the selected HR are drawn opposite to each other. The user can select the size of the clusters or draw single hits (such as genes). Interaction is limited to non-continuous panning and selection by clicking at annotations on the sequence.

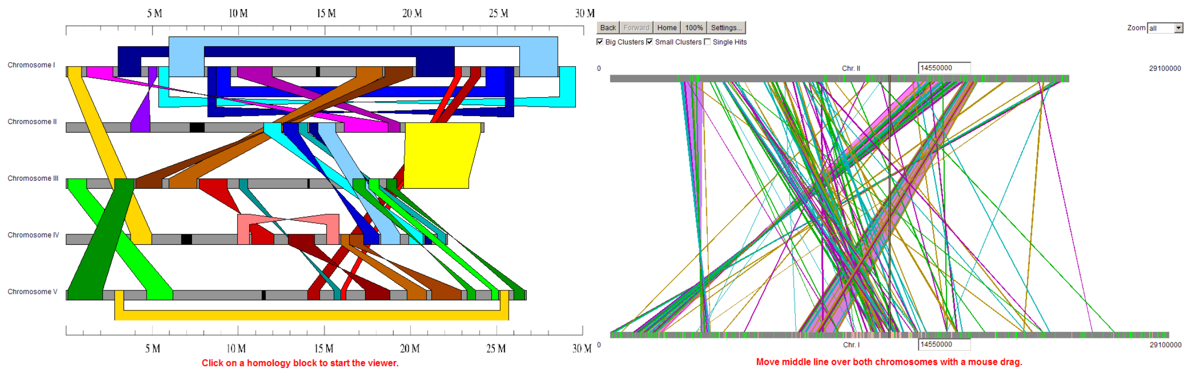


Figure 4.9. MAtDB Redundancy Viewer [28]. On the left the initial overview is displayed of duplicated segments of the *Arabidopsis thaliana* genome. The right side is a detailed view of chromosome I and II.

Because this visualization is limited to the *Arabidopsis thaliana* data set, none of the data requirements hold. The Redundancy Viewer is strong in providing an overview, but individual bases are not visible. The large clusters are probably not generated by the program, therefore the overview capabilities should not be overestimated.

	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	XIII	XIV
MAtDB	✗	✗	✗	✓	∽	✗	✗	✓	✗	∽	✗	∽	∽	✗

4.8 Vista

The Vista plot [29] (figure 4.10) is a graph which displays the relative amount of identity in an adjustable window size between two DNA sequences. On the horizontal axis the base index is displayed and the vertical axis represents the percentage of identity. To compare multiple sequences, a number of graphs are displayed in parallel. Exceeding a threshold value identifies HRs, which are colored under the graph curve. By providing annotation data to Vista, semantics can manually be attributed to HRs.

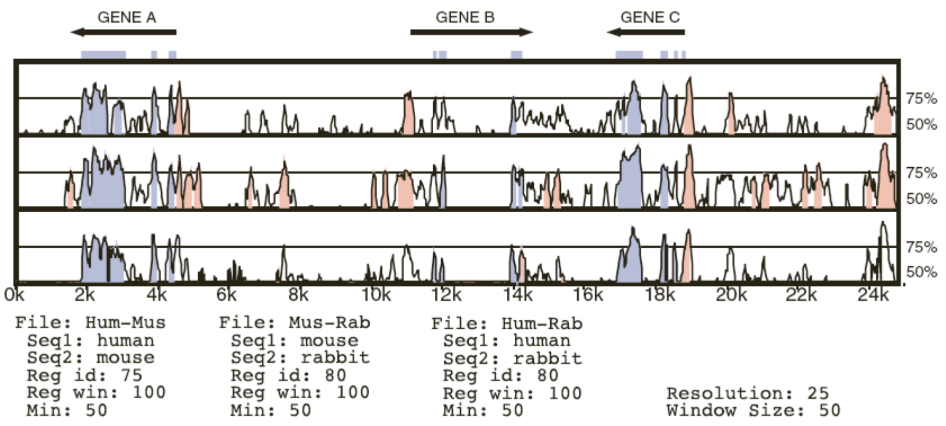


Figure 4.10. Comparison of human, mouse and rabbit in Vista [29]

In the Vista Browser separate bases are distinguishable by means of fragments of gapped alignments. Overview level is created by the zoomable graph. Non-continuous zooming and panning are the only interaction mechanisms. Vista requirers no pre-computed data sets, because it can do this preprocessing by itself.

	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	XIII	XIV
Vista	∽	✓	✓	✓	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗

4.9 Discussion

Various MSA tools are discussed in this chapter, resulting in table 4.1. From this table it is likely to conclude that if a tool fulfills all requirements then it is unique, since none of the tools fulfills all requirements. Hence, the tool is useful to build. By inspecting the table vertically, opportunities arise.

The "Interaction" column contains a lot of crosses indicating that continuous navigation and linked views are not widely used. This observation is not a surprise, because none of the tools is build on top of graphics hardware. Some of them (like Circos and SynBrowse) are even web-based, which is disastrous for a real-time experience.
From the "Interest" column it is fair to conclude that current tools are not flexible enough to

	DNAVis2 preservation						Domain specific							
	Data sets		LoD		Interaction		Data sets		Interests			Manipulation		
	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	XIII	XIV
Circos	✓	✓	✗	✓	✗	✗	✓	✓	✗	∽	∽	✗	✗	✗
GenomePixelizer	✓	✓	✗	✓	✗	✗	✓	✓	✗	∽	✗	✗	∽	∽
ACT	✓	✓	✓	✓	∽	✗	✓	✓	∽	✓	∽	✗	✗	✗
SynBrowse	✓	✓	✓	✓	✗	✗	✓	✓	✗	✗	✗	∽	∽	✗
M-GCAT	✓	✓	✓	✓	✗	✗	✓	✓	✗	∽	✗	✗	∽	✓
BARD	.	.	✗	✓	.	.	∽	✓	✗	✗	✗	✗	✗	✗
MAtDB	✗	✗	✗	✓	∽	✗	✗	✓	✗	∽	✗	∽	∽	✗
Vista	∽	✓	✓	✓	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗

Table 4.1. An overview table of existing comparison methods. Each column corresponds with one requirement of appendix C. Each row verifies all requirements of the method as described in the first column. A checkmark (✓) indicates that the requirement holds, a cross (✗) indicates that the requirement is not fulfilled, a reversed tilde (∽) indicates partial fulfillment of a requirement and a dot (.) indicates that the requirement is not applicable, because the method is a concept instead of a tool.

query for a group of interesting HRs. Filtering is limited implemented in current tools, however it may be the case that our definition of filtering is too exhausting. Since an HR is quite artificial, because of its fuzzyness, it might be the case that more filtering criteria do not result in more information from a biological perspective. For authors of current tools, it seems not so obvious that interesting HRs should catch more the eye to allow a biologist to draw a conclusion from relevant data.

Most promising requirement is the Uncluttering requirement of the "Manipulation" column, because none of the investigated tools deal with that issue. Some of the tools (like SynBrowse) avoid the problem by leaving out crossing HRs, but that does not solve the problem. It is possible that the sequence does not represent a healthy organism. Due to an ill sequenced organism, incomplete or erroneous sequencing, the order of the sequence may be disturbed. If crossing HRs are hidden, then using this visualization a biologist may easily draw a wrong conclusion.
Abstraction is also a requirement that is not implemented properly in any of the tools. A lot of HRs try to express a larger HR, but current tools do not try to combine them to get a better overview.
The last manipulation requirement is the Derivation requirement. Because similarity is - up to a certain degree - a transitive relation, it seems to be obvious that it should be possible to connect HRs between pairwise alignments. Only a few of the tools support derivation of HRs.

There is another notion not discussed so far. Some authors are aware of annotations and try to use them as boundaries of HRs. Tools like GenomePixelizer, ACT, M-GCAT, Redundancy Viewer, Vista and SynBrowse use some types of annotations, like predictions of genes. This is clever, because HRs matching annotations have semantics. However, due to fuzzyness of HRs a biologist should be aware that a match of annotations may be wrong.

4.10 Conclusion

Having seen the tools discussed in this chapter a number of conclusions can be drawn. The input of the user is the first lesson learned. Interactively expressing knowledge by a biologist could improve the way of viewing a visualization, resulting in a more sophisticated conclusion. Patience is needed when comparing multiple sequences of a couple of million base pairs, even when the comparison is pre-computed.

A main improvement which can make a difference with other tools is the removal of cluttering of HRs in MSA visualization. So far, none of the tools solves this problem. It is unclear whether this is unmentioned as a problem by biologists or is avoided by redefinition of homology (e.g. synteny). A fact is, that cluttered HRs are disturbing a visualization. We will try to solve this problem in the next chapter.

Chapter 5

Multiple sequence alignment visualization in DNAVis2

As became clear in chapter 4, no multiple sequence alignment tool that allows expression of interesting HRs, reduction of noise and uncluttering of crossing HRs exists. This chapter presents a novel visualization based on an existing visualization concept, which is integrated in the existing DNAVis2 application. The visualization is called *MSAView* (see figure 5.1).

We have chosen to decompose the MSA visualization in pairwise alignments which can be joined. For a pairwise alignment the Hierarchical Edge Bundles (HEB) visualization concept [34] is used, which resolves uncluttering of straight lines (see figure 5.2). This concept matches our problem to create order in a chaotical distributed set of HRs. The chosen visualization concept requires hierarchical relations between HRs, which are not available in the given data. Therefore, in section 5.1 an hierarchical cluster algorithm is presented, which creates a cluster tree of HRs based on a distance function on HRs. Having a cluster tree, in section 5.2 the HEB concept is adapted in such a way that it fits a pairwise alignment. In section 5.3 the pairwise alignments are joined, the multiple sequence alignment is applied to real world data, and techniques are introduced to reduce the data set. By embedding the visualization in DNAVis2 in section 5.4, the MSAView is completed. A summary of all parameters of the MSAView is given in section 5.5.

5.1 Distance based hierarchical clustering

In [35] a hierarchical cluster algorithm is found, which hierarchically clusters HRs. This algorithm is based on a hierarchical cluster scheme [36]. We have extended this algorithm such that clusters can be treated as large HRs, by aggregation of a type and a score. The algorithm is described using HRs as defined in section 3.1 on page 19.

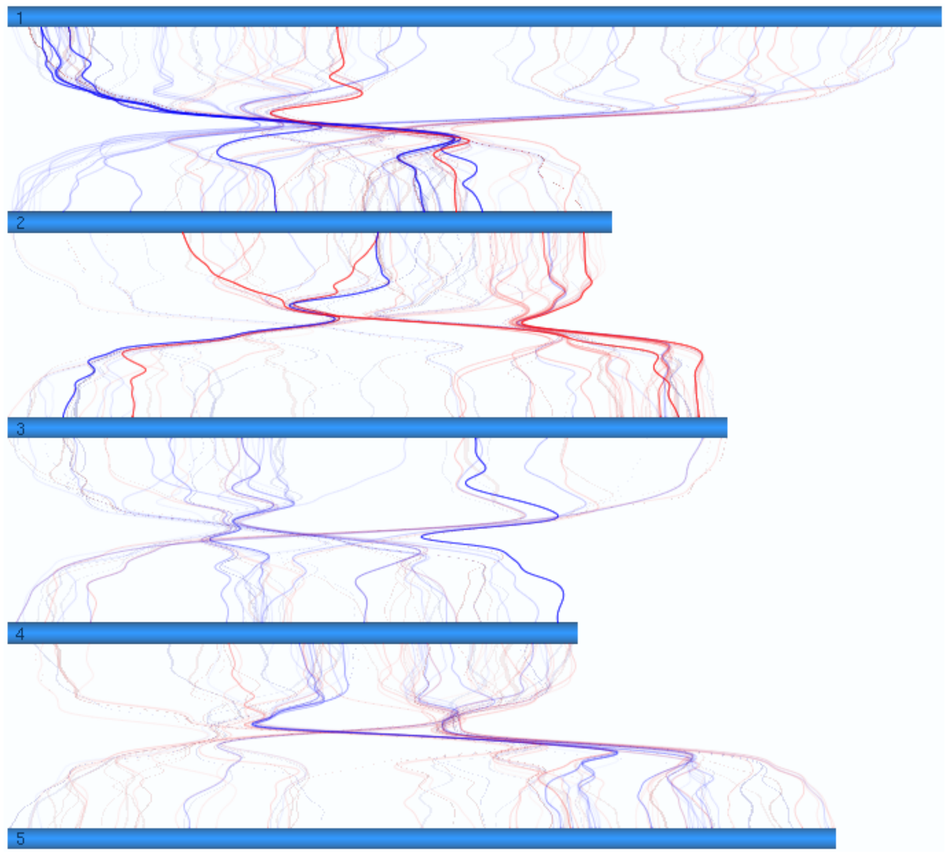


Figure 5.1. An MSAView of an *Arabidopsis thaliana* data set of The Institute for Genomic Research (TIGR) [22].

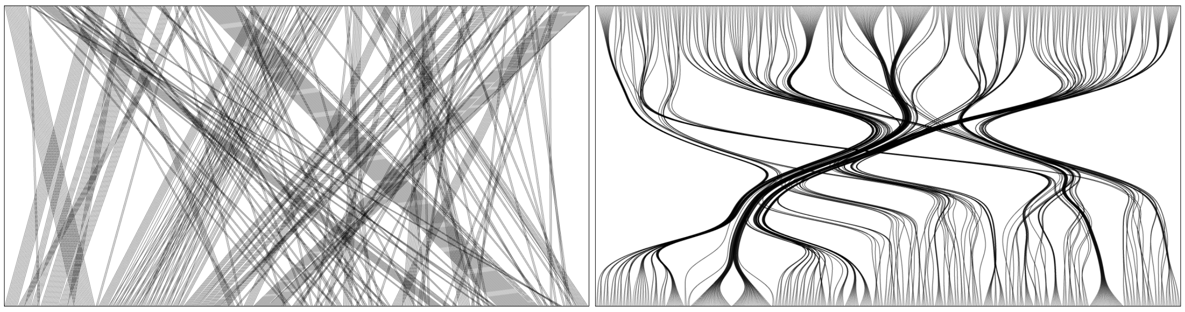


Figure 5.2. A preview of new HEB results: TreeComparer [33]. Left: straight line visualization. Right: hierarchical edge bundles.

5.1.1 Grouping by distance between HRs

Given a set U of N HRs, the following hierarchical cluster algorithm computes a cluster tree $\mathcal{T}$ consisting of vertices (HRs) V and edges E_T (see figure 5.3). Until $\mathcal{T}$ is completed, the two closest partial trees (clusters of HRs) are joined in one new partial tree:

i. $V := U$.
ii. Copy all elements of V to the set of roots of partial trees R: $R := V$
iii. Let $\alpha, \beta \in R$ be a pair with the smallest distance $\Delta(\alpha, \beta)$ (defined below).
iv. Aggregate : $\gamma := \alpha \bowtie \beta$ ($\bowtie$ is defined in subsection 5.1.2).
v. Add γ to $\mathcal{T}$: $V := V \cup \{\gamma\}$; $E_{\mathcal{T}} := E_{\mathcal{T}} \cup \{(\gamma, \alpha), (\gamma, \beta)\}$.
vi. Remove α and β from the roots of partial trees: $R := R \setminus \{\alpha, \beta\}$
vii. Add γ to the roots of partial trees: $R := R \cup \{\gamma\}$
viii. Repeat steps iii...viii until R contains only one HR.

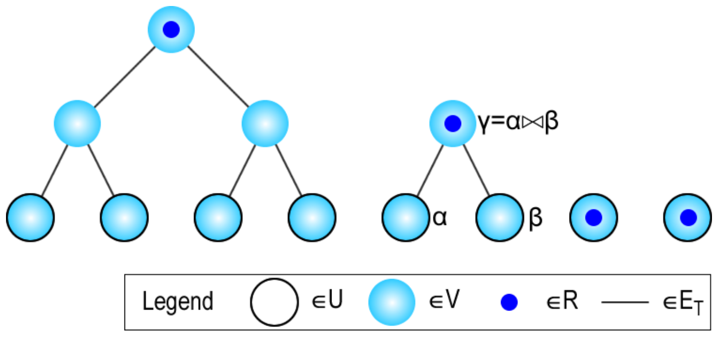


Figure 5.3. A schematic overview of variables in an intermediate state of the clustering algorithm.

Let $H_{\mathcal{T}}(\alpha)$ be height of a subtree of $\mathcal{T}$ rooted at $\alpha \in V$:

$$H_{\mathcal{T}}(\alpha) = \begin{cases} 0, & \text{if } isLeaf_{\mathcal{T}}(\alpha); \\ (\mathbf{max}\ \beta : (\alpha, \beta) \in E_{\mathcal{T}} : H_{\mathcal{T}}(\beta)) + 1, & \text{otherwise.} \end{cases} \tag{5.1}$$

$$isLeaf_{\mathcal{T}}(\alpha) = \neg(\exists \beta : \beta \in V : (\alpha, \beta) \in E_{\mathcal{T}}) \tag{5.2}$$

Let $L_{\mathcal{T}}(\alpha)$ be the leafs of a subtree of $\mathcal{T}$ rooted at $\alpha \in V$:

$$L_{\mathcal{T}}(\alpha) = \begin{cases} \{\alpha\}, & \text{if } H_{\mathcal{T}}(\alpha) = 0; \\ (\cup \beta : (\alpha, \beta) \in E_{\mathcal{T}} : L_{\mathcal{T}}(\beta)), & \text{otherwise.} \end{cases} \tag{5.3}$$

Assume δ is the distance between two HRs (see subsection 5.1.4). The distance $\Delta(\alpha, \beta)$ between two subtrees as defined in [35] is:

$$\Delta(\alpha, \beta) = \frac{(\Sigma \gamma, \zeta : \gamma \in L_{\mathcal{T}}(\alpha) \wedge \zeta \in L_{\mathcal{T}}(\beta) : \delta(\gamma, \zeta)))}{|L_{\mathcal{T}}(\alpha)| + |L_{\mathcal{T}}(\beta)|} \tag{5.4}$$

5.1.2 Aggregation $\bowtie$

So far, a cluster tree of HRs is build based on locations. We would like to use the clustering tree as an abstraction mechanism, to simplify the alignment: Individual HRs should be replaceable by an ancestor cluster, represented as an HR. Therefore, the top and bottom (sub)sequence, type , and score should be derived from its children.

From HRs α and β between sequences S_0 and S_1, HR γ can be derived, i.e. $\gamma := \alpha \bowtie \beta$. The aggregated HR γ is defined as follows:

$$\gamma.S_i = \alpha.S_i = \beta.S_i \tag{5.5}$$
$$\gamma.\sigma_i = \text{extend}(\alpha.\sigma_i, \beta.\sigma_i) \textbf{ where} \tag{5.6}$$
$$\text{extend}(S[i,j], S[k,l]) = S[\min(i,k), \max(j,l)] \tag{5.7}$$
$$\gamma.\text{Score} = \frac{\alpha.\text{Score} \cdot W_{HR}(\alpha) + \beta.\text{Score} \cdot W_{HR}(\beta)}{W_{HR}(\gamma)} \cdot 100\% \tag{5.8}$$

The top and bottom sequence of γ are equal to those of α and β as defined in (5.5). In (5.6) is defined that the aggregated subsequences $\gamma.\sigma_i$ are the smallest subsequence including both its children's σ_i. The score function in (5.8) represents the 'real' score of the HR: the number of known matching bases divided by the length of the new HR.

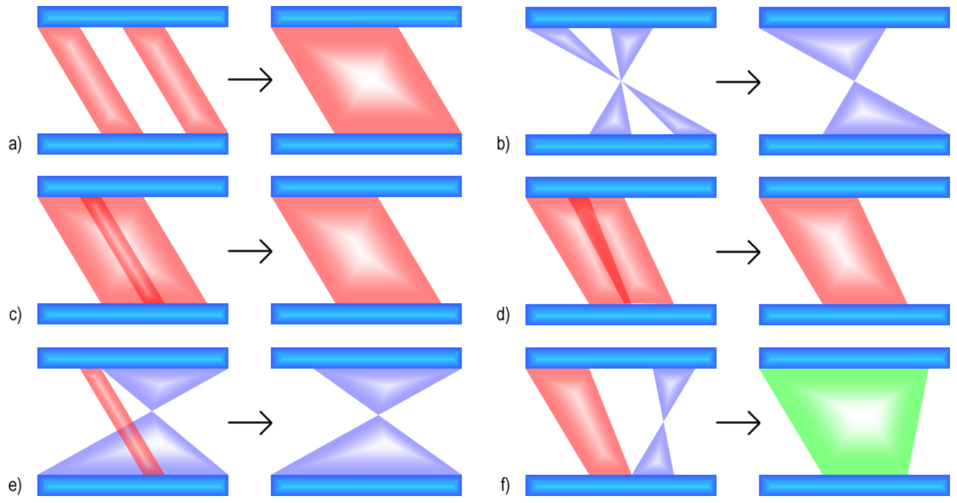


Figure 5.4. A case distinction on the degree of crossing, resulting in a type for the aggregated HR.

The Type is derived based on the degree of crossing as distinguished in figure 5.4: In case 5.4a, two disjunct matches result in a new match. Two fully crossing inversions in 5.4b result in a new inversion. In general, two crossing matches or disjunct inversions do not result in something meaningful. In 5.4c two HRs are fully contained in each other. In this case we take the widest HR, which contains the most matching bases, including those of the smallest. In case 5.4d two matches are partial contained in each other, on both top and bottom sequence. Because of the partial overlap, the order is preserved and only the difference in length of the overlapping parts are not matching. In case 5.4e a large and a relative small HR are aggregated: Let α be such that $W_{HR}(\alpha) \geq W_{HR}(\beta)$, if α.Score $> \mathfrak{S}$ and $|W_{HR}(\alpha) - W_{HR}(\beta)| > \mathfrak{W}$, then γ.Type becomes α.Type, otherwise the unknown type. We have chosen $\mathfrak{S} = 50\%$ and $\mathfrak{W} = \frac{1}{4} \cdot W_{HR}(\alpha)$. In this case the score is adapted, because the bases of the eliminated HR are not matching anymore, i.e. β is eliminated from the dividend in the score function. Finally, in 5.4f, all other cases results in an unknown type, because it is not fair to conclude that the result is either a proper match or inversion.

5.1.3 Example

In figure 5.5 on the right, a cluster tree $\mathcal{T} = (V, E_{\mathcal{T}})$ rooted at $\omega \in V$ is displayed for the data set as depicted left. The cluster tree is displayed *twice*, with one common root. One tree is displayed from the center to the top sequence displaying $\alpha.\sigma_0$ for each HR $\alpha \in V \setminus \{\omega\}$. The bottom tree displays $\alpha.\sigma_1$. Root ω is displayed once, as the longest subsequence of $\omega.\sigma_i$, to use vertical screen space as efficient as possible. Equal labeled subsequences represent *one HR* and vertical lines represent edges in $E_{\mathcal{T}}$. The color of a subsequence indicates the Type of the HR. In this example, we observe that typing of aggregated HRs is converging fast to the 'Unknown' type, because we have chosen to be conservative with respect to typing of an aggregated HR as match or inversion. A more important observation is that the quality of the size of the aggregated HRs is pretty good, because these are not unnecessary large. Note that the size of this data set is too small to make a fair conclusion about quality.

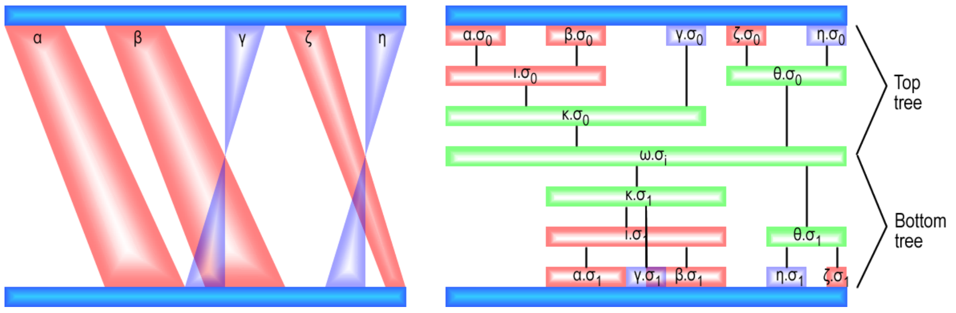


Figure 5.5. Left: an example data set of HRs. Right: a manually calculated cluster tree of the example data set at the left.

5.1.4 HR distance δ

In (5.4), we postponed the choice for a distance δ between two HRs. The choice for any δ is difficult, because δ influences locally the global result. The local decision may be conflicting with the global goal, such as allowing crossing HR locally, but not globally.

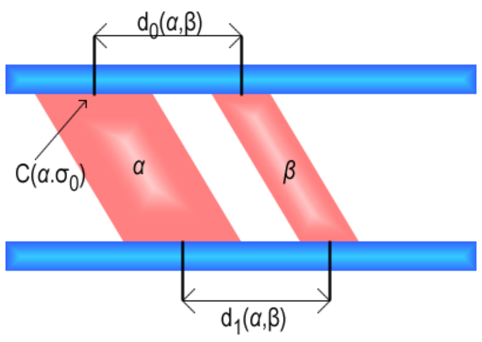


Figure 5.6. Graphical representation of d_i.

Let $d_i(\alpha,\beta)$ be the difference between the geometric centers of $\alpha.\sigma_i$ and $\beta.\sigma_i$ (see figure 5.6):

$$d_i(\alpha,\beta) = |C(\alpha.\sigma_i) - C(\beta.\sigma_i)| \tag{5.9}$$

In the clustering algorithm presented in [35] the following distance δ is chosen between two HRs α and β:

$$\delta(\alpha,\beta) = d_0(\alpha,\beta) + d_1(\alpha,\beta) \tag{5.10}$$

Using this distance we noticed that globally HRs at a high-level in the cluster tree diverge, i.e. $d_0(\alpha,\beta) \ll d_1(\alpha,\beta)$, or vice versa (see figure 5.7 left). Diverging HRs are unwanted to find bands such as in the left part of figure 3.4, as we will see in chapter 6.

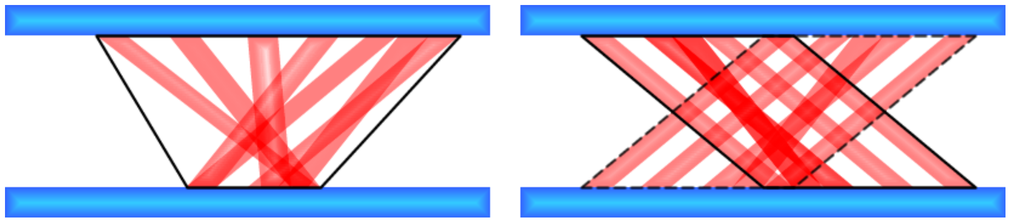


Figure 5.7. Left: a high-level diverging HR indicated with a fat line. Right: two colinear high-level HRs indicated with a fat line.

We have defined an alternative[1] for δ inspired by [37]:

$$\delta(\alpha,\beta) = 2\cdot\mathbf{max}(d_0(\alpha,\beta), d_1(\alpha,\beta)) - \mathbf{min}(d_0(\alpha,\beta), d_1(\alpha,\beta)) \tag{5.11}$$

[37] searches for colinear clusters. We use this distance to find colinear high-level HRs. A set of HRs is colinear if they, not necessarily in the same order on σ_i, together express a large coherent high-level HR (see figure 5.7 right). In a biological sense, a colinear HR suggests a recent duplication: A large subsequence is duplicated, but parts (low-level HRs) are not in the same order, due to rearrangements. By penalizing a large difference in $d_i(\alpha,\beta)$, high-level HRs tend to diverge less in the cluster tree.

In the end, the choice for δ is the crucial part of the clustering, with radical effects for the clustering and hence the visualization. Searching for a suitable δ is a discipline on its own, and further research is out of the scope of this project.

5.1.5 Further alternatives

So far, a complete cluster tree is defined. This tree will serve as input for the visualization as defined in 5.2. This subsection discusses some alternative design decisions for the cluster tree.

[1]Actually, δ is not a distance, since the triangle equality does not hold.

5.1.5.1 Tree height reduction

Because the cluster tree $\mathcal{T}$ is a binary tree, it is fairly high. Around the root and leafs, the binary tree structure has little intrinsic meaning, since these subtrees only exist because nodes can have only two children. Furthermore, these subtrees are not distinguishable from meaningful subtrees.

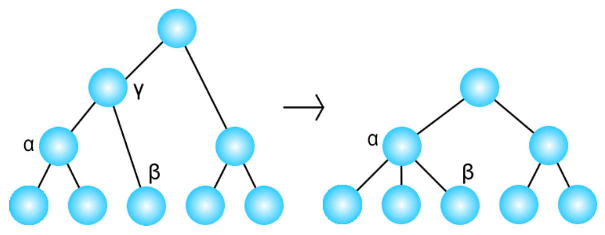


Figure 5.8. Left: a binary cluster tree. Right: a cluster tree with height optimization at the leafs.

No longer a new tree node is introduced when aggregating a parent of a leaf node and a leaf node. This results in non-binary nodes near the leafs of the cluster tree:

```
   i. $V := U$.
  ii. Copy all elements of $V$ to the set of roots of partial trees $R$: $R := V$
 iii. Let $\alpha, \beta \in R$ be a pair with the smallest distance $\Delta(\alpha, \beta)$, such that $H_{\mathcal{T}}(\alpha) \geq H_{\mathcal{T}}(\beta)$.
  iv. **if** $(H_{\mathcal{T}}(\alpha) = 1 \wedge H_{\mathcal{T}}(\beta) = 0)$
   v.     Add $\beta$ to $\alpha$: $\alpha := \alpha \bowtie \beta$; $E_{\mathcal{T}} := E_{\mathcal{T}} \cup \{(\alpha, \beta)\}$
  vi.     $\beta$ is no longer a root of partial tree: $R := R \setminus \{\beta\}$;
 vii. **else**
viii.     Aggregate : $\gamma := \alpha \bowtie \beta$
  ix.     Add $\gamma$ to $\mathcal{T}$: $V := V \cup \{\gamma\}$; $E_{\mathcal{T}} := E_{\mathcal{T}} \cup \{(\gamma, \alpha), (\gamma, \beta)\}$.
   x.     Remove $\alpha$ and $\beta$ from the roots of partial trees: $R := R \setminus \{\alpha, \beta\}$
  xi.     Add $\gamma$ to the roots of partial trees: $R := R \cup \{\gamma\}$
 xii. **fi**
xiii. Repeat steps iii...xiii until $R$ contains only one HR.
```

We noticed that, the height of the improved cluster tree is not reduced dramatically. By approximation, only a constant number of levels is removed.

5.1.5.2 Group distance Δ

[35] does not document why we should use Δ as defined in (5.4). The function Δ is a kind of average group linkage distance [38], which is more often used in clustering methods. But, the distance is higher for larger groups than the group average, because the divisor contains an addition instead of a multiplication. We think, an addition is preferred, because small clusters are enforced to group first, resulting in a more or less balanced binary tree. This is a nice property for visualization, because screen space can be filled more efficiently. However, we have calculated a data set with a multiplication, resulting in no remarkable differences.

5.1.5.3 Score aggregation

In (5.8) on page 42, a score is defined for HR γ, which is aggregated from α and β. We observed that this score strongly decreased, when traversing to the root. This is unwanted, because the score is an indication for the importance, hence aggregated HR become quickly unimportant. Therefore, we have introduced an alternative score.

$$\begin{aligned} S_{leaf}(\alpha) &= (\Sigma\beta : \beta \in L_{\mathcal{T}}(\alpha) : \beta.\text{Score} \cdot W_{HR}(\beta)) && (5.12)\\ W_{\mathcal{T}}(\alpha) &= (\Sigma\beta : \beta \in L_{\mathcal{T}}(\alpha) : \cdot W_{HR}(\beta)) && (5.13)\\ \gamma.\text{Score} &= \frac{S_{leaf}(\alpha) + S_{leaf}(\beta)}{W_{\mathcal{T}}(\gamma)} \cdot 100\% && (5.14) \end{aligned}$$

The score in (5.14) does not take into account what happens between initial HRs, because we do not know what is between them. It might contain matching bases, therefore (5.8) may underestimate the aggregated score.
Note that, if any pair of sequences $\alpha.\sigma_i$ and $\beta.\sigma_i$ overlap, matching bases may be counted double, in both (5.8) and (5.14). So, we truncate all scores exceeding 100%

5.1.6 Concluding remarks

The running time of the cluster algorithm is at least cubic in N: in each iteration, the number of elements in R decreases with one, using one computation of a minimal pair. Searching for a minimal pair for Δ takes at least quadratic time, since we have not considered any optimization. This naive implementation results in an optimal solution with respect to minimizing the distance function Δ for each pair, since in each step the minimal distanced pair of partial cluster trees is chosen. Jin et al. [35] try to improve the running time, by decreasing the search space for a minimal pair of partial cluster trees, but they do not indicate how and why it should be correct. Therefore, we have used the non-optimized solution, which is not a problem, because the clustering is pre-computed.

5.2 Visualization

This section applies and extends the HEB visualization method [34] to pairwise sequence alignments. The HEB visualization method displays an adjacency graph $\mathcal{G} = (V, E_A)$ using a tree $\mathcal{T} = (V, E_T)$. HEB is developed in the context of software metrics, where $\mathcal{G}$ is a method call-graph and $\mathcal{T}$ is the hierarchical structure of methods, classes, packages, etc. This results in the right part of figure 5.9, where a caller method at the green side of the edge and a callee method at the red side of the edge are connected. By guiding the edge of the call-graph over the hierarchical structure, bundles appear. The fat red bundle at the right bottom of figure 5.9 indicates that 'Unit 16' is a software library, because these methods are called often from various places in the code.

The general overview of HEB applied to pairwise alignments is introduced in subsection 5.2.1. Because the representation differs from [34], we reconstruct the geometry and apply the bundling

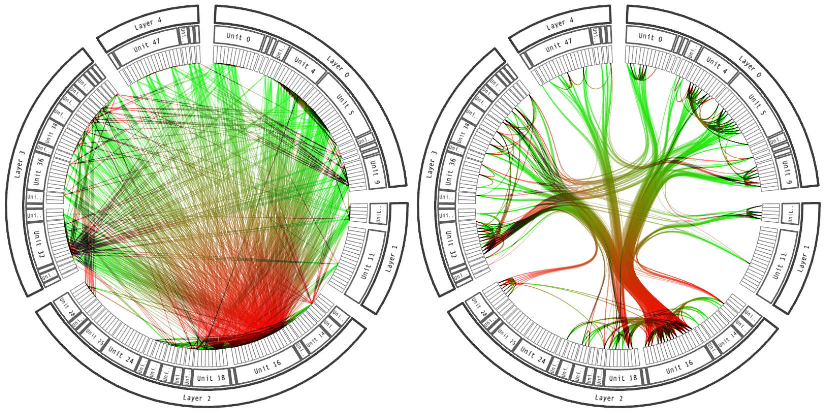


Figure 5.9. A call-graph visualized using HEB [34]. Left: a node-to-node visualization. Right: node-to-node visualization using hierarchical relations.

principle in subsection 5.2.2. Curves are transformed in to bands in subsection 5.2.3. Ambiguity problems are solved in subsection 5.2.4. Types of curves are considered in subsection 5.2.5.
In this section we denote user-specified visualization parameters with bold variables, i.e. **b**, **q**, etcetera. Table 5.2 contains default values of these parameters, which are the assumed values if a parameter is omitted.

5.2.1 Application

A pairwise alignment of two sequences is created by adapting the HEB method (see figure 5.10). For each HR $\alpha \in L_{\mathcal{T}}(\omega)$ of cluster tree $\mathcal{T} = (V, E_{\mathcal{T}})$, rooted at $\omega \in V$, subsequences $\alpha.\sigma_i$ on the sequences S_i are connected by a curve. The shape of the curve is guided by the cluster tree. Each node in the cluster tree denotes a group of HRs with minimal distance Δ. Hence, the curves of HR α and β with a closest common ancestor γ are guided by the same nodes between γ and ω. This implies that, the closer the HRs are on the sequence - in terms of δ - the longer the shared bundle is of HRs.

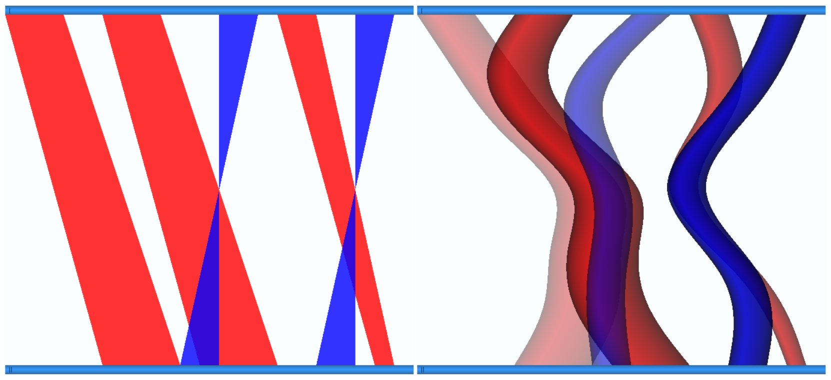


Figure 5.10. A pairwise alignment visualized. Left: a σ_0-to-σ_1 straight line visualization of HRs. Right: a σ_0-to-σ_1 visualization of HRs using a cluster tree and HEB.

5.2.2 Geometry

Before the bundling principle can be applied, each non-root cluster tree node needs two positions between the sequences (see figure 5.11). For cluster tree $\mathcal{T} = (V, E_{\mathcal{T}})$ rooted at $\omega \in V$, all HRs $\alpha \in V \setminus \{\omega\}$ are positioned using two points $\alpha.\mathrm{p}_0$ and $\alpha.\mathrm{p}_1$. The root ω is positioned at only one point $\omega.\mathrm{p}$.

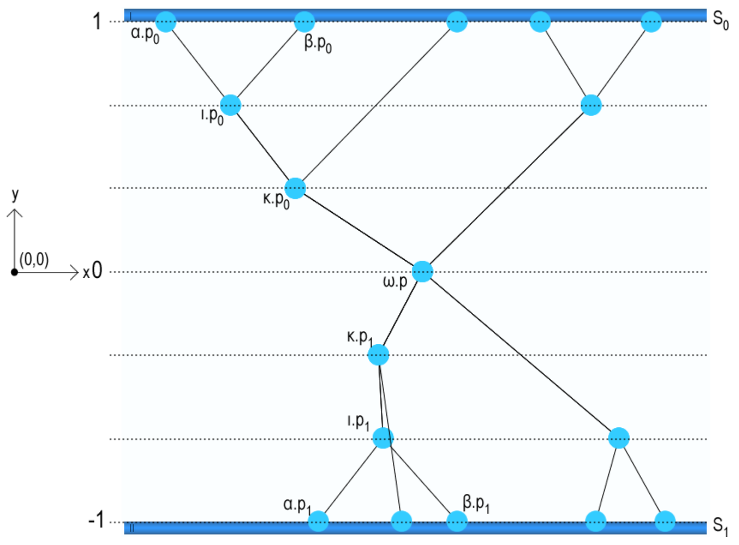


Figure 5.11. Layout of the cluster tree.

Let $B(b)$ be the function which returns the x-coordinate of b-th base. Let 1 and -1 be the y-coordinates of sequence S_i (see figure 5.11). We define the positions for HRs $\alpha, \omega \in V$ by:

$$\omega.\mathrm{p}.x = \tfrac{1}{2} \cdot B(C(\omega.\sigma_0)) + \tfrac{1}{2} \cdot B(C(\omega.\sigma_1)) \quad (5.15)$$

$$\omega.\mathrm{p}.y = 0 \quad (5.16)$$

$$\alpha.\mathrm{p}_i.x = B(C(\alpha.\sigma_i)) \quad (5.17)$$

$$\alpha.\mathrm{p}_i.y = (-1)^i \cdot (1 - \tfrac{H_{\mathcal{T}}(\alpha)}{H_{\mathcal{T}}(\omega)}) \quad (5.18)$$

Let $\alpha, \beta_0, \ldots, \beta_{n-1}, \omega$ be the rootpath of HR α. For every leaf HR α, we define a polyline $P = \alpha.\mathrm{p}_0, \ldots, \beta_{n-1}.\mathrm{p}_0, \omega.\mathrm{p}, \beta_{n-1}.\mathrm{p}_1, \ldots, \alpha.\mathrm{p}_1$. Bundles appear if P is used as the control graph of a curve (see figure 5.12 left). The color of the curve is related to the Type of the represented HR.

5.2.3 Width

Because wide HRs are more important than small ones, it is not fair to represent each HR with a single curve. Optionally, the user can render bands as wide as the represented HR to emphasize large HRs (see figure 5.12 right). The width at each point of the curve is the linear interpolation between the width of the start and end point.
Inversions not displayed as an hourglass, because then this type of band takes less screen space than matches, so matches are then unnecessarily more emphasized than inversions.

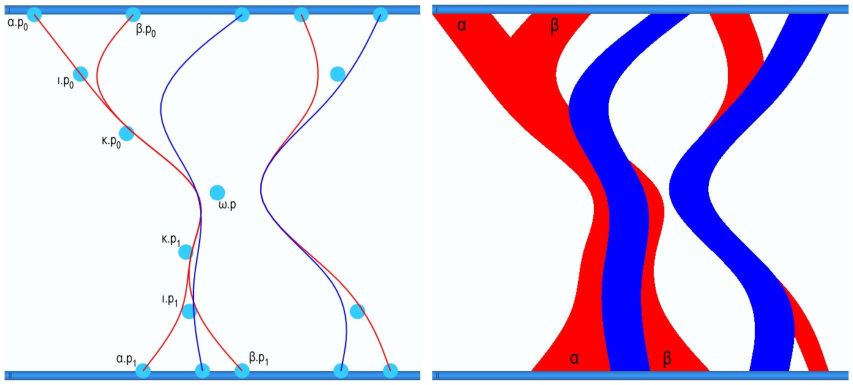


Figure 5.12. Left: bundling curves. Right: bundling bands.

5.2.4 Resolving ambiguity

In the right part of figure 5.12, it is impossible to determine whether HRs α and β are crossing or disjunct, because they share a bundle. In this subsection we introduce techniques to resolve this ambiguity.

5.2.4.1 Transparency and texture

Not all HRs are equally important. The score of an HR is a good indication of the importance. Optionally, the user can render the bands with the transparency proportional to the score, i.e. the higher the score, the opaquer the band (see figure 5.13 left). Hence, important HRs catch more the eye than unimportant ones. If the score of α and β differ, we are able to determine that α does not cross β.

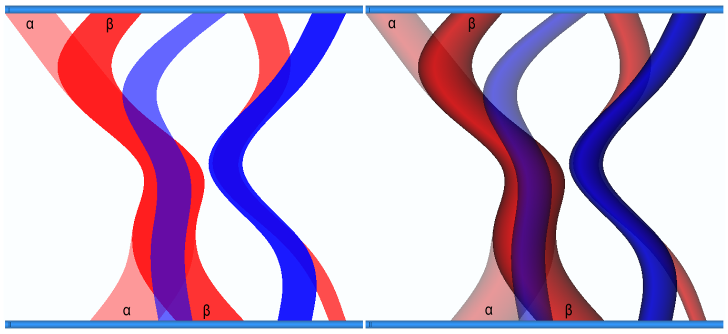


Figure 5.13. Left: transparency. Right: cushion texture.

Unfortunately, the score of two occluding bands may be the same, hence transparency will not resolve ambiguity. Therefore, the user can decide to render a band with a cushion texture [39] (see figure 5.13 right). This resolves ambiguity of HRs with equal Type and Score, because the borders are colored differently from the band itself.

5.2.4.2 Bundling strength

Holten [34] resolves ambiguity by means of a bundling strength $\mathbf{b} \in [0, 1]$. By altering $\mathbf{b}$ firm bundles are weakened. The bundling strength adapts the control graph P as defined in subsection 5.2.2 consisting of M points:

$$Q_i = P_0 + \frac{i}{M-1}(P_{M-1} - P_0) \tag{5.19}$$

$$P'_i = \mathbf{b} \cdot P_i + (1 - \mathbf{b}) \cdot Q_i \tag{5.20}$$

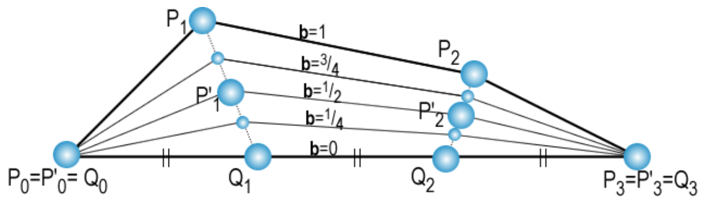


Figure 5.14. The bundling principle for $P_0 \ldots P_3$ at bundling strength $\mathbf{b} = \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$.

In (5.19), the straight line (P_0, P_{M-1}) is divided in $M - 1$ equal length line segments to obtain a polyline Q with M points (see figure 5.14). Each point P_i matches point Q_i in (5.20), resulting in point P'_i after linear interpolation with a factor $\mathbf{b}$. A typical value for $\mathbf{b}$ is between $\frac{3}{4}$ and $\frac{8}{9}$, resulting in coherent bundles (see figure 5.15). Ambiguity is solved, because bands do not fully occlude anymore.

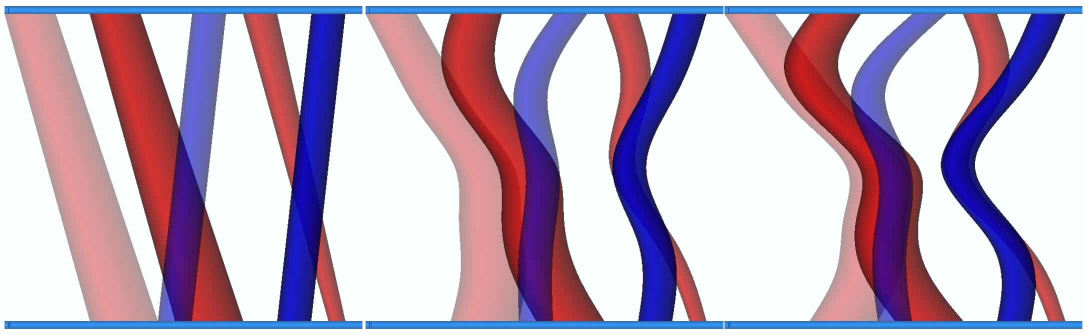


Figure 5.15. Weakened bundling bands $\mathbf{b} \in \{0, \frac{1}{2}, \frac{6}{7}\}$.

5.2.5 Variations and extensions

In subsection 5.2.2 we have not mentioned what type of curve is used. The type of curve is important for the effect of the bundling principle. Holten observes [34] that uniform cubic B-splines (degree=3) result in the most coherent and distinct bundles. We have considered quadratic B-splines (degree=2) too, because these are cheaper to compute. We have measured a speedup of up to 20% for large amounts of curves (> 30.000). In figure 5.16, the geometric

difference between the types is noticeable, but does not influence the bundling effect.
Because a curve defined by a uniform B-spline does not end in both end control points, we add $d-1$ extra equal end points to both sides of the control graph, where d is the degree of the spline.

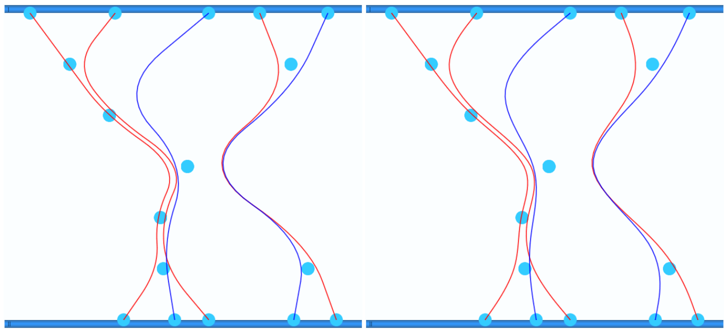


Figure 5.16. Quadratic versus cubic B-splines.

5.3 Real world data

So far, a clustering algorithm and a visualization is defined. Both are applied to a toy data set of five HRs in one pairwise alignment. The size of real world data sets is much larger. Data sets of thousands of HRs are common for a single pairwise alignment.

Note that we are creating a *multiple* sequence alignment visualization. Our pairwise alignments easily expand to multiple sequences by joining common sequences: A pairwise alignment $S_0 \asymp S_1$ of sequence S_0 and S_1 can be joined with $S_1 \asymp S_2$, resulting in MSAView $S_0 \asymp S_1 \asymp S_2$.

In figure 5.17, the clustering algorithm and visualization is applied to the whole TIGR *Arabidopsis thaliana* data set [22], which we have seen before in section 3.3 on page 21. The visualization has a couple of disadvantages:

- All curves visit the root of the cluster tree.
- Around the leafs, the curves are almost straight and do not express much information.
- The scene is still chaotical.

But, we already notice an advantage of the bundling: the pattern of the curves differs between each pairwise alignment, which is an indication that these bundles have something to tell.

From figure 5.17, we conclude that too much HRs are displayed. In this section we define two methods to reduced the number of HRs. In subsection 5.3.1 a method is described which hides details without loosing the big picture. Traditional filtering is defined in subsection 5.3.2.

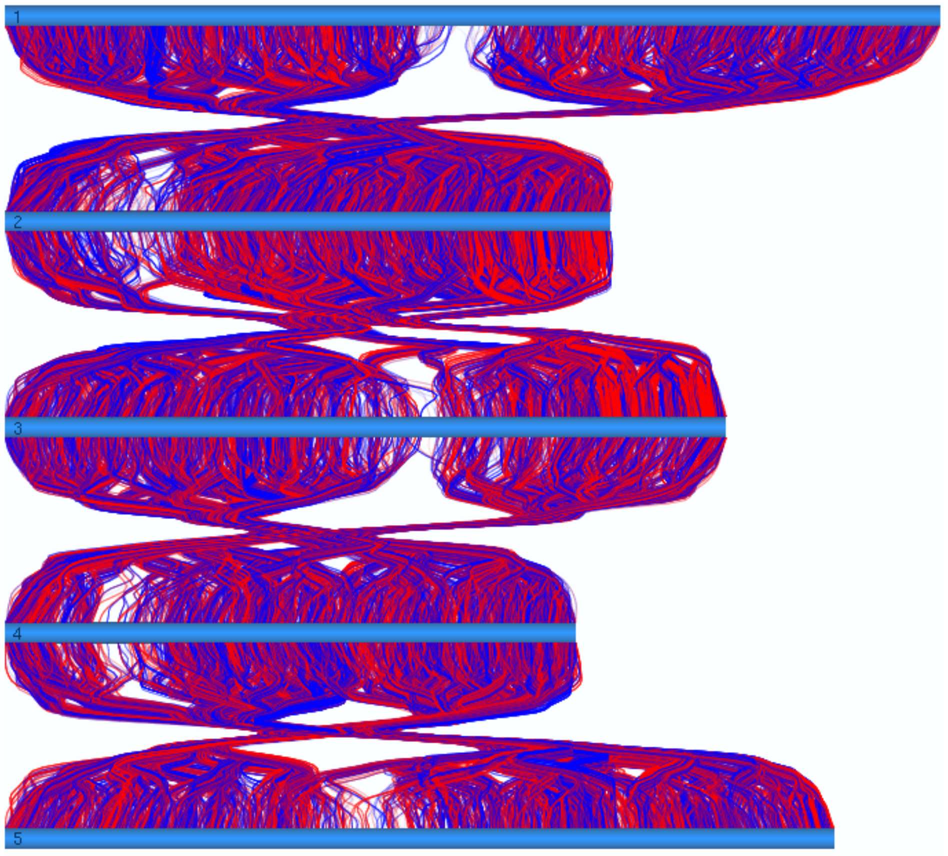


Figure 5.17. Five sequences of *Arabidopsis thaliana* with their homologous regions in the MSAView.

5.3.1 Level of detail

Nodes near the root tend to be unknown-typed, and hence are not very meaningful. Nodes near leafs are very small and can be represented by aggregated ones, which allows recognition of patterns more easily. We can use this to simplify the scene. A Level Of Detail (LOD) is defined by tree cuts at the root and around the leafs, which enables to emphasize the middle of the tree (see figure 5.18 left). To define tree cuts, we consider a value $\alpha.E$, which should be descending[2] for each path from root to a leaf. The tree cut at root ω is defined by a threshold value b_{max} and around the leafs by b_{min}.

The tree cut at top in (5.21) is just removing nodes from the cluster tree with values higher than b_{max}. The tree cut around the leafs is more subtle. In the right of figure 5.18, a tree is displayed left from the arrow. For $b_{min} = 2$, an ordinary tree cut would display the node with value 2, since only leaf HRs are rendered. But that means that we loose information, namely the left side of the HR with value 1, therefore we choose to make the HR with value 3 a leaf. In general, this means that the middle HRs in (5.22) are from the root node characterized in two ways: either all children are larger than b_{min} or it is a leaf, which implies that the HR is larger than b_{min} and at least one child is smaller than b_{min}.

[2]or ascending, with comparison operators flipped

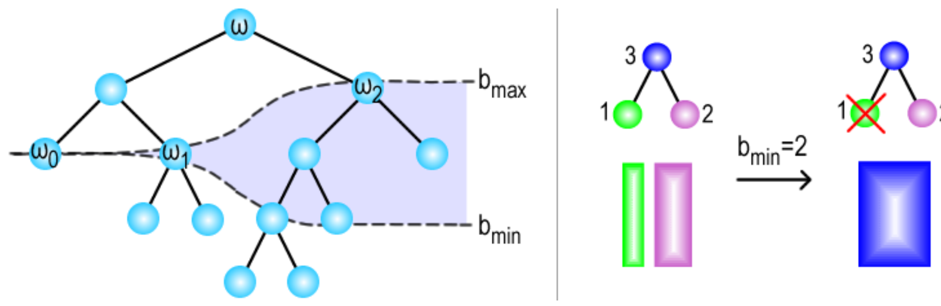


Figure 5.18. Left: two tree cuts. Right: level of detail, the numbers at the cluster tree represent the value E of the HRs. Below the visible HRs are displayed.

This results in a set of disjunct cluster subtrees $\mathcal{T}' = (V', E'_T)$:

$$W_t = \{\alpha \in V \mid \alpha.\mathrm{E} > b_{max}\} \tag{5.21}$$

$$W_m(\alpha) = \begin{cases} \{\alpha\} \cup (\cup\beta : (\alpha,\beta) \in E_T : W_m(\beta)), & \text{if } AL(\alpha); \\ \{\alpha\}, & \text{if } \neg AL(\alpha) \wedge \alpha.\mathrm{E} \geq b_{min}. \\ \emptyset, & \text{otherwise.} \end{cases} \tag{5.22}$$

$$AL(\alpha) = (\forall\beta : (\alpha,\beta) \in E_T : \beta.\mathrm{E} \geq b_{min}) \tag{5.23}$$

$$V' = W_m(\omega) \setminus W_t \tag{5.24}$$

$$E'_T = \{(\alpha,\beta) \in E_T \mid \alpha \in V' \wedge \beta \in V'\} \tag{5.25}$$

We have considered two instantiations for b_{min}, b_{max} and α.E as defined in table 5.1.

Parameter	*Depth-oriented LOD*	*Width-oriented LOD*
α.E	RD(α)	$W_{HR}(\alpha)$
b_{min}	$\mathbf{l}_{\text{end}}$	$\mathbf{w}_{\text{TCmin}}$
b_{max}	$\mathbf{l}_{\text{start}}$	$\mathbf{w}_{\text{TCmax}}$

Table 5.1. Instantiations of levels of detail.

In the first alternative - called Depth-oriented LOD -, RD(α) is the root distance of α. Hence, α.E is an ascending value, displaying subtrees between depth $\mathbf{l}_{\text{start}}$ and $\mathbf{l}_{\text{end}}$. In figure 5.19, two examples are given with a depth-oriented LOD. The maximal root distance of these cluster trees is approximately 25. At the left, HRs are displayed using score-related transparency for 9 visible levels of the tree, such that only a small number of HRs are visible. At the right, the score is too low to use it as transparency, hence a constant transparency is used, for 5 visible levels. Depth-oriented LOD reduces the cluster tree in an effective but a not so sophisticated way, because the root distance has no biological meaning.

The second alternative - called Width-oriented LOD - considers the width of an HR α, bounded by $\mathbf{w}_{\text{TCmin}}$ and $\mathbf{w}_{\text{TCmax}}$. Width-oriented LOD reduces the tree in a more sophisticated way, because the width of an HR indicates (a kind of) importance. In the left of figure 5.20, both the advantage and disadvantage become clear: because of $\mathbf{w}_{\text{TCmin}} = 5$Kbp (kilo base pair), small HRs ($\pm$1Kbp) are represented by aggregated HRs. Only a few matches and inversions survive aggregation, which is nice, because these large HR are really similar and hence interesting to investigate. A disadvantage is the vertical center of the pairwise alignments, because these are

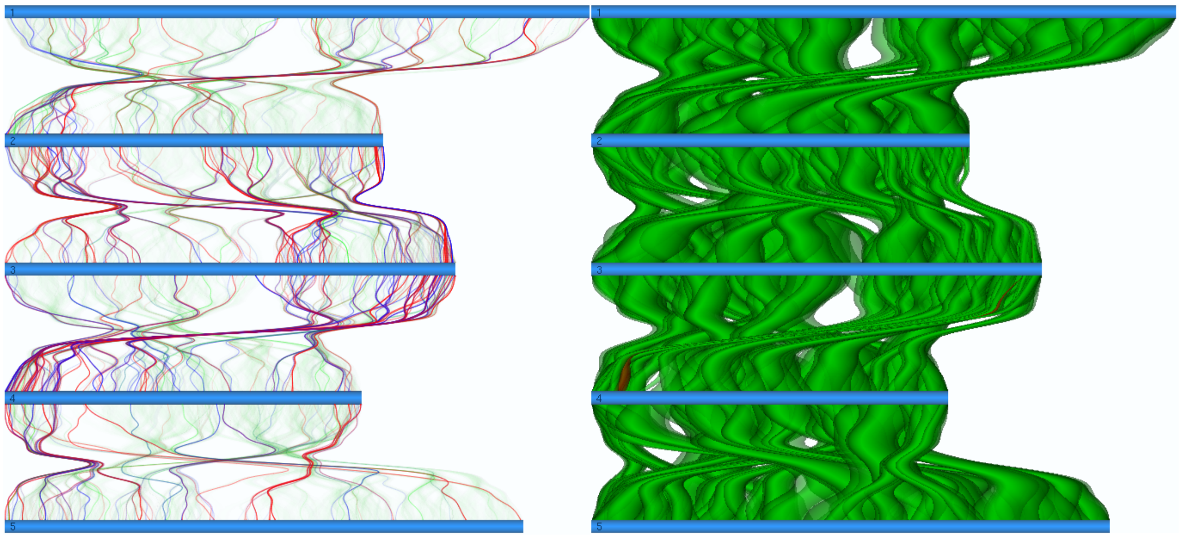


Figure 5.19. Example of depth-oriented level of detail in the TIGR *Arabidopsis thaliana* data set. Left: $\mathsf{l}_{\mathsf{start}} = 2$ and $\mathsf{l}_{\mathsf{end}} = 11$ with score-related transparency. Right: $\mathsf{l}_{\mathsf{start}} = 3$ and $\mathsf{l}_{\mathsf{end}} = 8$ with a constant transparency of 56%.

more cluttered than in the left of figure 5.19. If the right sides of figure 5.19 and 5.20 are compared, we notice that figure 5.20 contains only wide HRs. So, if the number of small HRs is too high, these can be represented by wider HRs using width-oriented LOD.

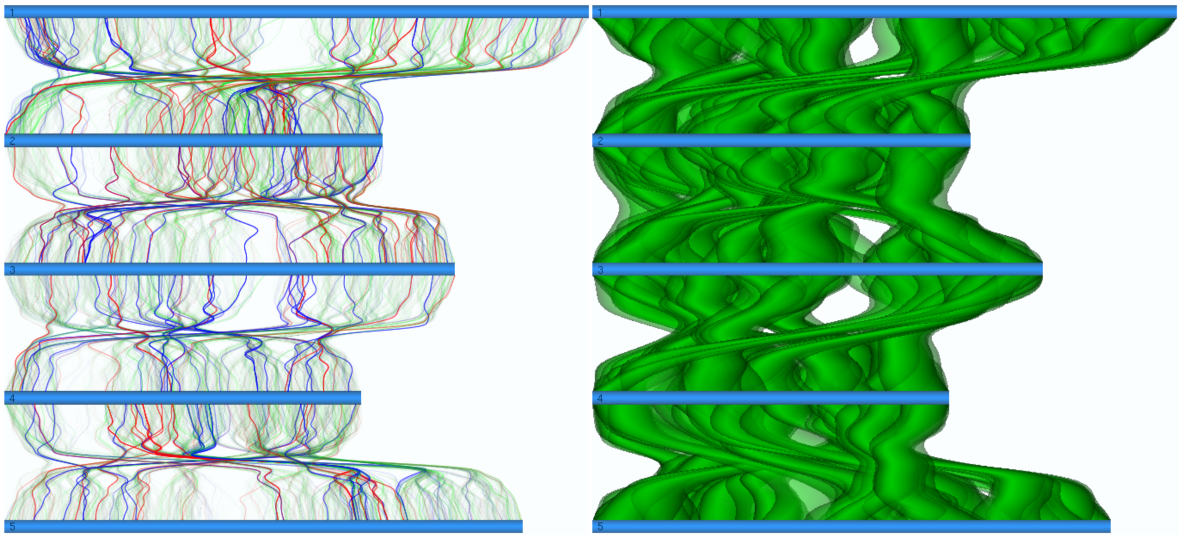


Figure 5.20. Example of width-oriented level of detail in the TIGR *Arabidopsis thaliana* data set. Left: $\mathbf{w}_{\mathsf{TCmin}} = 5\text{Kbp}$ and $\mathbf{w}_{\mathsf{TCmax}} = 9\text{Mbp}$ with score-related transparency. Right: $\mathbf{w}_{\mathsf{TCmin}} = 2\text{Mbp}$ and $\mathbf{w}_{\mathsf{TCmax}} = 13\text{Mbp}$ with a constant transparency of 56%.

5.3.2 Leaf filtering

By means of filtering, interesting elements - specified by a set of criteria - are emphasized. This filtering method is only applied to the leafs. If a leaf does not meet all criteria, then the complete curve is hidden.
For cluster tree $\mathcal{T} = (V, E_T)$, the following conjunction of criteria for leaf HRs $\alpha \in V$ is implemented, with user-specified parameters printed in boldface:

- The score and width of subsequences of an HR are bounded.

$$\mathbf{s}_{\min} \leq \alpha.\text{Score} \leq \mathbf{s}_{\max} \tag{5.26}$$

$$\mathbf{w}_{\min} \leq W(\alpha.\sigma_0) \leq \mathbf{w}_{\max} \vee \mathbf{w}_{\min} \leq W(\alpha.\sigma_1) \leq \mathbf{w}_{\max} \tag{5.27}$$

- Display only HRs in a given set of types **T**.

$$\alpha.\text{Type} \in \mathbf{T} \tag{5.28}$$

- Display only HRs which have a sequence in a given set of sequences **S**.

$$\alpha.\text{S}_0 \in \mathbf{S} \vee \alpha.\text{S}_1 \in \mathbf{S} \tag{5.29}$$

- Display only HRs which have a subsequence which is fully contained in at least one interval of the set of intervals **I**. Intervals in **I** are extended with $\mathbf{f}_\text{t}$ extra bases, which is known as a flanking sequence.

$$M(\alpha.\sigma_0, \mathbf{I}) \vee M(\alpha.\sigma_1, \mathbf{I}) \tag{5.30}$$

$$M(\text{S}[l, r], K) = (\bigvee [p, q] : [p, q] \in K : [l, r] \subseteq [p - \mathbf{f}_\text{t}, q + \mathbf{f}_\text{t}]) \tag{5.31}$$

In figure 5.21, leaf filtering is applied to the TIGR *Arabidopsis thaliana* data set. The intervals in **I** are indicated with a blue rectangle between the sequences, and the flanking sequences with a green rectangle. Leaf filtering allows fast reduction of HRs, by means of highly specialized criteria.

5.3.3 Composition

LOD and leaf filtering become interesting when put together. Typically, a level of detail is defined, unknown typed leafs are hidden and the width and score have an underbound to emphasize locations of strong homology. In figure 5.22, two examples are given. In the left, aggregated matches and inversions of at least 100Kbp are displayed in an interval. The score is at least 2%, but they are not much higher. But, these aggregated HRs still are a match or an inversion, indicating that these HRs are interesting to inspect. Note that these regions are approximately a hundred times larger than the given input HRs. In the right of figure 5.22, all HRs with a score larger than 5% are displayed.

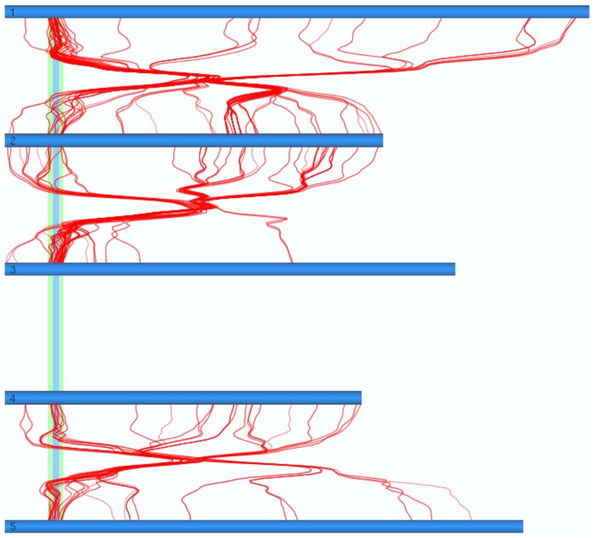


Figure 5.21. Example of leaf filtering in the TIGR *Arabidopsis thaliana* data set: $\mathsf{s}_{\mathsf{min}} = 80\%$, $\mathsf{w}_{\mathsf{min}} = 1.5\text{Kbp}$, $\mathbf{T} = \{\text{Match}\}$, $\mathbf{S} = \{1, 2, 5\}$, $\mathbf{I} = \{[2.5\text{Mbp}, 2.8\text{Mbp}]\}$ and $\mathbf{f}_t = 250\text{Kbp}$.

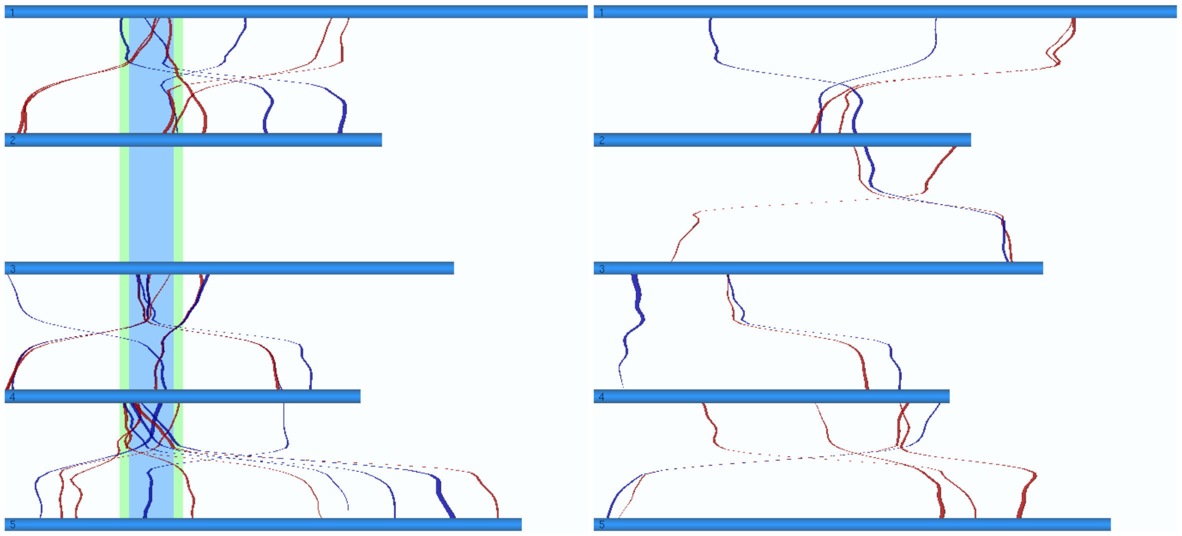


Figure 5.22. Examples of data reduction in the TIGR *Arabidopsis thaliana* data set with constant transparency. Left: $\mathsf{I}_{\mathsf{start}} = 5, \mathsf{w}_{\mathsf{TCmin}} = 100\text{Kbp}, \mathsf{s}_{\mathsf{min}} = 2\%, \mathbf{T} = \{\text{Match, Inversion}\}, \mathbf{S} = \{1, 4, 5\}, \mathbf{I} = \{[6.5\text{Mbp}, 8.8\text{Mbp}]\}$ and $\mathbf{f}_t = 500\text{Kbp}$. Right: $\mathsf{I}_{\mathsf{start}} = 5, \mathsf{w}_{\mathsf{TCmin}} = 100\text{Kbp}, \mathsf{s}_{\mathsf{min}} = 5\%, \mathbf{T} = \{\text{Match, Inversion}\}$ and $\mathbf{S} = \{1, \ldots, 5\}$.

5.3.4 Bundle coloring

So far, bundles are visible, but to see which subsequences are involved in a bundle we have to highlight them. In subsection 5.4.2 we define selection, but this can highlight only one bundle at the same moment. Inspired by [40], we color each distinct tree in $\mathcal{T}'$ with another color (see figure 5.23). This enables to see the distributions of subsequences of all bundles simultaneously.

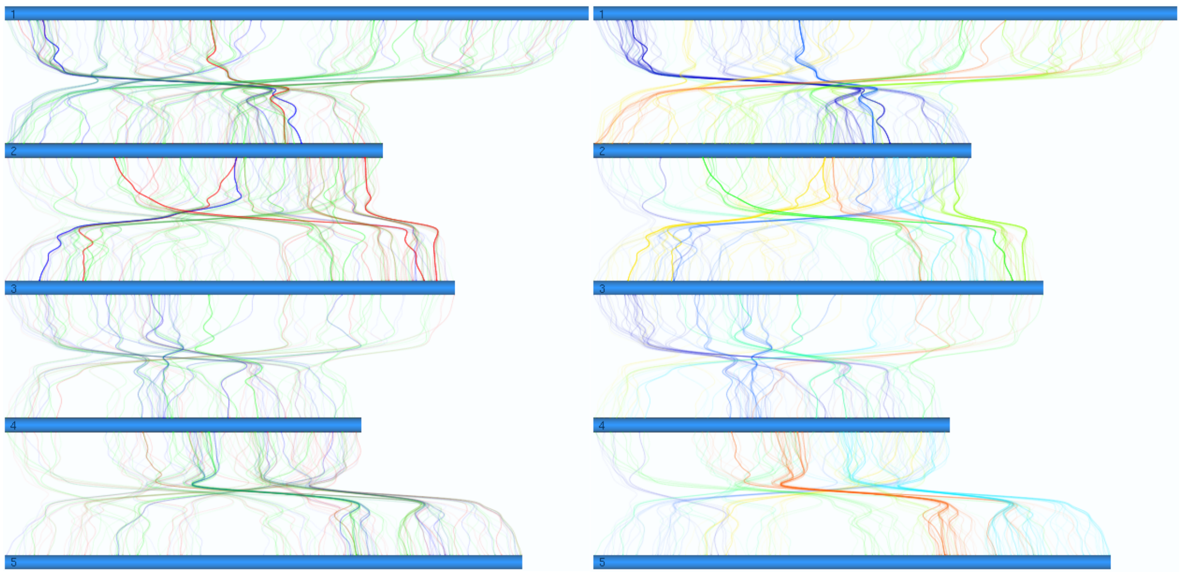


Figure 5.23. The TIGR *Arabidopsis thaliana* data set with score-related transparency, with $\mathbf{l}_{\mathrm{start}} = 3$, $\mathbf{w}_{\mathrm{TCmin}} = 10$Kbp and $\mathbf{s}_{\mathrm{min}} = 5\%$. Left: Typed coloring. Right: Bundle coloring.

5.4 Embedding in DNAVis2

The MSAView is integrated in the interactive genome browser DNAVis2 (see section 8.1). Before the visualization can be used (see figure 5.24), a computational comparison tool should generate HRs and a separate script calculates cluster trees, which are outputted in a special GFF3 files (see appendix E.3).

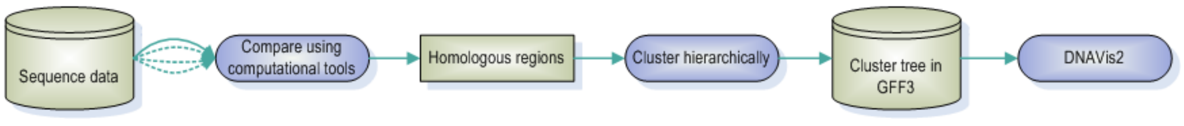


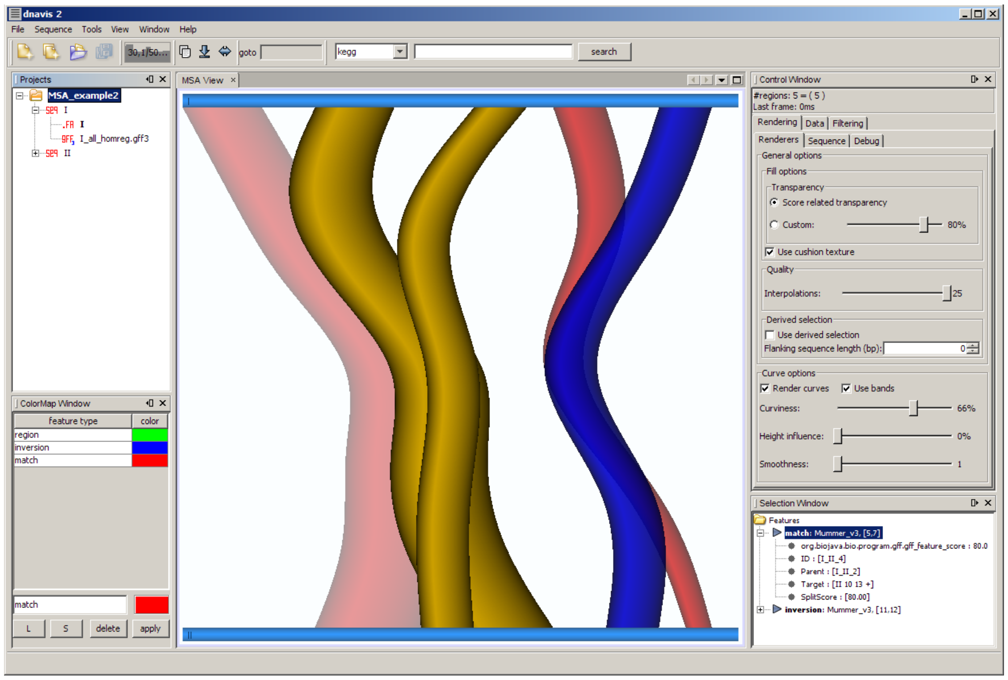
Figure 5.24. The data flow of the MSAView.

This section describes the integration of the MSAView in DNAVis2. The user interface is described in subsection 5.4.1, selection in subsection 5.4.2 and linking in subsection 5.4.3. Implementation details of the MSAView with respect to DNAVis2 are denoted in appendix D.

5.4.1 User interface

In figure 5.25, the user interface of DNAVis2 is displayed. At the left of the center, a project tree structure is listed in the 'Projects' window. A project is build of multiple sequences, where each sequence consist of sequence data (a FASTA file) and annotation data (GFF files), according the formats given in appendix E.

An MSAView can be created by clicking right on the project node and select 'open all in Multiple Sequence Alignment View', which loads all sequences of the project. Individual sequences can



MSA_example2
- SEQ I
  - .FA I
  - GFF3 I_all_homreg.gff3
- SEQ II

#regions: 5 = (5)
Last frame: 0ms

Rendering | Data | Filtering
Renderers | Sequence | Debug

General options
- Fill options
  - Transparency
    - [✓ Score related transparency]
    - Custom: 80%
  - [✓ Use cushion texture]
- Quality
  - Interpolations: 25
- Derived selection
  - [ ] Use derived selection
  - Flanking sequence length (bp): 0

Curve options
- [✓ Render curves] [✓ Use bands]
- Curviness: 66%
- Height influence: 0%
- Smoothness: 1

feature type	color
region	🟩
inversion	🟦
match	🟥

match 🟥

Features
- **match**: Mummer_v3, [5,7]
  - org.biojava.bio.program.gff.gff_feature_score : 80.0
  - ID : [I_II_4]
  - Parent : [I_II_2]
  - Target : [II 10 13 +]
  - SplitScore : [80.00]
- **inversion**: Mummer_v3, [11,12]

Figure 5.25. The graphical user interface of DNAVis2 with an MSAView.

be selected when holding the control key down. By clicking right and choosing 'open selected in Multiple Sequence Alignment View' the selected sequences are loaded.

At the right top a 'Control Window' is displayed, which accommodates all user-specified parameters, such as **b** and $\mathbf{s}_{\text{min}}$. At the left bottom, the 'ColorMap Window' allows to assign a color to each type[3] of HR.

5.4.2 Selection

By means of clicking and dragging, individual curves can be selected (see figure 5.26). When the control key is down, selections can be joined. Selected curves are highlighted and the annotations of the HRs are displayed in the 'Selection Window' (see figure 5.25). The beauty of selection becomes clear when selecting a bundle, because it allows straightforward selection of related HRs.

Because similarity between subsequences is a transitive relation and a subsequence can resemble its upper and lower neighbor, a selection of a set of curves can be extended to its neighbors until the uppermost and lowermost sequence is reached (see figure 5.27). Suppose we have an MSAView $S_{i-1} \asymp S_i \asymp S_{i+1}$. Let HR α be selected in $S_{i-1} \asymp S_i$ and β be a non-selected HR in

[3]Note that an unknown-typed region is an annotation of type 'region'

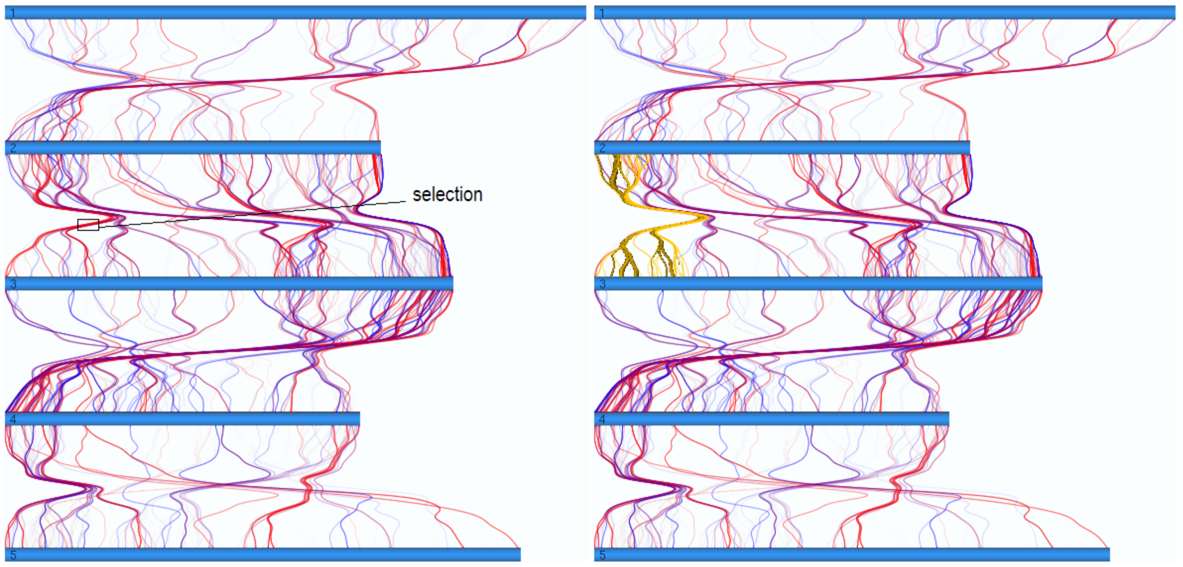


Figure 5.26. An example of a selection in the TIGR *Arabidopsis thaliana* data set, with $\mathbf{l}_{\mathrm{start}} = 2, \mathbf{l}_{\mathrm{end}} = 11$ and $\mathbf{T} = \{\text{Match, Inversion}\}$.

$\mathsf{S}_\mathsf{i} \asymp \mathsf{S}_{\mathsf{i}+1}$. HRs α and β resemble, if they meet the following:

$$[l_\beta, r_\beta] \subseteq [l_\alpha - \mathsf{fl}, r_\alpha + \mathsf{fl}] \tag{5.32}$$

where

$$\alpha.\sigma_i = \mathsf{S}_\mathsf{i}[l_\alpha, r_\alpha] \wedge \beta.\sigma_i = \mathsf{S}_\mathsf{i}[l_\beta, r_\beta] \wedge \mathsf{fl} = \mathbf{f}_\mathsf{d} \cdot W(\alpha.\sigma_i) \tag{5.33}$$

In (5.32), subsequence β is fully contained in α, that is extended with a flanking sequence, which is a fraction $\mathbf{f}_\mathsf{d}$ of $\alpha.\sigma_i$.

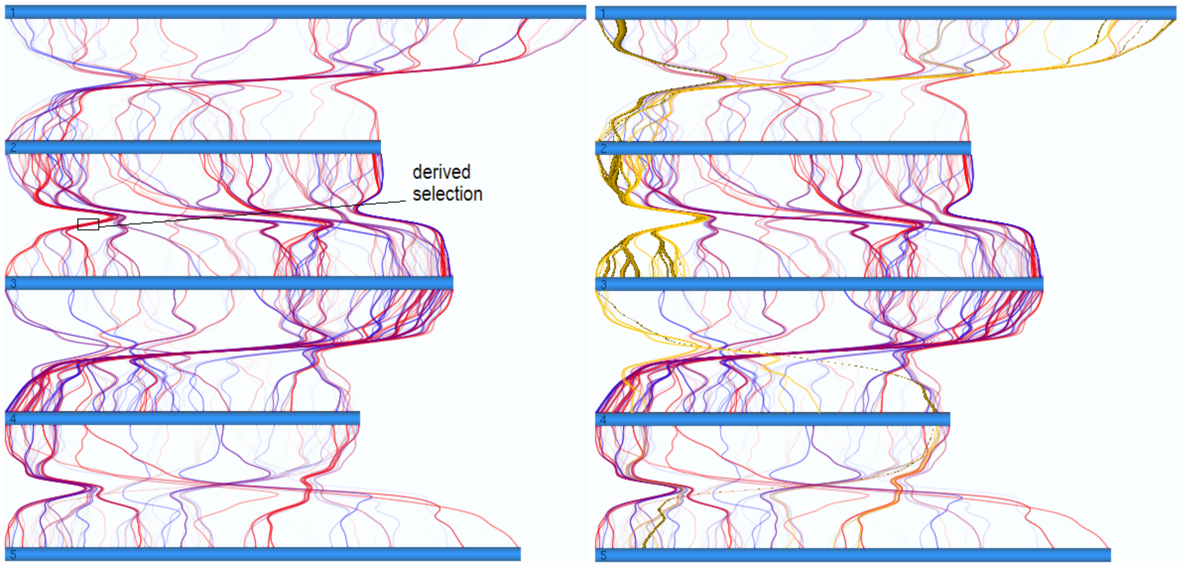


Figure 5.27. An example of a derived selection in the TIGR *Arabidopsis thaliana* data set, with $\mathbf{l}_{\mathrm{start}} = 2, \mathbf{l}_{\mathrm{end}} = 11$, $\mathbf{T} = \{\text{Match, Inversion}\}$ and $\mathbf{f}_\mathsf{d} = \frac{1}{2}$.

5.4.3 Linking

A biologist may want to focus on multiple parts of a sequence. Therefore, it is not sufficient to link one Linear view to a single sequence of the MSAView. Our solution allows the user to add intervals to a sequence, which can be linked (see figure 5.28).

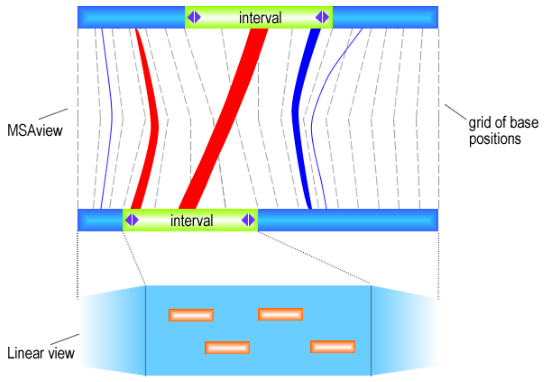


Figure 5.28. A sketch of an interval linked to a Linear view

An interval is a subsequence of one sequence. An interval matches a Linear view, because they both cut the sequence in three subsequences: a heading subsequence, the sequence itself and a tailing subsequence. An interval can be resized in base coordinates by means of zooming and in window coordinates by means of stretching. This implies that the grid of bases is divided in several densities of bases, i.e. function B of subsection 5.2.2 is piecewise linear. Stretching is an extra dimension of linking: Because the size of a sequence is fixed in an MSAView, intervals become too small when zoomed in, since it represents only a tiny part of the sequence, if a regular grid is chosen. In a Linear view it is common to zoom in up to a couple of hundred bases, which corresponds with less than one pixel of the sequence of an MSAView, with a regular grid. A single pixel is not enough to emphasize HRs in an interval, therefore the extra dimension in window coordinates is added.
The piecewise linear densities of two sequences are connected to each other by linearly interpolate the densities of the sequence to a regular grid in the center of the pairwise alignment, i.e. $y = 0$. This results in curves which create space around intervals to emphasize curves in an interval.
As a final touch, bases in the interval can be drawn not equally wide. For example, in figure 5.29, a lens [41] can be used, which emphasizes bases in the center of an interval.

5.5 Summary

This chapter has introduced a new MSA visualization called MSAView. The visualization draws bundled curves to point out dense locations of HRs. Various techniques enable manipulation to browse through a data set. The visualization is highly flexible, as indicated by parameters and options in tables 5.2 and 5.3. In the next chapter, we will evaluate the MSAView by visualize real world data and compare it with knowledge about that data.

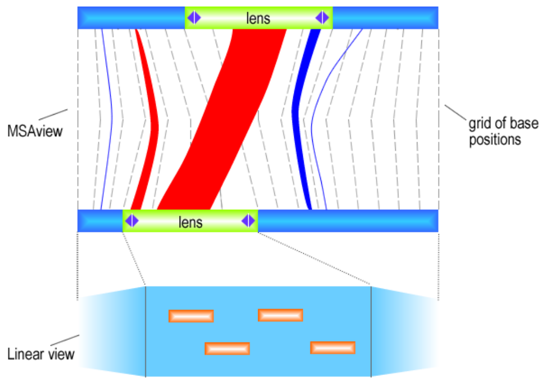


Figure 5.29. A sketch of a lens linked to a Linear view

Parameter	*Type*	*Default*	*Description*
$\mathbf{b}$	float $\in [0, 1]$	0	Bundling strength
$\mathbf{f}_\text{d}$	integer	0	Flanking sequence length of a derived selection in terms of a fraction of the interval
$\mathbf{f}_\text{t}$	integer	0	Flanking sequence length of an interval
$\mathbf{I}$	set of Interval	$\emptyset$	Intervals of interest
$\mathbf{l}_\text{start}$	integer	0	First visible height level of the cluster tree
$\mathbf{l}_\text{end}$	integer	∞	Last visible height level of the cluster tree
$\mathbf{q}$	integer	1	Curve quality: the number of approximation points of one segment of the curve
$\mathbf{S}$	set of Sequence	$\emptyset$	Sequence of an HR
$\mathbf{s}_\text{min}$	integer $\in [0, 100]$	0	Minimal score of an HR
$\mathbf{s}_\text{max}$	integer $\in [0, 100]$	100	Maximal score of an HR
$\mathbf{T}$	set of Type	*All types*	Type of an HR
$\mathbf{w}_\text{min}$	integer	0	Minimal width of an HR
$\mathbf{w}_\text{max}$	integer	∞	Maximal width of an HR
$\mathbf{w}_\text{TCmin}$	integer	0	Minimal width of an HR in a tree cut
$\mathbf{w}_\text{TCmax}$	integer	∞	Minimal width of an HR in a tree cut

Table 5.2. A list of user-influenceable visualization parameters of the MSAView

Other options:	
Curve type	Cubic or quadratic B-spline
Bands	Use bands as wide as the HR or just an one pixel curve
Color	The color curve of equally typed HRs
Transparency	The transparency of the curve related to the score or a constant transparency
Texture	Usage of cushion texture

Table 5.3. A list of options of the MSAView

Chapter 6

Evaluation

This chapter evaluates the MSAView as described in the previous chapter. First, the requirements of section 3.4 are verified in section 6.1. Experimental results are described in section 6.2.

6.1 Verification of requirements

The requirements as enumerated in appendix C describe necessary functionality of an MSA visualization.

Requirement I (Data size) The *Arabidopsis thaliana* data set at TIGR contains 258,916 HRs between 5 sequences. This data set can be explored easily with the default parameters.

Requirement II (Own data sets) Because MSAView uses FASTA and GFF3 data sets as input, anyone can use its own data set. To have useful input, the GFF3 file needs to be structured as defined in appendix E.3.

Requirement III (Bases) Via the selection window, exact positions of bases can be derived. Unfortunately, the individual bases are invisible in the visualization.

Requirement IV (Overview) The full sequence with its HRs to neighbor sequences is always visible. In section 6.2, we argue why the user is able to retrieve patterns at large scale.

Requirement V (Navigation) Unfiltered, complete genome data sets are a step too far for current desktop computers to render in real-time. But, with some filtering criteria the data set is quickly reduced such that it is as responsive as a Linear view. Due to a large number of parameters, the visualization is highly interactive.

Requirement VI (Link preservation) It is possible to link lenses on a sequence to Linear and Matrix views. However, the implementation of the linking concept as described in subsection 5.4.3 is still under development.

Requirement VII (Homologous regions) The MSAView displays HRs as curves.

Requirement VIII (Number of sequences) Because pairwise alignments of the MSAView can be joined, the number of sequences in the MSAView is unrestricted. Because of limited space on the screen, it is recommend to display up to seven sequences at once.

Requirement IX (Region of interest) The user can define locations for stretchable lenses, which define a region of interest. These regions are emphasized by drawing HRs within a lens larger, such as a real lens would do.

Requirement X (Filtering) The user can filter using the stated criteria in subsection 3.4.2.

Requirement XI (Emphasizing) In the MSAView four types of emphasizing are implemented: filtering, selection, bundle coloring and lens magnification.

Requirement XII (Uncluttering) By means of bundling, close related HR are drawn close to each other. If the overview is still cluttered, the user may reduce the size of the tree by means of a tree cut, level filtering and leaf filtering. All these techniques unclutter a straight line visualization (see figure 3.3 on page 22), resulting in bundles of similar HRs.

Requirement XIII (Abstraction) By defining a minimal width of an HR in a tree cut, aggregated HRs replace individual HRs.

Requirement XIV (Derivation) Using derived selection in one pairwise alignment, HRs can be propagated to all other visible pairwise alignments.

Summary

The validation of these requirements result in the following summary:

	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	XIII	XIV
MSAView	✓	✓	∽	✓	∽	∽	✓	✓	✓	✓	✓	✓	✓	✓

Although not all columns contain a checkmark, we fulfill more requirements than all other tools in chapter 4. Requirement III is partial fulfilled because the bases itself are invisible in the visualization. Requirement V is partial fulfilled because unfiltered data sets are still a step too far. We partially fulfill the sixth requirement because the implementation of linking is still under development. By concentrating on flexibility, a multiple sequence alignment is uncluttered and exploration is allowed at various levels of detail.

6.2 Results

Due to absence of a hierarchical structure over HRs, we needed to computed a cluster tree. So far, we have not answered the question: “What do I see?”. By comparing dotplots (see subsection 2.1.2.3) to an MSAView, we will answer this question.

6.2.1 Setup

We investigate the pairwise alignment of chromosome 1 and 2 and the pairwise alignment of chromosome 2 and 3 of the *Arabidopsis thaliana* genome. To do so, we have two sources [21, 22] providing information about homology in the *Arabidopsis thaliana* genome. The article in Nature [21] contains a stylish representation (see figure 6.1) of large scale homology. In figure 6.2 dotplots are depicted of a TIGR *Arabidopsis thaliana* data set [22]. Each dot represents one HR of at least 1000 bases. The score of the HR is not taken into account, so these dotplots may be deceiving. The large scale homology of figure 6.1 is expressed in the dotplots with diagonal lines of dots. From these dotplots, we conclude that figure 6.1 is really stylish, because chromosome 2 should be shorter than chromosome 3 and the green HR should be more at the front of chromosome 2. From now, we only compare our visualization to the dotplots, because figure 6.1 provides no extra information.

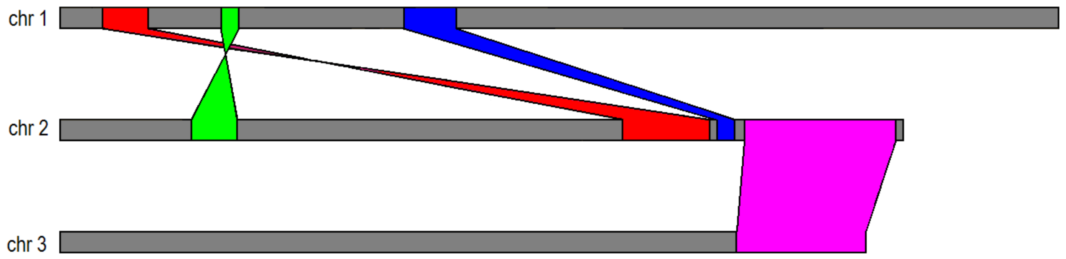


Figure 6.1. High-level homologous regions in the *Arabidopsis thaliana* genome found by [21]

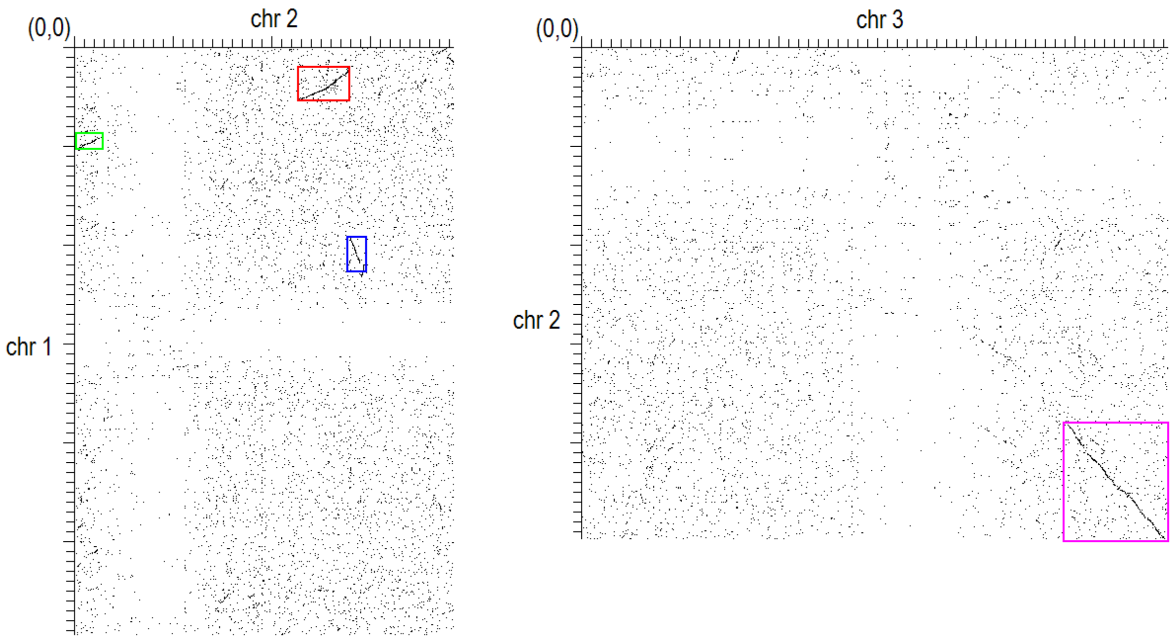


Figure 6.2. Dotplots of pairwise alignments of the *Arabidopsis thaliana* genome [22] generated with a tool provided with the data. Left: chromosome 1 and 2. Right: chromosome 2 and 3.

We put the same TIGR data set in the MSAView, once with distance (5.10) and once with distance (5.11) (see page 44). Figure 6.3 plots these distance functions in detail: The left graph displays (5.10) and the right displays (5.11). The darker the pixel the lower the value. The colored dots correspond with the examples of HRs below. The orange HRs are physically far from each other, the blue HRs are closer than the orange ones and the green are diverging ($d_0(\alpha, \beta) \ll d_1(\alpha, \beta)$). These examples indicate the application of the distances: With (5.10), blue and green have equal distance, which indicates that these HRs are physically close to each other. With (5.11) blue and orange have almost equal distance, which indicates colinear HRs (the diagonals in the dotplot).

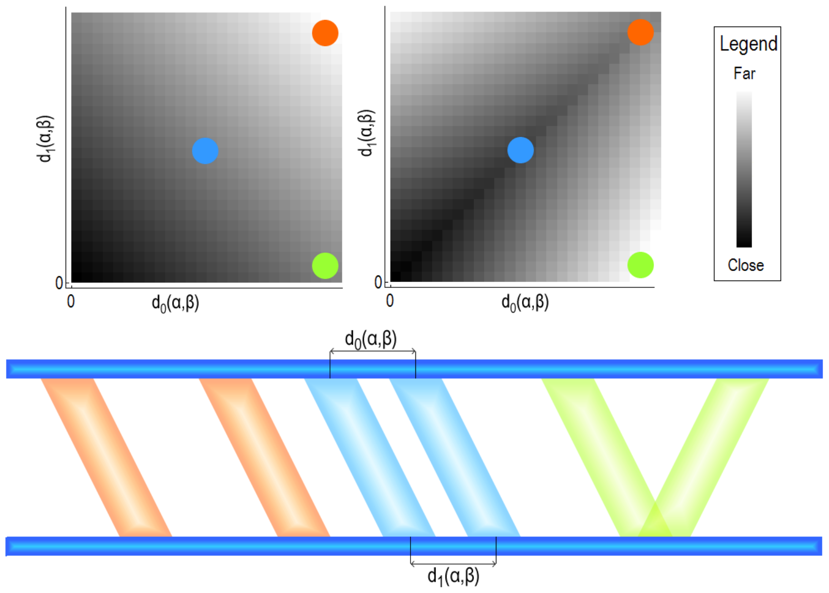


Figure 6.3. Distance δ between two HRs in the hierarchical clustering algorithm. Left: distance (5.10). Right: pseudo-distance (5.11).

To reveal what a bundle represents, we have rendered the TIGR data set with the following parameters: $\mathbf{W}_{\mathrm{TCmin}} = 10\mathrm{Kbp}$, $\mathbf{L}_{\mathrm{start}} = \{1, 2, 3\}$ and $\mathbf{b} = 0.9$. Furthermore, we used score-related transparency and bundle coloring. In figure 6.4 to 6.6, distance (5.10) is used, and in figure 6.7 to 6.9 distance (5.11) is used. We have highlighted in the dotplot the area which is covered when selecting the equally colored bundle.

6.2.2 Distance (5.10)

From comparisons of distance (5.10) in figures 6.4 to 6.6, we conclude that dense areas of HRs are bundled. Only the pink band of figure 6.1 is detected in figure 6.6. From a biological perspective the visualization provides insights in the distribution of duplicates. Where areas in the dotplot appear to be noise, the cluster tree divides these areas in non-equally sized areas, hence the dots likely not to be noise. These dense areas could provide a biologist insights in old evolutionary duplications, since these subsequences are likely to be more shuffled, but still close to each other.

6.2.3 Distance (5.11)

Distance (5.11) in figures 6.7...6.9 is developed especially for diagonals in dotplots. It finds the pink and red bands of figure 6.1. In figure 6.8 the bundling proves its strength, since the dark blue bundles in both pairwise alignments are more powerful than the others, even if more HRs are included in a cluster in the bottom one. This is not without a reason, because the areas are highly colinear. From a biological perspective, these colinear areas suggest recent evolutionary duplications, which are still almost in the same order.

6.2.4 Conclusion

Depending on the distance function, different possible relationships (for example evolutionary) can be visualized. The advantages of the MSAView with respect to the dotplot are a more intuitive metaphor to browse through the data and the possibility to extend the MSAView with pairwise alignments. However, the dotplot has no clutter at all, but degrades an HR quickly to noise.

We expect that the MSAView is a useful tool for analysis of sequences of multiple species of tomato, which is the point of departure of our research. Especially when distance (5.11) is used the MSAView is useful, because species of tomato are likely to consist of recent evolutionary duplications.

In the end, we should conclude that the HEB visualization method is data dependent: A more valuable tree structure results in a more attractive visualization.

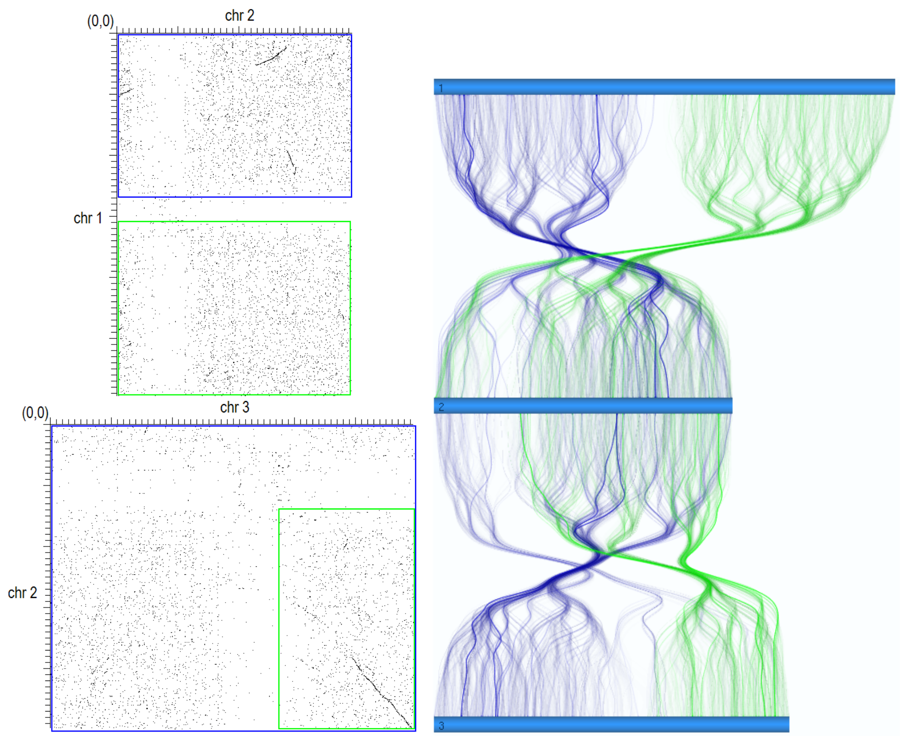


Figure 6.4. Dotplot vs MSAView using distance (5.10), with $\mathbf{L}_{\text{start}} = 1$

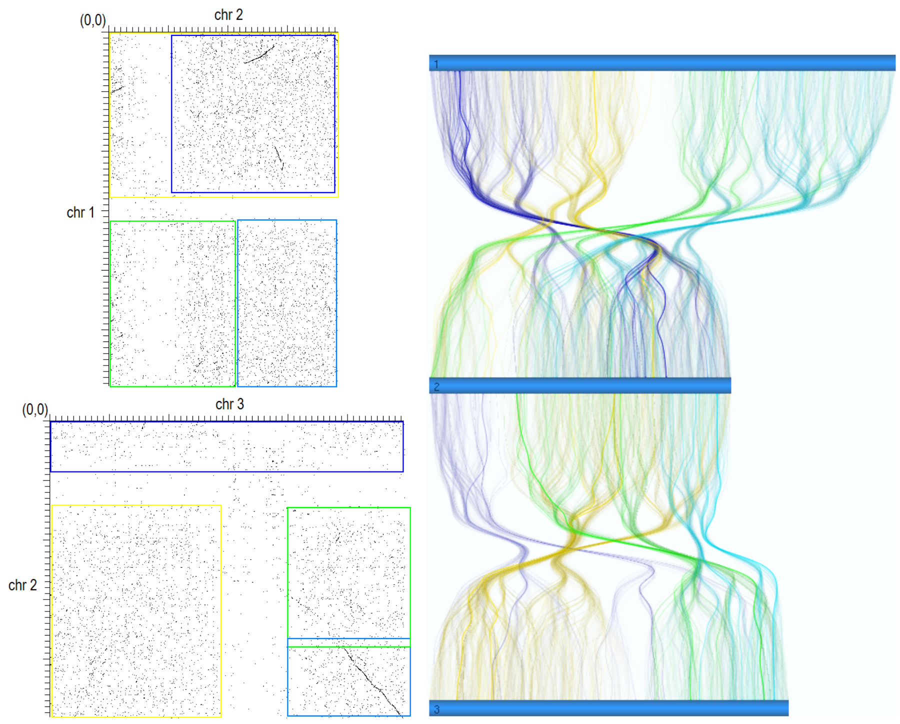


Figure 6.5. Dotplot vs MSAView using distance (5.10), with $\mathbf{L}_{\text{start}} = 2$

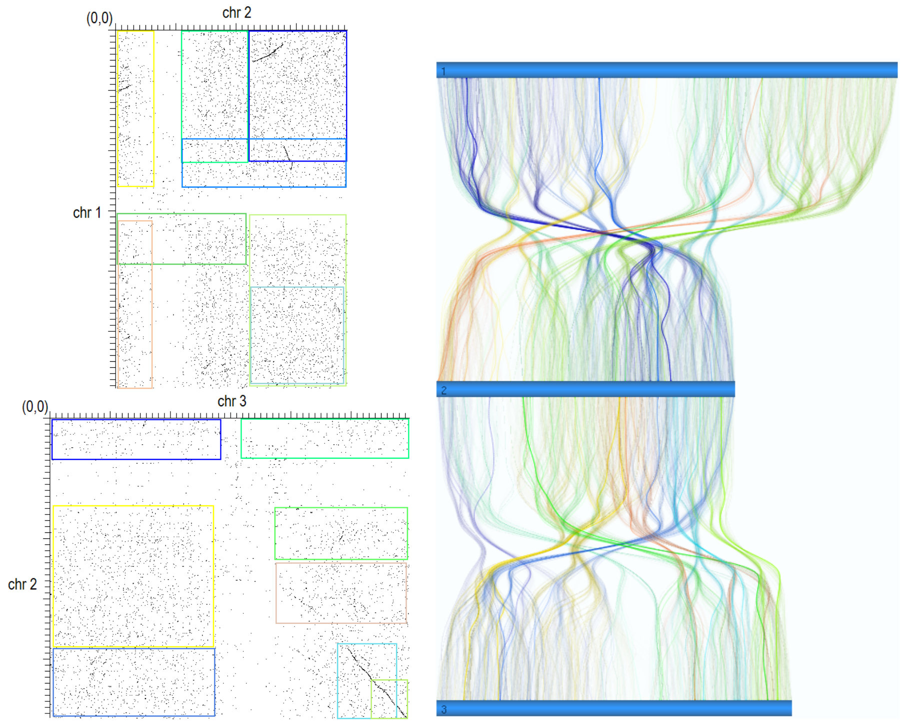


Figure 6.6. Dotplot vs MSAView using distance (5.10), with $\mathbf{L}_{\text{start}} = 3$

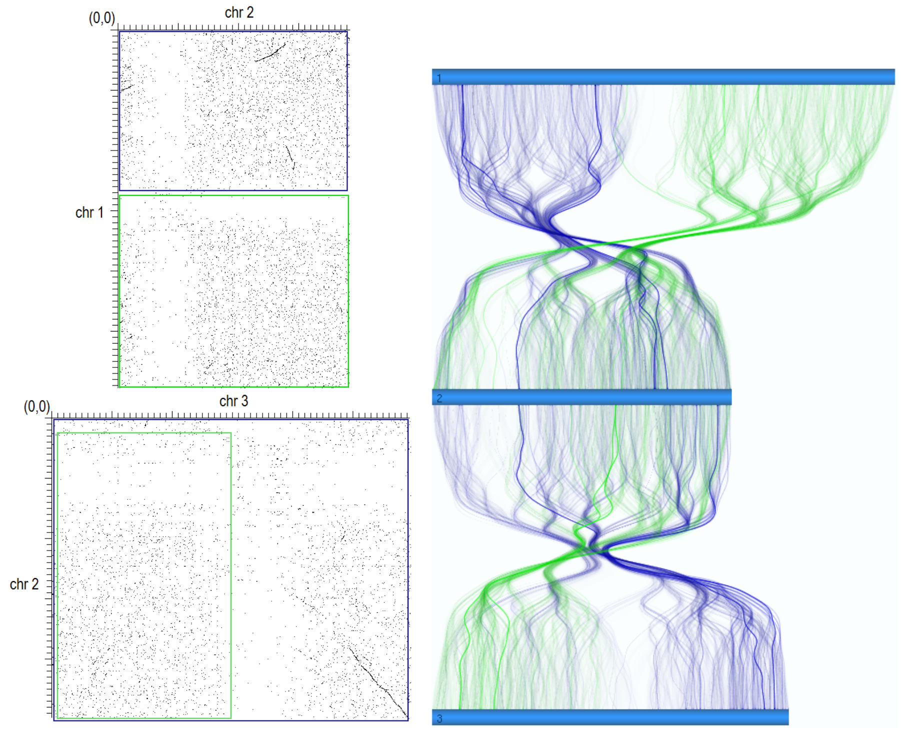


Figure 6.7. Dotplot vs MSAView using distance (5.11), with $\mathbf{L}_{\text{start}} = 1$

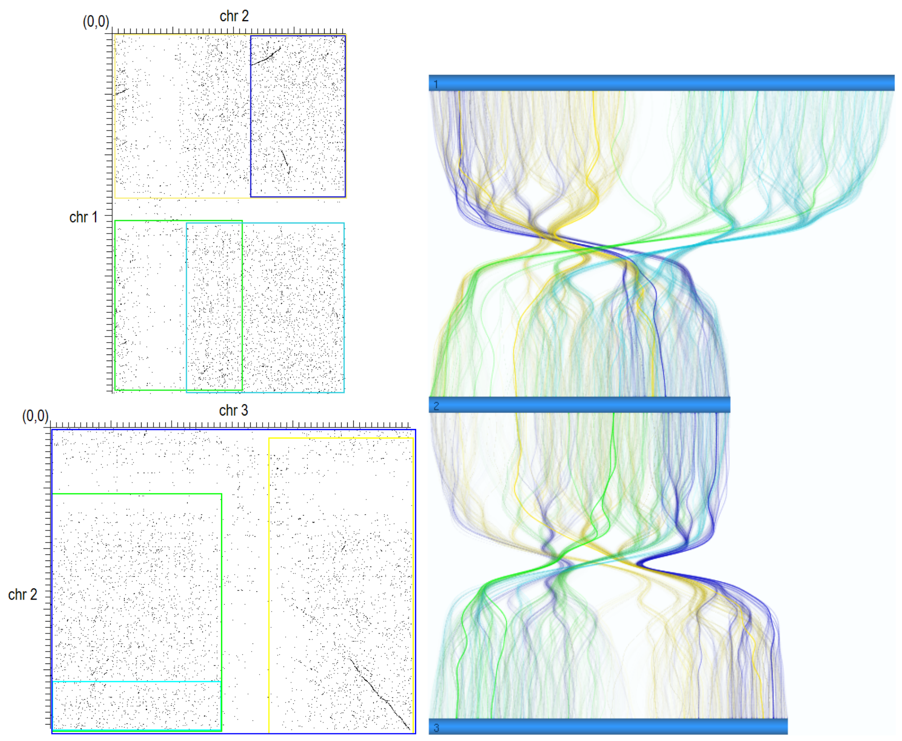


Figure 6.8. Dotplot vs MSAView using distance (5.11), with $\mathbf{L}_{\text{start}} = 2$

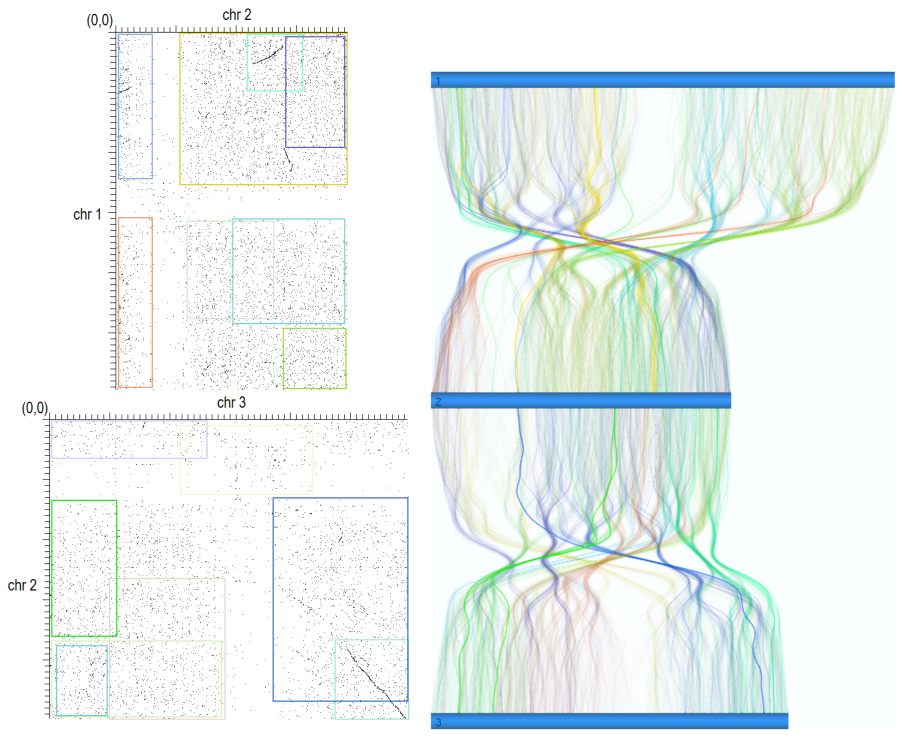


Figure 6.9. Dotplot vs MSAView using distance (5.11), with $\mathbf{L}_{\text{start}} = 3$

Chapter 7

Conclusion

The MSAView has been an attempt to emphasize dense locations of homologous regions in full sequence scale multiple sequence alignments, which do not express an obvious pattern, when rendered as straight lines between the sequences. Motivated by the fact that Hierarchical Edge Bundles (HEB) could unclutter a straight line chaos, we decided to setup an experiment with this visualization concept.

7.1 Achievements

Our explorative case study of hierarchical edge bundles has resulted in the following achievements:

- An MSA visualization is created in the context of DNAVis2, which does not make any assumption about the number of sequences.
- By means of bundling, dense locations of HRs are emphasized and cluttering has become more manageable.
- The biologist has a large number of techniques which allows him to tune the visualization. These techniques include various kinds of filtering, selection and adjustable visualization parameters.

7.2 Limitations

The results generated by the MSAView look a bit messy in comparison with Holten's results [34]. There are a number of reasons for that:

- Our data set is more chaotically distributed. The straight line versions of figure 5.9 and 5.2 (page 47 and 40) are quite orderly in contrast to our data as displayed in figure 3.4 on page 23.

- To emphasize width, HRs can be represented by bands, instead of single lines. This gives rise to a dilemma for the user: Single lines do not emphasize wide HRs and bands occlude, because clusters at the same level in the cluster tree are not disjunct.
- Holten's HEB method allows uncluttering by means of a permutation of leafs in the hierarchical structure, because their position is not fixed. Our implementation of HEB heavily relies on sequence alignments, due to locations of subsequences of HRs on the sequences. Therefore, the position of the leafs is fixed, hence the amount of clutter is unavoidable higher.
- The number of curves is at least a factor of ten higher with respect to figures in [34]. Because screen space is fixed, there is an upper bound on visible curves. Actually it depends on the distribution of curves, the more orderly distributed, the more curves can be displayed. We may have exceeded this upper bound.

We could have asked ourselves whether the plan to create order in chaotical alignments has not been too ambitious for a *computer science* Master project. Biologists with years of experience are struggling for decades to solve alignment problems, and still no general method exists which provides an overview at large scale. We have made a step forward in comparative genomics visualization: If a valuable hierarchical structure exists for HRs, then our visualization method can provide insights in that structure. By means of parameterization, the structure can be displayed as best as possible using techniques like filtering and selection. Future improvements in clustering will only add to the value of our visualization.

7.3 Verdict

To conclude the MSAView, we highlight some of van Wijk's ingredients of a valuable visualization [42].
The *costs* of the MSAView are not ideally low: There are a large number of potential users, which use an MSA visualization frequently. Development of the MSAView is minimal, but to use the MSAView in a foreign environment is costly: the user needs graphics hardware and a set of HRs has to be determined and clustered. Complex computations are common in bioinformatics, but graphics hardware is rare. The acquired knowledge is valuable, since it is not well-expressed in the data itself. However, at least for a single pairwise alignment, the *existing* dotplot [11] *method* is superior to the MSAView, because it has no crossing lines at all. Hence, the MSAView may not be cost effective.
The MSAView contains *negative knowledge*. The user should be aware that the top of cluster tree is only a binary tree structure to connect groups of close located HRs. But, we have introduced parameters to prevent visualization of this knowledge.
The ratio of *presentation* and *exploration* seems to be balanced, due to attractive pictures and a flexible model. As a side effect, the MSAView allows generation of a piece of *art*. We are looking forward to the moment somebody has a printing of his own genome in his living room, such as happened with a SequoiaView [43] visualization.
HEB was almost instantly successful for software metrics. Our MSAView needs some more improvements, mainly in the field of biology, to become a success, but, the fundamentals are realized.

Chapter 8

Recent and future work

Development of DNAVis has not been finished after [1]. Various publications [44, 45] and encouraging reactions from PRI gave a cause for further development of DNAVis. Improvements of DNAVis are described in section 8.1. Open problems of the MSAView are described in section 8.2.

8.1 DNAVis2

The initial version of DNAVis is a promising concept, however the implementation is not user-friendly. This may be the reason why DNAVis is not widely used, therefore further development is necessary. Improvements described in this section are mainly implemented by Huub v.d. Wetering and undersigned. The C++ implementation of DNAVis is ported to NetBeans Platform (NBP) [46]. This new implementation is called DNAVis2. Porting to Java is needed for the following reasons:

- NBP provides a common rich-client layout, which is assumed to be more familiar to the user than the old DNAVis user-interface.
- NBP provides a project-structure like in an Integrated Development Environment (IDE), which allows composition of various files. In DNAVis composition of FASTA and GFF files was controlled by a plain text file.
- NBP is a loosely coupled module-based architecture, which is easy extensible.
- NBP allows distribution of modules to running instances of an application via internet or by separated files.
- NBP provides a mechanism to handle various file types. This abstraction provides a way to use files on a remote location.
- Java allows usage of BioJava [47], an open-source framework for manipulation of sequences. This includes a reliable GFF2 parser (which is comparable with GFF3, appendix E.2).
- Java allows easy cross-platform development. Only few design issues have to be taken into account to run at least on both Windows and Linux platforms.

- Java allows distribution of applications via Java Web Start. This is practical, because installation is not needed for an application to run. Another advantage is that an application is always up-to-date.

After porting, other issues are tackled and new problems are solved. In the remainder of this section the introduction of GFF3 is discussed in subsection 8.1.1. Subsection 8.1.2 describes feature renderers in the Linear view.

8.1.1 GFF3

A major problem of DNAVis is the usage of GFF2. For example, genes, introns and exons have hierarchical relations, namely a gene consists of introns and exons. These are all separate annotations, which cannot be related to each other in GFF2 properly. Until now the convention is used, that a gene annotation is directly followed by its exon and intron annotations, but this is not a robust solution

GFF3 contains a proper definition of hierarchical relations [48] and support of GFF3 (appendix E.2) is therefore included in DNAVis2. Each annotation consist of a unique ID-tag and if necessary a Parent-tag containing the ID-tag of its parent annotation. After parsing a valid GFF3 file, children of an annotation are programmatically available, which allows information compression (see subsection 8.1.2).

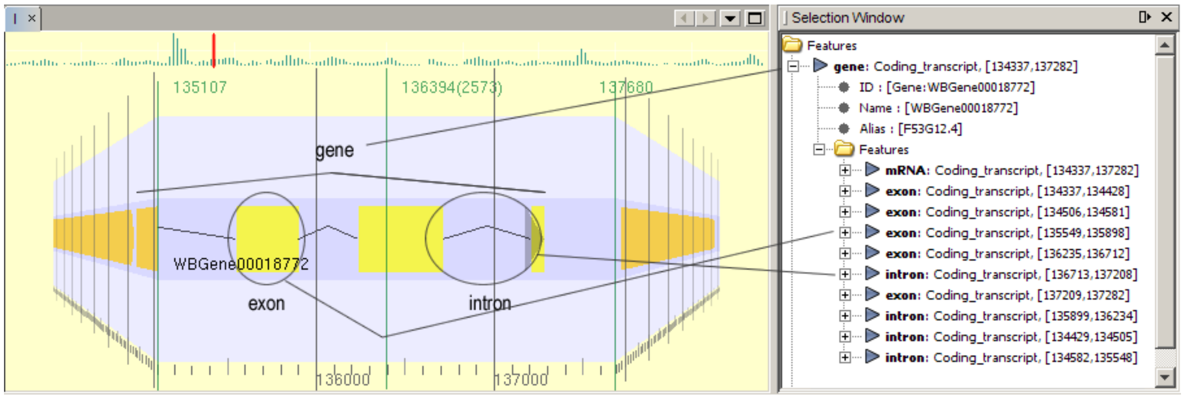
Usage of GFF3 provides other interesting features. First of all, it is possible to include a Dbxref-tag to an annotation, which allows to refer to an external data source containing more information about the annotation. This mechanism is primitive and not cleverly defined in the GFF3 specification yet. However, we expect this will improve soon, because this part of the GFF3 specification is still under development.
The type of each GFF3 annotation is based on a Sequence Ontology [49] term. The ontology describes the semantics of each annotation type and prevents proliferation. Hence, all genes are of type 'gene' and not some variants like 'Gene' or 'GENE', or another annotation type such as 'exon'. Validity of hierarchical relations are also recorded. So, a 'gene' is an ancestor of 'exon' but not the other way around.

Concluding, support of GFF3 has improved DNAVis2 by a well-defined, externally managed specification. This allows anyone to supply data, without undocumented assumptions.

8.1.2 Feature renderers

Availability of hierarchical relations in the GFF3 file type provides an opportunity to compress information in the Linear view. Based on aggregators [50] of the GBrowse Genome Annotation Viewer of the Generic Model Organism Database (GMOD) Toolkit a mechanism is developed, which allows to define a glyph per annotation type with the ability to access children of this annotation: *a feature renderer*.

Feature renderers allow developers to define glyphs using feature data, OpenGL commands [51], pre-calculated distances and parameters. Because each feature renderer is an NBP module, distribution is possible via the update functionality of the NBP.



```
Selection Window
Features
gene: Coding_transcript, [134337,137282]
    ID : [Gene:WBGene00018772]
    Name : [WBGene00018772]
    Alias : [F53G12.4]
    Features
        mRNA: Coding_transcript, [134337,137282]
        exon: Coding_transcript, [134337,134428]
        exon: Coding_transcript, [134506,134581]
        exon: Coding_transcript, [135549,135898]
        exon: Coding_transcript, [136235,136712]
        intron: Coding_transcript, [136713,137208]
        exon: Coding_transcript, [137209,137282]
        intron: Coding_transcript, [135899,136234]
        intron: Coding_transcript, [134429,134505]
        intron: Coding_transcript, [134582,135548]
```

Figure 8.1. An example gene feature renderer

In figure 8.1, an example feature renderer is depicted of a gene annotation. On the left, a Linear view is visible with on the front wall one partly visible annotation: a gene with ID 'WB-Gene00018772'. This gene consist of 5 exons and 4 introns, as visible in the list of features of this gene in the Selection window on the right. The gene annotation is used to keep child annotations together. On the other hand, while zoomed out, abstraction over child annotation takes place by drawing only an orange rectangle, such as on the perspective walls. Exon annotations are drawn as yellow rectangles and intron annotations are drawn between exons as a bent line, such as common in GBrowse.

Feature renders are a major improvement of the Linear view, because the number of needed annotation bars is decreased. However, this is not a cure-all, because a large number of annotations do not have hierarchical relations.
Annotation creators are challenged to include data in an annotation which allows a visualization. There are no limits to a feature renderer, even a complete graph or alignment is possible to draw in a glyph as long as the data is available in an annotation. Examples of possible interesting data may be Quantitative Trait Loci (QTL) data or 454 data. QTL data describes the contribution of an interval of bases to a certain property, such as earnings per square meter of grain. 454 data is a new sequence method which improves sequencing of close related species.
Biological data often are multi-dimensional consisting of both quantitative and qualitative values. Visualizing both dimensions at once could be interesting, e.g. one of the dimensions as geometry (size, position) and others as color, transparency, or texture.

8.2 Open problems of the MSAView

We have found a number of open problems of the MSAView. In subsection 8.2.1 open problems of the hierarchical clustering algorithm are listed. Suggestions for improvements for the visualization are enumerated in subsection 8.2.2

8.2.1 Clustering

It would be nice if the following items could be implemented in the clustering:

- *Further exploration of δ.* In subsection 5.1.4, we have considered two alternatives for distances between HRs. This degree of freedom should be researched further to get better outcomes for a cluster tree.
- *A non-binary cluster tree.* The MSAView tend to be chaotical, even with bundles. We expect a major improvement if the cluster tree is non-binary on all nodes, because it reduces the number of nodes. One option is to use nodes with a fixed number of children, but it is more natural to have a criterium that tells that no more children can be added to a node.
- *Less pre-computation.* The cluster tree is fully pre-computed and cannot be adapted at run-time. It is more attractive to allow more computations on the fly, which allows more parameters and thus a better expression of the hypothesis. In [35] a method is proposed which reduces the clustering algorithm with a factor N, for N HRs, but they forgot to describe it. It indicates that a method could exists, which allows near on-the-fly calculations. But, a powerful computer is needed, which decreases the amount of potential users.
- *Duplication typed HRs.* It is possible to introduce new types of HRs. We have noticed that the result of MUMmer can contain two HRs, which have one subsequence in common: a duplication (see 2.3.2.2). If these two HRs are grouped together, which is likely because the distance is small, the aggregated type can be 'duplication'.

8.2.2 Visualization

The following list suggest improvements for the visualization:

- *Base level alignments.* For in-depth research, like in the Linear view, individual bases of a subsequence of an HR should be visible, in a kind of alignment fashion. Alignment data can be accommodated in the predefined column 9 tag 'Gap' of a GFF3 file (see appendix E). We have chosen not to implement this feature for the moment, because the amount of annotation data would be too much, even for modern desktop computers. A single pairwise alignment easily consist of 100MB of alignment data.

- *A minimap.* An MSA data set for N sequences consist of N^2 pairwise alignments. A set of $N-1$ alignments is visible, hence $N^2 - N + 1$ pairwise alignments are invisible. It may be that something interesting happens at invisible pairwise alignments. A minimap - an $N \times N$ table - should indicate using colors, a heuristic indicating interesting pairwise alignments. Such a heuristic can be: the number of visible curves after filtering or the number of selections derived from the current selection.

- *Usage of a geometry shader for curves.* Large, unfiltered data sets are expensive to render, due a large amount of curves. From a profile session with Netbeans Profiler became clear that computation of a curve is the most expensive part of the rendering. We expect that real-time experience can be improved using a curve geometry program [52] running on a programable geometry shader. A geometric shader enables hardware-accelerated computations, which is currently only available on high-end graphics hardware.

- *Individual parameters per pairwise alignment.* So far, we only have used parameters which hold for all pairwise alignments. Especially separate filtering parameters, allow the biologist to more freely express himself. It allows - for example - to express the following query: Which matches overlap with its neighbor's inversions?

- *Instantiation of generic data reduction methods.* In subsection 5.3, two generic data reduction methods are presented. It may be, that more delicate criteria help the user to better express himself. It might be hard for a biologist to express its hypothesis using our set of parameters, because of the limited information included in an HR.

- *Wild twisting curves.* A single curve twists and turns a lot, because it is guided by a binary tree. A path in a binary tree is a list of decisions of going left or right. Therefore, a curve may take a detour. We have tried to minimize detours by reducing the number of control points, i.e. use only every i-th control point, but that destroys the bundling effect. Another try uses a height dependent bundling strength: the deeper the points in the tree the less the bundling strength is applied. This leaves the bundles in the center undisturbed, but straightens the curve at the leafs. The bundling effect is preserved, but the curve is still twisting and turning a lot.
  Question is: "Can we discard parts of the tree to minimize a detour and still keep bundling"?

- *More advanced selection.* The current selection highlights all curves touching a ray shot straight into the window. By selecting a bundle, all curves in that bundle are selected. But, the ray may also touch other curves, which are not in the bundle. Hence, too many curves might be selected.

Appendix A

Introduction to genomics

This appendix contains an adapted version of unpublished work which has been written as a preliminarily exercise to this Master project by undersigned.

In biology, genomic data as used in this thesis is only a tiny part of a larger world. This appendix explains what genomic data is, and its context. Be aware that this chapter gives only a rough introduction. It may be that vital information is not there. Among other things, [53] is used to obtain knowledge to write this appendix.

A.1 The cell

Every organism is built of one or more cells. An organism is a living creature (e.g. plant, animal, bacterium, mould, but not a virus) which has its own metabolism. A cell is a little factory: it consumes nutrients by converting them into energy, executes tasks and reproduces itself if necessary.

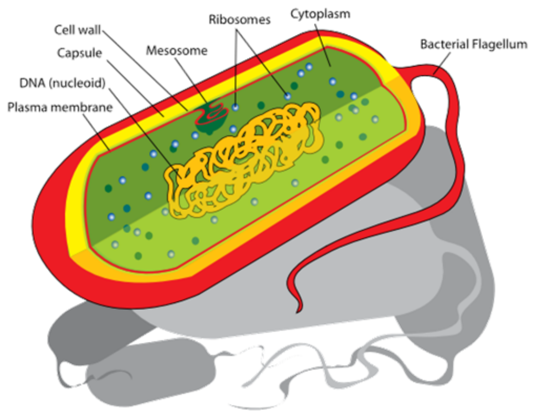


Figure A.1. Prokaryotic cell [54]

A cell is composed of several different parts, called organelles. Each organelle has its own task in the cell, such as energy production, transport of molecules, waste disposal or storage of genetic material. The mixture of organelles is in such a way, that characteristics of the cell correspond with the usage of the cell in the body of the organism. For example, a muscle cell contains a lot of mitochondrion organelles, which convert nutrients in energy and building materials. All organelles are located in the cytoplasm, a gel-like material. The cytoplasm is enclosed by a kind of skin, called the plasma membrane. In the very beginning of cell research one thought the membrane was passive. With the invention of the electron microscope it is known that the membrane has the function of filtering and transportation of materials.

There are two kinds of cells, prokaryotic cells (figure A.1) and eukaryotic cells (figure A.2). Prokaryotic cells are small and have simple structures whereas eukaryotic cells are complex and large. Bacteria are composed of prokaryotic cells, whereas animals consist of eukaryotic cells.

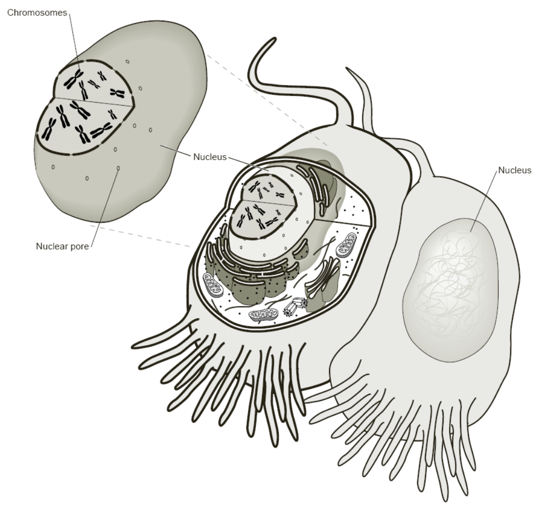


Figure A.2. Eukaryotic cell [55]

The main difference between prokaryotic and eukaryotic cells is the absence of a membrane enclosed nucleus and a number of other organelles in prokaryotic cells. Functions of the 'missing' organelles in the prokaryotic cells are taken over by the membrane of the cell. The only kind of organelle in both eukaryotic and prokaryotic cell is the ribosome. Ribosomes are responsible for translation of Messenger RNA (mRNA) in proteins. In section A.2 mRNA is described in more detail. In stead of a nucleus, the prokaryotic cell has a nucleoid. The nucleus is really an organelle, since it has a membrane, whereas the nucleoid is just a location within the cell. Both nucleus and nucleoid contain genetic material of the cell, but they are organized in different ways. A piece of circular DNA and possible some copies of it are located in the nucleoid. In a nucleus, DNA is divided in linear chains which are called chromosomes. A chromosome is composed of genes which as a whole are called the genome of the cell. The task of the nucleus is to maintain integrity of the genes and control the cell using gene expression.

A.2 Nucleic acids

In the previous section nucleic acids (like DNA and mRNA) are not well-defined yet. Nucleic acids are (bio)polymers, which are large molecules consisting of repeating molecular structures. The repeating structures are called nucleotides consisting of a sugar group, a phosphate group and a nucleobase. The phosphate group and sugar form the backbone (or strand) of the polymer and the nucleobase dangles on that backbone.

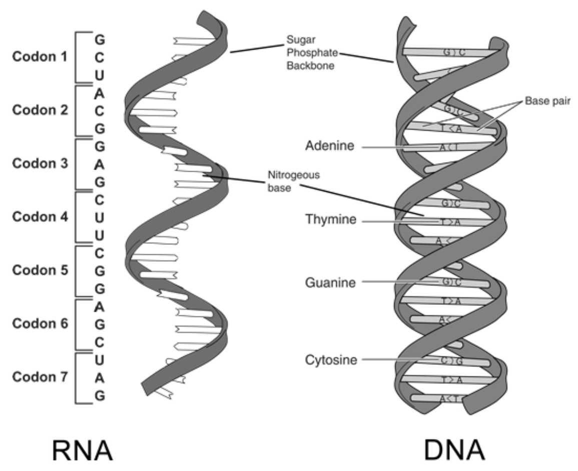


Figure A.3. RNA vs. DNA [55]

There are two main kinds of nucleic acids, Deoxyribonucleic Acid (DNA) and Ribonucleic Acid (RNA). The relation between DNA, RNA and proteins are defined in “The Central Dogma of Molecular Biology” [56]: For organisms, DNA can copy itself, DNA can be transcribed in Ribonucleic Acid (RNA), RNA can be translated in proteins and proteins control the cell. Conclusion: DNA controls the cell.

A.2.1 Deoxyribonucleic acid

The DNA molecule (figure A.3) consist of deoxyribose as sugar group and Adenine (A), Thymine (T), Guanine (G) and Cytosine (C) as nucleobases. DNA has two backbones with the property that the bases of one backbone are the complement of the other. The C is complementary to G and A to T. Besides being complementary, bases are 'connected' using hydrogen bounds.

The two backbones with bases inside is a double helix structure. The model of the DNA-molecule is described in the early fifties by Crick and Watson [13]. However, Wilkins and Franklin have experimentally proven the correctness of the model before the publication of Crick and Watson.

A backbone has polarity, due to a 5'-hydroxyl (or 5'-phospho) group at the begin of the backbone and a 3'-hydroxyl group at the end. For the complementary (anti-parallel) backbone, the polarity is the other way around. Therefore, the backbone starting with a 5'-hydroxyl group and ending with a 3'-hydroxyl group is called a 5'→3'-strand. The other backbone is called 3'→5'-strand.

The 5'→3'-strand is the coding (or sense) strand. The other strand is the template (or antisense) strand.

Cell division is an important, but complicated process. There are two kinds of cell division, called mitosis and meiosis. From a DNA perspective, both cell divisions are the same. This process is called DNA replication (figure A.4). DNA replication is one of the various processes in cell division.

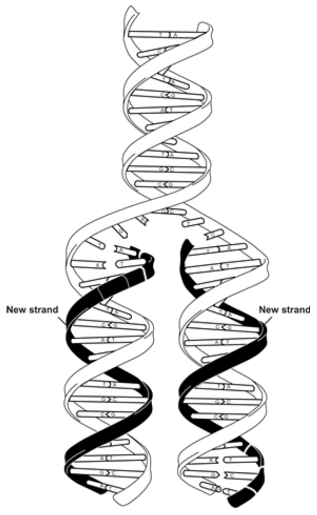


Figure A.4. DNA replication [55]

The purpose of DNA replication is to provide both original and new cell the same piece of DNA. To make a second DNA molecule, the double helix is split in the 3'→5' direction. An enzyme creates two new backbones by means of adding nucleobases on the positions of the removed strands. Since nucleobases are complementary, there is only one nucleobase that fits the new stand. This results in two identical DNA molecules.

A.2.2 Ribonucleic acid

A Ribonucleic Acid (RNA) molecule (figure A.3) is less complex as DNA. RNA has a single strand with as sugar group ribose. The nucleobases of RNA are the same as in DNA, except for thymine, which is Uracil (U) in RNA. There are many kinds of RNA, like Transfer RNA (tRNA), Ribosomal RNA (rRNA) and mRNA. The latter is the most important, but the others are also necessary. Messenger RNA is a medium to transfer orders from the DNA to the rest of the cell, outside the nucleus. This process is called gene expression.

Messenger RNA is built directly from DNA using DNA transcription (figure A.5). An enzyme - called RNA polymerase - moves in the 3'→5' direction on the antisense strand of the DNA sequence and adds complementary nucleobases to the mRNA strand. Where thymine is expected to be on the mRNA sequence - to be fully complementary - uracil is placed, since RNA has no thymine. The resulting mRNA strand is the same as the sense strand of the DNA sequence except from thymine bases.

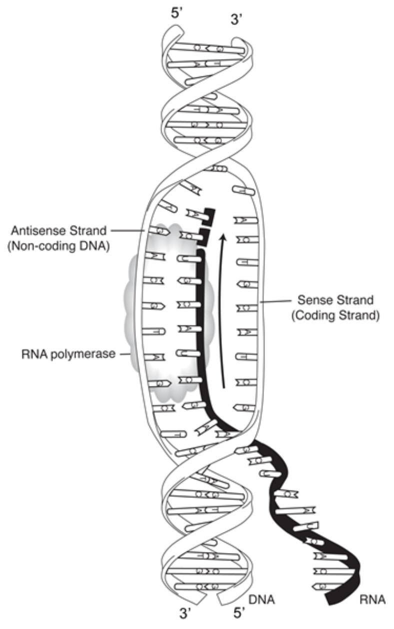


Figure A.5. DNA transcription [55]

Not the whole DNA is transcribed in mRNA, only genes. The transcription starts on a special subsequence, called the promotor region. This region is located on the antisense strand of the DNA sequence. Termination of transcription is still partly an open problem. However, two methods are well known. The first method is by a terminator subsequence. The other method is using a so called ρ factor, which is too technical to discuss now.

The process of gene expression is not complete yet. The next section finishes gene expression.

A.3 Chromosomes, genes and proteins

Return to the nucleus of an eukaryotic cell. Until now, chromosomes are vaguely defined as a mixture of genes. A chromosome is a very large strand of DNA where a gene is part of. A gene has a position on a chromosome, called the locus. Apart from the genes, a chromosome consist of DNA which is not yet defined in research, called 'junk' DNA. Junk DNA might be useless (which is quite unexpected to be) or might be a mechanism to defend against mutations.

A gene consist of two types of regions, introns and exons. Intron regions contain junk DNA, but there are indications that these have more intelligent functions. Exons consist of instructions to encode a protein, which is a chain of amino acids. Many proteins are enzymes, which control processes in the cell.

A gene can have several versions, called alleles. For example, one allele can code a brown eye, another allele a blue eye, but both are the same gene.

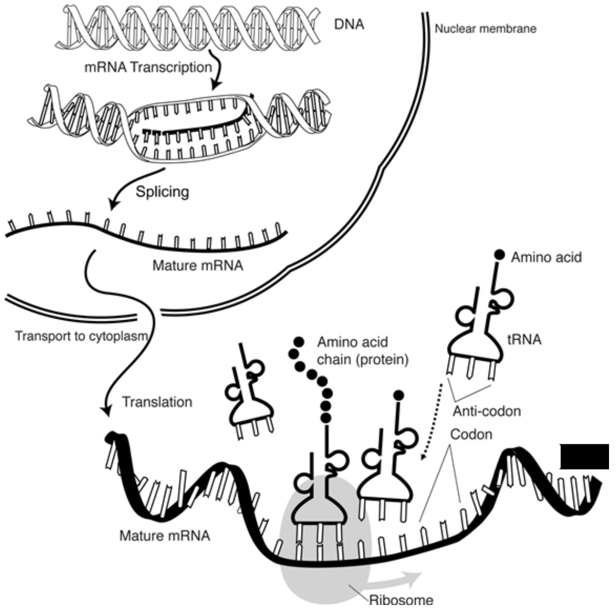


Figure A.6. mmRNA translation [55]

The process of gene expression is finished in this section. When mRNA is made out of DNA using transcription (figure A.6), all intron regions are removed using a process called splicing. This results in Mature mRNA (mmRNA), containing only exons. The mmRNA is moved outside the nucleus into the cell where using the mmRNA a protein is built.

Each amino acid of the protein to be produced is coded in a triplet of nucleobases (codon). Each codon corresponds to one amino acid, the genetic code (table 2.1). An organelle called ribosome reads the mmRNA and assembles the amino acids to the protein. If the protein is assembled it can do its work in the cell.

Appendix B

Data flow in biological research

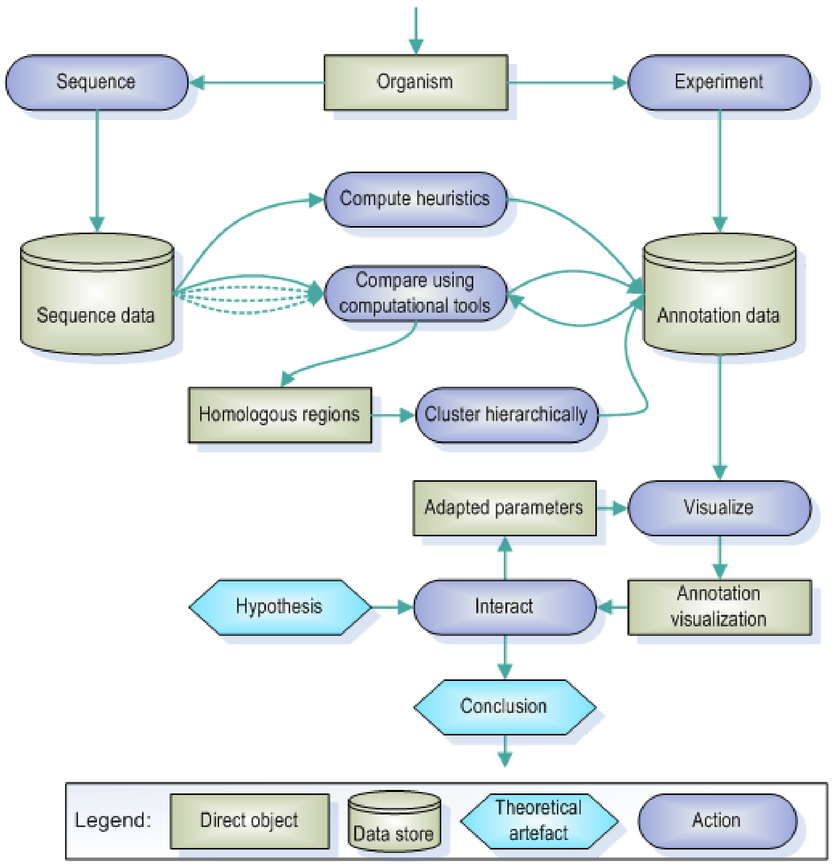


Figure B.1. Overview of the data flow in biological research using annotation visualization applications.

In figure B.1, an overview of the data flow in biological research using annotation visualizations is displayed. More specific using DNAVis2, therefore no distinction is made between genome browsers and comparative visualizations, because both are annotation-driven in DNAVis2.

The shape of each node in the diagram equals the type as stated in the legend:

- *Direct object*: An object subject to actions.
- *Data store*: A set of direct objects, preferably managed by an authorized organization.
- *Theoretical artefact*: A specialized direct object, to keep research separated from data.
- *Action*: Execution of the described task, taking all incoming shapes as input and result in one connected output.

Sequencing of an organism results in a database of sequences. A database of annotations is created by means of experiments, a heuristic computation on one sequence or a comparative computation based on multiple sequences and optionally other annotations. A comparative computation can be clustered, resulting in a cluster tree, which is stored using annotations.
Annotations are retrieved from the annotation data store by means of one or more GFF3 files. This data is visualized, resulting in a Linear view, Matrix view or MSA view. Having the hypothesis in mind, interaction results in a conclusion or the parameters are adapted resulting in another view on the visualization.

Appendix C

Requirements MSA view

This appendix contains all requirements as extensively defined in chapter 3. The developed Multiple Sequence Alignment (MSA) visualization $\mathcal{M}$ satisfies the following requirements:

Requirement I (Data size)
The user should be able to visualize data sets of arbitrary size.

Requirement II (Own data sets)
The user should be able to use its own data set.

Requirement III (Bases)
The user should be able to view individual bases of a DNA sequence.

Requirement IV (Overview)
The user should always have an overview of a full data set.

Requirement V (Navigation)
The user must be able to navigate through the data set in an interactive and continuous way.

Requirement VI (Link preservation)
The user must be able to link $\mathcal{M}$ to other DNAVis2 *views.*

Requirement VII (Homologous regions)
The user should be able to view locations on sequences which are similar to each other.

Requirement VIII (Number of sequences)
No assumption should be made for the number of sequences in the visualization.

Requirement IX (Region of interest)
The user should be able to specify one or more intervals on sequences which are more interesting than others.

Requirement X (Filtering)
The user should be able to specify - using criteria i to v of subsection 3.4.2.2 - a subset of homologous regions which are of interest.

Requirement XI (Emphasizing)
Homologous regions of interest should be emphasized by $\mathcal{M}$.

Requirement XII (Uncluttering)
The user should be able to unclutter homologous regions of highly conserved sequences.

Requirement XIII (Abstraction)
The user should be able to join homologous regions to get a better overview.

Requirement XIV (Derivation)
The user should be able to propagate homologous regions of interest in one pairwise alignment to other pairwise alignments by means of overlap of homologous regions.

Appendix D

Implementation

This appendix describes the development environment and architecture of the MSAView.

D.1 Development environment

DNAVis2is written in Java, therefore is should run on any operating system with a Java virtual machine. Windows XP and Fedora Linux distributions both are supported to run DNAVis2.

Java Development Kit version 6 [57] is required to run DNAVis2. The seventh milestone of NBP 6 [46] serves as rich-client application framework, which is also used by Boeing and Nokia. JOGL (version 1.0) [58] provides Java bindings for OpenGL [59], which enables usage of graphics hardware. The BioJava (version 1.4) [47] library provides data structures and parsing routines for sequence and annotation data sets.

The MSAView is developed on one machine, with the following hardware specifications:

CPU	A hyper-threaded Intel Pentium 4 3.0GHz
RAM	1.5 GB
GPU	NVidia GeForce 6600
OS	Microsoft Windows XP SP2

D.2 Software metrics

This project has extended the DNAVis2 application with the following modules:

Module	*Java files*	*LOC*	*Description*
`gff3`	8	503	A GFF3 parser
`intervalsbrowser`	6	949	A generic view containing sequences with linkable intervals
`featurerenderer exon`	2	107	A test feature renderer
`featurerenderer gene`	2	183	A feature renderer displaying gene and exon annotations in one glyph
`MSA view`	12	4651	The MSA view itself
Total	30	6393	

The `browser` and `groups` modules are modified to allow linking of intervals, which are a couple of hundred lines of code. Furthermore, the `linear view` and `browser` modules are adapted to create a feature renderer infrastructure, resulting in at most 1000 lines of code. The Java program which computes a cluster tree from a set of HRs, resulting in GFF3 input takes 1000 lines of code. Finally, in various places of the DNAVis2 application bugs are fixed and improvements are carried out. This has resulted in at most 500 lines of code. Roughly speaking, in total about 8500 lines of code are written.

D.3 Architecture

This section briefly describes the global software architecture of the MSAView using Unified Modeling Language (UML) class diagrams. To enable expression of modular legacy software, two extra dimensions are added to these class diagrams: A color indicates to which module the class or interface belongs. A texture indicates whether the class or interface is newly created or already existed in DNAVis2.
The MSAView consist of two modules: `intervalsbrowser` and `MSA view`. The `intervalsbrowser` module is described in subsection D.3.1 and the `MSA view` module is explained in subsection D.3.2.

D.3.1 Intervalsbrowser module

The `intervalsbrowser` module is a generic module, which accommodates data structures and linking capabilities for a sequence visualization consisting of multiple sequences. A sequence may contain linkable intervals and the view itself is not linkable. Note that this module has nothing to do with alignments, only with sequences. The `intervalsbrowser` module is similar to the existing `browser` module, which accommodates a model of a single sequence visualization (such as the Linear view).

In figure D.1, a class diagram is displayed of the `intervalsbrowser` module and the dependencies with modules `browser` and `view`. The latter is general purpose module containing ingredients for (OpenGL-based) windows. The main idea of `intervalsbrowser` is easy: the visualization (IntervalsDnavisTopView) consist of one model (IntervalSortedSequences) with multiple sequence (IntervalSortedSequence) which may contain multiple intervals (IntervalModel). The

IntervalsDnavisTopView class inherits OpenGL capabilities from the GLTopView class and implements the IntervalsSeqeunceBrowser interface to state that its model contains more than one sequence. The IntervalsSortedSequence inherits from SortedSequence which contains both sequence and annotation data from one sequence.

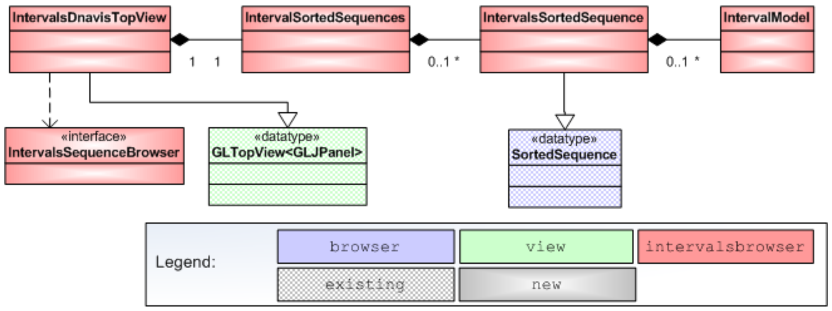


Figure D.1. A class diagram of the `intervalsbrowser` module

The linking capabilities (see figure D.2) of DNAVis2 heavily rely on the Singleton and Observer patterns [60]. A linkable can be either a view or an interval. All views are accommodated in the DnavisViews singleton, and all intervals are contained in the AllIntervals singleton. The Groups singleton observes in DnavisViews and AllIntervals for new or removed linkables and observes classes implementing LinkedView for changes in link connections. Linkables are grouped; each time a linkable is the leader (yellow) and not contained in the active group, a new group is created.
The SequenceBrowserModel guards the range and position of one visualized sequence, which - for example - is the state of the front wall of a Linear view. An IntervalModel is exactly the same, but with an extra dimension: the length in - for example - in world coordinates. If such a model changes, the Groups class notices a change and dispatches it to the group of linkables of the changed model.

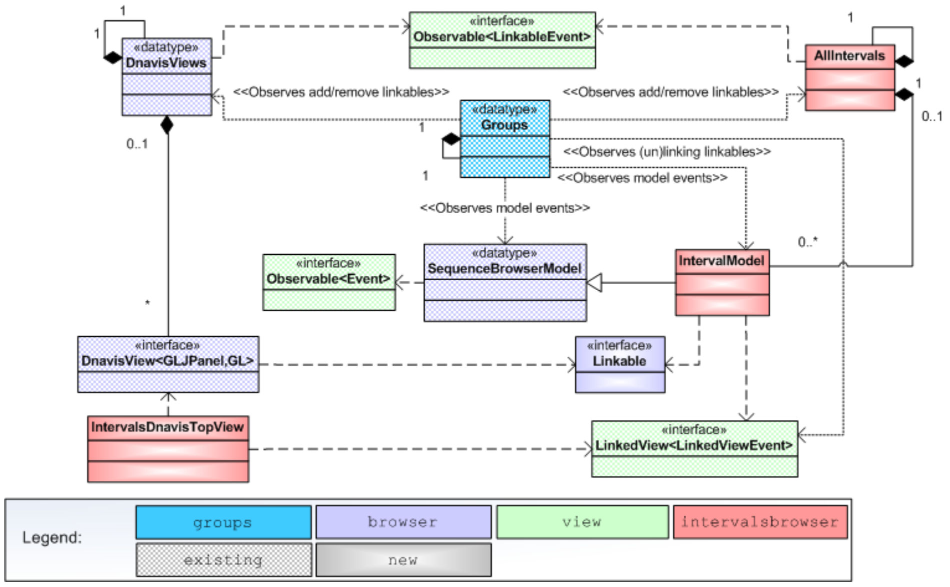


Figure D.2. A class diagram of linking in the `intervalsbrowser` module

D.3.2 MSA view module

The `MSA view` module (see figure D.3) contains the specific implementation of an MSA visualization. Using the MSAViewAction class the NBP loads data from selected files in the project tree and creates with loaded data a new instance of MSAView. The MSAView class accommodates cluster trees - for each pair of alignments one - by means of ClusterNode. The attributes of an HR of a ClusterNode are encoded in the Region class. The state of all parameters of the visualization - like the bundling strength - are included in the MSASettings class, which can be adapted with the GUI in the MSAController class.

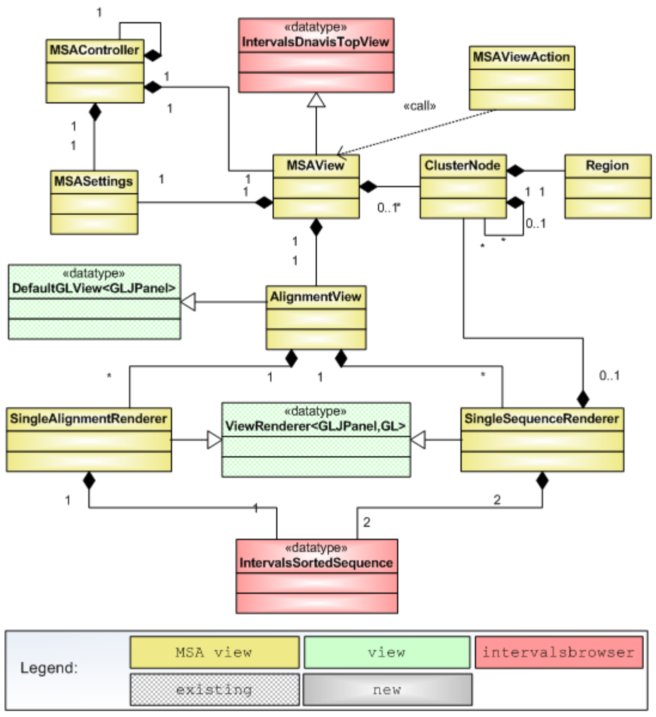


Figure D.3. A class diagram of the `MSA view` module

The MSAView may consist of multiple canvasses with its own view ports. We only used one canvas, namely AlignmentView. The AlignmentView contains mouse interaction routines and methods to transform an interval to a lens. If you want to include an extra canvas, for example a minimap (see chapter 8), only add an extra class to MSAView which derives from DefaultGLView and adapt the view port of AlignmentView. For each visible sequence, the AlignmentView consist of one SingleSequenceRenderer, which renders a sequence with its intervals within the view port of AlignmentView. For each visible alignment pair, the AlignmentView consist of one SingleAlignmentRenderer, which is the class where the most interesting visualization methods are inhabited. The SingleAlignmentRenderer class contains one ClusterNode object which is the cached filtered cluster tree, which always is rendered and only updated if a filtering parameter is changed.

Appendix E

Input specification

This appendix describes the file formats used in DNAVis2. Plain DNA sequences are encoded in the FASTA file format as described in section E.1. The semantics of a sequence are captured in DNA annotations which are stored in a GFF file (section E.2). Finally, in section E.3 is specified how to format a file which is displayable in the MSA view of DNAVis2.

E.1 FASTA

A FASTA file [61] is a plain text file with lines not longer than 80 characters. Officially, FASTA files have no file extension, but .fa, .mpfa, .fna, .fsa, or .fasta are most used. An example of a piece of tomato DNA sequence [3] in FASTA format:

```
>C01HBa0003D15.1 AC193776.1 submitted_to_sgn_as:100406.C143 100406.assem.ace.1 (whole contig)
ATAACGTAGAATTAAAACAAATGAATGTTACAACTCACATTCATGTAATT
TGTTATTTCAAAACTTTTTCAACTGATGAATTGTTTCAAAATCAATTGAA
ACACTAACTCTTACAATTTCCCACTTGTTTCAAGAAATTTATCTTTCAGA
AATCCAAAAAATATTTTAAGACGTTAATCTAGCAACATGCATCAATAATG
GTGTCTTTTGGACTTGAACCATTAGCTAGTGAAGGTTTTGTAAATCATTG
ACTACTTAGTGAACAAATCTTGAACTAAGTGGTTCATTTAATGAAGTTAG
AAAAATACTCCACGTATTTCATTATGTTCCAGCGAAAATCGGAATCCTTT
```

A FASTA file is in the language $\mathcal{L}(G)$, where $G = (V, T, S, P)$ is a Linz' style grammer [62]. Variables V are unquoted words in the productions P, terminals T are single quoted (' ') characters in P, S is the start variable and P is the following list of productions:

```
S                → Sequence*
Sequence         → Header (Comment | Data)*
Header           → '>' SequenceID Description
SequenceID       → ('a'|..|'z'|'A'|..|'Z'|'0'|..|'9'|'|')*
Description      → ('a'|..|'z'|'A'|..|'Z'|'0'|..|'9'|'|'|' ')*
Comment          → ';' ('a'|..|'z'|'A'|..|'Z'|'0'|..|'9')*
```

```
Data               → (DataCapitals|DataNonCapitals|DataMisc)*
DataCapitals       → 'A'|'C'|'G'|'T'|'U'|'R'|'Y'|'K'|'M'|'S'|'W'|'B'|'D'|'H'|'V'|'N'
DataNonCapitals    → 'a'|'c'|'g'|'t'|'u'|'r'|'y'|'k'|'m'|'s'|'w'|'b'|'d'|'h'|'v'|'n'
DataMisc           → '-'

Note:  SequenceID, Description and Comment may contain other characters
```

Multiple DNA sequences may be stored in a FASTA file. Each sequence begins with a header containing the sequence identifier and optionally a description. The actual data encodes one base or possibilities of bases. The latter is needed for draft versions of sequences which are not complete yet. The following mapping is used:

A	→	adenosine	M	→	A C (amino)
C	→	cytidine	S	→	G C (strong)
G	→	guanine	W	→	A T (weak)
T	→	thymidine	B	→	G T C
U	→	uridine	D	→	G A T
R	→	G A (purine)	H	→	A C T
Y	→	T C (pyrimidine)	V	→	G C A
K	→	G T (keto)	N	→	A G C T (any)

Apart from bases it is possible to encode a gap, which occurs in alignments. A gap is denoted by `-`.

A FASTA file may contain a protein, then the formatting differs. In this project the protein FASTA format is not used, therefore it is skipped. More information about protein formatted FASTA files can be obtained in [61].

E.2 General Feature Format Version 3

Semantics of a DNA sequence are stored in the General Feature Format (GFF) Version 3 [48]. A GFF3 file is a plain text file which encodes one annotation per line.

An annotation consist of the following columns which are separated by tab characters:

Column	*ID*	*Description*
1	SeqID	The identifier of the annotation's sequence which is equal to the SequenceID in the corresponding FASTA file.
2	Source	Indicates the program used to obtain the annotation or the authority which provides the annotation.
3	Type	The type of the annotation. For example a gene, exon, intron or mRNA. These types are restricted by the Sequence Ontology to prevent proliferation.
4	Start	The index of the first base involved in the annotation.
5	End	The index of the last base involved in the annotation.
6	Score	Indicates a floating point value for fuzzy annotations with respect to quality or precision.
7	Strand	Indicates on which strand the annotation is located (+,-,.[=unstranded]).
8	Phase	Indicates the offset with respect to the first codon. This applies only to Coding Sequence (CDS) annotations.
9	Attributes	A list of key-value pairs.

A list of reserved keywords is available for column 9. Attributes are separated by a semi-column. A key-value is formatted as ⟨key⟩ = ⟨value⟩, where ⟨value⟩ may be a comma-separated list of values. Any key-value pair may be inserted in column 9, but a number of case-sensitive keywords have predefined semantics:

Key	*Description*
ID	An unique identifier of the annotation in the scope of the GFF3 file.
Name	A not necessary unique name of the annotation.
Alias	A secondary name.
Parent	A pointer to the ID attribute of a parent annotation. An example relation is the mRNA(parent)-exon(child) relation.
Target	A 4-tuple (ID,start,end,strand) indicating a relation from this annotation to another interval on a possible other sequence. Matches and inversions are encoded using this key.
Gap	Provides space for a gapped alignment.
Derives_from	Indicates a temporal relation.
Note	Space for notes.
Dbxref	A reference to an external data source.
Ontology_term	A reference to an ontology term.

An example of some tomato annotations (→ = continue line):

```
C04HBa0308B07.2 FGENESH_Tomato mRNA 14941 16297 .  + .  →
ID=FGENESH_Tomato-C04HBa0308B07.2-gene_4-mRNA_1


C04HBa0308B07.2 FGENESH_Tomato CDS 14941 15120 -0.68 + .  →
ID=FGENESH_Tomato-C04HBa0308B07.2-gene_4-CDS-1_1;→
Parent=FGENESH_Tomato-C04HBa0308B07.2-gene_4-mRNA_1
```

```
C04HBa0308B07.2 FGENESH_Tomato CDS 15971 16267 23.38 + .  →
ID=FGENESH_Tomato-C04HBa0308B07.2-gene_4-CDS-2_1;→
Parent=FGENESH_Tomato-C04HBa0308B07.2-gene_4-mRNA_1
```

These annotations indicate the following: On sequence C04HBa0308B07.2 is using the FGENESH tool an mRNA with two CDS regions found. The mRNA ranges from 14941 to 16297 on the positive strand. The middle annotation is a CDS which ranges from 14941 to 15120, note that this range is occluded by the mRNA annotation. This CDS has a score of -0.68. The other CDS is also occluded by the mRNA annotation and has a score of 23.38. The Parent keys of both CDS indicate that these have the mRNA annotation as parent.

E.3 MSAView view input

The input of the MSAView consist of N sequences and N^2 pairwise alignments. For each sequence one FASTA file is needed. Each pairwise alignment requires a pre-computed cluster tree encoded in GFF3 format. Each cluster tree should be stored in the same GFF3 file, but a GFF3 file may contain multiple cluster trees.
HRs are defined in section 3.2. In a cluster tree $\mathcal{T} = (V, E_T)$ each node $\alpha \in V$ is formatted as follows, where $\alpha = (\mathrm{S}_0[l_0, r_0], \mathrm{S}_1[l_1, r_1], \mathrm{Type}, \mathrm{Score})$:

Column	*ID*	*Description*
1	SeqID	The id of sequence $\alpha.\mathrm{S}_0$ in its FASTA file.
2	Source	A free to choose source; we have chosen 'Mummer_v3' and 'BlastP', equal to the tool used.
3	Type	If α.Type is a match, then 'match'. If α.Type is an inversion, then 'inversion'. If α.Type is is unknown, then 'region'. All these types are terms in the Sequence Ontology [49].
4	Start	l_0.
5	End	r_0
6	Score	α.Score, assumed a floating point percentage.
7	Strand	'+', all HRs match with the positive top sequence.
8	Phase	'.', has no meaning at the moment.
9	ID	An unique identifier of the annotation in the scope of the GFF3 file.
9	Parent	The column-9 ID of the parent β of α, such that $(\beta, \alpha) \in E_T$
9	Target	A 4-tuple $(\mathrm{ID}, \mathrm{start}, \mathrm{end}, \mathrm{strand})$, such that ID is the id of sequence $\alpha.\mathrm{S}_1$ in its FASTA file, start is l_1, end is r_1 and strand is '-' if α.Type is an inversion or '+' otherwise.
9	SplitScore	Optional, an alternative score.

List of acronyms

A Adenine
ACT Artemis Comparison Tool
BARD Biological Arc Diagram
BLAST Basic Local Alignment Search Tool
C Cytosine
CDS Coding Sequence
DNA Deoxyribonucleic Acid
DP Dynamic Programming
G Guanine
GFF General Feature Format
GMOD Generic Model Organism Database
HEB Hierarchical Edge Bundles
HR Homologous Region
IDE Integrated Development Environment
LOD Level Of Detail
M-GCAT Multiple Genome Comparison and Alignment Tool
MAtDB MIPS Arabidopsis thaliana Database
mRNA Messenger RNA
mmRNA Mature mRNA
MSA Multiple Sequence Alignment
MUM Maximal Unique Match
NBP NetBeans Platform
PRI Plant Research International
QTL Quantitative Trait Loci
rRNA Ribosomal RNA
RNA Ribonucleic Acid
T Thymine
TIGR The Institute for Genomic Research
tRNA Transfer RNA
TU/e Eindhoven University of Technology
U Uracil
UML Unified Modeling Language
WUR Wageningen University and Research Centre

References

[1] T.H.J.M. Peeters. Interactive visualization of annotated dna sequences. Master's thesis, Technische Universiteit Eindhoven, Department of mathematics and computer science, 2004.

[2] J.J. van Wijk. Bridging the gaps. *IEEE Computer Graphics and Applications*, 26(6):6–9, 2006.

[3] International Solanaceae Genomics Project (SOL). International tomato sequencing project overview.
http://www.sgn.cornell.edu/about/tomato_project_overview.pl.

[4] M.R.V. Chaudron, J.F. Groote, K.M. van Hee, C. Hemerik, L.J.A.M. Somers, and T. Verhoeff. Software engineering reference framework. Technical Report CS-Report 04-039, Department of Mathematics and Computer Science, Eindhoven University of Technology, 2004.

[5] Anne Kaldewaij. *Programming: The derivation of Algorithms*. Prentice Hall, 1990.

[6] George W. Burns and Paul J. Bottino. *The science of genetics*. Macmillan Publishing Company, 6 edition, 1989.

[7] Mark Ridley. *Evolution*. Blackwell Scientific Publications, 1993.

[8] David P. Clark. *Molecular biology: Understanding the genetic revolution*. Elsevier Academic Press, 2005.

[9] Andrzej K. Konopka and M. James C. Crabbe, editors. *Compact handbook of computational biology*. Marcel Dekker, 2004.

[10] Mark Fiers. *Bioinformatics for plant genome annotation*. PhD thesis, Wageningen University and Research Centre, 2006.

[11] Adrian J. Gibbs and George A. McIntyre. The diagram, a method for comparing sequences. Its use with amino acid and nucleotide sequences. *Eur. J. Biochem*, 16(1):1–11, 1970.

[12] R.A. Fisher. The correlation between relatives on the supposition of mendelian inheritance. *Transactions of the Royal Society of Edinburgh*, 52:399–433, 1918.

[13] J.D. Watson and F.H.C. Crick. Molecular structure of nucleic acids. A structure of deoxyribose nucleic acid. *Nature*, 171(4356):737–738, April 25 1953.

[14] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, March 1970.

[15] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[16] Todd J. Treangen and Xavier Messeguer. M-GCAT: interactively and efficiently constructing large-scale multiple genome comparison frameworks in closely related species. *BMC Bioinformatics*, 7:433, 2006.

[17] Stefan Kurtz, Adam Phillippy, Arthur L. Delcher, Michael Smoot, Martin Shumway, Corina Antonescu, and Steven L. Salzberg. Versatile and open software for comparing large genomes. *Genome biology*, 5(2):R12, 2004.

[18] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[19] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman. *Readings in information visualization: Using vision to think*. Morgan Kaufmann Publishers, Inc., 1999.

[20] Eric C. Rouchka. Dynamic programming.
http://www.sbc.su.se/∼pjk/molbioinfo2001/dynprog/dynamic.html.

[21] The Arabidopsis Genome Initiative. Analysis of the genome sequence of the flowering plant Arabidopsis thaliana. *Nature*, 408:796–815, December 2000.

[22] Arabidopsis thaliana Genome Project. Segmental genome duplications of Arabidopsis, 2007.
http://www.tigr.org/tdb/e2k1/ath1/Arabidopsis_genome_duplication.shtml.

[23] FlyBase.
http://flybase.net/cgi-bin/gbrowse/dmel/.

[24] Martin Krzywinski. Circos, 2007.
http://mkweb.bcgsc.ca/circos/.

[25] A. Kozik, E. Kochetkova, and R. Michelmore. Genomepixelizer - a visualization program for comparative genomics within and between species. *Bioinformatics*, 18(2):335–336, 2002.
http://www.atgc.org/GenomePixelizer/.

[26] Tim J. Carver, Kim M. Rutherford, Matthew Berriman, Marie-Adele Rajandream, Barclay G. Barrell, and Julian Parkhill. Act: the artemis comparison tool. *Bioinformatics*, 21(16):3422–3423, 2005.
http://www.sanger.ac.uk/Software/ACT.

[27] Rhazes Spell, Rachael Brady, and Fred Dietrich. Bard: A visualization tool for biological sequence analysis. In *INFOVIS*. IEEE Computer Society, 2003.

[28] Heiko Schoof, Paolo Zaccaria, Heidrun Gundlach, Kai Lemcke, Stephen Rudd, Grigory Kolesov, Ronald Arnold, H.W. Mewes, and Klaus F.X. Mayer. MIPS Arabidopsis thaliana database (MAtDB): an integrated biological knowledge resource based on the first complete

plant genome. *Nucleic Acids Research*, 30(1):91–93, 2002.
http://mips.gsf.de/proj/thal/db/gv/rv/viewer.html.

[29] Chris Mayor, Michael Brudno, Jordy R. Schwartz, Alexander Poliakov, Edward M. Rubin, Kelly A. Frazer, Lior S. Pachter, and Inna Dubchak. Vista: visualizing global dna sequence alignments of arbitrary length. *Bioinformatics*, 16(11):1046–1047, 2000.

[30] Xiaokang Pan, Lincoln Stein, and Volker Brendel. Synbrowse: a synteny browser for comparative sequence analysis. *Bioinformatics*, 21(17):3461–3468, 2005.

[31] Lincoln D. Stein, Christopher Mungall, ShengQiang Shu, Michael Caudy, Marco Mangone, Allen Day, Elizabeth Nickerson, Jason E. Stajich, Todd W. Harris, Adrian Arva, and Suzanna Lewis. The generic genome browser: A building block for a model organism system database. *Genome Research*, 12(10):1599–1610, October 2002.

[32] Martin Wattenberg. Arc diagrams: Visualizing structure in strings. In *INFOVIS '02: Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, page 110, Washington, DC, USA, 2002. IEEE Computer Society.

[33] Danny Holten. Treecomparer. Unpublished, usage with explicit permission of Danny Holten., 2007.

[34] Danny Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

[35] Hee-Jeong Jin, Hye-Jung Kim, Jeong-Hyeon Choi, and Hwan-Gue Cho. Alignscope: A visual mining tool for gene team finding with whole genome alignment. *Advances in Bioinformatics & Computational Biology*, February 2006.

[36] Stephen C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, September 1967.

[37] Klaas Vandepoele, Yvan Saeysa, Cedric Simillion, Jeroen Raes, and Yves Van de Peer. The automatic detection of homologous regions (adhore) and its application to microcolinearity between Arabidopsis and rice. *Genome Research*, 12(11):1792–1801, November 2002.

[38] Inc. Resampling Stats. Hierarchical clustering.
http://www.resample.com/xlminer/help/HClst/HClst_intro.htm.

[39] Jarke J. van Wijk and Huub van de Wetering. Cushion treemaps: Visualization of hierarchical information. In G. Wills and D. Keim, editors, *Proceedings 1999 IEEE Symposium on Information Visualization*, pages 73–78, 1999.

[40] Jimmy Johansson, Patric Ljung, Mikael Jern, and Matthew Cooper. Revealing structure within clustered parallel coordinates displays. *Proceedings of the 2005 IEEE Symposium on Information Visualization*, pages 125–132, October 2005.

[41] T. Alan Keahey. *Nonlinear magnification*. PhD thesis, Indiana University, 1997.

[42] Jarke J. van Wijk. Views on visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):421–432, July/August 2006.

[43] SequoiaView.
http://www.win.tue.nl/sequoiaview.

[44] Tim Peeters, Mark Fiers, Huub van de Wetering, Jan-Peter Nap, and Jarke J. van Wijk. Case study: Visualization of annotated dna sequences. In O. Deussen, C. Hansen, D. A. Keim, and D. Saupe, editors, *VisSym*, pages 109–114. Eurographics Association, 2004.

[45] Mark W. E. J. Fiers, Huub Van De Wetering, Tim H. J. M. Peeters, Jarke J. Van Wijk, and Jan-Peter Nap. Dnavis: interactive visualization of comparative genome annotations. *Bioinformatics*, 22(3):354–355, 2006.

[46] NetBeans Platform.
http://www.netbeans.org/products/platform/.

[47] BioJava Project.
http://www.biojava.org.

[48] The Sequence Ontology project. Generic feature format version 3.
http://www.sequenceontology.org/gff3.shtml.

[49] The Sequence Ontology project.
http://www.sequenceontology.org/.

[50] Lincoln Stein. Generic genome browser: A tutorial.
*http://gmod.cvs.sourceforge.net/*checkout*/gmod/Generic-Genome-Browser/docs/tutorial/tutorial.html#aggregators*.

[51] Opengl api documentation overview.
http://www.opengl.org/documentation/.

[52] NVidia. Simple geometry program.
http://developer.download.nvidia.com/SDK/10/Samples/simple_geometry_program.zip.

[53] Carol Leth Stone. *The basics of biology*. Basics of the hard sciences. Greenwood Press, 2004.

[54] Mariana Ruiz.
http://en.wikipedia.org/wiki/Cell_(biology).

[55] National Human Genome Research Institute. Talking glossary of genetics.
http://www.genome.gov/.

[56] Francis Crick. Central dogma of molecular biology. *Nature*, 227:561–563, August 1970.

[57] Java Development Kit 6.
http://jdk6.dev.java.net/.

[58] JOGL: Java bindings for OpenGL.
http://jogl.dev.java.net/.

[59] OpenGL.
http://www.opengl.org/.

[60] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: elements of reusable object-oriented software*. Addison-Wesley, 1995.

[61] National Center for Biotechnology Information. Fasta format description. *http://www.ncbi.nlm.nih.gov/blast/fasta.shtml*.

[62] Peter Linz. *An introduction to Formal Languages and Automata*. Jones and Bartlett Publishers, Inc., third edition, 2001.