A Knowledge-Based Framework for the Alignment of Prokaryotic Genomes

von

Tom H. Wetjen

Dissertation

zur Erlangung des Grades eines Doktors
der Ingenieurwissenschaften
- Dr. Ing. -

Vorgelegt im Fachbereich 3 (Mathematik und Informatik)
der Universität Bremen
im Februar 2005

Datum des Promotionskolloquiums: 21.06.2005

Gutachter:
Prof. Dr. Otthein Herzog (Universität Bremen)
Prof. Dr. Jens Stoye (Universität Bielefeld)

Für Katja und Johanne

Danksagung

Ich möchte mich an dieser Stelle bei allen bedanken, die mich beruflich oder privat bei der Durchführung dieser Arbeit unterstützt und bestätigt haben. Die vorliegende Arbeit entstand im Rahmen meiner Tätigkeit als Doktorand und wissenschaftlicher Mitarbeiter am Technologie-Zentrum Informatik an der Universität Bremen.

Mein besonderer und sehr herzlicher Dank gilt meinem Doktorvater Prof. Dr. Otthein Herzog (Universität Bremen) für seine uneingeschränkte Unterstützung dieser Arbeit, seine Diskussionsbereitschaft, seine Hinweise und Anregungen und geduldige Betreuung. Genauso herzlich möchte ich mich bei Prof. Dr. Jens Stoye (Universität Bielefeld) für die vielen und langen anregenden Diskussionen bedanken. Die Diskussionen, Anmerkungen und Kritiken von beiden haben erheblich zur Qualität der Arbeit beigetragen. Ferner Danke ich beiden für die Übernahme der Begutachtung. Auch möchte ich mich bei Prof. Dr. Christoph Schlieder (Universität Bamberg) bedanken, der mich bei der Auswahl der Techniken zum räumlichen Schließen mit unterstützt hat.

Ich danke ebenfalls allen jetzigen und ehemaligen Kolleginnen und Kollegen der Arbeitsgruppe Künstliche Intelligenz für das tolle Arbeitsumfeld. Besonderer Dank gilt dabei, Thomas Wagner, Renzo Kottmann, Dr. Björn Gottfried, Sebastian Hübner und Dr. Holger Wache für ihre Neugier und Diskussionsbereitschaft, sowie Dr. Uta Bohnebeck, die mich in die Bioinformatik eingearbeitet hat und Priv. Doz. Dr. Ubbo Visser, der mir den Einstieg in die Informatik ermöglicht hat. Unserem Techniker Mathias Unger danke ich für die Unterstützung bei vielen technischen und organisatorischen Problemen. Ganz besonders möchte ich mich auch bei Renate Post-González für die orthograpische und idiomatische Überarbeitung bedanken.

Ganz besonders möchte ich auch meiner ganzen Familie herzlich für ihren Rückhalt und die Bestätigung danken, insbesondere meinen Eltern und Schwiegereltern. Der größte Dank gilt aber meiner Frau Katja ohne deren Verständnis und Beistand diese Arbeit nicht möglich gewesen wäre sowie meiner Tochter Johanne für ihr sonniges Gemüt ('Appelé').

Tom Wetjen

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation and Objective

Since the sequencing of the entire human genome (*Homo sapiens*) was completed in April 2000, researchers in molecular biology are discussing the beginning of the *post*-genomic era. Prior to this, the ambition of molecular biology and of bioinformatics was the sequencing, processing, and quality assurance of genomic sequence data. To-date, however, not only the human genome has been sequenced, but also that of other eukaryotic organisms like mouse (*Mus musculus*), rat (*Rattus norvegicus*), yeast (*Saccharomyces cerevisiae*), a nematode (*Caenorhabditis elegans*), a fruitfly (*Drosophila melanogaster*), and of some higher plants (e.g., thale cress *Arabidopsis thaliana*, and rice *Oryza sativa japonica*). Furthermore, about 216 bacteria (e.g., *Escherichia coli*), and 22 archaea (e.g., *Methanococcus jannaschii*) have been sequenced so far (`http://www.ncbi.nlm.nih.gov/genomes/Complete.html`).[1]

As scientists and companies are furthermore interested in specific parts of these genomes (e.g., in genes coding for a disease), there exists in addition to the sequence data of the complete genomes a vast amount of data of genome fragments. The raw sequence data is stored in data bases accessible to the public, e.g., at the National Center for Biotechnology Information (NCBI, data base GenBank (Benson et al., 2003)), or the European Bioinformatics Institute (EBI, data base EMBL (Stoesser et al., 2003)). In the past decade, the sequence data stored in e.g., GenBank increased exponentially (Figure 1.1), thus giving an idea of the potential it is awarded by scientists and industry in the field of biotechnology.

Apart from the effort invested in this sequencing and interpreting of the human genome, work on prokaryotes (bacteria and archaea) receives a lot of attention. In comparison to higher organisms, prokaryotes usually have small genomes (e.g., the bacteria *Escherichia coli* is about 4.5 million nucleotides long whereas the human genome is about 3.5 billion long), thus allowing the sequencing of

[1] At the 02/15/2005.

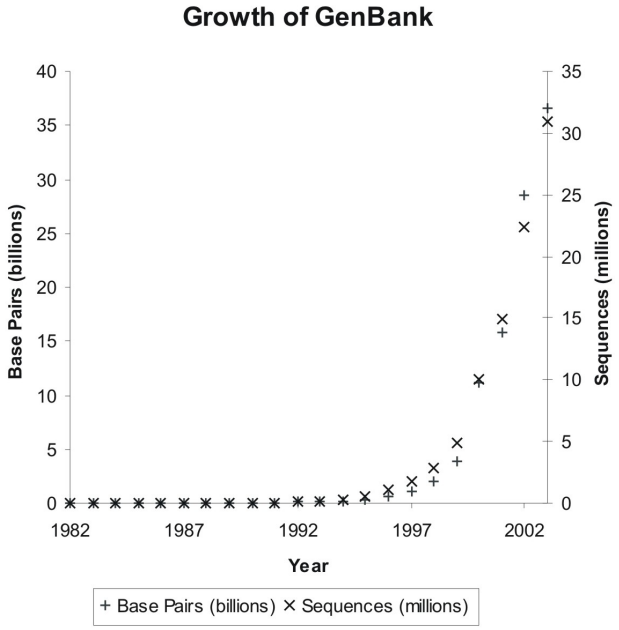


Figure 1.1: The increase in sequence data in GenBank between 1982 to 2003 (Figure was adapted from `http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html`, at the 09/06/2004).

their genomes in relatively short time. Prokaryotes play an important rule in many aspects of human life. They are pathogens (e.g., meningitis, caused by the bacteria *Haemophilus influenzae*) as well as, on the other hand, useful for food production (e.g., in the dairy industry *Lactococcus lactis* for cheese production). Moreover, prokaryotes are involved in almost all nutrient cycles of the ecosystems of our planet. Understanding the metabolism of prokaryotes is consequently essential to a better control of diseases, food production, and the understanding of our environment (Buckley, 2004). In addition, prokaryotes were probably the first organisms to appear on earth, and comparing their genomes can shed light on the processes which have taken place during the evolution. Thus, sequencing and comparing the genomes of prokaryotes have become an important field in biological science during the last decades. The additional task of bioinformatics in the *post*-genomic era is the interpretation of the sequence data of these genomes, in particular, the comparison of newly sequenced genomes to sequences for which knowledge exists in order to predict biological function and phylogenetic relationships.

The comparison of biological sequences is one of the oldest problems in bioinformatics, and "*early work on the problem resulted in what were arguably the first highly successful and widely adopted applications of computer science to biology*"

(Bray et al., 2003), p. 97. The basic technique for comparing sequences is the sequence alignment which is a way of placing one sequence above the other in order to illustrate the correspondence between similar subsequences of the sequences. The interpretation of the results is based on the central dogma of molecular biology which determines the flow of information from the DNA to the RNA to the protein. In other words, the sequence of nucleotides within the DNA codes for the structure of the protein-molecule which in turn determines biological function. Thus, similar sequences are assumed to have similar biological functions. Moreover, the predominant opinion in molecular biology assumes that sequence similarity between different species is caused by speciation from a common ancestor during evolution and thus can be used for phylogenetic reconstructions.

Owing to the rapidly growing amount of complete prokaryotic genomes publicly available, interspecies comparisons of genomes have now become possible. The task of the alignment is the identification of both coding (i.e., genes) and regulatory regions (the 'on/off switches' of genes) by comparing a newly sequenced genome to a genome knowledge already existing (Bansal et al., 1998). Furthermore, interspecies genome comparisons can find an answer to the question regarding the change of genomes in the course of time, and this will help further refining the phylogenetic tree (Overbeek et al., 1994; Mira et al., 2003). In detail, this includes the identification of the function and the evolution of gene families, the rate of gene conservation, variations of gene functionality in various organisms, and mechanisms of evolution (Bansal et al., 1998; Frazer et al., 2000; Frazer et al., 2003). In addition, information received from gene cluster analysis will help identifying functionally related genes and thus support the understanding of metabolic pathways (Bansal et al., 1998; Overbeek et al., 1998). All these analysis tasks require the comparison of species at different evolutionary distances (Frazer et al., 2003).

Existing comparative genomic tools at hand search for local similarities of the genome sequences and align the identified subsequences. This kind of procedure is necessary since a global (end-to-end) alignment strategy would align unrelated regions for the frequent case of genome rearrangements (i.e., permutation of subsequences), gene duplications, gene acquisitions, and gene losses shaping the genome during evolution (Mira et al., 2003; Snel et al., 2002). In order to avoid inconsistencies in the resulting alignment caused by wrong subsequence assignments (i.e., *false positive* hits), comparative genomic tools usually take only the subsequences with a local similarity value above a usually high threshold. In the case of similarities showing different relative orders between two genomes, the strongest similarity (i.e., the longest similarity with the highest value) is normally kept (Roytberg et al., 2002). Accordingly, all software-tools at hand require that the genomes to be aligned are phylogenetic closely related, i.e., that only a few rearrangements have occurred and that biologically corresponding regions within the genome are collinear (Miller, 2001; Chain et al., 2003). When comparisons are made among species (or even within some species) which are

non-collinear, however, this limits the possibilities for the identification of less conserved genomic features (Miller, 2001; Chain et al., 2003). Consequently, local similarities reflecting a true phylogenetic relationship (i.e., *true positive* hits) will be missed by these approaches. Miller (Miller, 2001) and Chain *et al.* (Chain et al., 2003) described the need to estimate the reliability of each region within a computed alignment to gain higher accuracy (i.e., increasing the specificity) and more information (i.e., increasing the sensitivity) perhaps at the cost of increased computational time.

In addition to the strength of local similarities, their biological context (e.g., their occurrence in a conserved gene cluster) could be used in order to support their biological evaluation. The motivation to introduce knowledge of genome architecture into the procedure of the genome alignment is that there act evolutionary pressures on the genome which do not allow for all possible permutations of its genomic features. Thus, there exist a number of regularities in prokaryotic genomes which increments with increasing phylogenetic relationship of the species. Rogozin *et al.* (Rogozin et al., 2002a) emphasized to go beyond straightforward genome alignment or local similarity search and to introduce the gene order into this analysis.

Altogether, this would allow:

1. to lower the threshold of the local similarity value for the comparison which in turn could make the identification of less conserved genomic features possible (i.e., increasing the sensitivity of the comparison), and

2. to align non-collinear genomes, i.e., genomes of phylogenetic further distanced species.

In this work we will discuss the following proposition:

Proposition 1.1.1 *Using the biological context of local similarities found between genomic sequences can support their biologically feasible assignment for interspecies genome alignments.*

In other words, the main objective of the knowledge-based alignment approach is to maximize the number of letters aligned (coverage) between the two genomes by increasing the sensitivity without losing specificity.

1.2 Structure of the Thesis

This thesis is structured as follows:

Chapter 2 explains the basic structure of biological sequences and their mechanisms of information encoding. Furthermore, the structure of prokaryotic genomes (i.e., genome sizes, geometry, and gene order) and their contents

(genomic features like genes and regulatory regions) are introduced. It also describes the evolutionary forces shaping prokaryotic genomes over time. Altogether, the biological background for understanding the concepts and ideas of the following Chapters is given with respect to the requirements to a modeling technique of prokaryotic genome architecture in a computer which is able to support the alignment procedure.

Chapter 3 is concerned with the basis of the computational sequence alignment. The basic concepts like sequence similarity and distance as well as the fundamental algorithms for a sequence comparison are briefly explained. In the following, the problems of the alignment of whole prokaryotic genomes is discussed before the state of the art of the existing genome alignment approaches is given.

Chapter 4 introduces the concepts and algorithms of the knowledge-based genome alignment. After introducing the principle idea of the alignment approach, we motivate the selection of a representation language for genome structures with respect to the domain concepts introduced in Chapter 2. Afterwards, the representation and its reasoning capability is described in detail. Furthermore, the assembling of local similarities to putative genomic features is explained. Eventually, different approaches for their evaluation are given in detail.

Chapter 5 describes in detail the approach to discover common genome architecture in a set of prokaryotic genomes.

Chapter 6 describes the prototypical implementation of the described concepts and algorithms. The system architecture (design and data models) are explained as well as the application of the system.

Chapter 7 compares and discusses the quality of genome alignments of prokaryotic genomes generated by the knowledge-based genome alignment and other approaches. Furthermore, the benefits and limitations of the approaches are discussed.

Chapter 8 gives a summary of the thesis and an outlook for future work.

Chapter 2

Architecture of Prokaryotic Genomes

Introduction of the biological basics, starting with the basic structure and functions of biological sequences like DNA, before the content and architecture of prokaryotic genomes is discussed. Eventually, evolutionary forces shaping prokaryotic genomes over time are described. This biological knowledge constitutes the general conditions of the computational techniques applicable for the representation of prokaryotic genome architecture.

Prokaryotes are small, unicellular organisms which lack the membrane-bound nuclei of eukaryotes and thus their DNA is free floating in the cell. Furthermore, their DNA is not bound to proteins as it is in eukaryotes and they lack organelles like chloroplasts or mitochondria. Furthermore, all metabolic reactions take place in the cytoplasm. They reproduce usually by binary fission, i.e., their DNA molecule is replicated and then the cell splits into two identical cells, each containing an exact copy of the original cell's DNA. Prokaryotes emerged at least 3.5 billion years ago and are arguably the oldest life forms on earth. Prokaryotes are divided into two phylogenetic groups, Archaea and (Eu)bacteria, belonging to the kingdom Monera.[1]

2.1 Structure of Biological Sequences

Molecular biological sequences are divided into two types based upon the molecules building up the sequence. Nucleic acid chains form the macromolecules deoxyribonucleic acid (DNA) and ribonucleic acid (RNA), whereas amino acid chains

[1]There are several theories about the exact phylogenetic relationship (what was derived from what) of archaea, eukaryotes, and eubacteria (for further discussion see e.g., the Tree-of-life at `http://tolweb.org/tree/` or for a more detailed overview (Doolittle, 2000)).

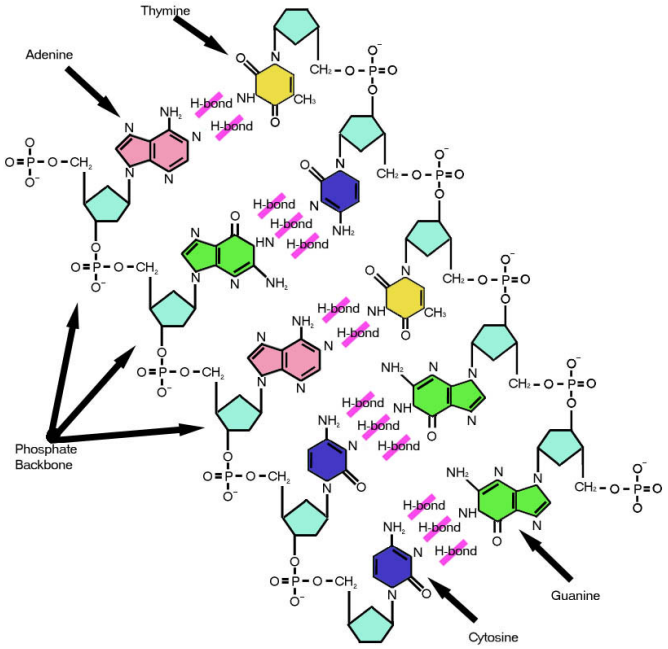


Figure 2.1: A schematic structure of the DNA molecule (Figure was taken from `http://en.wikipedia.org/wiki/Image:DNA_labels.jpg`, at the 02/15/2005).

form the proteins. The following excursion on sequence structures and functions follows mainly Alberts *et al.* (Alberts et al., 1994).

The DNA is a threadlike molecule consisting of many deoxyribonucleotides forming a double helix. A deoxyribonucleotide itself is built up by a sugar molecule (2'-deoxyribose), a phosphate residue, and a base (nucleotide). The sugar molecule attached to the phosphate residue forms the backbone unit of the DNA molecule, and the nucleotides code the genetic information. The sugar molecule contains five carbon atoms, labelled from 1' to 5'. The chemical bond that creates the backbone is between the 3' carbon of one unit, the phosphate residue, and the 5' carbon of the next unit. Thus, DNA molecules have an orientation, which by convention, starts at the 5' end and finishes at the 3' end (5' to 3' direction). Attached to each 1' carbon by a chemical bond is one nucleotide (Figure 2.1).

There exist four different bases: adenine (A), cytosine (C), guanine (G), and thymine (T). This nitrogenous bases are derivates of either purine or pyrimidine and thus can be classified into adenine and guanine as purines, and cytosine and thymine as pyrimidines. DNA molecules are double stranded, consisting of two single strands of the described deoxyribonucleotide strands. The two strands are tied together via hydrogen bonds (a bond between polarized and adjacent H and O-atom or H and N-atom) between two nucleotides, which is called a base pair. Base A is always paired with base T, and C is always paired with G (complemen-

tary base pairs)(Figure 2.1). This is called the Watson-Crick base pairing, after James Watson and Francis Crick who discovered these pairings in 1953 (Watson and Crick, 1953). Base pairs (bp) or sometimes kilo base pairs (kbp, corr. 1000 bp) provide one unit of length of DNA molecules. In 3-dimensional space the DNA forms a helical structure, i.e., the molecule winds in a spiral. However, the 3-dimensional structure is of no importance for our further discussion.

The RNA molecule is much like the DNA molecule, with some compositional and structural differences. The sugar in the RNA is ribose instead of 2'-deoxyribose, the nucleotide thymine (T) is replaced by uracil (U), which bonds as well to adenine (A), and RNA does not form a double helix, it is single-stranded.

In order to synthesize proteins, subsequences of the DNA acting as a template are transcribed (transcription) into complementary single-stranded messenger-RNAs (mRNA) which subsequently are translated (translation) to amino acid chains forming the proteins (Figure 2.4). This is called the molecular biological dogma stating that the structure and thus the function of proteins is exclusively encoded by the sequence of nucleotides in the DNA. The transcription of the DNA is catalyzed by the enzyme RNA polymerase. The RNA polymerase molecules in a cell collide randomly with the DNA, sliding along it and, if contacting a promotor region (see 2.2.2.1), bind very tightly to the DNA. Accordingly, a short stretch of the DNA double helix is opened by the enzyme and the strand acting as a template is 'read' (3' to 5' direction) in single nucleotides by the RNA polymerase synthesizing a mRNA chain in the 5' to 3' direction until a terminator region (see 2.2.2.1) is reached. This causes the RNA polymerase to release both, the DNA template and the newly synthesized mRNA. By convention, the non-template strand of the DNA in the 5' to 3' direction is called a gene sequence, since this strand corresponds to the mRNA sequence that is made.

The rules by which the mRNA sequence is translated into amino acid chains is called the genetic code. In the process, three consecutive nucleotides, called a codon or base-triplet, specify one amino acid. Since there exist 4 different nucleotides in the RNA (DNA), there are $4^3 = 64$ possible codons (Figure 2.2). However, proteins commonly contain 20 different amino acids, so that most amino acids are coded by several codons. This is called the degeneration of the genetic code. In particular, the exchange of a nucleotide at the third position within a codon can change the coding of the amino acid. Therefore, this position is called the wobble position. The start- and the stop-position of the translation are encoded by a start-codon (AUG) which codes as well for the amino acid methylene and one of three possible stop-codons (UAA, UAG, or UGA). In (Eu)bacteria alternative start-codons (GUG, UUG, AUU, CUG) have been observed. The start-codon determines the reading frame, i.e., the phase where nucleotides are read in sets of three. The generated proteins carry out different functions within a cell, part of which are: enzymes, transporters, linkers, and receptors.

1st position (5' end)	2nd position U	2nd position C	2nd position A	2nd position G	3rd position (3' end)
U	Phe	Ser	Tyr	Cys	U
	Phe	Ser	Tyr	Cys	C
	Leu	Ser	STOP	STOP	A
	Leu	Ser	STOP	Trp	G
C	Leu	Pro	His	Arg	U
	Leu	Pro	His	Arg	C
	Leu	Pro	Gln	Arg	A
	Leu	Pro	Gln	Arg	G
A	Ile	Thr	Asn	Ser	U
	Ile	Thr	Asn	Ser	C
	Ile	Thr	Lys	Arg	A
	Met	Thr	Lys	Arg	G
G	Val	Ala	Asp	Gly	U
	Val	Ala	Asp	Gly	C
	Val	Ala	Glu	Gly	A
	Val	Ala	Glu	Gly	G

Figure 2.2: The standard genetic code (Figure was taken from (Alberts et al., 1994), p. 106, Figure 3-16).

2.2 Structure of Prokaryotic Genomes

The genome is all the DNA contained in an organism or a cell and the genomic sequence codes for all the hereditary characteristics of an organism. In prokaryotes, the genome is often partitioned to one chromosome (sometimes called replicon) and possibly one to many plasmids (sometimes called extrachromosomal elements, or small chromosomes). The definitions of these terms have become fuzzy as new paradigms have emerged in the last years (Casjens, 1998). However, throughout this thesis we will always refer to the genome as the chromosome of prokaryotes.

2.2.1 Genome Size and Geometry

The genome size of prokaryotes can range from 580 kbp for *Mycoplasma genitalium* up to 9200 kbp for *Myxococcus xanthus*, the smallest and largest genomes known so far. Thus, genome size can differ over a tenfold range. If this range of genome size is compared to members of other kingdoms of life it can be seen that the size of prokaryotic genomes overlap the largest viruses and the smallest eukaryotes (Figure 2.3)(Shimkets, 1998; Casjens, 1998).

The size of the chromosome can be highly variable even within the same

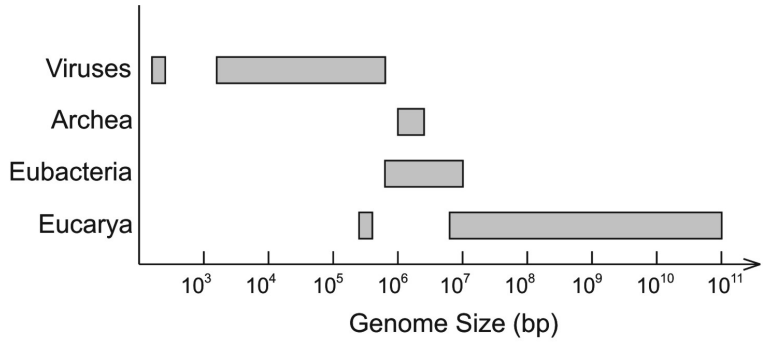


Figure 2.3: Known genome size range for extant life forms on earth. (Figure was modified from (Casjens, 1998), p. 344, Figure 2)

genus (e.g., *M. genitalium*, 580 kbp, and *M. mycoides*, 1350 kbp). However, this variability, although apparently typical, is not universal (e.g., the *Borrelia* species studied so far differ only by 15 kbp in size) (Casjens, 1998). Most chromosomes are circular in shape. However, an increasing number of linear chromosomes have been identified during the last years (e.g., in *Borrelia*), and there exist at least two types of chromosome linearity, which at present lead to the suggestion that linearity arose twice from circular progenitors (Casjens, 1998). As already mentioned, most prokaryotes contain a single large chromosome and, in addition, some plasmids. Although, these plasmids are not universally present they appear to be very common.

2.2.2 Genome Content

In general, the biological functions coded in the genome can be divided into genes (regions coding for a protein or a RNA) and regulatory regions (regions enable molecules to attach and, thus, to regulate gene expression). Furthermore, there exist some structures of higher levels combining more basic functions into an overall concept. In the following we will use the term *genomic feature* to describe all DNA regions of a complete genomic DNA sequence which carry a biological function.

2.2.2.1 Genomic Features

We will briefly browse through the main genomic features - their definitions, structures, and functions. However, this is not a comprehensive survey of all genomic features known so far, since this would go beyond the scope of this thesis.

Gene: A gene is the fundamental physical and functional unit of heredity. It is an ordered sequence of nucleotides located in a particular position on a

particular chromosome or plasmid that encodes a specific functional product (i.e., a protein or RNA molecule).

Protein Gene: Genes coding for proteins (via mRNA) consist of an open reading frame (orf) which in turn consists of codons coding for the amino acid sequence of the protein, a start-codon, and a stop-codon which trigger the beginning and the termination of the translation, respectively (Figure 2.4).

RNA Gene: An RNA gene is a DNA region which codes for a RNA product other than mRNA.

rRNA Gene: A DNA region that codes for a ribosomal RNA (rRNA) molecule, i.e., a RNA component of the ribosome which assembles amino acids into proteins.

tRNA Gene: A DNA region that codes for a transfer RNA (tRNA) molecule (about 75-85 bp long) which mediates the translation of nucleic acid sequences into an amino acid sequence.

Operon: A set of genes transcribed under the control (co-regulation) of an operator gene is named operon. The adjacent genes combined within an operon are co-expressed and normally encode functionally related proteins or are transcribed to a single transcriptional unit (Figure 2.5). If the genes are transcribed to the same mRNA transcript, the mRNA is considered polycistronic.

Regulon: Regulons are global regulation systems consisting of either a set of genes or operons which can be spatially separated within the genome, but are regulated in a coordinated fashion by a common regulator gene (protein).

Regulatory Region: A regulatory region is a DNA region that controls (activating or repressing) gene expression.

Promotor: The promotor is the DNA site that RNA polymerase binds to and initiates the transcription. It consists of two important subregions which are highly conserved throughout different species.

-10-region: The -10-region (a.k.a. Pribnow box) is a conserved region located about 10 bp upstream of the initiation site of transcription and is involved in binding the RNA polymerase (consensus = TAtAaT)[2].

-35-region: The -35-region is a conserved hexamer (region of 10 bp) located about 35 bp upstream of the initiation site of transcription and is involved in binding the RNA polymerase (consensus = TTGACa or TGTTGACA).

[2]lower letters indicate sites of variability

Upstream activating site: The upstream activating site (UAS) is a DNA sequence that regulates protein gene expression by increasing its transcription into mRNA. UASs are located upstream of the promotor.

Transcription factor binding site: The transcription factor binding site (TFS) is a conserved DNA region which helps to control gene expression. It is located upstream of a protein gene and often upstream of the promotor a protein binds to. TFSs often control regulons.

Terminator: The terminator region is a subregion of the transcribed mRNA which forms a 3-dimensional structure called hairpin loop. This structure causes the RNA polymerase to release the newly produced mRNA molecule, i.e., terminates the transcription.

Operator: The operator is a short conserved region which is recognized by a gene regulatory protein (repressor). It overlaps or is contained in the promotor region. If the repressor protein binds to the operator, the promotor region is blocked for the RNA polymerase and transcription cannot take place. The level of expression can either be stimulated or repressed, depending on the ecological pressure (e.g., nutrient availability) acting on the organism.

Ribosomal Binding Site: The ribosomal binding site (RBS) is the mRNA (DNA) subregion where the ribosome binds to the mRNA to initiate translation.

Attenuator: The attenuator is a DNA region located between the promotor and the first protein gene in prokaryotic operons. It regulates the level of expression of some operons by a partial termination of transcription.

Repeats: Repeats (repetitive sequences) are sequences of varying lengths which occur in multiple copies in the genome. There exist various kinds of repeats (e.g., tandem repeats, insertion sequences, and inverted repeats) in prokaryotes, all caused by different evolutionary forces (see 2.3). Repeats can bear a biological function as e.g., regulatory region like the terminator which is an inverted repeat.

The composition of genes and regulatory regions is not strict, but depends on the metabolic role (essential *vs.* ecological specific) of the gene. A protein gene - if not part of an operon - is always flanked upstream by a promotor and downstream by a terminator, however, depending on its function, further repressing or enhancing regulatory regions can occur upstream of the gene. This is as well the case for most operons know so far, but there is recently an increasing number of operons in which more than one promotor have been observed.

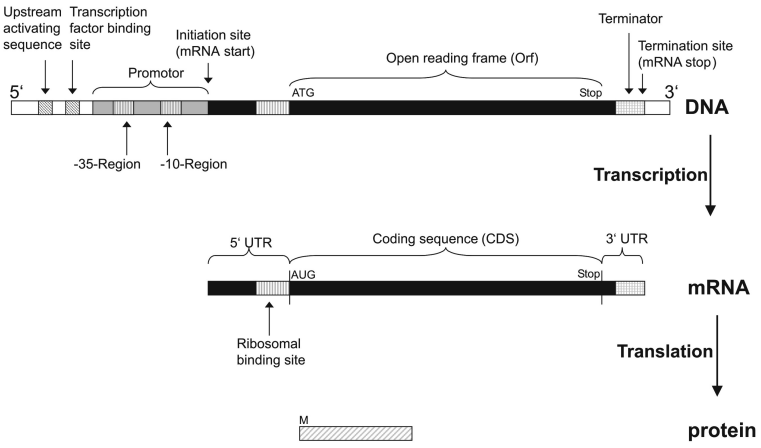


Figure 2.4: Schematic view on a typical gene and its regulatory sites in prokaryotic genomes. The lengths of genomic features shown are not in scale.

2.2.2.2 Evolutionary Origin of Genomic Features

If comparing genomes across species, the relationship of genomic features can further be characterized based on their origin, i.e., based on their evolutionary history. In general, genomic features can appear in a genome either because of divergent evolution (descent from a common ancestor) or because of convergent evolution (two previously unrelated genes which became similar by acquiring new, related functions). Although a few examples of a convergent evolution of genes in prokaryotes have been reported, most genomic features evolve by divergent evolution. Such genomic features are called homologs.

Homology: Homology is the relationship of a genomic feature related to a second genomic feature in different species by descent from a common ancestral DNA. Homology is either true or false (with respect to a given time in the past).

The term homology may apply to the relationship between genomic features separated by the event of speciation (orthology) or by the event of duplication (paralogy) (see 2.3.2). This assumes a single universal ancestor of all existing life forms. However, there is an ongoing discussion about the existence of this universal ancestor (for a more detailed discussion see e.g., (Woese, 1998) and (Doolittle, 2000)).

Orthology: Orthology is the relationship of any two homologous genomic features in different species that evolved from a common ancestral by speciation.

Paralogy: Paralogy is the relationship of any two homologous genomic features related by an ancestral duplication event.

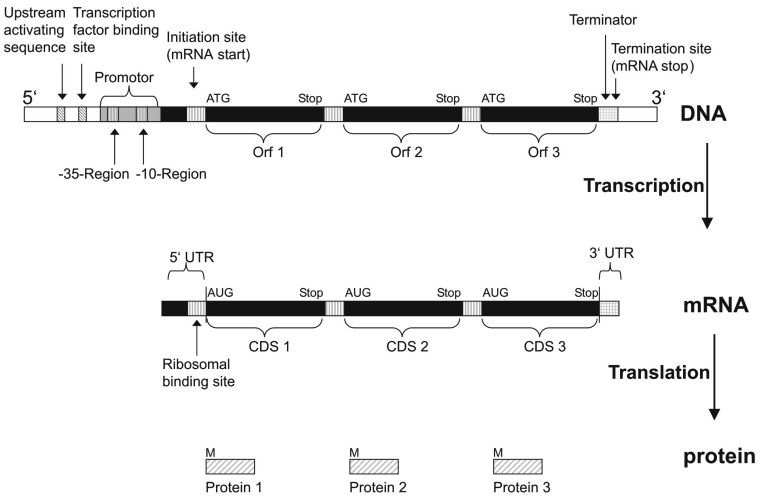


Figure 2.5: Schematic view on a typical operon structure in prokaryotic genomes. The lengths of genomic features shown are not in scale.

Furthermore, if the relationship of genomic features is based on a 'sexual' origin this is determined as xenology.

Xenology: Xenology is the relationship of any two genomic features (in particular genes) related by an ancestral event of a horizontal gene transfer (see 2.3.2).

Orthologous genomic features (so called orthologs), in particular protein genes, are the pertinent features to use for the reconstruction of phylogenetic trees. Paralogous and xenologous genomic features (so called paralogs or xenologs, respectively), in particular protein genes, are helpful for understanding the course of protein evolution (Zouine et al., 2002).

2.2.2.3 Gene Content

The average protein gene size for all prokaryotes sequenced so far is uniform, about 900 to 1000 bp, and genes appear to be similarly closely packed in prokaryotic genomes (Casjens, 1998). In correspondence to the described genome sizes (see 2.2.1), prokaryotic genomes have approx. 470 up to nearly 10,000 genes, and larger prokaryotic genomes have more genes than smaller ones, and it appears that the gene number reflects their ecological 'lifestyle' (Casjens, 1998; Mira et al., 2003). In general, prokaryotes with smaller genomes are specialized for an ecological niche (e.g., parasites like *M. genitalium* (Fukuda et al., 1999), or endosymbiontic prokaryotes like *Buchnera* sp. (Wixon, 2001)) whereas those with larger genomes are metabolic generalists (e.g., *M. xanthus*) (Casjens, 1998). Moreover, the degree of genome variability is to some extent related to

their 'lifestyle', because there is a trend that gene order is less conserved in larger genomes (Ermolaeva et al., 2001). In particular, parasites like *M. genitalium* show the highest degree of stability in comparison to free-living species which undergo frequent rearrangements and gene content variability (see 2.3.2) between and within species which is probably due to gene transfer (Mira et al., 2003). Large-scale studies on the presence and absence of genes have shown that the number of shared genes between genomes depends on the size of genomes and their evolutionary distance (Snel et al., 2002).

There is an evolutionary pressure towards smaller prokaryotic genomes (Mira et al., 2003). The reason is that smaller chromosomes can replicate faster, and prokaryotes with smaller genomes can grow faster, out-populating other prokaryotes with larger chromosomes. The colonization of a new environment by a prokaryotic species can lead to a shrinkage of its genome size if some genes contained in the genome are not required. For example, in an environment where a special amino acid is always provided, some members of the population will lose their ability to produce enzymes to synthesize this amino acid, e.g., through any mutation within a gene of an enzyme of its metabolic pathway (see 2.3). A gene which has lost its function is called pseudogene. After many generations, the ability to synthesize the amino acid is lost throughout the entire population and eventually each member will lose the complete DNA sequence coding for an enzyme for its metabolism. Evolution by deletion, however, is a very slow process (i.e., it takes many generations), in part to prevent prokaryotes from extinction by atrophy of the genome, but is thought to be one of the major evolutionary forces shaping prokaryotic genomes during evolution (Snel et al., 2002; Wolf et al., 2002; Mira et al., 2003)(see 2.3.2).

There exists a set of essential genes common to all prokaryotes consisting of RNA genes and protein genes. Taken together, they mainly code for ribosomal proteins and ribosomal RNA subunits, translation initiation factors, proteins associated with the ribosome or protein modification, or proteins and RNAs associated with transcription (tRNAs) and replication of DNA (Mushegian and Koonin, 1996; Harris et al., 2003). These essential genes are more evolutionarily conserved than are non-essential genes over both microevolutionary (having local mutations to a lesser extent, see 2.3.1) and macroevolutionary (having global mutations to a lesser extent, see 2.3.2) time scales (Jordan et al., 2002).

2.2.2.4 Gene Order and Clustering

The first comparisons of prokaryotic genomes of *Escherichia coli* and *Salmonella tryphimurium*, two close relatives which have been deviated only 120-160 million years ago, gave reasons towards the conclusion that gene order is highly conserved within closely related prokaryotes (Kolstø, 1997). Recent and more comprehensive genome comparisons, however, showed significant differences of genome architecture in closely related species and even in strains of the same

species (Andersson, 2000; Horimoto et al., 2001; Suyama and Bork, 2001), and there exists evidence that a substantial proportion of these gene rearrangements in gene order are caused by recombination sites (indicated by the positions of the replication forks) (Tillier and Collins, 2000). The usual spatial configuration observed so far for genes in prokaryotes is that they are disjoint (Bruijn et al., 1998; Casjens, 1998), however, recently an increasing number of intersections (i.e., overlapping genes) and joints (genes directly adjacent) likely to correspond to gene fusion events have been reported (Snel et al., 2000; Yanai et al., 2002).

Nevertheless, genes in prokaryotic genomes commonly form operons (see 2.2.2.1 and Figure 2.5). Operons observed so far have on average a length (number of genes included) which remains constant around three (Zheng et al., 2002). There exist two primary models for this operon formation, the co-regulation model and the Selfish operon model (Lawrence and Roth, 1996). The co-regulation model of operon formation claims that co-regulation of genes that are co-adaptive (i.e., functionally coupled) provide a selective advantage to species (Overbeek et al., 1999; Andersson and Eriksson, 2000). For example, genes coding for enzymes related to a particular metabolic pathway are often physically linked and co-transcribed. Moreover, genes in operons such as the *trp* or *his* operon in *E. coli* are linked according to the order of their deduced metabolic function (Andersson and Eriksson, 2000). However, there are some problems with the co-regulation model, namely, that almost all genes in prokaryotes essential for their survival are not found in operons (with some exceptions like the ribosomal operon).

The Selfish Operon model of Lawrence and Roth (Lawrence and Roth, 1996) claims why and how non-essential genes primarily form operons. The main argument of this model is that gene clustering facilitates the spread of functional related genes among organisms via horizontal gene transfer (see 2.3.2) (Lawrence and Roth, 1996; Andersson and Eriksson, 2000). Thus, approaches to find operons often searches for conserved gene clusters (see Chapter 5), where such gene clusters are usually considered as

Gene Cluster: A gene cluster is a set of at least two adjacent genes that occur in a required minimal number of genomes.

However, the concrete definition of gene clusters varies for such approaches sometimes requiring genes to be transcribed into the same direction or allowing for non-common genes to be included (i.e., common adjacent genes to be separated by a usually small number of genes). Such a cluster of genes from one organism can horizontally be transferred to another organism in one step, and therefore all these newly acquired genes are adjacent. Only some of these genes serve a purpose in the species which obtain them, i.e., give the species a new ability like the ability to synthesize a special amino acid. This amino acid related genes might be already adjacent or they could be spread out among the newly acquired genes. If the species moves to an environment lacking this amino acid, this bacterium is the only one of its kind able to synthesize it for its survival. Inside genes

without useful functions deletions will occur which may lead to the deletion of the complete genes in future generations of this bacterium. This leaves the genes required for the synthesis of the amino acid adjacent after several generations. The main difference to the co-regulation model is that co-transcription of the genes is not required in order to form gene clusters. The genes in an operon can, if not transcribed to the same mRNA transcript, be rearranged again (Lathe III et al., 2000).

Beyond the level of operons, however, gene order in prokaryotes is little conserved over the major phyla (Rogozin et al., 2002a) and tends to decrease with increasing genome size (Ermolaeva et al., 2001). In a study by Wolf *et al.* (Wolf et al., 2001) on 25 prokaryotic genomes, the total coverage of genomes by conserved gene strings ranged between <5% (for *Synechoxystis* sp.) and 24% (for *M. genitalium*), and the conserved gene strings consist mainly of known operons. Taken together, gene order conservation can be used as a genomic measure to study the phylogenetic relationships between prokaryotes (Tamames, 2001; Mira et al., 2003).

2.2.2.5 Non-coding DNA Content

Non-coding DNA regions are all those regions which do not code for either protein genes or RNA genes. In the majority of genomes of prokaryotes studied so far, 6% to 14% of the genome corresponds to non-coding DNA (Rogozin et al., 2002b). This non-coding regions are thought to typically contain regulatory regions like promotors, attenuators, and terminators etc. (see 2.2.2.1). The structure, in this case especially the curvature, of the DNA molecule at such non-coding DNA regions plays a well-characterized role in many regulation mechanisms like DNA replication, recombination, transposition, and transcriptional regulation and curvature for some region is significantly conserved across prokaryotic genomes (Jáurequi et al., 2003).

Non-coding regions can as well contain repeats (see 2.2.2.1) some of which are known to bear a biological function, whereas others are assumed to be non-coding 'junk' DNA. For example, inverted repeats are compatible with the formation of hairpin loops in the mRNA, thus leading to the termination of the translation (Lillo et al., 2002) (see 2.2.2.1). Furthermore, repeats provide the potential for altering and destabilizing the genome and thus can lead to global mutations (Weinstock and Lupski, 1998; Mira et al., 2003) (see 2.3.2). In particular, close repeats have been proven to generate duplications and large scale deletions and thus to change prokaryotic genomes in the evolution (Rocha, 2003).

As for gene content (see 2.2.2.3), there is an evolutionary pressure to retain the minimal amount of non-coding DNA regions in prokaryotic genomes that is essential for preserving regulatory signals (Rogozin et al., 2002b). In contrast to the evolution of protein genes, however, the evolution of non-coding DNA seems to evolve in a different regime (Rogozin et al., 2002b). The evolutionary pressure

can generate the overlap of genes in parasitic prokaryotes like *M. genitalium* and *M. pneumoniae*, mainly caused by the loss of a stop-codon (Fukuda et al., 1999). Furthermore, the deletion of the non-coding DNA between two genes can lead to the fusion of genes (which will then form one mRNA molecule during transcription) which in turn is evolutionary beneficial if genes are functionally coupled (Snel et al., 2000; Yanai et al., 2002). The number of gene fusion events increases with increasing genome size (Snel et al., 2000), and genes linked by fusion events are generally of the same functional category (Yanai et al., 2001).

2.3 Genome Evolution in Prokaryotes

There exist various mechanisms having an effect on prokaryotic genome architecture during evolution which are summarized under the term mutations. A mutation is a permanent structural alteration in the DNA. In most cases, DNA changes either have no effect or cause harm, but occasionally a mutation can improve an organism's chance of surviving and passing the beneficial change on to its descendants. Because of the reproduction of prokaryotes by binary fission, they mainly rely on these mutations in order to generate novel functions. Mutations can be caused by different chemical and biophysical processes changing the DNA during replication or repairing.

Mutations can be further classified into local and global, depending on their spatial expansion within the genome. Local mutations have to be considered when comparing short sequences of genomic features (e.g., single genes), whereas global mutations have to be taken into account when comparing either larger fragments (e.g., an operon) or complete genomic sequences.

We will take a closer look at the changes of genome architecture caused by mutations on the sequence and the structural level, however, we will skip the chemical and biophysical level, since they are of no importance to our discussions.

2.3.1 Local Mutations

Local mutations (sometimes referred to as point mutations) are changes of one or a few consecutive bases caused by errors during replication or repairing processes of the DNA. If occurring in a coding region of the DNA, these mutations can cause the change of a single or a few amino acids within the translated protein which then can cause the loss of the biological function of this protein. However, local mutations do not necessarily change the protein chain, since different codons can code for the same amino acid (see 2.2). If either occurring at the wobble position of a codon (silent mutation) or if the exchange of an amino acid is without the loss of the biological function (neutral mutation) then this mutation will have no further effect. Local mutations can be further divided into:

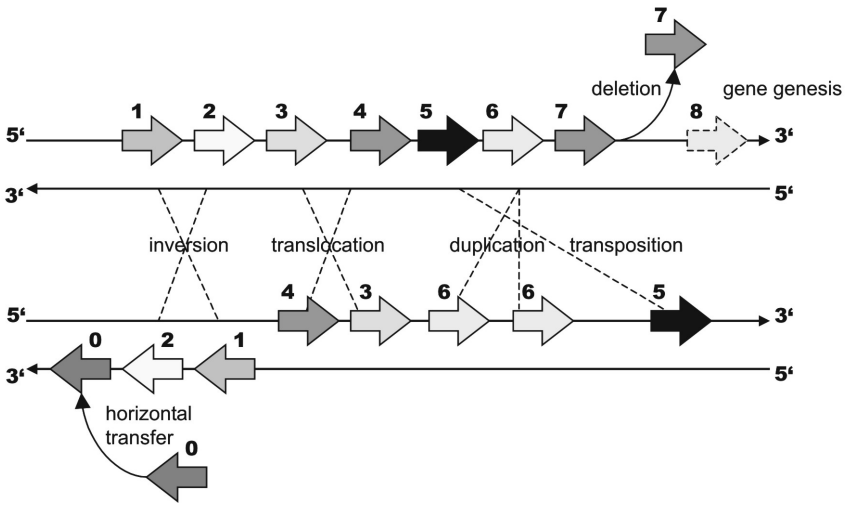


Figure 2.6: Schematic illustration of different global mutations and how they effect genome architecture (Figure was modified from (Andersson, 2000), p. 134, Figure 1)

Deletion: One or a few consecutive bases are removed from the sequence, e.g., ATG becomes AG if T is deleted.

Insertion: One or a few consecutive bases are introduced into the sequence, e.g., AG becomes ATG if T is inserted.

Substitution: An exchange of one or a few consecutive bases occurs in the sequence, e.g., ATG becomes AGG if T is substituted by G.

Local mutations do not directly affect the architecture (e.g., the gene order) of a genome. However, when the biological function of a region is lost through local mutations (e.g., reading frame shift through insertion or deletion which generates a so called pseudogene), the prokaryotes either dies out or its genome compensates for the loss which in turn can result in a change in genome architecture (see 2.2.2.3).

2.3.2 Global Mutations

In contrast to local mutations, global mutations do effect larger regions of the genome and can directly change genome architecture (e.g., by permutating genes).

Global mutations are caused by a crossover of DNA-segments (i.e., of the chromosome) during cell division. In addition, there exist DNA segments in prokaryotic genomes able of an 'active' translocation of DNA-regions. Such mobile genetic elements are Insertion Sequences (IS) and Transposons (sometimes referred to as 'jumping' genes). Furthermore, prokaryotes exhibit the mechanism

of horizontal gene transfer, which is an active way of distributing genes between organisms. Altogether, global mutations have the consequence of either the increase in DNA content, the decrease in DNA content, or the rearrangement of DNA segments which can affect complete genes or even operons. Global mutations (see Figure 2.6) can be classified into:

Decrease in Genome Size: Removal of subsequences of the genome caused by

Deletion: A subsequence is removed from the genome.

Deletions tend to occur more often than large insertions (e.g., by horizontal transfer) in prokaryotic genomes (Mira et al., 2003), or are at least balanced with the gain of subsequences in the genome (Kunin and Ouzounis, 2003). If the deleted subsequence contains a complete gene, this process is called gene loss or gene deletion. Gene loss is one of the major evolutionary forces shaping prokaryotic genomes (Snel et al., 2002; Wolf et al., 2002; Mira et al., 2003) and correlates, unlike other global mutational processes, fairly well with time (Snel et al., 2002). It has been shown that gene loss occurs up to three times more frequently than horizontal gene transfer and up to two times more frequently than gene genesis (Kunin and Ouzounis, 2003). If a deletion of non-coding DNA between two genes occurs this can cause gene fusion (Snel et al., 2000; Yanai et al., 2002).

Rearrangement within the Genome: The genome sequence is restructured by permutating subsequences. In general, it has been observed that genome rearrangements are often symmetrically organized around the origin and terminus of replication as a result of high recombination frequencies at the open replication forks (Tillier and Collins, 2000; Andersson, 2000). One of the following events can occur:

Inversion: The orientation (5' to 3') of a subsequence within a genome is changed (3' to 5') while it is transferred to the contrary strand.

Often single genes are reversed which in some species correlates to the presence of IS elements (Mira et al., 2003).

Transposition: A subsequence is transferred to a new position in the genome.

Translocation: Two subsequences within a genome are exchanged.

Like Inversions, often single genes are translocated correlate in some species to the presence of IS elements (Mira et al., 2003).

Increase in Genome Size: Extension of the genome can occur caused by:

Duplication: A subsequence of a genome is duplicated.

If the duplicated subsequence contains a complete genomic feature like a gene, this process is called gene duplication resulting in a paralogy (see 2.2.2.2). Paralogous genes play a major role in the evolution by generating new functions of proteins (Jordan et al., 2001; Kondrashov et al., 2002). For example, in *E. coli* 30% of the coding sequences could be grouped into gene families (i.e., cluster of genes with similar functions consisting mostly of paralogs), and such gene families contained two to 30 copies in nearly every prokaryotic genome (Pushker et al., 2004). Beside their role in protein evolution, recently gene copies are assumed to contribute to a proper gene dosage or to provide different specificities of similar chemical reactions
(Pushker et al., 2004).

Horizontal Gene Transfer: Horizontal gene transfer (a.k.a. lateral gene transfer) is the lateral transfer of genetic information among contemporary generations of organisms.

It occurs either by:

- conjugation, i.e., the direct transfer of a plasmid or a partial genome sequence via a pilus (a membrane tube between cells),
- transduction, i.e., transfer of genomic fragments via viral infections, or
- transformation, i.e., uptake and incorporation of DNA fragments from the environment (Jain et al., 2002).

There exists growing evidence that horizontal gene transfer is the primary route of acquiring new genes even between further distanced prokaryotic species (Nelson et al., 1999; Ochman et al., 2000; Mira et al., 2001; Jain et al., 2002). In addition, there exists first evidence that a large fraction of pseudogenes in prokaryotes occur because of failed horizontal gene transfer events (Liu et al., 2004). Genes obtained through horizontal gene transfer are classified as xenologous (see 2.2.2.2). Horizontal gene transfer could provide gene families with members already divergent in sequence and function (Lawrence and Hendrickson, 2003).

Gene Genesis: The de novo origin of genes is determined as gene genesis.

The de novo origin of genes is used to explain the existence of genes unique to a specific species, i.e., those for which no orthologous genes occur in their closest relatives. Large prokaryotic genomes as well as genomes whose closest relatives are relatively distant have the most genesis events (Snel et al., 2002) which as well could reflect ecological 'lifestyle'.

Only recently the described evolutionary forces were explored in more detail. An overall evolutionary model describing the degree of change of prokaryotic genomes over time is not available yet and it can be doubted that such an overall model will exists in a near future (Wolf et al., 2002).

Chapter 3

Sequence Alignment

The basis of the sequence alignment is introduced and motivated. After introducing the basic algorithms for a pairwise sequence comparison and summarizing some approaches for a multiple alignment, we discuss the problems arising with respect to the alignment of whole prokaryotic genomes. Finally, the existing approaches of pairwise and of multiple genome alignments are described in more detail.

The alignment of molecular biological sequences is an essential basic technique in the field of bioinformatics and serves as a preprocessing step for many more complex analysis such as phylogenetic reconstructions and molecule structure predictions. An alignment is the comparison of two (pairwise alignment) or many (multiple alignment) sequences in order to identify similar and non-similar subsequences. With respect to the molecular biological dogma (see 2.1), by analyzing sequence similarity one can draw conclusions about the structure and the function of the sequences. Furthermore, this analysis allows the determination of phylogenetic relationships (i.e., to construct taxonomies) and here similar sequences are assumed to have been passed on from a common ancestor (i.e., to be homologous, see 2.2.2.2).

3.1 Basis of the Sequence Alignment

We will explain the basic ideas and algorithms of sequence alignments, where we concentrate on the simpler case of the pairwise alignment problem for DNA sequences. The principles described, however, remain the same for the multiple alignment problem and work as well for protein sequences. For a more detailed introduction to the principle of the sequence alignment the interested reader may consult e.g., (Gusfield, 1997) or (Mount, 2001).

3.1.1 Alphabet and String: Basic Notation

For computational reasons, biological sequences can be represented as strings over an alphabet of letters. Note that we will always represent only a single strand of the DNA, since strands are antiparallel (see 2.1) and can be recomputed from each other. Furthermore, if not marked differently, DNA sequences are shown in the 5' to 3' direction.

DNA Alphabet: The DNA alphabet denoted by Σ is a set of four letters $\Sigma = \{A, C, G, T\}$. For proteins the alphabet usually consists of the 20 letters of the single letter notation of amino acids. With $\alpha, \beta \in \Sigma$ we denote variables for a single letter of the alphabet Σ, and with Σ^* we denote the set of all finite sequences over Σ.

String: A string $s \in \Sigma^*$ is an ordered sequence of letters over an alphabet Σ. The letters of a string s are indexed $\alpha_1\alpha_2 \ldots \alpha_n$. The index starts at 1 (first letter) and continues to n (last letter). The length of a string s is denoted by $|s|$ which is the total number of letters in s, with $|s| = n$. The string of length $|s| = 0$ is called the empty string and is denoted by ϵ. The letter at an index i of a string s is determined by $s(i)$. For our further considerations, the alphabet is always the DNA alphabet and we will use the terms string and sequence interchangeably.

Substring: A substring of a string s is a sequence of letters starting at an index i and ending at an index j. The substring is denoted by $s[i, j]$, with $1 \leq i \leq j \leq |s|$, and its length is $|s[i, j]| = j - i + 1$.

Prefix: A prefix of a string s is a substring $s[1, j]$, with $0 \leq j \leq |s|$. If $j = 0$ the prefix represents the empty string ϵ.

Suffix: A suffix of a string s is a substring $s[i, |s|]$, with $1 \leq i \leq |s| + 1$. If $i = |s| + 1$ the suffix represents the empty string ϵ.

For example, $\Sigma^* = \{\epsilon, A, C, G, T, AA, AC, AG, AT, CA, \ldots\}$, then let $s \in \Sigma^*$ be a string $s = AATGACT$. The length of s is $|s| = 7$, the letter at index $s(3) = T$, a substring is $s[3, 6] = TGAC$, a prefix is $s[1, 3] = AAT$, and a suffix is $s[4, 7] = GACT$.

3.1.2 Measurement of Sequence Correspondence

Two concepts can be distinguished when comparing short sequences: similarity and distance. The similarity of two sequences is a function associated with a numerical value which increases if the similarity between the sequences increases. Contrary to the similarity, the distance is a function associated with a numerical value which decreases with increasing sequence similarity. Mostly, these two

concepts are interchangeable since high similarity indicates low distance and vice versa. The measurements of sequence correspondence include two aspects: a quantitative one which represents the degree of similarity as a numerical value, and a qualitative one which is the alignment of the sequences in order to identify similar and non-similar regions. In the following, we will describe the basic concepts of the sequence alignment on the example of sequence similarity, since similarity is used as the measurement for the local alignment which is of further importance for this thesis.

The alignment procedure is provided with three edit operations, namely

- Insertion$(-, \alpha)$: inserts a letter α into s,
- Deletion$(\alpha, -)$: deletes a letter α from s, and
- Substitution(α, β): exchanges a letter α for a letter β in s

which are similar to the local mutations (see 2.3.1)).

Alignment: An alignment $\mathcal{A}$ of two strings s and t is an arrangement of s and t by position, which refletcs the sequence of edit operations (insertions, deletions, and substitutions) necessary to transform s into t.

Each edit operation is associated with a fixed score value denoted by σ. These score values vary depending on the sequence type (DNA, RNA, or amino acids) and the underlying evolutionary model (see 3.1.2.1). A simple scoring scheme is the Unit-Cost-Model with the following score values

- $\sigma(\alpha, \alpha) = 1$ is called a match,
- $\sigma(\alpha, \beta) = -1$, with $\alpha \neq \beta$ is called a mismatch, and
- $\sigma(\alpha, -) = \sigma(-, \alpha) = -1$ is called a deletion or insertion, respectively.

Alignment Score: Given an alignment $\mathcal{A}$, its alignment score $S(\mathcal{A})$ is the sum of all score values for all edit operations (insertions, deletions, and substitutions).

Biologists search for the alignment of the two sequences requiring the minimal number of edit operations, since there exists the evolutionary theory that two sequences emerged from a common ancestor sequence through the minimal number of mutation events. The alignment representing this is called the optimal alignment.

Optimal Alignment: An optimal alignment of s and t is an alignment of all possible alignments from s and t with a maximal alignment score. The maximal alignment score is called the optimal score value (Figure 3.1).

s =	G A - T G T A A - T G	G A - T G T A A T G
t =	G A C T - T A A T T -	G A C T - T A A T T
Scores	1+1-1+1-1+1+1+1-1+1-1 = 3	1+1-1+1-1+1+1+1+1+1-1 = 4

Figure 3.1: Different alignments of two strings s = GATGTAATG and t = GACTTAATT, with an optimal score value = 4.

(a)

	A	G	T	C
A	2			
G	-5	2		
T	-7	-7	2	
C	-7	-7	-5	2

(b)

	A	G	T	C
A	2			
G	-6	2		
T	-6	-6	2	
C	-6	-6	-6	2

Figure 3.2: Scoring schemes for DNA: (a) A scoring scheme which weights transition threefold higher than transversion and (b) a PAM scoring scheme with uniform mutation rate among nucleotides (Figure was modified from (Mount, 2001)), p. 91, Table 3.5.

3.1.2.1 Scoring Schemes and Gap Penalties

For the comparison of sequences the weighting of the edit operations (i.e., mutation events, see 2.3) is a critical step. The score σ (sometimes referred to as cost or weight) of an alignment is the sum of scores for each aligned pair of residues, plus scores for each introduced gap (see 3.1.2). A scoring scheme lists the likelihood of change from one nucleic (or amino) acid to another nucleic (or amino) acid, i.e., it weights substitution events, whereas gap penalties weight the likelihood of deletions and insertions during evolution. Selecting the adequate scoring scheme and gap penalties with respect to the evolutionary distance of the sequences is a crucial task for any such comparison (Durbin et al., 1998; Mount, 2001).

DNA Scoring Schemes: For alignments of DNA sequences there exist two scoring schemes that are frequently used. The first is based on physico-chemical characteristics of nucleic acids, which reveal that transitions (substitutions between purines A and G or between pyrimidines C and T (see 2.1)) are more likely to occur than transversions (substitutions between a purine and a pyrimidine or vice versa). Thus, this scoring scheme weights transitions 'better' than transversions (Figure 3.2 a). The second is based on an evolutionary model trained from empirical data. The PAM (Percent Accepted Mutation) matrices list the likelihood of change from one nucleic acid to another in homologous DNA sequences during evolution (Figure 3.2 b).

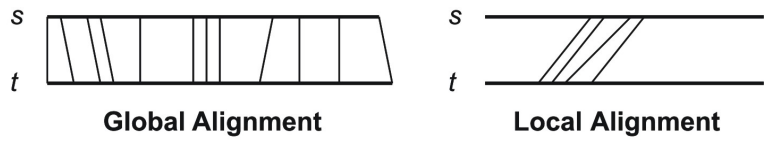


Figure 3.3: Schematic view on the global *vs.* the local alignment strategy (s and t indicating two sequences).

Gap Penalties: The introduction of gaps into the alignment (often indicated by the symbol '-') is necessary in order to account for the local mutations deletion and insertion (see 2.3.1). However, determination whether a deletion or an insertion has occurred during evolution in one of the sequences compared is mostly not possible, since the common ancestor of the sequences is normally not available. Usually, there are different gap penalties (affine gap costs) used for a gap opening g and for each gap extension r which are summed up to give the total gap cost w_x for a gap of length x, calculated as $w_x = g + r(x-1)$. The gap extension penalties are often set to smaller values than for gap opening, which allows a lower penalization of long insertion and deletions than by using a linear gap cost function (Durbin et al., 1998). This accounts for global mutations (2.3.2), where sometimes complete subsequences can be shuffled. Gap penalties have to be balanced with the scoring scheme in use in order to avoid that gaps either never appear in the case of too high penalties in comparison to the values in the scoring scheme or, vice versa, they will appear everywhere if the gap penalty is too low compared to the values in the scoring scheme.

3.1.3 Pairwise Alignment

Pairwise sequence alignment is the problem of comparing two input sequences for similarity. The strategies of sequence alignments can be classified into local and global. The global alignment strategy aligns sequences in the whole (end-to-end), i.e., a global alignment contains all letters of the two sequences. This kind of alignment is used when comparisons are made among relatively similar sequences of about the same length. In contrast to the global alignment, the local alignment strategy aligns the subsequences of these input sequences showing the highest similarity. The result of the strategy is an alignment of regions of high similarity and regions which are not aligned at all. This strategy is used when sequences of different length are compared or when searching for conserved domains (Figure 3.3).

3.1.3.1 Global Alignment

The global alignment approach aligns two sequence in the whole. If the sequences are of different sizes, gaps (sometimes referred to as indels) are introduced into

the shorter of the two sequences to bring the sequences to the same lengths. After Waterman (Waterman, 1995), the total number of possible global alignments of two strings s and t, both having the same length n is:

$$\binom{2n}{n} = \frac{(2n)!}{(n!)^2} \cong \frac{2^{2n}}{\sqrt{2\pi n}}.$$

For example, there exist 252 possible global alignments for two strings of length 5, and already 184,756 possibilities for strings of length 10. Realistic sequence lengths in molecular biology for a comparison of proteins are about 20 to 100 letters and for the comparison of the DNA sequences of prokaryotic genes about 1000 letters (see 2.2.2.3). Therefore, computing and evaluating all alignments of two long sequences in order to find the optimal alignment of these sequences is not possible in an interesting time, even with the computer generations at hand.

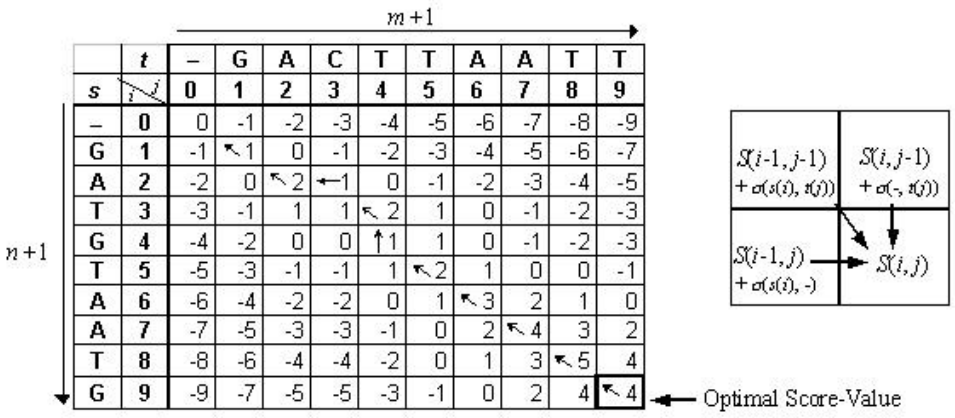
The traditional algorithm to compute the global alignment of two strings is based on the concept of dynamic programming and was invented by Needlemann and Wunsch (Needlemann and Wunsch, 1970). The basic idea of the algorithm is to construct the optimal alignment of the strings s and t by using previous optimal alignment solutions for the prefixes of these strings. This is the principle of dynamic programming, namely to lead back the actual solution on solutions from previous computation steps. If an optimal alignment of two prefixes of the strings s and t is known, there exist three possibilities for the expansion (see as well 3.1.2), namely

1. Insertion(-, $t(j)$),
2. Deletion($s(i)$, -), or
3. Substitution($s(i)$, $t(j)$),
   either a mismatch if $s(i) \neq t(j)$ or a match if $s(i) = t(j)$.

Assuming that the score values for all edit operations are known, the sequence of operations leading to an optimal score value has to be chosen in order to calculate the similarity S between the two sequences. The recursive call is

$$S(i,j) = max \begin{cases} S(i-1,j-1) & + & \sigma(s(i),t(j)), \\ S(i-1,j) & + & \sigma(s(i),-), \\ S(i,j-1) & + & \sigma(-,t(j)) \end{cases} \tag{3.1}$$

The results of the calculations can be stored in a $(n+1) \times (m+1)$ alignment matrix if the length of $|s| = n$ and that of $|t| = m$, because the number of all prefixes including the empty string ϵ is $(n+1)$ for s and $(m+1)$ for t. This alignment matrix contains all score values $S(i,j)$. The optimal score value of the two sequences can be found in the cell indexed (n,m) of the matrix. Initialization of the row $i = 0$ and the column $j = 0$ is done by calculating the score values for the alignment of either setting $s = \epsilon$ or $t = \epsilon$, respectively. There are three base cases:



	t	–	G	A	C	T	T	A	A	T	T
s		0	1	2	3	4	5	6	7	8	9
–	0	0	-1	-2	-3	-4	-5	-6	-7	-8	-9
G	1	-1	↖1	0	-1	-2	-3	-4	-5	-6	-7
A	2	-2	0	↖2	←1	0	-1	-2	-3	-4	-5
T	3	-3	-1	1	1	↖2	1	0	-1	-2	-3
G	4	-4	-2	0	0	↑1	1	0	-1	-2	-3
T	5	-5	-3	-1	-1	1	↖2	1	0	0	-1
A	6	-6	-4	-2	-2	0	1	↖3	2	1	0
A	7	-7	-5	-3	-3	-1	0	2	↖4	3	2
T	8	-8	-6	-4	-4	-2	0	1	3	↖5	4
G	9	-9	-7	-5	-5	-3	-1	0	2	4	↖4

Figure 3.4: Left: Alignment matrix for two strings s = GATGTAATG and t = GACTTAATT. The scoring scheme is the Unit-Cost-Model. The optimal score value of the strings can be found in the cell indexed (n, m) (i.e., cell (9, 9) = 3). An arrow in a cell (i, j) represents a pointer to the cell holding the maximal score value of the alignment of the prefixes. Right: General calculation scheme of the score value for a cell (i, j). To the maximal score value of either $(i-1, j)$, $(i-1, j-1)$, or $(i, j-1)$, the corresponding score value for either a deletion $\sigma(s(i), -)$, insertion $\sigma(s(i), t(j))$ or substitution $\sigma(-, t(j))$ (match or mismatch, respectively) is added.

1. $S(0, 0) = 0$,
2. $S(i, 0) = S(i-1, 0) + \sigma(s(i), -)$, for $i \leftarrow 1, ..., n$, or
3. $S(0, j) = S(0, j-1) + \sigma(-, t(j))$, for $j \leftarrow 1, ..., m$.

The computational complexity for calculating the edit distance using dynamic programming is $O(nm)$, since all calculations up to cell (n, m) have to be performed (Figure 3.4).

In order to find the corresponding alignment to the optimal score value, a further step is required. The path in the matrix describing the path to the cell (n, m) has to be identified. Pointers stored during the calculation of each score value $S(i, j)$ of the matrix are used to reconstruct the path from the cell (n, m) to the cell $(0, 0)$. For example, the pointer in the cell (i, j) points either to the cell $(i-1, j)$, $(i, j-1)$, or $(i-1, j-1)$ depending on the calculation performed to get its maximal score value $S(i, j)$. In case of more than one maximal score value can be calculated, a set of pointers is stored for the cell. Subsequently, a procedure called traceback follows a pointer from the actual cell (i, j) to either cell $(i-1, j)$, $(i, j-1)$, or $(i-1, j-1)$. In the case of $(i-1, j)$ or $(i, j-1)$ as the precursor cell a gap is inserted in the alignment, whereas in the case of $(i-1, j-1)$ as the precursor cell the letters $s(i)$ and $t(j)$ are aligned. The time complexity of the traceback procedure is $O(n + m)$ (Figure 3.4).

3.1.3.2 Local Alignment

The procedure to compute a global alignment of two sequences using dynamic programming can be adopted for the local alignment problem. Local alignment describes the problem of identifying all similar substrings of two strings s and t. Furthermore, two variants of local alignment approaches can be distinguished.

Approximate Pattern Search: The approximate pattern search of two strings s and t is an optimal alignment of s to a substring of t, where $|s| < |t|$. It is important that s needs to be considerable shorter than t, otherwise the result would be a global alignment of s and t.

For example, approximate pattern search can be used to identify -10-signals (see 2.2.2.1) often showing small differences.

Local Similarity Search: The local similarity search aligns all substrings $s[i, j]$ and $t[k, l]$ exhibiting the highest similarity with respect to a scoring scheme.

For example, local similarity search can be used to identify conserved functional domains within a protein.

For both variants, an optimal alignment (sometimes called the best local alignment) can again be computed using dynamic programming. The local alignment algorithm for the local similarity search was introduced by Smith and Waterman (Smith and Waterman, 1981). The recursive call is:

$$S(i,j) = max \begin{cases} 0, & & \\ S(i-1,j-1) & + & \sigma(s(i),t(j)), \\ S(i-1,j) & + & \sigma(s(i),-), \\ S(i,j-1) & + & \sigma(-,t(j)) \end{cases} \tag{3.2}$$

The score for the alignment of the string s to a suffix $t[j, |t|]$ as well as for the string t to a suffix $s[i, |s|]$ is 0 which in comparison to the global alignment corresponds to starting a new alignment. As a consequence, the row $i = 0$ and the column $j = 0$ are initialized with the score value = 0. Taken together, this ensures that only the best local similarities will be found.

Another difference to the global alignment is that now the best alignment will not necessarily end at the cell (n, m), but can end anywhere in the matrix. Therefore, the maximal score is searched over the whole matrix and the traceback starts from there and ends at a cell with a score value = 0.

3.1.4 Multiple Alignment

The comparison of more than two sequences in one analysis is called multiple sequence alignment. There are some challenges for the multiple sequence alignment which do not occur in the pairwise alignment problem. First, finding an

optimal alignment for more than two sequences by taking into account the degree of variation in all sequences at the same time, i.e., to judge whether a match, mismatch, or a gap needs to be introduced at a specific position of the alignment, is a computational problem. Second, finding biologically reasonable cumulative costs for the substitution events and for the weighting of the gaps in the alignment is another challenge.

In principle, the dynamic programming approach of the pairwise sequence alignment can be extended for a multiple sequence alignment approach, where the distance matrix is then extended to a k-dimensional hyper lattice for k sequences, with $k > 2$. However, the computational complexity to find the optimal alignment of the sequences $s_1, \ldots, s_k$ using dynamic programming is

$$O(2^k \prod_{i=1}^{k} |s_i|),$$

i.e., it grows exponentially with the number of sequences. Thus, for three sequences or a small number of short sequences, dynamic programming can still be used to generate an optimal alignment, but for more and longer sequences the computation time needed makes this approach unattractive. For example, if the time needed to compute one cell in the hyper lattice is 1 nanosecond, then for 8 sequences of length 100, the running time would be $2.6 \cdot 10^9$ seconds which are ~ 82.5 years. The problem of multiple sequence alignment has been proven to be NP-complete (Wang and Jiang, 1994). Thus, different heuristics of different strategies have been developed which can be classified into

- progressive methods like CLUSTALW (Thomson et al., 1997), where most alike sequences are aligned first and the multiple alignment is generated by adding more sequences (e.g., in their descending order of a pre-distance measure) into the existing alignment using a 'guide' tree. The guide tree is a phylogenetic tree generated with the neighbor joining method (Pearson et al., 1999) in order to support the selection of the sequences according to their pre-distance measure,

- iterative methods, i.e., out of the alignment of groups of sequences a revise multiple alignment of the sequences is generated (e.g., SAGA (Notredame and Higgins, 1996) based on a genetic algorithm),

- pattern-based methods, i.e., methods searching for locally conserved patterns occurring in the same order in all sequences which are then aligned first (e.g., DIALIGN (Morgenstern et al., 1996)), and

- statistical methods and probabilistic models (e.g., HMMER (Durbin et al., 1998) based on a hidden Markov model).

Furthermore, the approaches can be divided into global and local strategies as the pairwise alignment approaches. The classification of any approach is not unique, since a local or a global alignment strategy is always in combination with another approximation strategy (for a more detailed overview see (Thompson et al., 1999)).

3.2 Alignment of Whole Genomes

The alignment of genomes with up to some million nucleotides causes problems not well addressed by traditional alignment programs, which were typically designed for the alignment of proteins or short DNA sequences (Miller, 2001). Recently, a number of newer tools for the comparison of genomic DNA sequences have arisen, usually designed for different specific goals such as: to find all similar subsequences, to identify specifically target coding sequences (e.g., genes) and their conserved order, or to identify intergenic regions to detect conserved regulatory regions (Chain et al., 2003). First alignment tools available for whole genome alignments were concerning a pairwise alignment, whereas recently first tools for a multiple genome comparison have appeared (Chain et al., 2003).

In the following, we will discuss the problems of prokaryotic genome alignments with the example of pairwise genome comparisons. The problems discussed, however, hold as well for the problem of a multiple genome alignment.

3.2.1 Problems of Prokaryotic Genome Alignments

Although all approaches available show different procedures, all of them are based on the concept of local similarity search (sometimes referred to as anchor- or seed-based)(see 3.1.3.2). This is necessary since a global (end-to-end) alignment strategy would align unrelated regions for the frequent case of global mutations (see 2.3.2) (Miller, 2001; Chain et al., 2003). In principle, any approach which is able to identify local similarities (see 3.1.3.2) between two genomic sequences could be used for a pairwise alignment of such sequences, and existing approaches differ in their strategies used. In order to process long genome sequences efficiently, however, the traditional method using dynamic programming is too computational demanding. Therefore, many approaches are concerning heuristics in order to find local similarities efficiently (Miller, 2001; Chain et al., 2003).

In bioinformatics, (local) similarity is used to determine homology and functional equivalence of genomic features, whereupon homologous regions are thought to be functionally equivalent but not necessarily vice versa. The terms 'homology' and 'similarity' are often incorrectly used interchangeably (Reeck et al., 1987; Zouine et al., 2002). However, homology is either true or false with respect to a given time date back in evolution (see 2.2.2.2), while there are gradations of local similarity depending on the corresponding cost scheme and gap penal-

ties used (see 3.1.2.1). In order to conclude phylogenetic relationships between genomic features (2.2.2.1), their evolutionary origin (orthologous, paralogous, xenologous, or convergent evolution, see 2.2.2.2) has to be clarified. This requires going beyond the prediction of homology and of functional equivalence. For the discrimination between orthologous and paralogous genomic features, however, the use of sequence similarity is ambiguous, since paralogy is the outcome of a duplication event (see 2.3.2) and, thus sequences will be similar to some extent. The same problems arise when comparing xenologous genes which occur in one of the species because of inheritance from its ancestor and in the second species because of a horizontal transfer event in an ancestor (not necessarily with the species compared to). Horizontal transfer events are thought to be a primary route of acquiring new genes in prokaryotes (see 2.3.2). For paralogs that have descended from the last common ancestral species it is assumed that they do not reveal as many evolutionary conserved regions as corresponding orthologs, because those paralogs have been apart longer and are thus more divergent (Frazer et al., 2003). There exist approaches based on this assumption to determine (putative) orthology for protein-genes only by similarity comparisons, usually using the reciprocal best hits between at least three protein sequences (e.g., (Tatusov et al., 1997), see 3.3.1).

However, up to now there exists no commonly accepted model for the evolution of complete genomic DNA sequences of prokaryotes (i.e., a model listing the likelihood of change of nucleic acids for the comparison of whole genomes), since different DNA regions like orthologs, paralogs, or xenologs are expected to have different mutation rates. Moreover, the rates of duplication events (leading to paralogs) and of horizontal gene transfer events remain unclear at the moment (see 2.3.2) (Wolf et al., 2001). Therefore, each local similarity will have an individual statistical significance depending on its location in the genomes (Roytberg et al., 2002). This makes the selection of an adequate cost scheme and gap penalties a demanding task for genome comparisons of prokaryotes. In many cases the task of the alignment is the comparison of a newly sequenced genome to a genome which is already annotated, indeed then the selection of adequate cost values is even impossible.

In order to avoid inconsistencies in the resulting alignment caused by falsely assigned local similarities to biologically unrelated regions because of inadequate cost schemes or gap penalties, comparative genomic tools usually take only local similarities with a similarity value above a usually high threshold. In the case of local similarities showing different relative orders between the two genomes, normally the strongest similarity (i.e., the longest similarity with the highest similarity value) is kept in order to identify the true chain of local similarities which reflects homology. Accordingly, almost all software tools available require that the genomes to be aligned have a collinear order of their biological corresponding regions (Figure 3.5, left) (Miller, 2001; Chain et al., 2003). In turn, this implies that the genomes to be compared are phylogenetically closely re-

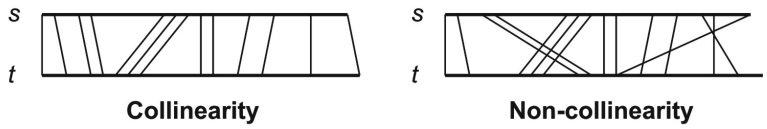


Figure 3.5: Schematic view on collinearity and non-collinearity (s and t indicating two sequences).

lated and that only a few global mutations have occurred since their speciation. When comparisons are made among species (or even within some species) which are non-collinear (Figure 3.5, right), however, this limits the possibility for the identification of less conserved genomic features because true local similarities of biologically related but rearranged genomic features will not necessarily be incorporated into the alignment. This is in particular of importance for the comparison of a newly sequenced genome to known genomes, since here the existence of paralogs and of xenologs is of special interest in order to infer evolutionary relationships and gene functions. Hence, allowing for non-collinearity is a strong requirement for an inter-species alignment approach (Miller, 2001; Chain et al., 2003; Frazer et al., 2003).

3.2.2 Approaches for a Pairwise Genome Alignment

3.2.2.1 BLAST

The Basic Local Alignment Search Tool (BLAST) (Altschul et al., 1990) and the resulting 'family' of BLAST-tools[1] (e.g., Gapped BLAST and PSI-BLAST (Altschul et al., 1997)) are widely used tools in order to search for sequence similarity within large protein or DNA data bases. BLAST is a heuristic which attempts to optimize a specific local similarity measure. The heuristic can be used in a way that by setting a threshold parameter T, BLAST permits a trade-off between speed and sensitivity, i.e., a higher value of T increases speed by decreasing sensitivity whereas a lower value of T decreases speed but increases the probability to find weaker similarities.

The original algorithm is based on the notion of a maximal segment pair (MSP) which is the highest scoring pair of identical length segments chosen from two sequences. An MSP can be of any length as its boundaries are chosen to maximize its score. This score is heuristically calculated providing a measure of local similarity of a pair of sequences. The MSP score is defined to be locally maximal, i.e., a score cannot be improved by either extending or shortening the MSP, in order to identify all conserved regions between two sequences. In contrast to the standard local alignment approach of Smith and Waterman (Smith

[1] Available at `ftp://ftp.ncbi.nih.gov/blast/`

and Waterman, 1981), MSP scores are calculated using a rapid approximation based on a statistically significant estimation of the highest MSP score so that by chance similarities are unlikely to appear. This measurements is called the p-value (based on the Poisson distribution) which is the chance that a score as high as the one observed between the two sequences will be found by chance. However, for a better understanding of the results generated by BLAST by a user during database searches so called expectation values (E-value) instead of the p-values are given. An E-value gives the probability that a score as good as the one found between a query sequence and the sequences in the database will be found by the same number of comparisons to randomly generated sequences. For example, an E-Value = 1 means that there is a chance that one unrelated hit will be found. BLAST searches only for local similarities below a given E-value.

An example of a BLAST result can be found in Figure 3.6. Each BLAST hit constitutes a local similarity, where the substrings in two sequences subject (Sbjct:) and the query (Query:) are indicated by their indices, together with the declaration of their strands; the alignment is the specification of matched letters (indicated by a cross), mismatched letters (aligned without cross), and gaps (indicated by a dash in one of the sequences); and the similarity value is indicated by a score (Score in bits).

3.2.2.2 MUMmer

The Maximal Unique Match(mer) (MUMmer) (Delcher et al., 1999) and the newer versions MUMmer2 (Delcher et al., 2002) and MUMmer 3.0 (Kurtz et al., 2004) is a system to align two closely related genome sequences (i.e., here closely related refers to as homologous and collinearity of the sequences). It has been shown by Delcher *et al.* (Delcher et al., 1999; Delcher et al., 2002), that their approach can produce biologically relevant alignments of closely related species like two *Mycoplasma* species or two strains of *Mycobacteria tuberculosis*. The output of the approach is a base-to-base alignment of the input sequences, highlighting the exact differences in the genomes. In detail this includes: SNPs (single nucleotide polymorphism), large insertions, significant repeats, tandem repeats and reversals, and the exact matches between the genomes.

The algorithm of MUMmer is based on a data structure named suffix tree, which contains all suffixes of the two input genomes (Delcher et al., 1999). Suffix trees can be constructed in linear time regarding the sum of the length of the two genomes (Gusfield, 1997; Kurtz, 1999). Furthermore, suffix trees allow to quickly find all subsequences shared by the two input genomes. In a first step, the approach constructs the suffix tree for the two input genomes and uses it to find all maximal unique matches (MUMs) between both strands of the input genomes of a minimum user-specified length. A MUM is defined as a subsequence that occurs exactly once in both genomes and is not contained in any longer such sequence, i.e., a MUM is delimited by mismatches. The MUMs identified are sorted using

```
 Score = 56.0 bits (28), Expect = 4e-005
 Identities = 76/92 (82%)
 Strand = Plus / Minus

Query: 595634  gagaaggattatggaaaggttaaagctctaaggggggtaagctttcagatagaagaaggg 595693
               ||||| ||||||||||||||| ||||| || |||||  | | ||||   ||||| |||||
Sbjct: 1364273 gagaaagattatggaaaggtgaaagcactcaggggaattacctttccatagaggaaggt 1364214

Query: 595694  gagatatttggtttgataggccctaacggtgc 595725
               |||||||||||| ||||| ||||| ||||||||
Sbjct: 1364213 gagatatttggattgattggcccaaacggtgc 1364182

 Score = 54.0 bits (27), Expect = 2e-004
 Identities = 131/165 (79%), Gaps = 3/165 (1%)
 Strand = Plus / Plus

Query: 405034  agggcatttccaacgaaaaacaatcccattaaaatgtcgctaacgaaccaggccttatag 405093
               |||||||| ||| |||| || || ||||| || || |||||||  ||||| |||||||||
Sbjct: 1605522 agggcattaccagcgaagaagaaacccatcaatatatcgctaataaaccatgccttatag 1605581

Query: 405094  cttaggaatatcttccagctttttctggcaacccctaaaagggctctaagctc---agta 405150
               |||| || |||||||||  | ||||| ||||| |||   ||||||   |||||   | ||
Sbjct: 1605582 cttatgagtatcttccaattctttcttgcaacacctttagggcttcgagctcttcaata 1605641

Query: 405151  gcctccatatcccatcatcctcgttatcttttccgcccacttgaa 405195
                || ||||| |||||||||||||||||| | || ||||||||||
Sbjct: 1605642 acccccataccccatcatcctcgttatcctctcggcccacttgaa 1605686

 Score = 48.1 bits (24), Expect = 0.010
 Identities = 129/164 (78%)
 Strand = Plus / Minus

Query: 791499  ggaatagagataatggaaggcaaagcttacatcttcacagagaagattaaaggctgggga 791558
               |||||||||| ||| |||| || || ||||||| ||| ||||| ||||| || |||||
Sbjct: 1182803 ggaatagagatcatgcaaggaaaggcctacatctacaccgagaaaattaagggttggggt 1182744

Query: 791559  ggattacctattggaacccagggaaagatggttggcttactgcaagatgaactttcggct 791618
               ||  | || || |||||| |||| |  ||| | ||  |||| || |||||||| || |||
Sbjct: 1182743 ggccttccaataggaaccgaggggagaatgatcgggatacttcatgatgaactctcagct 1182684

Query: 791619  ttggcgatattcctgatgatgaagagaggagttgaggttatacc 791662
               || || ||||| || |||||||| || |||||||||| |||||
Sbjct: 1182683 ttagcaatatttctaatgatgaaaaggggagttgaggtaatacc 1182640
```

Figure 3.6: Part of a BLAST (blastn) result for the genomes of Sbjct: *Pyrococcus horikoshii* (Pho) *vs.* Query: *Pyrococcus abyssi* (Pab) with an E-value (Expect) cutoff = 1.0.

the longest increasing subsequence algorithm in order to identify the longest possible set of matches occurring in ascending order in both genomes (Delcher et al., 1999). Gaps between these MUMs are closed using a standard alignment approach (dynamic programming) and by recursively applying the entire matching procedure using a reduced minimum MUM length.

The strong requirement that the two input genomes have to be closely homologous restricts this approach to a small subset of complete genomes presently available. This requirement is the result of two steps of the approach: the use of MUMs as anchors and their sorting procedure. When comparisons are made among further distanced species, MUMs restrict the alignment by their demanded

minimum length, the required exact similarity, and the required uniqueness. The length of a MUM should never be less than 18-20 bp, otherwise random matches will possibly be contained within the result set. It is well known, however, that SNPs do accumulate between evolutionarily distant species (Bruijn et al., 1998) and homologous subsequences can be carved up by SNPs (occurring especially at the wobble position) in smaller similar subsequences than the required minimum MUM length. Furthermore, exact matches occurring more than once in the genomes may also be meaningful and the coverage of the sequences increases if this requirement is relaxed (Chain et al., 2003). However, since the release 2.1, MUMmer can also find distant matches, and the latest MUMmer 3.0 release[2] can also accept non-unique maximal matches in order to find all matches for a repetitive substring (Kurtz et al., 2004). Nevertheless, the sorting of the MUMs does not account for rearrangements which frequently occur within prokaryotic genomes since collinearity of similar segments is required. Moreover, the resulting chains of MUMs can contain overlapping MUMs which leads to inconsistencies in the resulting alignment (Chain et al., 2003). To overcome this problem, MUMmer simply removes the overlapping parts from such MUMs (Höhl et al., 2002; Chain et al., 2003).

3.2.2.3 PipMaker

The Percentage Identity Plot Maker (PipMaker) (Schwartz et al., 2000) is an alignment approach to compare two long genomic DNA sequences which can be accessed via the world-wide web. The approach is able to align e.g., *Escherichia coli vs. Salmonella typhimurium*. The output of the approach is a percent identity plot (pip), which shows both: the position in one sequence and the degree of similarity for each aligned segment between two sequences. Furthermore, a dot plot of the sequences is available for the user.

PipMaker[3] is based on the BLASTZ algorithm (Schwartz et al., 2003), which is an entirely new implementation of the Gapped Blast program of Altschul *et. al* (Altschul et al., 1997) (see 3.2.2.1). This implementation guarantees that computer memory will never be the limiting factor. Instead of the Smith-Waterman (Smith and Waterman, 1981) notion of locally optimal alignment, it uses the X-drop approach (Miller, 2001).

In addition to the sequence similarity, the approach can visualize interspersed repeats, which are provided by an separate file computed by the program RepeatMasker[4]. There are further options available which can be used separately and subsequently to the BLASTZ search in order to increase the specificity of the approach by eliminating duplicated hits. The 'chaining' option remains only

[2]Available at `ftp://ftp.tigr.org/pub/software/MUMmer/`

[3]Available at `http://pipmaker.bx.psu.edu/pipmaker/`

[4]Available at `http://ftp.genome.washington.edu/RM/RepeatMasker.html`

the BLASTZ-hits which appear in the same relative order in both genomes. The 'single coverage' option allows only the highest scoring hits to form the alignment.

3.2.2.4 WABA

The Wobble Aware Bulk Aligner (WABA) (Kent and Zahler, 2000) is an algorithm which has been shown to be able to align an 8 million bp genomic fragment of *Caenorhabditis briggsae* against the complete genome of *Caenorhabditis elegans* containing 97 million bp. Both species are nematodes, but the approach is as well applicable for the task of an interspecies alignment of prokaryotes, since the implementation is capable of dealing with small and large insertions and copes with SNPs. However, the proof of reliability for this task still has to be given.

The novelty of the algorithm implemented in WABA[5] is that it copes with the wobble position problem, i.e., that the third nucleotide in a codon is often redundant in order to maintain the amino acid sequence (see 2.1) and thus tends to diverge freely. This is a problem for an interspecies comparison of evolutionary distanced species when relatively long conserved seeds (anchors) for the alignment are required like for Gapped Blast ((Altschul et al., 1997)) or MUMmer (see 3.2.2.2). In a first phase homologous regions between the two inputs are determined. This is done by breaking the shorter input sequence into fragments and applying a Gapped Blast-like (see 3.2.2.1) approach. However, WABA does not require such long anchors (high scoring pairs) as Gapped Blast, but allows at every third base position a mismatch to occur in order to account for the wobble position. In the second phase, the homologous regions are aligned in an extended window using a pairwise Hidden Marked Model (Durbin et al., 1998). In the third phase, the possible short alignment fragments are merged, if possible, into longer ones if they overlap by at least 15 bp.

3.2.2.5 DIALIGN

DIALIGN (for DIAgonal ALIGNment) (Morgenstern et al., 1996; Morgenstern et al., 1998) and the newer version DIALIGN-2 (Morgenstern et al., 1999) is an algorithm capable of a pairwise or a multiple alignment of both nucleic and amino acid sequences. The basic idea of this approach is to construct the alignment of long sequences by the comparison of long segments of these sequences which are not interrupted by any gap rather than by the comparison of single bases. The resulting alignment is composed of gap-free pairs of segments of equal length which are called diagonals since they form diagonals in the corresponding dot-matrix of the sequences used for their identification. The diagonals are collected considering a consistency criterion, i.e., a diagonal is consistent if no double or crossover assignment of its bases occurs.

[5]Available at `http://www.cse.ucsc.edu/~kent/xenoAli/`

In the case of a pairwise alignment, the consistency of any two diagonals is given if diagonals are ordered so that the end positions of one of them are both smaller than the respective starting positions of the other one. A weight is assigned to each diagonal which measures the significance of a diagonal, i.e., the similarity of the respective segments and its probability of a random occurrence, and the algorithm takes into consideration the significance of diagonals of different length (it assigns e.g., high weights to shorter diagonals if they have a high rate of matches). Gaps are not considered in the calculation of the alignment score. In order to find the 'best' alignment of all diagonals found, the algorithm optimizes the set of diagonals by maximizing the sum of the weights of consistent diagonals using a modified dynamic programming scheme (see 3.1.3.1).

In the case of a multiple sequence alignment, a greedy procedure is used to find the best set of diagonals for the alignment. In a first step, all optimal pairwise alignments are formed. The diagonals contained in these alignments are sorted with respect to their weight scores and their degree of overlap to other diagonals in order to emphasize motives occurring in more than two sequences. The multiple alignment is generated by taking the diagonals in their order of decreasing weights which are added to the alignment if they are consistent with the already assigned diagonals. Once a diagonal has been introduced into the alignment it cannot be removed at any later stage.

The use of gap-free segments allows DIALIGN[6] to identify even small conserved regions that cannot be identified by a standard Smith-Waterman approach (Chain et al., 2003). Consequently, the approach is able to identify even less conserved functional regions on a large genomic scale necessary for the comparison of further distanced species. However, the approach assumes collinearity of the segments as a prerequisite which restricts DIALIGN to comparisons of genomes of highly conserved gene order (i.e., in which few rearrangements have occurred).

3.2.2.6 ASSIRC

The Accelerated Search for Similarity Regions in Chromosomes (ASSIRC) (Vincens et al., 1998) and its newer version D-ASSIRC (Vincens et al., 2002) which is able of a distributed processing is a pipeline which has been developed in order to find regions of similarity in nucleotide sequences >100 kbp. The method is split in three steps: (1) identification of pairs of identical common motifs called seeds (or k-mers) using standard hashing functions (as implemented e.g., in BLAST or FASTA), (2) extension of the identified seeds by a random walk procedure, and (3) final selection of regions of similarity by aligning these regions using a standard Smith-Waterman approach (Smith and Waterman, 1981) using the BESTFIT program of Dölz (Dölz, 1994).

Although this method has proven to be faster and more sensitive in com-

[6]Available at `http://bibiserv.techfak.uni-bielefeld.de/dialign/`

parison to BLAST (see 3.2.2.1), the approach is susceptible to large insertions and deletions (Vincens et al., 1998; Chain et al., 2003), i.e., to global mutations. The comparison between ASSIR[7], BLAST (see 3.2.2.1), and FASTA (Pearson, 1990) using data of the yeast (*Saccharomyces cerevisiae*) chromosome V and IX indicated that ASSIRC is clearly faster by nearly the same coverage (number of identified similarities) of all approaches.

3.2.2.7 LSH-ALL-PAIRS

The LSH-ALL-PAIRS algorithm (Buhler, 2001) finds ungapped local alignments with up to a specified fraction of substitutions in genomic sequences. This allows to overcome the problem of exact-matched-based comparisons like BLAST, FASTA, or MUMmer which require a minimum seed length which balances sensitivity to weak similarities against efficiency on long sequences to reduce hits by random chance. To find these ungapped local alignments efficiently, LSH-ALL-PAIRS uses a randomized search technique called Locality-Sensitive-Hashing (LSH). Since seeds can span substitutions, which accounts e.g., for the wobble base, the seed length can be much longer (typically 60-80 bp) which improves the sensitivity of a hit while maintaining sensitivity to weak similarities. Because of the randomized search the algorithm may by chance miss seeds, i.e., some significant similarities for the alignment. By iterating the LSH-algorithm multiple times along with some implementation heuristics, the approach minimizes the chance to miss such significant hits. LSH-ALL-PAIRS has been tested on genomic sequences such as the β-Globin-locus of human and mouse, a self-comparison of the human chromosome 22, and three *Pyrococcus* species. The approach was able to align these sequences with as little as 63% identity in both coding and non-coding sequences.

There are some drawbacks of the approach mentioned by the authors. Gapped similarities may cause difficulties if the segments between the gaps are too short and, therefore, can be missed by the initial random search. Furthermore, long gapped similarities may be broken into ungapped fragments if the similarities are on different diagonals. These ungapped fragments may by themselves not be considered significant, i.e., not considered for the alignment. Finally, the initial randomize search scores the seeds with a simple mismatch count instead of a more general alignment scoring function (Chain et al., 2003).

3.2.2.8 OWEN

OWEN (after Richard Owen who invented the concept of homology in 1848) (Ogurtsov et al., 2002; Roytberg et al., 2002) is an interactive software tool to pairwise align long DNA sequences. OWEN[8] has been tested on regions of

[7]Available at `http://www.biologie.ens.fr/perso/vincens/assirc.html`
[8]Available at `ftp://ftp.ncbi.nih.gov/pub/kondrashov/owen/extra`

vertebrate genomes like pufferfish *vs.* human, chicken *vs.* human, and murine (mouse-like) *vs.* human.

The algorithm tries to find the true chain of similarities (that reflects orthology) by a hierarchical greedy approach which keeps the strongest similarity (i.e., the longest similarity with the highest value) in case of a conflict, e.g., repeating similarities. To find local similarities OWEN uses a combination of a core-based approach (BLAST-like) to identify strong similarities and exhausting searches (Smith-Waterman-like) for weak similarities between strong similarities. The greedy approach to identify the evolutionarily true chain (called backbone chain) works as follows. Initially, the backbone chain is formed by the strongest similarity and then iteratively it tries to add the next strongest similarity not conflicting with the previous ones until all strong similarities are proceeded. In a second step, gaps between the strong similarities are closed by searching for weaker similarities in between them which are statistically significant because of the reduced length, and the algorithm assembles the true chain as in the previous manner. In addition, the resolution of conflicts can also, as an option, be manually be done by a user.

Comparisons of results of OWEN (using the previously described sequences) have been compared to results generated by PipMaker (see 3.2.2.3), and the comparison indicated that OWEN has a higher sensitivity (i.e., identifies more local similarities assigned as evolutionary true) as PipMaker.

3.2.2.9 Vmatch

Vmatch (Kurtz, 2003) cited in (Chain et al., 2003) is an index-based large scale matching approach. Vmatch[9] is an anchor-based alignment approach like MUMmer (see 3.2.2.2), however, instead of using a suffix tree it uses suffix arrays which speeds up the substring matching (about six to 24 times in comparison to other hashing methods) and reduces the space required in comparison to suffix trees. Here, maximal exact matches (MEMs) are pairwise computed that cannot be extended without a mismatch. Subsequently, seeds (MEMs) can be extended using either a greedy algorithm or alternatively by a method used in REPuter. It was shown that Vmatch can compare a 5 mb microbial genomes to a concatenation of other microbial genomes of total 800 mb. We were not able to find a more detailed description of the main algorithm, even though an extensive user manual is available at `http://www.vmatch.de`.

3.2.2.10 MaxMinCluster

MaxMinCluster (Wong et al., 2004) is an approach to perform pairwise genome alignments of closely related species in order to discover regions containing con-

[9]Available at `http://www.vmatch.de/`

served genes. MaxMinCluster[10] has been used to align ten pairs of human and mouse chromosomes, but not yet to compare prokaryotic genomes. In these comparisons the alignments generated by MaxMinCluster often had a higher coverage (up to 1.5 times) while preserving the sensitivity in comparison to the alignments generated by MUMmer 3.0 (see 3.2.2.2).

In a first step, the approach identifies all maximal unique matched substring pairs (equivalent to MUMs, see 3.2.2.2) using a suffix tree (compare 3.2.2.2). The identified matched substrings are ordered according to their occurrence in one of the two genomes. In a subsequent step, this sequence of uniquely matched substrings is the input for a dynamic programming algorithm which is used in order to identify k-noisy clusters out of them. A k-noisy cluster is a subsequence S of uniquely matched substrings which allows to remove up to k pairs of matched substrings, where k is supposed to be a small positive integer (default set to $k = 3$). Furthermore, to considered a cluster as k-noisy it is required that the strands of all matched substrings within S are identical, the distance between consecutive matched substrings does not exceed a given threshold, and the number of matched substrings contained in S is at least of a required minimum size. Then, the algorithm tries to maximize the size of the smallest of these clusters. The implementation of the MaxMinCluster approach is optimized according to its time and space requirements and allows to perform pairwise genome alignments on a standard PC.

The procedure of the MaxMinCluster approach is to some extent similar to MUMmer using suffix trees in order to identify matched regions and thus, has the same drawbacks as described in Section 3.2.2.2. In contrast to MUMmer, however, the selection of the substrings for the alignment differs in the way that it is not as strict as the longest increasing subsequence algorithm used there. Nevertheless, as for MUMmer the sorting of the regions does not account for rearrangements within prokaryotic genomes since their collinearity is required.

3.2.3 Approaches for a Multiple Genome Alignment

3.2.3.1 MGA

The Multiple Genome Aligner (MGA) (Höhl et al., 2002) is an anchor-based approach like MUMmer (see 3.2.2.2). However, MGA[11] is capable of coping with more than two genomes. MGA was tested on, e.g., three complete genomes of different strains of *E. coli*, and it was able to align 74% of these genomes.

The approach is divided into three phases. The first phase finds all maximal multiple exact matches (multiMEMs) of a user-defined minimum length. A multiMEM is the counterpart of a MUM (see 3.2.2.2), i.e., a sequence occurring in all input genomes which cannot be extended to the left or the right in any of these

[10] Available at `http://www.csis.hku.hk/~colly/maxmincluster/`.
[11] Available at `http://bibiserv.techfak.uni-bielefeld.de/mga/`

genomes. However, multiMEMs are more general than MUMs since they do not require to be unique, i.e., they can occur more than once in each genome. Searching for multiMEMs is done by an efficient implementation of a virtual suffix tree (Höhl et al., 2002). In analogy to MUMmer, the second phase sorts these anchors to retrieve the longest non-overlapping sequence of multiMEMs which occur in the same order in all genomes. The last phase tries to close gaps between the anchors either recursively applying the same method or by the standard multiple alignment program CLUSTALW (Thompson et al., 1999) (see 3.1.4).

The results are quite remarkable in comparison to MUMmer (multiple sequences, less computational time, less memory usage, alignment coverage), however, the strong requirement on closely homologous sequences for the comparison as in MUMmer generates the same problems.

3.2.3.2 MUMmer

The last version of MUMmer 3.0 (Kurtz et al., 2004) is able to a multiple genome alignment (see 3.2.2.2 for a detailed description of the basic approach).

3.2.3.3 Mauve

Mauve (Darling et al., 2004a) is a software tool for a multiple genome alignment in order to identify conserved genomic regions as well as rearrangements and inversions in such regions. Mauve[12] has been tested on nine enterobacterial genomes which among the usual recombination events included genome rearrangements and horizontal transfer events. Furthermore, Mauve was able to determine global rearrangements in three mammalia genomes (including mouse and human).

Like other methods described above (e.g., MUMmer, see 3.2.2.2), as a heuristic to speed up the alignment procedure Mauve searches for anchors of a minimum length k. Usually, the number of possible anchors of a required minimum length k increases exponentially with the number of genomes to be aligned while the number of 'true' anchors (i.e., hits between orthologous genes) remain constant. To overcome this problem, Mauve searches for Multiple Maximal Unique Matches (multi-MUMs) which are exact matching substrings common to at least two genomes, occur only once in each genome, and are flanked by mismatches at each side. In order to find all multi-MUMs, Mauve uses a seed-and-extend hashing method (see (Darling et al., 2004b) for details). From the resulting set of multi-MUMS a phylogenetic guide tree is calculated using Neighbor Joining.

The selection of the 'true' chain of local similarities (reflecting homology) from this set of multi-MUMs is more relaxed in comparison to other alignment approaches like MUMmer. Instead of selecting the highest scoring collinear chain of local alignments, Mauve constructs locally collinear blocks (LCB) which are then combined to the overall alignment. An LCB is an ordered collinear subset of

[12] Available at `http://gel.ahabs.wisc.edu/mauve/`

all multi-MUMs which meets some minimum weight criteria, i.e., a LCB reflects a homologous region shared by at least two of the genomes in the analysis and does not contain any rearrangements. In order to generate all LCBs and to filter out unlike local alignments from the resulting set of multi-MUMs Mauve uses a greedy breakpoint elimination algorithm with a minimum weight criterion as another input. The minimum weight criterion is the sum of the lengths of all multi-MUMs contained in a LCB. Its initial value is set to $3k$, i.e., three times the lengths of the initially required minimum seed length k.

In a next step, Mauve recursively uses the seed-and-extend hashing method with a progressively reduced minimum length k in order to identify additional anchors within and between LCBs. This step increases the sensitivity of the approach. Finally, like MGA (see 3.2.3.1) Mauve uses CLUSTALW (Thompson et al., 1999) (see 3.1.4) in combination with the previously generated guide tree in order to align closely related genomes first. Thus, the global alignment of a set of genomes is generated of locally conserved blocks of collinear subsequences. This procedure allows for genomes in the analysis to be non-collinear.

3.2.3.4 ABA

ABA (A-Bruijn Aligner) (Raphael et al., 2004) is a method for a multiple alignment of biological sequences, in particular, for protein sequences with shuffled or repeated domains or genomic sequences with duplications or inversions. ABA has been applied in different case studies including genomic sequences such as plant chloroplast genomes. It has been shown that ABA has some advantages over traditional alignment methods and partial order alignments. This method can as well be used for the alignment of complete prokaryotic genomes.

In ABA a multiple alignment is represented by a weighted directed graph which is allowed to contain cycles called A-Bruijn graph. For the alignment of n sequences ABA uses the sequences and all combinations of their pairwise alignments as the input for the construction of the A-Bruijn graph. In the graph, each sequence s is modeled by a directed path of m vertices, where m is the number of characters contained in s. In case of two sequences s_i and s_j have matched positions indicated by their pairwise alignment, they share vertices at this positions (in the process called 'gluing'). The resulting A-Bruijn graph can contain short cycles indicating 'weak' and inconsistent alignments. Short cycles are removed from the graph by a solving the subgraph with large girth problem. The remaining larger cycles indicate then e.g., multidomain organization. To draw the resulting multiple alignment ABA uses the Graphviz package.

Taken together, the representation of multiple alignments by the A-Bruijn graph an enables the flexible comparison of biological sequences with repeats and inversions. However, the major drawback of the approach is its long runtime which is required for the preprocessing step of constructing all combinations of pairwise alignments of the input sequences (Raphael et al., 2004). For genomic

sequences, memory space for the pairwise alignments as well as for the graph-based representation is the major constrained (Raphael et al., 2004).

3.3 Determining Evolutionary Origin

Determining the evolutionary origin of genomic features between genomes is an important step in order to a better understanding of evolution of prokaryotes including the reconstruction of taxonomies and the understanding of the dynamics of genes in different ecosystems (Buckley, 2004). Since the common ancestor of, e.g., two species is normally not known and its genome is not available, this frequently causes problems in the field of microbiology. Thus, usually sequence similarity is used in order to infer homology (see 2.2.2.2) of biological sequences which can cause problems (see the discussions in, e.g., (Reeck et al., 1987), (Zouine et al., 2002), and (Woese, 1998)). However, the strong need for a classification of the evolutionary origin of genomic features resulted in different approaches (many of them based on sequence similarity) to establish such a basis. In the following we will describe one such approach, since this will be the data basis used in the evaluation in Chapter 7.

3.3.1 Clusters of Orthologous Groups of Proteins (COGs)

In order to determine orthology (see 2.2.2.2) between protein coding genes within complete genomes, Tatusov *et al.* (Tatusov et al., 1997) performed an all-against-all BLAST search (see 3.2.2.1) on the amino acid sequences of the genes determined in their GenBank entries. Subsequently, genes were classified as orthologous if at least three reciprocal best BLAST hits existed, i.e., a gene was only labelled orthologous if this gene had its best hit in a gene of at least two other genomes, these genes had their best hits between each other, and vice versa. Thus, orthology was determined using the similarity of the amino acid sequences of the genes annotated in the genome entries. As a result, a COG consists of either individual orthologous genes (one-to-one relationship) or orthologous groups of paralogs (one-to-many, or many-to-many relationships) (Tatusov et al., 1997). Some of these annotated genes were divided into different COGs due to multidomain proteins (Tatusov et al., 1997). Yanai, Wolf and Koonin (Yanai et al., 2002) identified 405 fusion links between COGs corresponding to such multidomain proteins.

Altogether, the update version of the COG data base (`http://www.ncbi.nlm.nih.gov/COG/new/`) contains 63 prokaryotic genomes in the Unicellular Clusters divided into 13 Archaea species, ten species generally classified as Bacteria, four species classified as Actinobacteria, twelve species as Gramplus, six species generally as Proteobacteria, eleven species as γ-Proteobacteria, and seven as α-Proteobacteria. Furthermore, three eukaryotic species are included. In the data

base 4873 COGs are included for this species. The number of COGs in the genomes, however, does not necessarily reflect their true number of orthologous genes. First, orthology was determined using sequence similarity. The problem of using sequence similarity in order to determine orthology is discussed in 3.2.1. Second, since the approach of Tatusov *et al.* (Tatusov et al., 1997) requires at least reciprocal best hits to two other genomes to determine a COG, i.e., orthologous genes occurring in only two of the genomes in the COG database are not classified as orthologous (have no COG label). For example, on a larger data set including a genome containing this gene, it might have been classified as orthologous. Thus, in the following we will refer to orthology as determined by the COG data base as to *putative* orthology, because of the reasons described above.

Chapter 4

Knowledge-Based Genome Alignment

In this chapter, the approach of a knowledge-based genome alignment for prokaryotic genomes is described in detail. First, the selection of the representation language is motivated. The modeling of genome architecture, the introduction of knowledge, and the reasoning method for ensuring consistency of this knowledge is explained. Subsequently, the alignment procedure is given in detail and the algorithms to use the knowledge modeled in order to evaluate similarities is described.

The term knowledge-based paraphrases techniques of the field of Artificial Intelligence (AI), which in turn applies theories and techniques of three other fields: logic (provides the formal structure and rules of inference), ontology (defines the concepts that exist in the application domain) and computation (supports the applications that distinguish knowledge representation from ontology) (Sowa, 2000). A knowledge base is a dynamic collection of facts and rules that can be used for logical inferences, i.e., it contains the intrinsic capacity to compute new knowledge from stored knowledge. In contrast, relational databases or object-oriented databases are static collections of facts that can only be used for information retrieval.

4.1 Principle of the Approach

In order to admit the comparison of genomic DNA sequences of phylogenetically further distanced species an alignment approach is required which is able to deal with non-collinearity of local similarities (see 3.2.1). In particular, this task is of interest for the comparison of a newly sequenced genome (called the

query sequence) with a genome which is already annotated (called the *reference sequence*). Such a comparison gives a first impression of

- the existing metabolic functions encoded by the query sequence (and enables thus to infer the ecological relevance of this species, where ecological relevance subsumes aspects as pathogenicity or symbiotic and parasitic life styles, and
- the phylogenetic relationship of the species of the query sequence and the reference sequence (in order to get information about the rate of change of these genomes since their descent from a common ancestor.)

Altogether, this requires the identification of coding regions (i.e., genes) as well as of non-coding regions and of their conserved orders (Overbeek et al., 1994; Bansal et al., 1998).

A reduction to the identification of orthologous regions based on the measurement of sequence similarity at this time of the analysis would discount the identification of, e.g., horizontal transfer events (i.e., xenologous regions) and gene families (i.e., paralogous regions). Moreover, the problems emerging by deriving evolutionary relationships like orthology simply by comparisons of sequence similarity have already been discussed in Section 3.2.1. Therefore, a less strict standard of local similarity in order to identify functional equivalence of regions is a prerequisite of such an approach. This in turn requires an approach capable of assessing local similarities for biological plausibility which relies not exclusively on their strengths (i.e., their lengths, similarity values, and their number of appearances in the genomes). For sequences such as long genomic sequences containing repeats (including gene duplications), a local similarity needs to be long, to have a high similarity value, and to have a low number of appearances in order to exhibit a significant strength. However, this limits the identification of short and less conserved regions of interest like regulatory regions.

The basic idea of the knowledge-based alignment approach is to use the genomic context of the local similarities which is determined by the reference sequence in order to check for their biological plausibility (see Proposition 1.1.1 in Section 1.1) (Wetjen, 2003). The term 'genomic context' refers here to the knowledge about the genome architecture, i.e., the spatial configuration of genomic features like gene order and the appearances of their regulatory regions within a genome sequence.

For example, let us assume that the order of all genomic features of a reference sequence is known, e.g., by extracting their locations (starting positions and end positions) from the corresponding genome entry taken from GenBank (Benson et al., 2003). Furthermore, let us assume that we have found a set of local similarities between the reference sequence and the query sequence by using a low strength threshold of the local similarities, where the strength is a value that increases with the number of matches and decreases with the number of

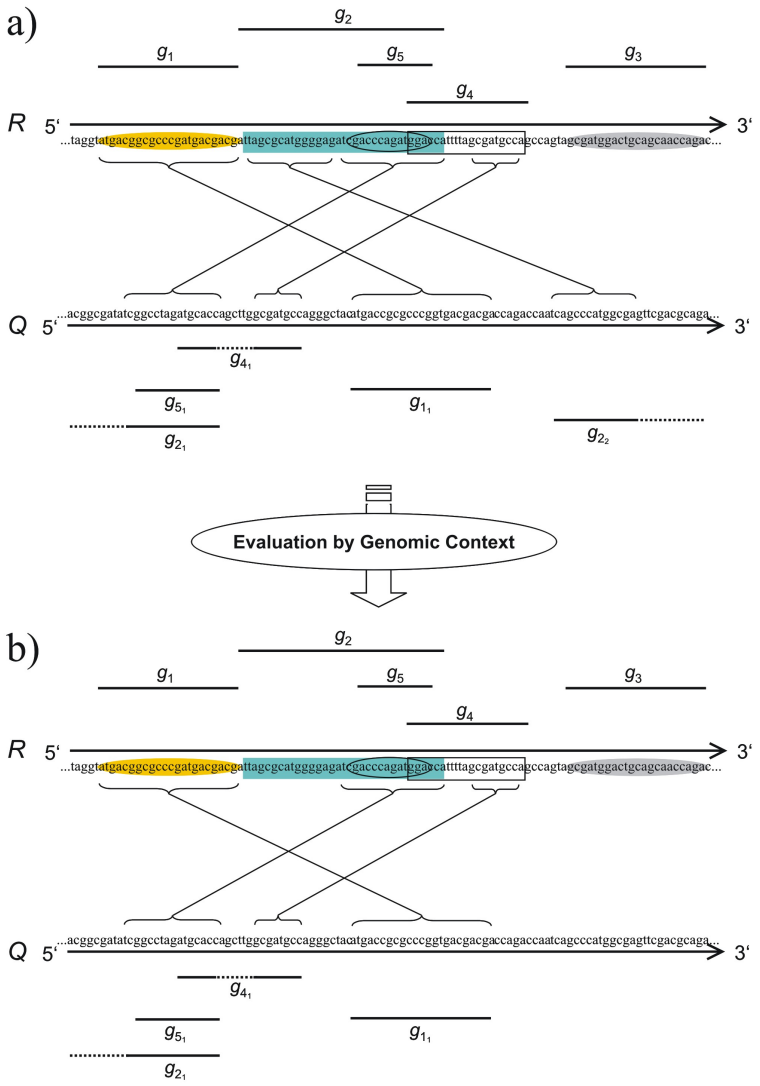


Figure 4.1: The basic idea of the knowledge-based alignment approach. a) A part of the reference sequence (R) containing five genomic features $\{g_1, \ldots, g_5\}$ and local similarity hits (indicated by the braces connected with the lines) to the query sequence (Q). b) The remaining local similarity hits after their evaluation (see text for explanation).

mismatches and gaps inside the alignment of a local similarity. As a result, this set will contain some strong similarities (e.g., very long ones with a high similarity value) and many weaker similarities (e.g., relatively short ones with a low similarity value). On the one hand, for a genomic feature in the reference sequence many hits to the query sequence could exist which would correspond to a one-to-many or even to a many-to-many allocation. On the other hand, for some

genomic features of the reference sequence no hit at all will exist, either because of the absence of this genomic feature in the query sequence or because of a lack of conservation of this feature on the DNA level. For the moment, however, let us consider each local similarity of this set to be a substring of exactly one genomic feature and that all local similarities associated with a single genomic feature are alternatives. Each local similarity within a genomic feature in the reference sequence implies the existence of a functionally equivalent genomic feature in the query sequence. Let us determine such a corresponding feature as a *putative* feature, since at the moment it remains unclear whether it is a correct hit and thus a true inference (Figure 4.1a).

The remaining question for this set of local similarities (putative genomic features of the query sequence) is which of them reflects a *true* functional equivalence. In order to assess these hits for biological plausibility, we check which of their combinations constitute a consistent spatial order in comparison to the reference sequence. Let us consider the example shown in Figure 4.1 which partially shows single strands of a reference sequence R and a query sequence Q. This part of the reference sequence contains five genomic features g_1 to g_5 and their spatial order could be denoted like

1. g_1 adjoins g_2,
2. g_5 is inside g_2,
3. g_4 is overlapped-by g_2 and overlapped-by g_5, and
4. g_3 is behind g_1, g_2, g_4, and g_5.

The set of local similarity hits between R and Q is indicated by braces connected by lines. As a result of the similarity search, the putative genomic features $g_{1_1}, g_{2_1}, g_{2_2}, g_{4_1}$, and g_{5_1} could be identified in Q, where their lowest index is a numeration of alternatives of the genomic feature, e.g., for g_2 in R there exist two alternative putative features g_{2_1} and g_{2_2} in Q. Note, that for the genomic feature g_3 no similarity hit exists. The two alternatives g_{2_1} and g_{2_2} are compared to the spatial order of the corresponding genomic features in R. Hence, the alternative g_{2_1} is selected, since a g_2 is not allowed to be located behind a g_1 (according to 1), but can contain a g_5 (according to 2) and can overlap a g_4 (according to 3) (Figure 4.1b).

However, if the genomic context is limited to the genome architecture of the reference sequence, this restricts the approach in its flexibility with respect to the evolutionary events which have shaped the genomes since their speciation. Therefore, the approach should be able to model a broader spectrum of genome architectures, e.g., of related species, and to decide during the process about its plausibility.

4.2 Formalizing Genome Architecture

There remain two research questions concerning the task of developing a knowledge-based alignment approach which is able to cope with the genome architecture of prokaryotic genomes, namely:

Question 4.2.1 *How can we represent knowledge about a genome architecture and reason about this knowledge?*

Question 4.2.2 *How can we acquire the knowledge of common genome architecture from real (multiple) genomes?*

We will answer Question 4.2.1 in the following Sections, whereas a method to answer Question 4.2.2 will be introduced in Chapter 5.

4.2.1 Evaluating Spatial Frameworks

With respect to known genome architectures of prokaryotes (see 2.2) and evolutionary forces shaping prokaryotic genomes (see 2.3), the requirements on a spatial framework (i.e., on their representation and reasoning abilities) are as follows:

- For the alignment, genome sequences are usually formalized as strings over an alphabet of letters (see 3.1.1), i.e., as a *one-dimensional* structure, even though, the majority of prokaryotic genomes sequenced so far is circular in its molecular structure (see 2.2.1).
- Prokaryotic genomes are shaped during evolution by processes of local mutations and global mutations which can result in highly variable genome architectures even within the same species. This requires a *qualitative representation* of genomic features, since a quantitative representation of concrete positions, lengths, and distances of genomic features would disregard these evolutionary processes.
- Each genomic feature is a substring of the complete genome (see 2.2.2.1 and Definition 4.2.1 in Section 4.2.2.1) which is not necessarily delimited by other genomic features, e.g., enclosures, intersections, and joints of genomic features frequently occur in prokaryotic genomes (see 2.2.2.1). In addition to the strict order of genomic features, the representation should admit modeling this kind of genome architectures.
- At present, the knowledge of genome architectures is almost limited to conserved gene clusters and all of these have been identified on subsets of the presently sequenced prokaryotic genomes which is a diminutive subset of all existing genomes in nature. In addition, some partial knowledge

exists for a few extensively studied regulation units and for some model organisms like *E. coli*. Therefore, representing and reasoning should be capable of dealing with *indefinite* knowledge of the spatial configuration of genomic features, i.e., in case the concrete spatial configuration between two genomic features is not known the approach should be able to model and reason on the set of all possible configurations in order to identify the subset of the feasible configurations.

- If the genome architecture of multiple prokaryotic genomes is compared different frequencies of spatial orders can be observed between genomic features. The weighting of these spatial relationships in order to generate an alignment could help to find the most likely assignment of local similarities while maintaining flexibility for unknown configurations.

- Prokaryotic genomes can have up to nearly 10,000 genes (see 2.2.2.3), and if their regulatory regions are included, the number of genomic features can be considerably higher, i.e., the reasoning component will have to deal with a large number of primitives and thus reasoning should be practicable for the approach described, i.e., it should be computable in a reasonable runtime.

Qualitative spatial frameworks (for an overview see (Cohn and Hazarika, 2001)) can cover different aspects of spatial representations like mereology (part to whole relationships), topology, orientation, and distance. In consideration of the requirements given, we will evaluate existing spatial knowledge representations and their reasoning abilities restricted to one-dimensional qualitative representations. Two kinds of such knowledge representations (an interval-based and a point-based framework) are well known in knowledge-based systems (for an introduction see (Beek, 1992)).

4.2.1.1 Point-Based Framework

The point-based framework of Vilain and Kautz (Vilain and Kautz, 1986) consists of three basic relations which can hold between two points p_i and p_j (the primitives in this framework) located on a directed line, namely *before* ($<$), *equals* ($=$), and *after* ($>$). Let P denote the set of all basic relations, indefinite information between two points can be represented as a disjunction of the basic relations. The set of all possible relations between two points is $\{\emptyset, <, \leq, =, >, \geq, \neq, P\}$, where $\leq$, for example, is an abbreviation of $\{<, =\}$. A network which represents points as nodes and where directed edges between nodes are labelled by sets of basic point relations, is called a point algebra network (PAN).

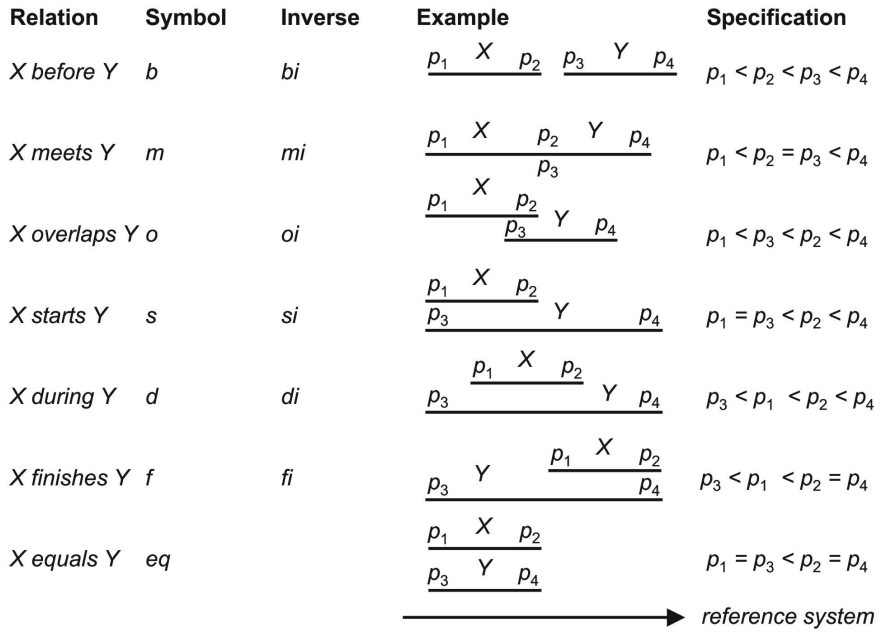


Figure 4.2: All 13 basic interval relations of Allen (Allen, 1983).

4.2.1.2 Interval-Based Framework

In Allen's (Allen, 1983) interval-based framework, intervals are the primitives. An interval is of the form of $p_i < p_j$ with p_i and p_j as points on a directed line. There are thirteen basic binary relations to describe the relative position of two intervals with respect to the reference system, namely *before* (b), *after* (bi), *meets* (m), *met-by* (mi), *overlaps* (o), *overlapped-by* (oi), *starts* (s), *started-by* (si), *finishes* (f), *finished-by* (fi), *during* (d), *contains* (di), and *equals* (eq) (Figure 4.2). We denote the set of all interval relations by $I = \{b, bi, m, mi, o, oi, s, si, f, fi, d, di, eq\}$. In order to represent indefinite information between two intervals, their relation is allowed to be a disjunction of the basic relations, i.e., any subset of I. For example, let A and B denote two intervals. In order to express that A is *before* or *after* B we write $A \overset{\{b,bi\}}{\longrightarrow} B$ which means $A \overset{\{b\}}{\longrightarrow} B \vee A \overset{\{bi\}}{\longrightarrow} B$. Since the 13 possible interval relations are mutually exclusive, there is no ambiguity about this notation. Networks with nodes representing intervals and labelled with subsets of I at their directed edges are called interval algebra networks (IAN) (Beek, 1992).

A restricted class of IANs - consisting of those subsets which can be expressed as conjunctions of the set of all possible point relations - can be translated into PANs without loss of information (Vilain and Kautz, 1986). However, a disjoint of two intervals, e.g., $A \overset{\{b,bi\}}{\longrightarrow} B$ (A is *before* or *after* B), cannot be expressed in this way (Beek, 1992), a restriction which will be important in the context of the intended biological application.

4.2.1.3 Reasoning in Spatial Frameworks

Given indefinite knowledge, there exist two main reasoning tasks: find all feasible relations between all pairs of primitives, and find a consistent scenario using the information provided. Both of them can be formalized for PANs and IANs using networks of binary constraints, i.e., a special case of constraint satisfaction problems (see Definition 4.4 in Section 4.4) (Beek, 1992; Dechter et al., 1991). For PANs there exist path consistency algorithms in order to find all feasible relations, and backtracking algorithms in order to find a consistent scenario based on this formalization which run in polynomial time (Beek, 1992). For IANs it was demonstrated that these problems are NP-hard (Vilain and Kautz, 1986; Golumbic and Shamir, 1993).

4.2.1.4 Applicability of Spatial Frameworks

In general, both frameworks are able to represent indefinite qualitative one-dimensional knowledge as required for our task. However, only the interval-based framework meets the requirements for the representation of genome architecture with respect to modeling the order, enclosure, intersection, and the joint of genomic features (as discussed in 4.2.1.2). Furthermore, only this framework enables to model a disjoint of genomic features (see 4.2.1.2) in order to express indefinite knowledge of their relative positions (i.e., accounting for permutations in the gene order). This framework has already been introduced to a problem of molecular biology of practical size, namely to check if the DNA molecule is linear in structure by modeling sequenced segments and testing whether they are disjoint or intersect (Golumbic and Shamir, 1993), and it has been demonstrated that reasoning is practicable for this problem (Beek and Manchak, 1996).

However, interval-based frameworks lack the ability to weight basic relations provided between two primitives with respect to, e.g., their frequencies of occurrence. This is of importance when different frequencies of relations, e.g., of genes in conserved clusters, are taken into account to find the most likely consistent scenario for the alignment. Thus, the existing spatial framework has to be extended with such a feature to meet this requirement.

4.2.2 Qualitative Modelling of Genome Architectures

We will show how the interval-based framework can be used in order to model genome architecture and how it is extended with weights of the spatial relationships reflecting their frequencies in a set of genomes.

4.2.2.1 Basic Reference Model

Let us denote the reference sequence for the alignment by R. Because of the DNA being a double stranded molecule (see 2.1), any genomic feature (see 2.2.2.1) of

a genome is located either on its plus or its minus strand (e.g., for a gene this constitutes its direction of transcription) with its complement located on the other strand. For modeling the qualitative genome architecture of genomic features, however, this direction can be neglected by using the complement of the genomic features of the minus strand. Usually, in data base entries (e.g., in GenBank) of a genome the sequence of the plus strand is stored and minus strand features are represented by its complementary sequence within the plus strand.

Before explaining how to model the genome architecture of the reference sequence, let us define more precisely some preconditions. First, let us define a genomic feature (see 2.2.2.1) as

Definition 4.2.1 *A **Genomic Feature** of a prokaryotic genome sequence s is a substring $s[i, j]$, with $1 \leq i < j \leq |s|$, which carries a specific biological function.*

Hence, each genomic feature represents an interval of the interval-based framework (see 4.2.1.2), i.e., it has a defined starting point i and an end point j with respect to a reference system which in this case is the string s of the plus strand of a genome inscribed in the 5' to 3' direction of length ≥ 2. Note, that in contrast to the substring definition used (see 3.1.1), a genomic feature requires the index i to be smaller than the index j in order to avoid any genomic feature to be a single nucleotide, since all genomic features known so far (see 2.2.2.1) consist of more than one base.

Second, let us denote the set of all basic interval relationships by

$$I = \{b, bi, m, mi, o, oi, s, si, f, fi, d, di, eq\},$$

and with $r \in I$ we denote a single interval relation. The function *inverse* (inv)

(4.1) $inv : I \to I,$

r	b	m	o	s	f	d	bi	mi	oi	si	fi	di	eq
$inv(r)$	bi	mi	oi	si	fi	di	b	m	o	s	f	d	eq

maps each relation onto its inverse relation. Regarding the domain specific reference system, some of the specifications of the basic interval relationships need to be adapted. In particular, the relation *meets* between two substrings $s[i, j]$ and $s[k, l]$ requires that the letters at the indices $s(j)$ and $s(k)$ are adjacent unlike the original specification where the end point p_j and the starting point p_k of two intervals would be situated at the equal coordinate (Figure 4.3).

In order to model the spatial configuration of the genomic features of the reference sequence R and, in addition, to introduce further configurations of biologically related sequences, we will use an IAN as introduced in Section 4.2.1.2. However, we will define here this framework more precisely as a graph data structure.

Let us describe the finite set of genomic features of R as a set of nodes $G = \{g_1, \ldots, g_n\}$, with $n \in \mathbb{N}_0$. By $\mathbb{N}_0$ we denote the set of natural numbers including zero. Between each pair of nodes $(g_i, g_j) \in G$, with $1 \leq i < j \leq |G|$, directed

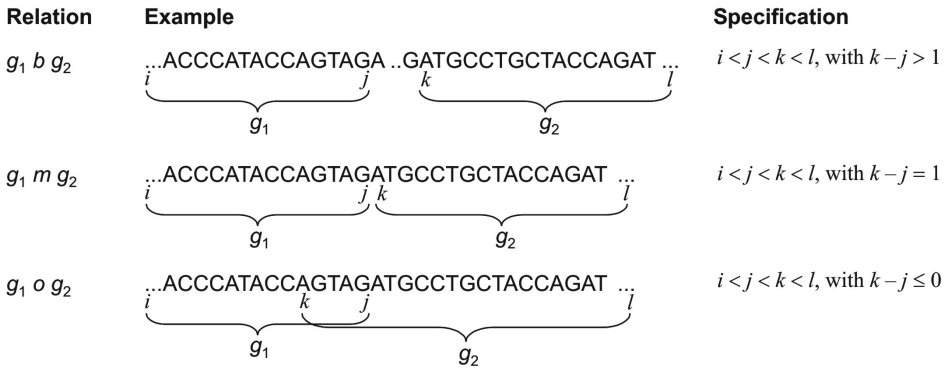


Figure 4.3: For DNA adapted specifications of the interval relationships *before* (b), *meets* (m), and *overlaps* (o).

edges (i, j) are introduced. Each edge is labelled with a set $C_{ij} \subseteq I$, i.e., any possible combination of the basic interval relations can constitute a set C_{ij}.[1] For each such pair of genomic features (g_i, g_j) there exists exactly one basic relation $r \in I$ corresponding to their spatial order in R, i.e., initially each set C_{ij} contains exactly one element.

Note, that in order to save memory space we always model only one directed edge (i, j) between two nodes g_i and g_j, because the inverse direction (j, i) would be labelled with the inverse of the label C_{ij} which can be computed using function *inv* (see Equation 4.1). Furthermore, we do not model directed edges (i, i), because they would always be labelled with the identity relation *eq*.

Definition 4.2.2 *A **Basic Reference Model** is a directed graph of the form* $\mathcal{R} = \langle G, H, C, c \rangle$*, where*

- G *is a set of nodes representing the genomic features of the reference sequence* R*,*
- $H = \{(i, j) | 1 \leq i < j \leq |G|\}$ *is a set of directed edges, and*
- $c : H \rightarrow C$ *is a function from each edge* (i, j) *to a set* $C_{ij} \subseteq I$*, with* $C_{ij} \in C$ *where* I *is the set of all basic interval relations as defined in Section 4.2.1.2.*

An example of a basic reference model generated for our previously used example of Figure 4.1 can be found in Figure 4.4b.

4.2.2.2 Weighted Reference Model

In order to account for different frequencies of basic interval relations (see 4.2.1.2) between genomic features (see 4.1 for explanation), a reference model is extended

[1] Remember that a set of basic relations corresponds to their disjunction (see 4.2.1.2).

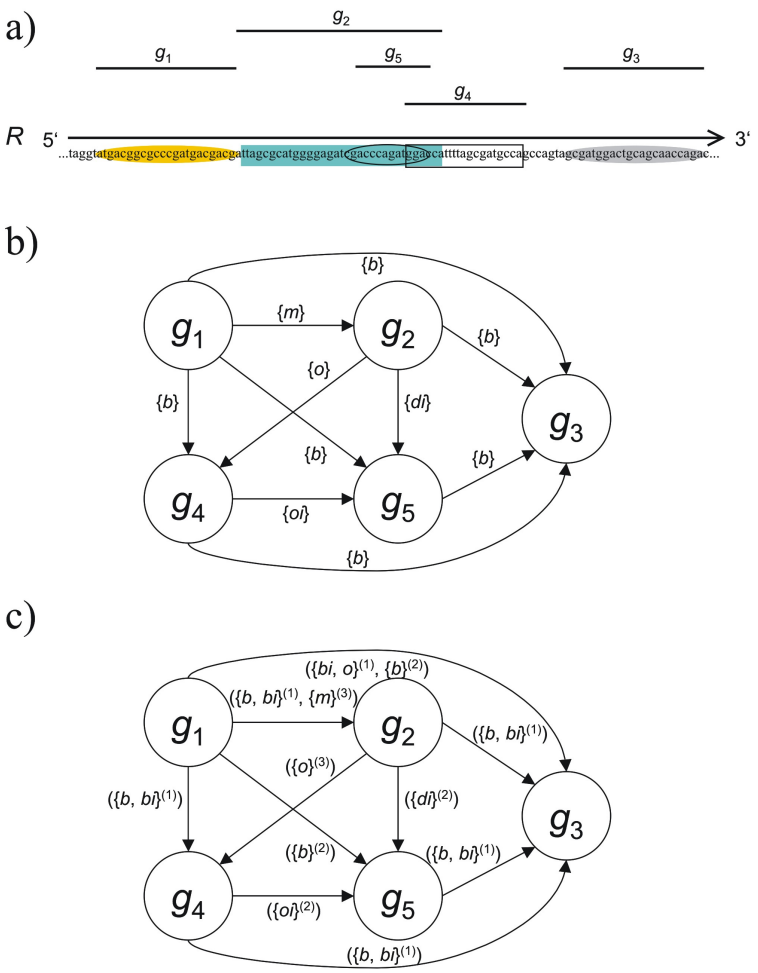


Figure 4.4: Graphical representation of a reference model. a) A schematic view of a part of a reference sequence. b) A graphical view of a basic reference model representing the five genomic features of a). c) A weighted reference model for a) containing further knowledge (empty relation sets in a vector are not shown).

with an ordering of the labels at its directed edges which reflects the weight of a label, i.e., of this set of basic interval relations. Therefore, the spatial relations at each edge (i, j) are organized as a vector $V_{ij} = (V_{ij}^{(1)}, \ldots, V_{ij}^{(k_{max})})$, where each $V_{ij}^{(k)} \subseteq I$ with $V_{ij}^{(k)}, V_{ij}^{(k')} \in V_{ij} : V_{ij}^{(k)} \cap V_{ij}^{(k')} = \emptyset$, and $k_{max} \leq 13$, because there exist 13 basic interval relations and thus maximal 13 different such elements. The weight of an element $V_{ij}^{(k)}$ is given by $w(V_{ij}^{k}) = k$. Each weight represents the importance of that element $V_{ij}^{(k)}$, ranging from the weakest $V_{ij}^{(1)}$ to the strongest $V_{ij}^{(k_{max})}$.

Definition 4.2.3 *A* ***Weighted Reference Model*** *is a directed graph of the*

form $\mathcal{R}_\diamond = \langle G, H, V, h \rangle$*, where*

- G *is a set of nodes representing the genomic features of the reference sequence* R*,*
- $H = \{(i,j) | 1 \leq i < j \leq |G|\}$ *is a set of directed edges, and*
- $h : H \rightarrow V$ *is a function from each edge* (i,j) *to a vector* $V_{ij} \in V$*.*

An example of a weighted reference model generated for our previously used example of Figure 4.1 containing further knowledge can be found in Figure 4.4c.

4.2.2.3 Integrating Knowledge into a Reference Model

Initially, a reference model represents only the genome architecture of the genomic features of the reference sequence R. An evaluation of putative genomic features in Q on this basis would be very strict, since only a set of putative features describing exactly this genome architecture would be allowed to constitute the alignment which for phylogenetically further distanced species is rarely the case (see 2.2). Therefore, more knowledge can be introduced into a reference model either by an expert, or by extracting it automatically from publicly available data bases. However, the manual introduction of knowledge is impractical considering the large number of genomic features that can occur in prokaryotic genomes (see 2.2.2). For instance, in order to identify locally conserved gene orders in a set of genomes which includes R (referring to Question 4.2.2 in Section 4.2), the approach described in the next Chapter 5 can be used. However, here we are not concerned with the sources of the knowledge, but in which way to integrate it into a reference model.

For a basic reference model $\mathcal{R}$, more spatial knowledge can simply be integrated by adding to a set C_{ij} more relations $r \in I$, i.e., representing alternative genome architectures of the genomic features g_i and g_j. This can be done for any pair of genomic features $(g_i, g_j) \in G$ of $\mathcal{R}$.

For a weighted reference model $\mathcal{R}_\diamond$, a relation $r \in I$ to be introduced between two genomic features g_i and g_j needs to be mapped to a weight w. On the one hand, the expert introducing the knowledge could carry out this mapping. On the other hand, if searching in a set of genomes for the frequency $fr(r)$ (see Equation 5.1 in Section 5.3) of the relation r between g_i and g_j a domain dependent function $f : F \rightarrow W$ can be specified, where F is the set of all possible frequencies and $W = \{1, \ldots, 13\}$ is the set of all possible weights. This function has to be monotone so that the value of the weight w increases with increasing frequency $fr(r)$. The relation r is then introduced into the corresponding element $V_{ij}^{(k)} \in V_{ij}$. For example, between gene g_i and g_j in a set of ten genomes the following interval relations have been found: four times *before* with $fr(b) = 0.4$, three times *meets* with $fr(m) = 0.3$, and two times *after* with $fr(bi) = 0.2$. If

the function f maps frequency $0.2 \mapsto 2$, $0.3 \mapsto 2$, and $0.4 \mapsto 4$ this results in a $V_{ij} = (V_{ij}^{(2)} = \{m, bi\}, V_{ij}^{(4)} = \{b\})$.

The general knowledge introduced can keep the flexibility of gene order of unknown genomes, e.g., by introducing the relation set $\{b, bi\}$ between all pairs of genes in order to allow for all permutations to occur. However, such knowledge should generally be of minor weight than knowledge generated from real data. Altogether, this allows for a flexible knowledge adjustment with respect to the analysis task (e.g., regarding the taxonomically composition of the genomes for the frequency measurements).

4.2.2.4 Ensuring Consistency of the Reference Model

In order to generate a biologically feasible alignment, prior to the usage of a reference model the interval path consistency has to be guaranteed. Inconsistencies can be caused by partial knowledge refinements, i.e., a relation set between two nodes g_i and g_j was either extended or reduced during the knowledge integration step (4.2.2.3). All feasible relations between all pairs $(g_i, g_j) \in G$, with $1 \leq i < j \leq |G|$, have to be found. For this task, in the field of temporal reasoning path consistency or transitive closure algorithms are used (Beek and Manchak, 1996).

The basic idea of the interval path consistency algorithm (Algorithm 1) is that for any three nodes g_i, g_j, and g_k in the graph, the labels on the edges (i, j) and (j, k) potentially constrain the label on edge (i, k) that completes the triangle. Concerning a basic reference model $\mathcal{R}$, if $g_i \overset{\{b\}}{\longrightarrow} g_j$, $g_j \overset{\{d\}}{\longrightarrow} g_k$, and $g_i \overset{\{b,bi,m,mi\}}{\longrightarrow} g_k$, then it is inferred that the relation set between g_i and g_k can be restricted to $g_i \overset{\{b\}}{\longrightarrow} g_k$. Let us denote by $\otimes$ the composition of labels C_{ij} and C_{jk}. In order to perform the deduction, the algorithm checks whether

$$C_{ik} \leftarrow C_{ik} \cap (C_{ij} \otimes C_{jk}), \tag{4.2}$$

The composition for labels of single interval relationships is a mapping given by the table in Figure 4.5. The composition of two labels C_{ij} and C_{jk} is then computed by the union of the composition for each $r \in C_{ij}$ with each $r \in C_{jk}$.

If a relation is deleted from the set C_{ik} by Equation 4.2, this may in turn constrain the labels at other edges. Therefore, the interval path consistency algorithm iterates until no more changes are possible. A detailed description of the basic algorithm can be found in (Allen, 1983). Here we describe this algorithm with the required adaptation in order to handle a weighted reference model $\mathcal{R}_\diamond$. In addition, the algorithm is extended by some heuristic skipping techniques proposed by Beek and Manchak (Beek and Manchak, 1996).

In a weighted reference model, the basic interval relations are organized in sets ordered by their weights. This ordering, however, has no influence on the consistency of $\mathcal{R}_\diamond$, i.e., we can ignore this ordering for the task of guaranteeing the consistency of the model. For instance, even if a relation has a high weight

A→B / B→C	b	m	o	s	d	f	eq	fi	di	si	oi	mi	bi
b	b	b	b	b	b m o s d	b m o s d	b	b	b	b	b m o s d	b m o s d	I
m	b	b	b	m	o s d	o s d	m	b	b	m	o s d	f eq fi	di si oi mi bi
o	b	b	b m o	o	o s d	o s d	o	b m o	b m o fi di	o fi di	o s d f eq fi di	di si oi	di si oi mi bi
s	b	b	b m o	s	d	d	s	b m o	b m o fi di	s eq si	d f oi	mi	bi
d	b	b	b m o s d	d	d	d	d	b m o s d	I	d f oi mi bi	d f oi mi bi	bi	bi
f	b	m	o s d	d	d	f	f	f eq fi	di si oi mi bi	oi mi bi	oi mi bi	bi	bi
eq	b	m	o	s	d	f	eq	fi	di	si	oi	mi	bi
fi	b	m	o	o	o s d	f eq fi	fi	fi	di	di	di si oi	di si oi	di si oi mi bi
di	b m o fi di	o fi di	o fi di	o fi di	o s d f eq fi di	di si oi	di	di	di	di	di si oi	di si oi	di si oi mi bi
si	b m o fi di	o fi di	o fi di	s eq si	d f oi	oi	si	di	di	si	oi	mi	bi
oi	b m o fi di	o fi di	o s d f eq fi di	d f oi	d f oi	oi	oi	di si oi	di si oi mi bi	oi mi bi	oi mi bi	bi	bi
mi	b m o fi di	s eq si	d f oi	d f oi	d f oi	mi	mi	mi	bi	bi	bi	bi	bi
bi	I	d f oi mi bi	d f oi mi bi	d f oi mi bi	d f oi mi bi	bi	bi	bi	bi	bi	bi	bi	bi

Figure 4.5: Composition table for all 13 basic interval relations. An entry in a cell of the table represents the composition of relations allowed between genomic features $A \xrightarrow{r} C$ resulting from the transitivity of the relations between $A \xrightarrow{r} B$ and $B \xrightarrow{r} C$.

this label will never be part of a solution if inconsistent with other relations. Therefore, all sets of basic relations $V_{ij}^{(k)} \in V_{ij}$ at an edge (i, j) can be combined into a single label C_{ij} by the union of these sets:

$$C_{ij} \leftarrow \bigcup_{k=1}^{k_{max}} V_{ij}^{(k)}. \tag{4.3}$$

Using the example of Section 4.2.2.3 where $V_{ij} = (V_{ij}^{(2)} = \{m, bi\}, V_{ij}^{(4)} = \{b\})$, then Equation 4.3 will return the label $C_{ij} = \{m, bi, b\}$ for this edge. We call the function of Equation 4.3 *GetEdgeLabel*. In order to proceed for a basic reference model, a function *GetEdgeLabel* would just return the label C_{ij}.

The skipping heuristics proposed by Beek and Manchak (Beek and Manchak, 1996) are based on the observation that the computation of Equation 4.2 can be skipped if the composition of the labels $C_{ij} \otimes C_{jk}$ will not constrain the label C_{ik}:

1. If the condition,

$$(b \in C_{ij} \wedge bi \in C_{jk}) \vee (bi \in C_{ij} \wedge b \in C_{jk}) \vee (d \in C_{ij} \wedge di \in C_{jk})$$

   is *true*, the result of the composition will be I (see Figure 4.5). This condition is tested in steps 12 and 21 (here C_{ij} is replaced by C_{ki} and C_{ki} is replaced by C_{ij}) of Algorithm 1 by the function called *Skipping*.

2. If either C_{ij}, C_{jk}, or $C_{ki} = I$ the result of the composition will be I. This condition is tested in steps 5, 11, and 20 of Algorithm 1.

The interval path consistency algorithm (Algorithm 1) generates a local consistency (a so called 3-consistency), since only triangles of edges are validated. On the one hand, the algorithm may return that a reference model is consistent when it is actually not (Allen, 1983; Beek and Manchak, 1996). On the other hand, 3-consistency can be used as a pruning technique prior to the search for a solution and because it is known to be the least expensive pruning technique for IANs (Ladkin and Reinefeld, 1992; Beek and Manchak, 1996).

The application of the interval path consistency algorithm, however, can be very time consuming for a reference model since there are at least

$$\binom{n}{2} = \frac{n!}{2 \cdot (n-2)!} = \frac{n \cdot (n-1)}{2} = O(n^2)$$

edges, with n being the number of nodes (genomic features) in the reference model which have to be tested for interval path consistency (referring to step 2 and 3 of Algorithm 1). This number does not account for further iterations required through the deletion of a relation from a label! Consider for example a reference model containing all $n = 4242$ genes of *Escherichia coli* K12 (see Table 7.2 in Section 7.1.3). Ensuring its interval path consistency requires at least 8,995,161 iterations. However, since genes in prokaryotes are mostly disjoint (see 2.2.2.4), i.e., their spatial configurations can be modeled by the interval relations *before* and *after*, its is likely that the first skipping heuristic can be applied and thus a minor number of further iterations caused by the deletion of a relation from a set is usually required. Furthermore, this consistency needs to be ensured only once for a reference model after its generation (until further knowledge is introduced), and then it can be used for many analysis.

Algorithm 1 *Interval Path Consistency*($\mathcal{R}_\diamond$)

Input: $\mathcal{R}_\diamond = \langle G, H, V, h\rangle$ \\ a weighted reference model
Output: An interval path consistent reference model.

```
1: L ← () \\ new empty list
2: for i ← 1, ..., (|G| − 1) do
3:    for j ← (i + 1), ..., |G| do
4:       C_ij ← GetEdgeLabel((i, j)) \\ see Equation 4.3
5:       if C_ij ≠ I then
6:          L ← L ∪ {C_ij}
7:          while L ≠ ∅ do
8:             C_ij ← L(0)
9:             L ← L\C_ij \\ delete C_ij from L
10:            for k ← 1, ..., |G|, k ≠ i ∧ k ≠ j do
11:               if C_jk ≠ I then
12:                  if Skipping(C_ij, C_jk) then
13:                     R_ik ← C_ik ∩ (C_ij ⊗ C_jk) \\ see Equation 4.2
14:                     if R_ik ≠ C_ik then
15:                        C_ik ← R_ik
16:                        L ← L ∪ {C_ik}
17:                     end if
18:                  end if
19:               end if
20:               if C_ki ≠ I then
21:                  if Skipping(C_ki, C_ij) then
22:                     R_kj ← C_kj ∩ (C_ki ⊗ C_ij)
23:                     if R_kj ≠ C_kj then
24:                        C_kj ← R_kj
25:                        L ← L ∪ {C_kj}
26:                     end if
27:                  end if
28:               end if
29:            end for
30:         end while
31:      end if
32:   end for
33: end for
```

4.3 Identifying Putative Genomic Features

The standard approach used in bioinformatics which is based on the concept of sequence similarity (3.1.2) is used in order to identify functionally equivalent genomic features between the reference sequence and the query sequence. Beyond functional equivalence, sequence similarity could even indicate homology (see Section 2.2.2.2) or orthology (see Section 2.2.2.2). Such identified genomic features

in the query sequence are marked as *putative*, because there exists no further proof (using laboratory techniques like expression profile analysis for genes) of their authenticity at this point of the analysis.

When comparing non-collinear genomes, however, less conserved regions will frequently appear in between conserved regions caused by global mutations and local mutations (see 2.3). Therefore, some genomic features of the reference sequence may be represented by more than one local similarity hit to the query sequence, i.e., conserved parts of it which have to be combined in order to represent the putative genomic feature in the query sequence. Furthermore, if such less conserved regions occur at the borders of a genomic feature, we would like to identify its starting and end indices in the query sequence. On the one hand, this gives a better prediction for further analysis, on the other hand, this ensures crisp arguments about its genomic context in order to validate its biological plausibility. Thus, the identification of functionally equivalent genomic features is performed in three steps:

1. a local similarity search between the reference and the query sequence using a local alignment technique,

2. collecting and chaining of all local similarities corresponding to a genomic feature in the reference sequence in order to assemble putative genomic features for the query sequence, and

3. an extension of putative genomic features with (putative) starting indices and end indices if not already represented by one of its local similarities.

In the following, this approach is described only for the plus strand (the one annotated in a genome entry at GenBank). However, for the minus strand the principle remains the same with the exception that starting and end indices are first converted to their analogs in the plus strand. In consequence, the complements of start and stop codons (e.g., TAG becomes CTA) are searched in their inverse order (e.g., start codons to the right).

4.3.1 Searching Local Similarities

There exist many local alignment (see 3.1.3.2) approaches to find local similarities between large sequences (compare 3.2). Formally, a local similarity is given by

Definition 4.3.1 *A **Local Similarity** between two sequences R and Q consists of a pair of substrings $R[b, e]$ and $Q[b', e']$ together with their alignment $\mathcal{A}$ and its corresponding similarity value c. We denote a local similarity by the tuple $Sim = \langle R[b, e], Q[b', e'], \mathcal{A}, c \rangle$.*

Note that the characteristics of the alignment and its similarity value depend on the cost scheme and gap penalties used (see 3.1.2.1) and that the substrings

in R and Q need not necessarily the same length, since gaps could be included in either one or both of them in order to maximize the similarity value (see 3.1.3).

In order to find all local similarities between R and Q, the stand alone BLAST version 2.2.6 (Altschul et al., 1997) (see Section 3.2.2.1 for a detailed description) is used. Equivalent to the described strength threshold for a local similarity, an expectation value cutoff (sometimes called E-value or Expect) is used. By lowering the E-value, the significance of the local similarities is increased, i.e., we implicitly allow for duplications, reduced lengths of hits, and more mismatches to occur. In the following, we will use the terms *hit* and *local similarity* interchangeably.

4.3.2 Assembling Putative Genomic Features

The local similarities identified by BLAST have to be assembled to putative genomic features representing the matches to the genomic features in the reference sequence (i.e., in the reference model). Let us define such a *putative* genomic feature more precisely by

Definition 4.3.2 *A **Putative Genomic Feature** (PGF) is a triple* $\mathcal{P} = \langle \mathcal{C}, start, end \rangle$*, where*

- $\mathcal{C}$ *is a set of local similarities called a chain, where similarities are ordered in ascending (5' to 3' direction) qualitative order of their appearance in both sequences (sometimes referred to as relative order) and*
- *start and end are indices which determine the upstream and downstream border in 5' to 3' direction in* Q*, respectively, with* $1 \leq start < end \leq |Q|$.

4.3.2.1 Collecting and Chaining Local Similarities

Remember that each node $g \in G$ represents a genomic feature which is a substring $R[i, j]$; and that each local similarity Sim contains two substrings $R[b, e]$ and $Q[b', e']$. Let us denote the resulting set of all m local similarities found by $S = \{Sim_1, \ldots, Sim_m\}$. For each genomic feature $g \in G$ we generate a set D of PGFs which may possibly be empty if no similarity hits exist. This set D associated to a genomic feature g is called its *domain*. In order to belong to the same PGF, local similarities have to occur in the same qualitative order in both sequences, and the length of their PGF in Q has to be almost identical to the length of the corresponding genomic feature.

The algorithm to assemble all PGFs (Algorithm 2) from local similarities in S starts to collect for each genomic feature $g \in G$ all $Sim \in S$ in an set F whose $b < j \wedge e > i$, i.e., all local similarities from S which at least partially represent this genomic feature in R are collected (Algorithm 2 step 7). This set is sorted in ascending order of their starting indices b' in Q (Algorithm 2 step 12).

Algorithm 2 *Assembling*(S, G)

Input: S \\ a set of local similarities
G \\ the set of genomic features of R
Output: $\mathcal{D}$ \\ a set of domains containing for each $g \in G$ a domain D

1: **for** $x \leftarrow 1, \ldots, |G|$ **do**
2: $g_x \leftarrow G(x)$
3: $F \leftarrow \{\}$ \\ new empty set
4: $D_x \leftarrow \{\}$ \\ new empty domain
5: **for** $y \leftarrow 1, \ldots, |S|$ **do**
6: $Sim_y \leftarrow S(y)$
7: **if** $b < j \wedge e > i$ **then**
8: $F \leftarrow F \cup \{Sim_y\}$
9: **end if**
10: **end for**
11: **if** $F \neq \emptyset$ **then**
12: $Sort(F)$ \\ ascending w.r.t. their starting indices b' in Q
13: $D_x \leftarrow Chaining(F, g_x)$ \\ see text for explanation
14: **end if**
15: $\mathcal{D} \leftarrow \mathcal{D} \cup \{D_x\}$
16: **end for**

Subsequently in step 13 of this algorithm, the domain D of the genomic feature is created by the function $Chaining(F, g)$ described in the following.

During the *Chaining*, each $Sim \in F$ is proceeded in order to generate PGFs where the following cases can be distinguished:

1. case: $b \leq i < j \leq e$ (Figure 4.6a)

   This local similarity completely represents this genomic feature. Therefore, the current PGF $\mathcal{P}_{cur}$ is 'closed', added to the domain D of g, and a new $\mathcal{P}_{cur}$ is initialized. In case the local similarity exceeds the genomic feature (either upstream or downstream, or both), the similarity is partitioned into parts (substrings) of the form

   - $R[i, j]$,
   - $R[b, (i-1)]$, and
   - $R[(j+1), e]$,

   hence constituting separated similarities. Note that string operations are here only described for the substrings of the reference sequence, but have to be conducted in similar fashion for the query sequence. Furthermore, in case one or both of the substrings of a local similarity contain gaps (see 3.1.3), the indices are adapted by the number of gaps in this substrings.

The Sim with $R[i,j]$ is added to the chain of $\mathcal{P}_{cur}$. Furthermore, $\mathcal{P}_{cur}$ is added to the domain D of g and is not considered for further extensions. A new $\mathcal{P}_{cur}$ is initialized. The other similarities are added to a set denoted as $Putty$, i.e., in this set we collect all local similarities which do not hit any genomic feature of the reference sequence. These similarities can later be used to close gaps between the local similarities of the validated PGFs (see 4.5). Proceed to the next Sim.

2. case: $b \leq i < e < j$ (Figure 4.6b)

   This local similarity starts a new PGF, i.e., the current PGF $\mathcal{P}_{cur}$ is 'closed', added to the domain D of g, and a new $\mathcal{P}_{cur}$ is initialized. Again, the similarity is partitioned if necessary (see above) and the Sim with $R[i,j]$ is added to the chain of $\mathcal{P}_{cur}$. Proceed to the next Sim.

3. case: $i < b < e < j$ (Figure 4.6c)

   This local similarity either extends $\mathcal{P}_{cur}$ or it starts a new PGF. We test if the qualitative order and length conditions described are both satisfied. If

   - $b_{pre} < b$,

     where b_{pre} denotes the starting index of the previous Sim in R, the qualitative order of the previous and the current local similarity are the same in R,

   - $b'_{pre} < b'$,

     where b'_{pre} denotes the starting index of the previous Sim in Q, the qualitative order of the previous and the current local similarity are the same in Q, and

   - $(|R[i,j]| + x) \leq (b' + b'_{pre})$,

     where x is a specified tolerance of bases which a putative genomic feature is allowed to be larger in Q

   this Sim extends $\mathcal{P}_{cur}$ and is added to its chain. The qualitative order is thus equivalent to the *total order* of two Sim. Otherwise, the current PGF $\mathcal{P}_{cur}$ is 'closed', added to the domain D of g, and a new $\mathcal{P}_{cur}$ is initialized with this Sim. Proceed to the next Sim.

4. case: $i < b < j \leq e$ (Figure 4.6d)

   This local similarity either extends $\mathcal{P}_{cur}$ or it starts a new PGF. However, in both cases this Sim results in the 'closing' of $\mathcal{P}_{cur}$ and the initialization of a new one. If both above described conditions are satisfied, this Sim extends $\mathcal{P}_{cur}$, otherwise it constitutes an own PGF.

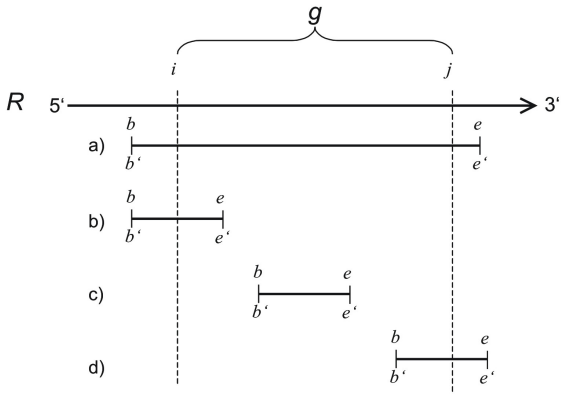


Figure 4.6: The four possibilities of how a PGF $\mathcal{P} = \langle \mathcal{C}, start, end \rangle$ of the query sequence Q may cover a genomic feature g in the reference sequence R.

Finally, all $Sim \in S$ which are not considered by the above algorithm, since they occurred in between the genomic features contained in G, are added to the set $Putty$.

4.3.2.2 Locating Putative Borders

During the search for a biologically feasible alignment it is tested for each PGF whether the relation between this feature and any other feature in the query sequence is a relation allowed between them by looking up the relations modeled in the reference model. In case that a PGF represents just a part of a genomic feature of the R this might cause imprecisions, e.g., if two genomic features in the reference sequence have the relation *meets*, a partial representation of the corresponding PGFs could have the relation *before*. If compared to the reference model, this relation might either not be included or it might be of a minor weight in the set of relations between these two features. Hence, identifying the 'real' borders of the PGFs in the query sequence allows for more precision and, thus, a biologically reasonable evaluation of PGFs.

Let us denote the starting index of a genomic feature g in the reference sequence R by i and the end index by j. Let us denote the starting index of the **first** local similarity Sim_1 of a chain $\mathcal{C}$ in the reference sequence R by b and in the query sequence Q by b'; correspondingly we denote the end index of the **last** local similarity $Sim_{|\mathcal{C}|} \in \mathcal{C}$ in R by e and in Q by e'. There are two possibilities of how a PGF may represent a genomic feature of the reference sequence:

1. **The PGF completely represents the genomic feature**. This is the case *iff*

   - $i = b < e = j$,

i.e., the borders of the chain are aligned to the defined starting and end index of the genomic feature in R. This representation of a genomic feature by a PGF still allows for (larger) gaps to occur between the local similarities.

2. **The PGF partially represents the genomic feature**. This is the case either *iff*

   - $i = b < e < j$,
   - $i < b < e = j$, or
   - $i < b < e < j$.

For PGFs representing a part of a protein gene (see 2.2.2.1), the identification of starting and end indices is a simple search for the corresponding start and end codon (this task is sometimes called orf finding). The algorithm used here is defined as follows:

1. Search for putative start codons (ATG or alternatively for GTG, TTG, ATT, and CTG), stop codons (TAA, TAG, TGA), or both in the query sequence Q by a simple string comparison. The search for start and stop codons is restricted to a region in Q close to the corresponding region of the start and stop codon in R. The borders of this search regions are calculated by

   Start Codon Search Region: $i' \leftarrow (b' - (b - i)) - x$; $j' \leftarrow i' + (2 \cdot x)$,

   Stop Codon Search Region: $j' \leftarrow (e' - (j - e)) + x$; $i' \leftarrow j' - (2 \cdot x)$,

   where x is a specified tolerance of bases upstream and downstream which are searched in Q for the codons (the same x as used for the assembly algorithm, Algorithm 4.3.2). This search results in two separate sets, one of starting indices for putative start codons and one of putative stop codons, both of which may be empty.

2. All combinations of putative start and stop codon indices are tested whether they constitute a valid reading frame. A combination is considered as valid *iff*

   - (*end codon index* - *start codon index*) mod 3 = 0, where mod is the modulo function.

3. From these valid combinations, the combination whose total length comes closest to the original length of the protein gene in the reference sequence is chosen. In case no valid combination could be found (either because of

no valid reading frame was found or no start or stop codon was identified), the borders of the PGF are set to either $start \leftarrow b'$ or $end \leftarrow e'$, or both, respectively.

For other genomic features such as regulatory regions (2.2.2.1) this task is more difficult, since they are usually conserved to a much lesser extent on the DNA level than protein genes. There exist data mining approaches to identify such features (e.g., (Pritsker et al., 2004)) which could be used in this context or already annotated data (e.g., for *E. coli* in the RegulonDB (Salgado et al.,)), however, their integration into this approach remains an interesting topic for future work, since their spatial configuration is as well constrained by their biological functions and thus could help to increase the precision of the reasoning task.

4.4 Evaluating Putative Genomic Features

As mentioned in Section 4.2.1.3, reasoning about the interval-based framework can be formalized as a constraint satisfaction problem of binary constraints, where constraints (the basic interval relations) are defined between pairs of variables (pairs of intervals). Such a constraint satisfaction problem is defined as

Constraint Satisfaction Problem (CSP): A CSP consists of a finite set of variables $X = \{x_1, \ldots, x_n\}$, each associated with a nonempty domain of discrete values $D_1, \ldots, D_n$ and a set of binary constraints on these variables. The domain D_i of a variable x_i is the finite set of values to which this variable may be instantiated. A binary constraint C_{ij} between two variables x_i and x_j is a relation $C_{ij} \subseteq D_i \times D_j$, i.e., a subset of the Cartesian product of the values of their domains. An *assignment* of a unique domain value to each member of some subset of variables is called an *instantiation*. An instantiation is said to *satisfy* a constraint C_{ij} if this partial assignment does not violate this constraint. An instantiation is said to be *locally consistent* if it satisfies all constraints of the subnetwork. A *solution* of the constraint satisfaction problem is a instantiation of all variables of X.

Any reference model, either a basic one $\mathcal{R}$ (see 4.2.2.1) or a weighted one $\mathcal{R}_\diamond$ (see 4.2.2.2), can be considered to be a CSP. Actually, the graph data structure of a reference model corresponds directly to the graphical representation of a CSP called constraint graph (see, e.g., (Dechter, 2003)). A binary constraint between a pair of genomic features $(g_i, g_j) \in G$ is given by the set of basic relations C_{ij} annotated at the directed edge (i, j) which specifies the valid combinations of values of their domains, i.e., $C_{ij} \subseteq D_i \times D_j$. For a weighted reference model $\mathcal{R}_\diamond$ the constraints between variables $(g_i, g_j) \in G$ are organized in a vector $V_{ij} = (V_{ij}^{(1)}, \ldots, V_{ij}^{(k_{max})})$ according to their weight reflecting their importance. Thus, any $V_{ij}^{(k)} \in V_{ij}$ constitutes a constraint of the form $V_{ij}^{(k)} \subseteq D_i \times D_j$.

Let G again denote the set of all genomic features g of the reference genome R (see 4.2.2). The possibility to identify at least one PGF $\mathcal{P}$ for each genomic feature $g \in G$ decreases when comparisons among phylogenetically further distanced species are made even with a high E-value cutoff (i.e., a lowered strength threshold). This is either because of the two genomes sharing only a subset of their genes (see 2.2.2.3) or because the evolutionary pressure for a gene was not on the DNA but on the protein level (i.e., many silent or neutral mutations, see 2.3.1, have occurred on the DNA level since their speciation). Therefore, only those genomic features of a reference model are considered to be variables of the CSP for which at least one PGF was identified between the reference sequence R and the query sequence Q, i.e., for which its domain D is not empty. In the case of no or only one similarity was found at all between the two genomes, no further processing is performed. However, in the following we discuss the case that some similarities have been found by the local similarity search (see 4.3.1). We denote the set of genomic features with a nonempty domain by G', with

$$G' \subseteq G,$$

$$\forall g \in G' \text{ its domain } D \neq \emptyset, \text{ and}$$

$$\forall g \in G \backslash G' \text{ its domain } D = \emptyset.$$

Consequently, this restricts the sets of directed edges, $H' \subseteq H$, and of their labels, either $C' \subseteq C$ for a basic reference model or $V' \subseteq V$ for a weighted reference model. This is due to the fact that each pair of nodes $(g_i, g_j) \in G$ is connected via a directed edge (i, j), and that all these edges are labelled (see Definitions 4.2.2 and 4.2.3, respectively). For instance, by the deletion of one node $g \in G$ the number of directed edges in e.g., H would decrease by $|G| - 1$. We denote the subgraph relation of a reference model by $\sqsubseteq$ and write $\mathcal{R}' \sqsubseteq \mathcal{R}$ or $\mathcal{R}'_\diamond \sqsubseteq \mathcal{R}_\diamond$, respectively.

A reference model $\mathcal{R}' = \langle G', H', C', c' \rangle$ or $\mathcal{R}'_\diamond = \langle G', H', V', h' \rangle$ is considered to be a *solution* denoted by $\mathcal{S}$ to the CSP *iff* all genomic features $g \in G'$ are instantiated with a single PGF from their domains $\mathcal{P} \in D$ of g which does not violate any of the constraints modeled. Furthermore, we consider a locally consistent network as a *partial solution* to the CSP, i.e., a solution which has yet not assigned all $g \in G'$ with valid values. The biological motivation here is the assumption that the genome architectures which are combined into the reference model represent a class of structural design of prokaryotic genomes all generated by the same evolutionary pressures (see Chapter 2), and that these forces have shaped as well the genome architecture of the query sequence (i.e., that it belongs to this class of structural design).

In the following we will introduce an approach to find such a solution. Furthermore, we will adapt (relax) the definition of a solution in order to increase the biological plausibility of it.

4.4.1 Backtracking Search Strategy

There exist many algorithms to solve CSPs (for an introduction see e.g., (Dechter, 2003)) where backtracking search being one of the most commonly used ones. Allen (Allen, 1983) was the first to propose the use of a backtracking algorithm for the task of finding a solution to an IAN. For IANs a solution is considered to be a consistent atomic labelling. It has been shown by Vilain and Kautz (Vilain and Kautz, 1986) that the search for a solution of an IAN using backtracking is NP-hard, however, a backtracking algorithm can still work well in practice depending on the concrete problem to be solved (Beek and Manchak, 1996).

Backtracking search is a depth first search strategy that chooses values for one variable at a time and backtracks when a variable has no valid assignment. There are several strategies to make this search more efficient than a straight uninformed depth first search (i.e., to prune the search space) concerning the order of variables to assign next, the order of values of their domains to choose, and the backtracking strategy if a variable has no legal value (Dechter, 2003). Backtracking will find a solution to the CSP if there is one consistent with the knowledge modeled in the reference model (i.e., the interval relations allowed) and the PGFs found for the query genome.

The search strategy (from now on denoted as *Backtracking Search*, Algorithm 3) to find a solution $\mathcal{S}$ can be divided into a pre-processing and a search phase with the following characteristics:

1. **Pre-processing Phase:** Prior to the backtracking search algorithm the following heuristic pruning techniques are used:

   (a) **Value Ordering:** For each genomic feature $g \in G'$ the PGFs in its domain D are ordered in decreasing order of their total length. The total length of a PGF $\mathcal{P}$ is calculated by summing up the lengths of its local similarities. We denote this function by $length(\mathcal{P})$. The idea behind this value ordering heuristic is that the longer a PGF the higher is the probability that it reflects a true functional equivalence (or even a homology) and thus, it is supposed that a longer PGF is a better assignment for a variable unlikely to cause a failure of the search.

   (b) **Variable Ordering:** Usually, in CSPs the variables having the minimum remaining values are most likely to cause a failure during the search process (Dechter, 2003). Therefore, we use the minimum remaining value heuristic, i.e., we construct a queue of variables to proceed, where variables are ordered by increasing number of PGFs in their domains. Furthermore, all variables with the same number of PGFs are ordered in decreasing order of the total length of their longest PGF (i.e., the first PGF in a domain). Thus, during the search

phase the variable with the longest PGF of the variables having the least values is processed next.

2. **Search Phase:** The search phase is structured into the following steps:

   (a) **Variable Instantiation:** Initially, from the queue of all variables the first one is selected and assigned with its longest PGF. For the rest of the search for a solution iteratively one variable at a time is assigned from this queue, i.e., one variable is added to a locally consistent network. For each of these variables we check

      i. if the relation between this PGF and all already instantiated PGFs is contained in the set C of relations modeled for their corresponding $g \in G'$ (Algorithm 4) and

      ii. if this assignment is interval path consistent (see 4.2.2.4) by using the *To-Add* function (Algorithm 5). This function is in part identical to Algorithm 1, however, here only triangles (3-nodes) including the current genomic feature g are checked for consistency by Equation 4.2. In case one of the edge labels becomes empty it returns *false* indicating that the instantiation of g with this PGF causes inconsistency.

   If one of the above conditions is violated the next longest PGF from the domain D of g is chosen and the validation starts again. In case there exist no further values in its domain the following backtracking step (see 2b) is conducted.

   (b) **Chronological Backtracking:** In chronological backtracking, it backtracks to the most recent instantiated variable, when a branch of a search fails, tries a different value for it, and continues the search from here. Chronological backtracking is the common technique used for solving IANs (Beek and Manchak, 1996). Since the order in which variables are assigned is in decreasing length of their values (for all variables having the same number of values), chronological backtracking backtracks always to the variable having the least longest PGF assigned, i.e., the variable having the assignment with the lowest probability of reflecting a true functional equivalence.

There are two reasons why the above described backtracking approach may fail in finding a solution. First, there may exist a spatial relation between genomic features of the query sequence (i.e., PGFs) which has not been modeled in the reference model. This can occur since relations can only be searched on a subset

Algorithm 3 *Backtracking Search*($\mathcal{R}'$)

Input: $\mathcal{R}' = \langle G', H', C', c' \rangle$ \\ a basic reference model
Output: a solution $\mathcal{S}$ *iff* found,
'failure' otherwise.

```
1: i ← 1  \\ the index of the current variable
2: j ← 1  \\ the index of the domain value of the current variable
3: k ← 1  \\ the index of the domain value of the previous variable
4: while 1 ≤ i ≤ |G'| do
5:    g_i ← G'(i) \\ select a variable
6:    if j > |D_i| then
7:       i ← i − 1
8:       j ← k + 1
9:    else
10:      while 1 ≤ j ≤ |D_i| do
11:         P_j ← D_i(j)
12:         instantiate g_i with P_j
13:         S add g_i
14:         if Valid Relation(R', S, i) then
15:            if To-Add(S, i) then
16:               i ← i + 1
17:               k ← j
18:               j ← 1
19:            else
20:               S remove g_i
21:               j ← j + 1
22:               if j > |D_i| then
23:                  i ← i − 1
24:                  j ← k + 1
25:               end if
26:            end if
27:         else
28:            S remove g_i
29:            j ← j + 1
30:            if j > |D_i| then
31:               i ← i − 1
32:               j ← k + 1
33:            end if
34:         end if
35:      end while
36:   end if
37: end while
38: if i = 0 then
39:    return 'failure'
40: else
41:    return S
42: end if
```

Algorithm 4 *Valid Relation*$(\mathcal{R}', \mathcal{S}, i)$

Input: $\mathcal{R}' = \langle G', H', C', c' \rangle$ \\ a basic reference model
$\mathcal{S}$ \\ a partial solution
i \\ the index of the last instantiated variable
Output: *true iff* the relations between the PGF of g_i and all other PGFs of $\mathcal{S}$ are modeled in $\mathcal{R}'$,
false otherwise.

1: **for** $j \leftarrow 1, \ldots, (i-1)$ **do**
2: $R_{ij} \leftarrow R_{ij} \cap C_{ij}$
3: **if** $R_{ij} = \emptyset$ **then**
4: **return** *false*
5: **end if**
6: **end for**
7: **return** *true*

Algorithm 5 *To-Add*$(\mathcal{S}, i)$

Input: $\mathcal{S}$ \\ a partial solution
i \\ the index of the current variable
Output: *true iff* interval path consistent,
false otherwise.

1: **for** $j \leftarrow 1, \ldots, (i-1)$ **do**
2: **for** $k \leftarrow 1, \ldots, (i-1)$, $k \neq j$ **do**
3: $R_{ik} \leftarrow R_{ik} \cap (R_{ij} \otimes R_{jk})$
4: **if** $R_{ik} = \emptyset$ **then**
5: **return** *false*
6: **end if**
7: $R_{kj} \leftarrow R_{kj} \cap (R_{ki} \otimes R_{ij})$
8: **if** $R_{kj} = \emptyset$ **then**
9: **return** *false*
10: **end if**
11: **end for**
12: **end for**
13: **return** *true*

of all genomes in nature (altogether, 216 prokaryotic genomes are available at the moment at the NCBI). For instance, if for these features only one (strong) hit in their domain exists, the approach is not able to find a valid instantiation. The reference model is so called *over-constrained*.

Second, we do not expect a valid assignment for each genomic feature $g \in G'$, since for some genomic features only *false positive* PGFs may exist. The reason is that on the one hand, the raised E-value cutoff helps to identify less conserved genomic features between the two genomes. On the other hand, however, it leads to more *false positive* hits, i.e., hits that occur just by chance and do not reflect a

functional equivalence. Therefore, backtracking is apparently doomed to fail for high E-Value cutoffs. A search strategy is required which does not rely on the instantiation of all variables as is for traditional constraint satisfaction problems. In the following Section we will discuss such an approach.

Finally, backtracking as a depth first search strategy is neither complete nor is it optimal (Dechter, 2003), i.e., using backtracking we cannot ensure that the best solution (the alignment which aligns the most letters of functional equivalent regions) will be found.

4.4.2 Incremental Search Strategy

There exist different techniques to find solutions for an over-constrained problem summarized under the term partial constraint satisfaction, e.g., Freuder and Wallace (Freuder and Wallace, 1992), Borning *et al.* (Borning et al., 1992), or Henz *et al.* (Henz et al., 2000). These techniques focus on the relaxation of a preferably minimal number of constraints in order to find an instantiation for all variables, i.e., they would admit values to be assigned to variables violating the defined constraints between them. On the one hand, this would help to overcome the over constrainment of a reference model as described in the previous Section. On the other hand, however, for our problem we are not willing to relax constraints but rather to discard variables if only *false positive* hits exist for them.

Therefore, we define the search for a biological feasible alignment as an incremental constraint satisfaction problem. We start the search with the genomic features having at least one PGF in their domain which exceeds a defined minimum length. The idea behind this strategy is to find a solution for those genomic features whose PGF are likely to belong to the alignment of the genomes. This is the essential strategy of all other approaches for a genome alignment (compare, e.g., MUMmer 3.2.2.2 or OWEN 3.2.2.8). Subsequently, we iteratively introduce those genomic features to the current solution not having a PGF exceeding the required minimum length and thus test their biological plausibility. This allows us to discard a genomic feature, if its instantiation is not valid with the existing solution.

Consider a set G of genomic Features g of a reference genome R and the subset $G' \subseteq G$ of all genomic features whose domain D contains at least one PGF $\mathcal{P}$. Then, we denote the subset of genomic features having at least one PGF longer than a required length len by G'', with

$$G'' \subseteq G' \subseteq G \text{ and}$$

$$G'' = \{g | g \in G' \wedge \exists\, \mathcal{P} \in D : length(\mathcal{P}) \geq len\}.$$

Hence, we get another subgraph which we denote by $\mathcal{R}'' \sqsubseteq \mathcal{R}' \sqsubseteq \mathcal{R}$, with $\mathcal{R}'' = \langle G'', H'', C'', c'' \rangle$. For the extension of the solution $\mathcal{S}$ we consider those genomic

features g not contained in G''. Let us denote this set of genomic features by G^*, with

$$G^* = G' \backslash G'' \text{ and}$$

$$G^* = \{g | g \in G' \wedge \exists\, \mathcal{P} \in D : length(\mathcal{P}) < len\}.$$

In this case, a solution $\mathcal{S}$ requires the instantiation of all genomic features $g \in G''$, but not necessarily the instantiation of variables in G^*.

We denote this search strategy from now on as *Incremental Search* (see Algorithm 6) which is defined as follows:

1. Before G' is partitioned into G'' and G^*, the same pre-processing steps as for *Backtracking Search* (4.4.1)(value ordering (step 1a) and variable ordering (step 1b)) are used. In the following, the partitioning is just a matter of dividing the queue of variables before the first element with its first PGF shorter than the required minimum length.

2. Initially the variable with the longest PGF is assigned first. Subsequently, all other $g \in G''$ are assigned with their longest PGF and we test if this instantiation is interval path consistent (again using the *To-Add* function, (Algorithm 5)). In case of an inconsistent instantiation, the next PGF of this variable is considered if it has the required minimum length. If not, this variable is removed from G'' and added to G^* in order to be considered by the next step.

3. We incrementally extend $\mathcal{S}$ by introducing a $g \in G^*$ which are selected in descending order of the total length of their first PGF in their domains. Subsequently, we check their consistency to all other instantiated variables as described in step 2a of the *Backtracking Search*. In contrast to *Backtracking Search*, however, if the instantiation of a g does not satisfy all constraints it is discarded from $\mathcal{S}$ and either the next value of this variable or the next variable if no further values exist is considered.

Algorithm 6 *Incremental Search*$(\mathcal{R}'', G^*)$

Input: $\mathcal{R}'' = \langle G'', H'', C'', c'' \rangle$ \\ a basic reference model
G^* \\ set of genomic features for which only PGFs shorter than a required min. length exist

Output: a solution $\mathcal{S}$, possibly empty.

1: $\mathcal{S}$ \\ new empty solution
2: *Instantiate Solution*$(\mathcal{S}, G'', G^*)$ \\ see Algorithm 7
3: *Incrementally Extend Solution*$(\mathcal{S}, G^*)$ \\ see Algorithm 8

Algorithm 7 *Instantiate Solution*$(\mathcal{S}, G'', G^*)$

Input: $\mathcal{S}$ \\ an empty solution
G'' \\ see Algorithm 6
G^* \\ see Algorithm 6
Output: a partial solution $\mathcal{S}$

```
 1: i ← 1   \\ the index of the current variable
 2: j ← 1   \\ the index of the domain value of the current variable
 3: while 1 ≤ i ≤ |G''| do
 4:    g_i ← G''(i) \\ select a variable
 5:    while 1 ≤ j ≤ |D_i| do
 6:       P_j ← D_i(j)
 7:       instantiate g_i with P_j
 8:       S add g_i
 9:       if To-Add(S, i) then
10:          i ← i + 1
11:          j ← 1
12:       else
13:          S remove g_i
14:          j ← j + 1
15:          if j > |D_i| then
16:             i ← i + 1
17:             j ← 1
18:             G* ← G* ∪ {g_i}
19:          end if
20:       end if
21:    end while
22: end while
```

4.4.3 Local Search Strategy

Until now we were discussing the search for a solution not concerning the quality of this solution. Both, *Backtracking Search* (see 4.4.1) and *Incremental Search* (see 4.4.2) will find a solution if one exists, but cannot guarantee to find the optimal one, i.e., the alignment excluding all false positive hits and in parallel aligning the most letters of the two genomes. Thus, an approach able to optimize a solution found by one of the above described approaches could increase the quality of the resulting alignment.

Local search strategies are in use in this context. Local search strategies (e.g., hill climbing, simulated annealing, or genetic algorithms, see (Russell and Norvig, 2003)) can be applied to a problem when the path to the solution (i.e., the order in which variables are assigned) is irrelevant and only the final configuration (i.e., their instantiation) matters. This is the case for CSPs, since problems modeled in this way are usually commutative (Dechter, 2003). Thus, a local search strategy

Algorithm 8 *Incrementally Extend Solution*$(\mathcal{S}, G^*)$

Input: $\mathcal{S}$ \\ a partial solution
G^* \\ see Algorithm 6
Output: a solution $\mathcal{S}$

```
1: i ← 1
2: j ← 1
3: while 1 ≤ i ≤ |G*| do
4:    g_i ← G*(i) \\ select a variable
5:    while 1 ≤ j ≤ |D_i| do
6:       P_j ← D_i(j)
7:       instantiate g_i with P_j
8:       S add g_i
9:       if Valid Relation(R', S, i) then
10:         if To-Add(S, i) then
11:            i ← i + 1
12:            j ← 1
13:         else
14:            S remove g_i
15:            j ← j + 1
16:            if j > |D_i| then
17:               i ← i + 1
18:               j ← 1
19:            end if
20:         end if
21:      else
22:         S remove g_i
23:         j ← j + 1
24:         if j > |D_i| then
25:            i ← i + 1
26:            j ← 1
27:         end if
28:      end if
29:   end while
30: end while
```

assigns a value to a current variable and subsequently moves to a neighbor of it to proceed for the search. In combination with a weight function (sometimes referred to as objective function or comparator), local search is commonly used to solve optimization problems, i.e., to find the best solution to a problem according to this function.

In our case there are two attributes of the genome alignment which we want to optimize: specificity (i.e., the precision for excluding *false positive* hits) and sensitivity (the precision for including *true positive* hits and thus, increasing the number of letters aligned). Usually, sensitivity and specificity are competing goals

for an optimization strategy - increasing one means decreasing the other. In the following we describe the (heuristic) weight function which is used for the task of optimizing a solution.

4.4.3.1 Heuristic Weight Function

The first task concerning the optimization is to increase the sum of the length of all assigned PGFs for the alignment. Therefore, the weight function should include a weight for the length of a PGF assigned to a variable during the search process, with a biologically improving result of the function with increasing sum of the length of all assigned PGFs. The second task is to exclude assignments of a PGF to a genomic feature which is functionally inequivalent to it (corresponding to a false positive hit). So far, the reference model has been used in order to evaluate each PGF by its genomic context and thus to identify false positive hits. However, this evaluation is flexible by allowing alternatives of relations between genomic features which are all assumed to be of similar quality. If, however, the gene order of the reference genome would be compared to a set of (related) genomes, either only a subset of these relations would occur or different frequencies of spatial relations between genes would occur due to common gene clustering (see 2.2.2.4). The weighted reference model (see Definition 4.2.3) encodes these frequencies of relations between genomic features discovered in a reference set of genomes. Hence, selecting the PGF for a variable satisfying the constraints of the highest weights to all other variables will increase the biological plausibility of the resulting alignment, i.e., with respect to the information available in the reference genome set. The crucial task is to define a weight function which optimizes the sum of the lengths of assigned PGFs while simultaneously satisfying the constraints of the highest weights.

First, in order to weight the total length of a solution $\mathcal{S}$ the sum of the lengths of all assigned PGFs is considered. This weight function is denoted by φ_{length}, where $0 \leq \varphi_{length} \leq 1$. The weight function can be calculated by

$$\varphi_{length} \leftarrow \sum_{i=1}^{|G''|} \frac{length(\mathcal{P}_i)}{length(g_i)}, \tag{4.4}$$

where $length(\mathcal{P}_i)$ is the sum of the lengths of all local similarities of the PGF $\mathcal{P}_i$ and $length(g_i)$ is the length of the genomic feature g_i in the reference genome R, i.e., of its corresponding substring. This function reflects the degree of coverage (with $\varphi_{length} = 1$ represents a 100% and $\varphi_{length} = 0$ represents a 0% coverage, respectively) of the PGFs found in the query genome Q to the genomic features in the reference genome R.

Second, in order to weight the frequencies of the relations between genomic features, the weights of the constraints can be considered. This weight function is

denoted by φ_{weigth}, where $0 \leq \varphi_{weight} \leq 1$. The weight function can be calculated by

$$\varphi_{weight} \leftarrow \frac{1}{|H''|} \sum_{1 \leq i < j \leq |G''|} \frac{w_{ij}}{k_{max}}, \tag{4.5}$$

where w_{ij} is the weight of the constraint satisfied between g_i and g_j, and k_{max} is the maximal weight of all constraints between g_i and g_j.

Finally, we can define the (heuristic) weight function φ_{global}, where $0 \leq \varphi_{global} \leq 1$, for a solutions $\mathcal{S}$ as

$$\varphi_{global} \leftarrow \frac{(\alpha \cdot \varphi_{weight}) + \varphi_{length}}{2}, \tag{4.6}$$

where α, with $0 \leq \alpha \leq 1$, is a weighting factor. In the case of $\alpha = 0$ the weights of the constraints are ignored for the optimization, i.e., the solution will be optimized for its total length, and in the case of $\alpha = 1$ the sum of weights of satisfied constraints are weighted just like the total lengths of assigned PGFs. The heuristic weight function φ_{global} indicates the quality of a solution $\mathcal{S}$, where a higher value indicates a better solution, e.g., a $\varphi_{global} = 1$ using an $\alpha = 1$ would indicate that always the highest weighted constraints were satisfied and in parallel all assigned PGFs have in sum the length of the smaller of the two genomes (identical to a 100% coverage). For the sake of clarity, we will refer to the heuristic weight function φ_{global} as the *Comparator*($\mathcal{S}$) in the following algorithm.

4.4.3.2 Optimization Procedure

The optimization algorithm (see Algorithm 9) starts with a solution $\mathcal{S}_{best}$ (subscribed *best*, because it is the best solution found so far) found by either backtracking or incremental search. This solution is copied to $\mathcal{S}_{cur}$ (a current solution). Subsequently, it randomly selects without replacement a variable from the list of variables of $\mathcal{S}_{cur}$. For a selected variable g its next longest PGF is selected and assigned to it. The algorithm considers only these PGFs, since any longer PGF in a domain of a variable was already tested for consistency by the previous search for $\mathcal{S}_{best}$ and thus can only result in an inconsistent instantiation. If this instantiation is interval path consistent (tested by the *To-Add* function), it is considered to be a solution. The quality of $\mathcal{S}_{cur}$ is evaluated by *Comparator*($\mathcal{S}_{cur}$) and compared to the quality of $\mathcal{S}_{best}$ (*Comparator*($\mathcal{S}_{best}$)). In case the current solution is better (i.e., has a higher value) then it is considered as the best solution $\mathcal{S}_{best}$. This procedure is repeated *max_tries* times, where *max_tries* is the specified number of steps allowed before giving up. We will refer to this search strategy as *Greedy Optimization*.

Algorithm 9 *Greedy Optimization*$(\mathcal{S}, max\_tries)$

Input: $\mathcal{S}$ \\ a solution
max_tries \\ the number of tries before giving up
Output: a solution $\mathcal{S}_{best}$, possibly identical to $\mathcal{S}$

```
1: S_best ← S
2: S_cur ← S
3: for m ← 1, ..., max_tries do
4:    g_i ← RandomlySelectVariable(S_cur)
5:    P_(j+1) ← D_i(j + 1) \\ with j being the index of the current P_j of g_i
6:    instantiate g_i with P_(j+1)
7:    if To-Add(S_cur, i) then
8:       if Comparator(S_best) < Comparator(S_cur) then
9:          S_best ← S_cur
10:      end if
11:   end if
12: end for
13: return S_best
```

4.5 Assembling the Genome Alignment

In order to complete the genome alignment, those local similarities have to be introduced that were not considered in the evaluation phase, because they were not part of a genomic feature. We have stored these hits during the identification of PGFs (see 4.3) in a set $Putty = \{Sim_1, \ldots, Sim_m\}$. The assembly of the genome alignment proceeds in two steps: first, all PGFs of the solution $\mathcal{S}$ found by the evaluation (see 4.4) are introduced into the alignment. Second, the gaps between the PGFs are tried to be closed with local similarities $Sim \in Putty$. Therefore, first for each gap between two PGFs $\mathcal{P}_i$ and $\mathcal{P}_j$ those $Sim \in Putty$ are collected and introduced between them which are the result of a partitioning step. Subsequently, those $Sim \in Putty$ are collected which were not considered for a PGF. Here, it is tested if their qualitative order in the query sequence is identical to their surrounding local similarities. If so, a Sim is introduced into the alignment.

Chapter 5

Discovery of a Common Genome Architecture (DCGA)

In which an knowledge discovery approach is described to find all frequent patterns of genomic features in a set of genomes. The knowledge discovered can be directly integrated into the reference model of the knowledge-based alignment procedure, since it searches on the same representation of genomes.

Knowledge discovery in databases (KDD) is the non-trivial process of identifying valid, novel, potentially useful and ultimately understandable information (knowledge) from digital data, i.e., KDD is concerned with the development of methods and techniques for making sense of these data (Fayyad et al., 1996). This process is based on the application of specific data-mining techniques (specific algorithms) for pattern discovery and extraction (Fayyad et al., 1996). Prior to the step of data-mining, some preprocessing has to take place, namely: data selection, data cleaning and data transformation (Fayyad et al., 1996). This comprises the choice of a suitable knowledge representation technique in order to admit the application of a data-mining algorithm. Following the data-mining, KDD can contain the process of pattern interpretation and evaluation to gain novel knowledge from patterns found. Consequently, data-mining is the analysis of observational data sets in order to find unsuspected relationships and to summarize the data in novel ways that are useful for the data owner (Hand et al., 2001).

5.1 Requirements to the Knowledge Discovery

For the introduction of knowledge of genome architecture into the knowledge-based genome alignment (see Chapter 4), we require a process which is able to identify common patterns of genome architecture in a set of genomes (see 4.2.2.3).

In general, a pattern is a local feature of the data, perhaps occurring only in a few instances. The instances for the task of identifying common genome architecture across different species are the genomes, however, not their DNA sequence but the spatial configuration of its genomic features encoded in it (see 2.2). In particular, the interspecies comparison of gene order of complete prokaryotic genomes got into focus in recent years in order to identify operons (see 2.2.2.1). Information received from gene cluster analysis (see 2.2.2.4) can help to support the understanding of functional coupling and metabolic pathways and to improve evolutionary hypotheses about the reasons of gene order (see 2.2.2.4). For the special task of finding operons, several methods have been proposed in order to find common gene clusters for pairs of genomes, multiple genomes, and for detecting local gene order conservation (e.g., (Bansal, 1999; Overbeek et al., 1999; Ermolaeva et al., 2001; Heber and Stoye, 2001; Mazumder et al., 2001; Wolf et al., 2001; Rogozin et al., 2002a)). These methods differ with respect to the amount of gene insertions/deletions and local rearrangements they allow to form gene clusters (Rogozin et al., 2002a). Other methods use the differences between inter- and intra-genic distances (i.e., number of bases) between genes (e.g., (Rogozin et al., 2002b; Salgado et al., 2000; Moreno-Hagelsieb and Collado-Vides, 2002)), and here the intra-genic distance (i.e., the distance between genes which are functionally coupled) remains usually shorter since no regulatory region is required between them.

However, all methods at hand do not concern the spatial configuration of gene clusters beyond the order of genes included. Therefore, an approach at a finer level of granularity which is able to distinguish, e.g., the overlap or the inclusion of features is required. With respect to the reference model (see 4.2.2) used for the knowledge-based alignment approach (see Chapter 4) it would be beneficial if the results of such an approach could be directly introduced into a reference model. Furthermore, the data mining approach should be robust in the pattern identification with respect to global and local mutations, gene acquisitions (i.e., horizontal gene transfer, gene duplication, and gene genesis), gene losses (i.e., loss of fragments encoding one or more genes, inactivation to form a pseudogene, and erosion by deletional bias), and gene fusion events shaping the genome during evolution (Mira et al., 2001). The basic ideas for the following approach were already described in Wetjen (Wetjen, 2002).

5.2 Modelling of Genome Architecture

A crucial task when searching for patterns is the representation of the instances (genomes), since only features of instances which have been explicitly modeled can be used to form patterns. We use a qualitative representation of the genome architecture (see 2.2), since a quantitative representation (i.e., concrete positions, lengths, and distances of genomic features) would disregard evolutionary forces

(see 2.3) shaping a genome sequence.

Let g denote a genomic feature (see Definition 4.2.1 in Section 4.2.2.1), and let $G = \{g_1, \ldots, g_n\}$ denote the ordered set of all genomic features of a genome. With $\preceq_G$ we denote the linear order of these genomic features, with $\preceq_G$ being transitive, reflexive, antisymmetric, and total (Bronstein et al., 2001). This linear order represents the order of succession of the genomic features in the 5' to 3' direction on the plus strand of this genome, i.e., for all $g_i, g_j \in G$, with g_i being a substring $s[k, l]$ and g_j being a substring $s[m, n]$ $k \leq m \Rightarrow g_i \preceq_G g_j$. Thus, again genomic features on the minus strand are 'projected' onto the plus strand, i.e., their complement is considered (see as well 4.2.2).

The interval relationships (see 4.2.1.2 for details) are used to qualitatively model the genome architecture. Let us denote the set of all interval relationships again by $I = \{b, bi, m, mi, o, oi, s, si, f, fi, d, di, eq\}$ (see 4.2.1.2). Note that since the relationships *before* and *after* do not require genes to be direct neighbors, they can be separated by genes in between. Given $|G| = n$ genomic features of a genome sequence we capture their qualitative positions to each other in a $N^{n \times n}$ matrix, where each cell $N(i, j)$ describes the relationship $r \in I$ between the genomic features $g_i, g_j \in G$. Thus, we can define a qualitative genome as follows

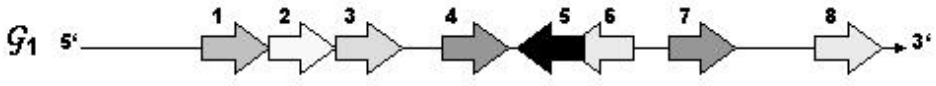
Definition 5.2.1 *A **Qualitative Genome** is a triple $\mathcal{G} = \langle G, \preceq_G, N \rangle$, where*

- *G is its set of genomic features,*
- *$\preceq_G$ is a relation describing the linear order of all genomic features $g \in G$, and*
- *N is a matrix holding the interval relations describing the spatial order between all pairs of genomic features.*

An example of a qualitative genome can be found in Figure 5.1.

5.3 Properties of Genomic Patterns

Let $\Gamma = \{\mathcal{G}_1, \ldots, \mathcal{G}_m\}$ be a set of qualitative genomes. We are interested in finding all patterns of genomic features in Γ occurring in a quorum q (i.e., minimum number) of this genomes, with $2 \leq q \leq |\Gamma|$. Furthermore, we want the genomic features contained in a pattern to be functionally equivalent in the respective genomes. Because of the architecture of prokaryotic genomes which are shaped by local and global mutations (see 2.3), we expect patterns to be spatially restricted to relatively short DNA regions corresponding to a finite number of genomic features. For example, the average size of operons is three genes (see 2.2.2.4). Thus, we can restrict the search for genomic patterns to a specific number w of adjacent genomic features. Let us define such a neighborhood of genomic features more precisely as follows



N	1	2	3	4	5	6	7	8
1		*m*	*b*	*b*	*b*	*b*	*b*	*b*
2			*m*	*b*	*b*	*b*	*b*	*b*
3				*b*	*b*	*b*	*b*	*b*
4					*b*	*b*	*b*	*b*
5						*o*	*b*	*b*
6							*b*	*b*
7								*b*
8								

Figure 5.1: Example of a qualitative genome. The empty cells of the matrix N are values not stored since they can be computed using function *inv* (see Equation 4.1).

Definition 5.3.1 *Given the set $G = \{g_1, \ldots, g_n\}$ of all genomic features of a qualitative genome, a* ***Genomic Neighborhood*** *is a pair $\mathcal{N} = (G', \preceq_{G'})$, where*

- *$G' \subseteq G$ is its set of genomic features with a size $w = |G'|$, and*
- *$\preceq_{G'}$ is a relation describing the linear order of all genomic features $g \in G'$.*

Subsequently, we can define a genomic pattern as follows:

Definition 5.3.2 *Consider a subset $\Gamma' \subseteq \Gamma$ of all qualitative genomes which contains at least q genomes. Given a gene neighborhood $\mathcal{N} = (G', \preceq_{G'})$, let $G'' = \{g_1, \ldots, g_k\}$ be a non-empty set of genomic features, with $G'' \subseteq G'$ and $G'' \neq \emptyset$. Furthermore, let $R^{k \times k}$ be a matrix, where each cell $R(\imath, \jmath)$ describes the relationship $r \in I$ between the genomic features $g_\imath, g_\jmath \in G''$. A* ***Genomic Pattern*** *is a triple $\pi = \langle \Gamma', G'', R \rangle$, where for each $\mathcal{G} \in \Gamma'$ there exists*

- *a set of genes in the gene neighborhood $\mathcal{N}$ so that $G'' \subseteq G'$ and*
- *an injective mapping $f : R \to N$ so that $f(R(\imath, \jmath)) = f(N(i, j))$.*

Let again $\Gamma = \{\mathcal{G}_1, \ldots, \mathcal{G}_m\}$ denote a set of qualitative genomes. The frequency of a genomic pattern $\pi = \langle \Gamma', G'', R \rangle$ found in a subset $\Gamma' \subseteq \Gamma$ of all qualitative genomes contained in Γ is a function

$$fr \leftarrow \frac{|\Gamma'|}{|\Gamma|}, \text{ with } 0 \leq fr \leq 1. \tag{5.1}$$

For instance, consider the example given in Figure 5.2. A genomic pattern π_x in a quorum $q = 3$ of these genomes, occurring in a neighborhood with a

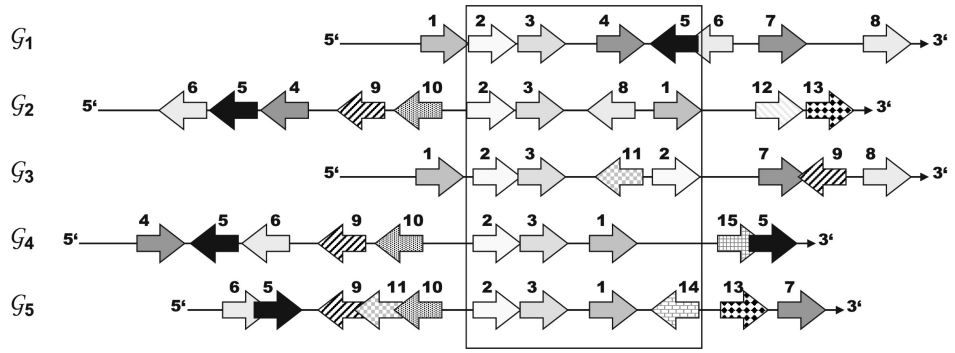


Figure 5.2: Example of a genomic pattern in a genomic neighborhood of size four (highlighted by the frame).

size $w = 4$, and having a number of $k = 3$ genes is $\pi_x = \langle \Gamma' = \{\mathcal{G}_2, \mathcal{G}_4, \mathcal{G}_5\}$, $G' = \{2, 3, 1\}, R = ((2\,eq\,2), (2\,m\,3), (2\,b\,1), (3\,mi\,2), (3\,eq\,3),$
$(3\,b\,1), (1\,bi\,2), (1\,bi\,3), (1\,eq\,1))\rangle$, with a frequency $fr(\pi_x) = 0.6$. We call such a genomic pattern a 3-pattern since it contains three genes ($k = 3$). There exists, however, no 3-pattern for 4, 5 and 6, since genes are rearranged or have different relationships.

Next, we define a partial order on genomic patterns. For any two genomic patterns $\pi_i = \langle \Gamma'_i, G''_i, R_i \rangle$ and $\pi_x = \langle \Gamma'_x, G''_x, R_x \rangle$ we say that π_i is a *subpattern* of π_x (denoted by $\pi_i \sqsubseteq \pi_x$) *iff*

- $G''_i \subseteq G''_x$,
- $\Gamma'_i \supseteq \Gamma'_x$, and
- there exists an injective mapping $f : R_i \rightarrow R_x$, such that $f(R_i(\imath, \jmath)) = f(R_x(k, l))$.

Note that the relation $\sqsubseteq$ is reflexive, transitive, but not antisymmetric (Höppner, 2001). Using our previous example, the genomic patterns $\pi_i = \langle \Gamma'_i = \{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4, \mathcal{G}_5\},\ G'_i = \{2, 3\}, R_i = ((2\ eq\ 2), (2\ m\ 3), (3\ mi\ 2), (3\ eq\ 3))\rangle$; $\pi_j = \langle \Gamma'_j = \{\mathcal{G}_2, \mathcal{G}_4, \mathcal{G}_5\},\ G'_j = \{2, 1\}, R_j = ((2\ eq\ 2), (2\ b\ 1), (1\ bi\ 2), (1\ eq\ 1))\rangle$; and $\pi_k = \langle \Gamma'_k = \{\mathcal{G}_2, \mathcal{G}_4, \mathcal{G}_5\},\ G'_k = \{3, 1\}, R_k = ((3\ eq\ 3), (3\ b\ 1), (1\ bi\ 3), (1\ eq\ 1))\rangle$ are subpatterns of the genomic pattern $\pi_x = \langle \Gamma'_x = \{\mathcal{G}_2, \mathcal{G}_4, \mathcal{G}_5\}, G'_x = \{2, 3, 1\}, R_x = ((2\,eq\,2), (2\,m\,3), (2\,b\,1), (3\,mi\,2), (3\,eq\,3), (3\,b\,1), (1\,bi\,2), (1\,bi\,3), (1\,eq\,1))\rangle$ (Figure 5.3 gives an example).

5.4 Discovering Frequent Genomic Patterns

In order to find all frequent genomic patterns in the genomes of Γ we start to construct all frequent 1-patterns ($k = 1$) that occur in the required quorum q. In

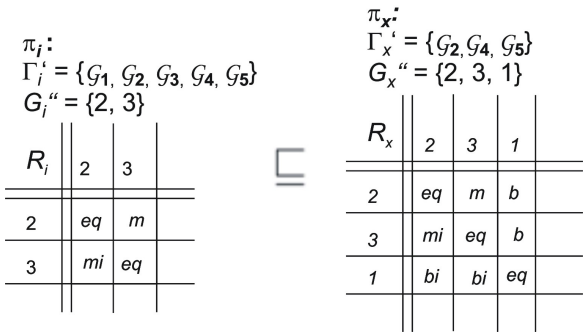


Figure 5.3: Example of a subpattern relation.

each further iteration k we try to create out of each frequent k-pattern a (k+1)-pattern by searching for frequent gene relationship combinations in Γ' in the 5' to 3' direction within a specific number w of adjacent genomic features. The fact that the frequency of a pattern is less than or equal to the frequency of any of its subpatterns

(5.2) $\forall\, \pi_i, \pi_x : \pi_i \sqsubseteq \pi_x \Rightarrow fr(\pi_i) \geq fr(\pi_x)$

guarantees that no frequent pattern will be missed. So far, the procedure is identical to association rule mining (Agrawal et al., 1996).

5.4.1 Merging Genomic Patterns

With the increase of k, the number of gene relationship combinations to be tested for frequency grows exponentially. Constructing all k-patterns as described above can thus be very time-consuming for large k. A pruning technique introduced by Höppner (Höppner, 2001) is used to keep the increase in the number of gene relationship combinations to be tested moderate.

Each subpattern of a (k+1)-pattern candidate is frequent according to Equation 5.2. Any two frequent k-patterns π_i and π_j can thus be merged into a (k+1)-pattern π_x if they occur in the same genomes and share a common (k-1)-subpattern. Let us denote the remaining genes beside the common subpattern in π_i and π_j with g' and g'', respectively. In order to build the relation matrix R_{π_x}, the relations for the common subpattern denoted by A can be taken from π_i or π_j. Furthermore, the relations between g' and g'' and the first (k-1) genes can also be taken from π_i and π_j. The only degree of freedom within π_x is the relation r between g' and g''.

The freedom in choosing r yields 13 different patterns which might become the candidate (k+1)-pattern because we allow 13 interval relationships. As the search for a frequent gene relationship combination is restricted to the 5' to 3' direction, we can reduce the possible values of r to a maximal number of six by

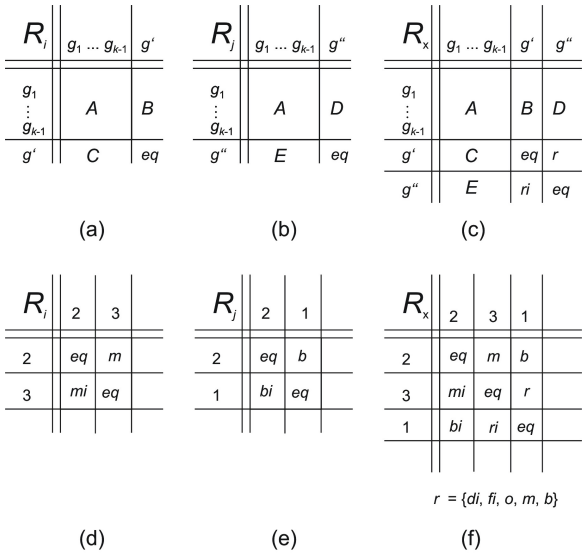


Figure 5.4: General procedure to merge the two relation matrices R_i (a) and R_j (b) into a (k+1)-pattern matrix R_x (c). An example of the merging procedure is given in (d), (e), and (f), where the common subpattern is A.

ignoring the inverse relations and the relation *equals* (*equals* would indicate the same gene) without any loss of generality. Before checking all genome instances of the k-patterns for one of the relations between g' and g'', another pruning technique based on the composition properties of the interval relations is applied (see 4.2.2.4). For instance, the 2-patterns '2 meets 3' and '2 before 1' share the $(k-1)$-pattern '2'. The missing relation r between 3 and 1 obtained by the composition table (Figure 4.5) is either $\{di, fi, o, m, b\}$. Only those relations which do not contradict the result obtained by the composition can be included in the (k+1)-pattern π_x. Searching for a common r is done by iterating over the associated set Γ' of genomes of one of the two k-patterns. Figure 5.4 illustrates how to generate the (k+1)-pattern matrix π_x by merging two patterns π_i and π_j.

For each possible combination of a list of n k-patterns, with $k \geq 2$, which occur in the same genomes and share a common subpattern, the algorithm tries to merge two k-patterns to a (k+1)-pattern as described above (Algorithm 10). All k-patterns sharing a common subpattern and occurring in the same set of genomes can recursively be merged to (k+1)-patterns until there exist no further two k-patterns which can be merged (step 1 of Algorithm 10). If in one of the iterations all (k+1)-patterns of a π_i and the π_j, for $j \leftarrow (i+1), \dots, n$, have been generated, the algorithm terminates since any subsequent merging of a $\pi_{(i+1)}$ with a π_j, for $j \leftarrow (i+2), \dots, n$ would generate a subpattern of the previously generated (k+1)-patterns (step 18 of Algorithm 10).

In the case that two k-patterns π_i and π_j cannot be merged to a (k+1)-pattern

Algorithm 10 *RecursivePatternMerging*(L_k, L_{res})

Input: $L_k = (\pi_1, \ldots, \pi_n)$ \\ List of n k-patterns
$L_{res} = ()$ \\ List of resulting patterns initially empty
Result: L_{res} \\ List of resulting patterns containing all patterns which cannot continuing be merged

```
1: if n < 2 then
2:   L_res ← L_res ∪ {π_1}
3: else
4:   for i ← 1, ..., (n − 1) do
5:     m ← 0
6:     L_(k+1) ← () \\ new empty list
7:     π_i ← L_k(i)
8:     for j ← (i + 1), ..., n do
9:       π_j ← L_k(j)
10:      π_x ← merge(π_i, π_j) \\ see Figure 5.4
11:      if π_x ≠ null then
12:        L_(k+1) ← L_(k+1) ∪ {π_x}
13:        m ← m + 1
14:      end if
15:    end for
16:    if m > 1 then
17:      RecursivePatternMerging(L_(k+1), L_res)
18:      if m = n − 1 then
19:        return
20:      end if
21:    else
22:      L_res ← L_res ∪ {π_i}
23:    end if
24:  end for
25: end if
```

π_x, since no common relationship r can be found between g' and g'' for all $\mathcal{G} \in \Gamma'$ the merging procedure returns *null* (in step 10 Algorithm 10). We search again for frequent gene relationship combinations in Γ' for these two k-patterns (again Equation 5.2 holds). For all subsets of Γ' with a frequent r, the recursive merging process of the k-patterns is restarted. Patterns whose number of genes k exceeds the given threshold are stored in decreasing order of their k and these k-patterns are sorted in decreasing order of their frequencies.

5.4.2 Generating a Non-Redundant Pattern Set

As all $(k+1)$-patterns are generated by extending any k-pattern in the 5' to 3' direction, the result set of the frequent patterns may contain some redundancy. This redundancy occurs if there exists a k-pattern π_i starting with a gene g_i and

a (k-1)-pattern π_j starting with the gene $g_{(i+1)}$, both having equal relationships between equal genes and occurring in the same genomes, i.e., $\pi_j \sqsubseteq \pi_i$ with $\Gamma'_{\pi_i} = \Gamma'_{\pi_j}$. In order to produce a non-redundant set of patterns, it is tested for each k-pattern whether there exists a (k+m)-pattern, for $m \leftarrow 1, \ldots, (w-k)$, of which this k-pattern is a subpattern. If so, this k-pattern is removed from the result set.

Consider again the example given in Figure 5.2. By moving the genomic neighborhood (i.e., the frame shown) one genomic feature to the right (corresponding to the genomic feature $g_{(i+1)}$) we would find the pattern 3 b 1 in $\mathcal{G}_2$, $\mathcal{G}_4$, and $\mathcal{G}_5$ which is a subpattern of the previously found pattern 2 m 3 b 1 in $\mathcal{G}_2$, $\mathcal{G}_4$, and $\mathcal{G}_5$ occurring in exactly the same genomes. Therefore, this pattern is removed, since it does not contribute new information.

5.5 Complexity Analysis

Let C denote the union of all genomic features of all p genomes of Γ. All frequent 1-patterns can be found by simply counting the distinct occurrences (i.e., ignoring gene duplications) of each genomic feature $g \in C$ in the $\mathcal{G} \in \Gamma$. Thus, it takes $O(pl)$ to find all frequent 1-patterns, where l denotes the average number of genomic features in the genomes $\mathcal{G} \in \Gamma$. Let us denote the resulting number of frequent 1-patterns with m. Generating all frequent 2-patterns takes $6mp(w-1)$ steps, since for each of the m 1-patterns there are $(w-1)$ extensions in a neighborhood of size w with six possible relationships which have to be tested for frequency in the p genomes. Subsequently, Algorithm 10 generates for a single 1-pattern out of its list of n frequent 2-patterns all frequent k-patterns, with a maximal $k = w$. In the worst case, there are $O(m(w-1)n^2(p+t))$ merging steps necessary for the merging of all n 2-patterns to frequent k-patterns, where t are the steps required for the merging procedure. This merging procedure can be performed in linear time requiring k steps by copying the genomic feature and its relations to the other genomic features from the relations matrix of one of the patterns into the relation matrix of the other pattern.

Chapter 6

The System KnowAlign

In this chapter, the realization of the approaches of the former chapters is described in detail with respect to its design and its implementation. Furthermore, the application of the system by a user in order to generate a reference model and to perform a knowledge-based genome alignment is explained.

The algorithms described in the former chapters have been implemented in the system KnowAlign. KnowAlign allows a user

- to collect and to store locally the necessary genome entries (given in the GenBank format),
- to create reference models which include the discovery and integration of common genome architectures,
- to align two genomes using the stand alone BLAST implementation, and
- to evaluate the local similarities using these algorithms.

In the following, we will describe the system in more detail with respect to its implementation and its usage.

6.1 System Architecture

There exist several requirements for a system for aligning genomic sequences besides its performance and the quality of the resulting alignments. In particular, a user-friendly displaying and browsing possibility of the alignments has been requested in the last years (Miller, 2001; Chain et al., 2003). Therefore, some alignment tools available which are based on standard alignment techniques have concentrated on the generation of elaborated user interfaces (e.g., Pipmaker (3.2.2.3) based on BLASTZ) while others have realized sophisticated alignment techniques

(e.g., MUMmer (3.2.2.2) based on virtual suffix trees) but have just recently (if at all) satisfied this request in update versions. For the design of KnowAlign, the requirements of displaying and browsing an alignment have thus been incorporated from the beginning, i.e., alignments can be displayed as dot-plots or alternatively linear with the possibility to zoom inside. Furthermore, more graphical support was necessary for a user in order to generate reference models, i.e., selecting a set of genomes, modeling spatial relations between genomic features, browsing the genomic features and their relations, etc. Altogether, a graphical user interface was a strong requirement for our system.

The large amount of data of the genome entries and their resulting reference models which presently cannot be stored in the RAM of a normal PC with approx. 512 MB required the usage of a database for the the implementation of the system. This database should be accessible to other users (e.g., in the same institute) in order to admit the central usage of the reference models regarding its reusability.

Therefore, the system KnowAlign was modeled as a three tier architecture consisting of an interface layer implementing the presentation to the user (GUI), a process management layer implementing all algorithms (the business logic), and a data management layer in order to store and access the data (Figure 6.1). A three tier architecture allows for "*an effective distributed client/server design that provides increased performance, flexibility, maintainability, reusability, and scalability, while hiding the complexity of distributed processing from the user*"(`http://www.sei.cmu.edu/str/descriptions/threetier.html`) and thus, accomplishes the requirements described above. Precisely, the system was implemented as a database-based (Microsoft SQL Server 2000)[1] JAVA application (Sun J2EE version 1.4.2) consisting of approx. 20,000 lines of code (including comments) which are distributed to 110 classes in four packages. As an external source for performing local alignments, KnowAlign uses the stand alone BLAST implementation (3.2.2.1). However, other local alignment tools could easily be integrated.

In the following, some implementation details of the data management layer and the process management layer will be given. Furthermore, we will explain the application of the system by a user.

6.1.1 The Data Management

The data management is divided into the data base (possibly located elsewhere) which stores the information on genome entries, reference models, and DCGA results (see 5.4), and an encapsulated data management consisting of data objects which communicate with the data base through JDBC 2.0 to make this information available for the business logic (e.g., the class `ReferenceModel` reads/writes

[1]The data base software can easily be exchanged by any other relational data base software, since only standard SQL commands were used.

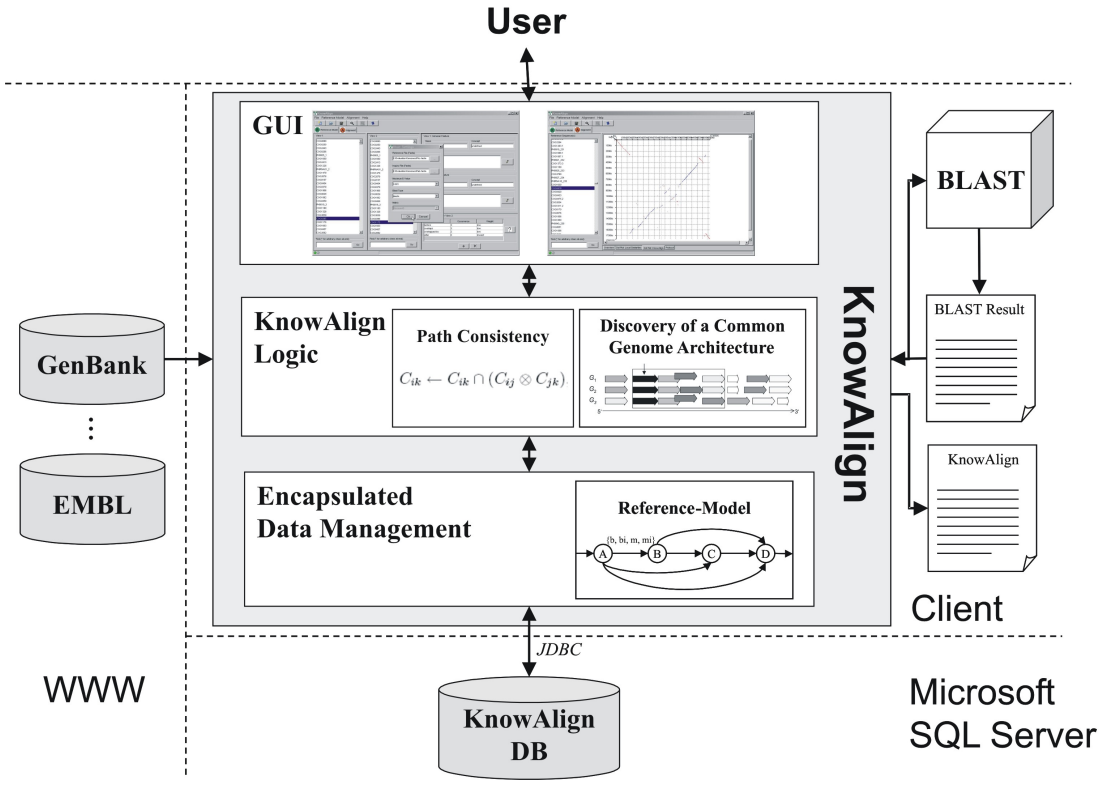


Figure 6.1: Overview of the system architecture of KnowAlign.

to the tables representing a reference model in the data base). In this way, the implementation is easily scalable for future extensions, e.g., concerning the integration of more knowledge of genome entries or the integration of other algorithms for the identification of genomic features.

6.1.1.1 The Data Base Model

The relational data base schema of KnowAlign is divided into three parts with respect to the content of the tables containing either information on

1. genome entries (**Genome**),
2. reference models (**Reference Model**), or
3. DCGA results (**DCGA**) (Figure 6.2).

In the tables of the first part **Genome**, the genome entries are stored by saving all genomic features as specified in the feature table of its GenBank entry.[2] All

[2]For a description of the format of the feature table of GenBank please refer to `http://www.ncbi.nlm.nih.gov/projects/collab/FT/index.html`.

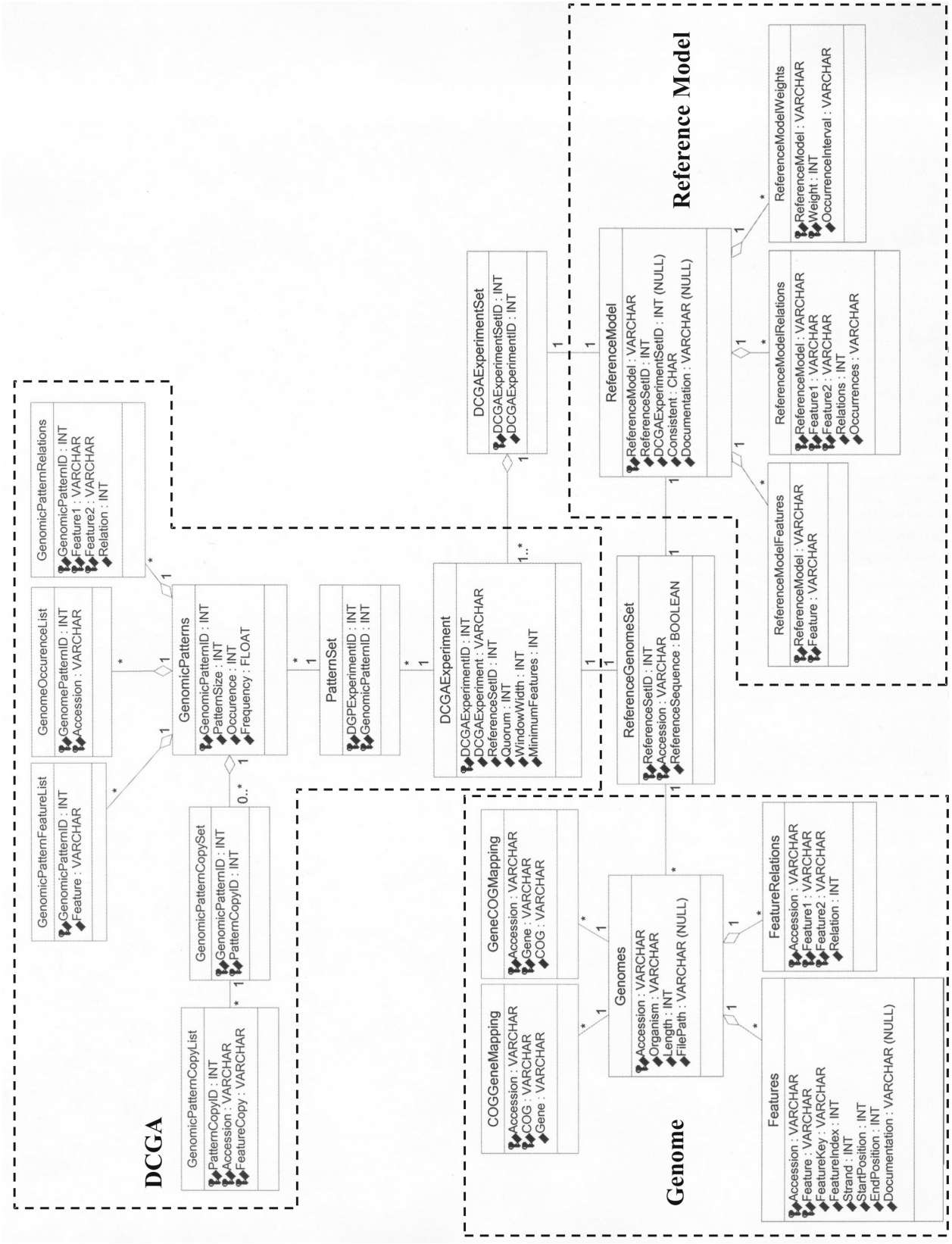


Figure 6.2: UML diagram (in Rational Rose) of the DB-schema of KnowAlign (including data types modeled in the Microsoft SQL Server 2000). The three parts (see the text for explanation) of the data base are highlighted by dashed boxes.

data of this part are linked via their *accession* (i.e., the unique key of a genome data base entry as specified in GenBank) as the primary key mapped to its species name in the central table `Genomes`. Sometimes a key cluster including the accession is used in order to allow for a faster data access. In the tables `Features` and `FeatureLocations` the annotated feature names, their feature keys (e.g., gene, tRNA, or promotor etc. as specified in the GenBank feature table), their index describing the total order of all genomic features contained in a genome, their positions (start-index and end-index), and their strands are stored. The table `FeatureRelations` then holds all spatial relations in 5' to 3' direction (see 4.2.2) of these genomic features which are determined during the writing of an entry to the data base using their annotated positions. The tables `COGGeneMapping` and `GeneCOGMapping` store the mapping of an annotated gene name to the determined COG name of the COG data base.[3] Here we store both directions of this mapping in order to admit an efficient data access regarding a possible many-to-many classification of gene names to COG labels (see 3.3.1). Because of their lengths, the DNA sequences of the genome entries are stored separately in files in the FASTA-format. In addition, this allows for the direct usage of the sequences by BLAST. The physical locations of these files are notated in the column FilePath of the table `Genomes`. In order to save memory space, no further information contained in a GenBank genome entry (e.g., the encoded product or the translated amino acid sequence) is stored in the data base, since none of the information is used during the alignment procedure. Even though, there exists a column Documentation in table `Features` in which some information could be stored if wished by a user.

In the tables of the second part **Reference Model**, all information of a reference model is stored. Here, the central table is `ReferenceModel` which stores the information of the configuration (i.e., the set of genomes and the DCGA results used) of a reference model as well as a flag indicating its status of consistency. In contrast to a value in the column Relation of the table `FeatureRelations` of **Genome** which constitutes a single spatial relation, a value in the column Relations of the table `ReferenceModelRelations` is allowed to be a set of spatial relations (then representing the disjunction of these relations, 4.2.2). Spatial relations are represented as integers to enable a memory efficient representation and, in addition, to admit an efficient computation of set operations (e.g., intersection and union) necessary for the described algorithms (e.g., Algorithm 1) which can then be implemented as logical operations (this has been proposed by e.g., (Beek and Manchak, 1996)). The column Occurrences in `ReferenceModelRelation` stores the number of occurrence of each spatial relation modeled between the corresponding genomic features in Feature1 and Feature2. Using the table `ReferenceModelWeights` which holds the general (user defined) allocation of weights to occurrence intervals of spatial relations

[3]This mapping can be retrieved from ftp://ftp.ncbi.nih.gov/pub/COG/COG/

(see 4.2.2.3) the weight for each relation can be computed. In this way, a later adaptation of weight-to-occurrence mapping by the user causes no problems.

In the third part **DCGA**, the result of a DCGA experiment (i.e., a run of the algorithm on a data set) on the genomes of a reference model is stored. In the table `DCGAExperiment`, the parameters (columns: Quorum, WindowWidth, and MinimumFeatures) of a DCGA run on a specified reference genome set (column: ReferenceSetID) are stored. The genomic patterns found (indicated by the unique key GenomicPatternID) are linked to these parameters via the table `PatternSet`, i.e., the parameters are stored once for all patterns found in order to avoid the storage of redundant data. Each such pattern is stored in the tables `GenomicPatternFeature` list (holding the included genomic feature names), `GenomicPatternRelations` (holding the spatial relations between these features), and `GenomeOccurrenceList` (storing the *accessions* of the genomes in which the pattern occurs). In case of multiple appearances (i.e., a copy in the same genome) of a pattern, its occurrence is stored in the tables `GenomicPatternCopySet` and `GenomicPatternCopyList`.

The three parts are linked via the table `ReferenceGenomeSet` which stores all *accessions* of the genomes belonging to a reference model (indicating the concrete reference sequence by a flag in column ReferenceSequence) and thus, can be used for a `DCGAexperiemnt` and for the alignment procedure; and the table `DCGAExperiementSet` which determines the relation of the DCGA results to a reference model.

6.1.2 The Process Management

In the process management layer the business logic is implemented which comprises all algorithms (methods) involved in the generation of a reference model, the knowledge discovery, the determination of PGFs (see Definition 4.3.2), and the evaluation of their biological plausibility. Exemplarily, we will describe here the realization of the classes involved in the alignment process described in Chapter 4 in more detail.

6.1.2.1 The Classes of the Alignment Procedure

Several classes are involved in the generation of a `KnowAlign` object (Figure 6.3). A `KnowAlign` object corresponds to the result of the alignment of two genomes by BLAST which was subsequently evaluated by one of the algorithms described (see Chapter 4). The main classes implementing the algorithms (i.e., for the assembling of PGFs and for the evaluation of these PGFs) for this process are `PGFCompilator` and `BiologicalContext`, both implemented as Java Threads in order to admit continued working while waiting for the results and to allow for the update of the graphical user interfaces without any time delay (e.g., the progress bar). The input of this process is a Blast result file which is parsed by

the class `BlastResultParser` generating a `BlastResult` object. Such an object consists of `LocalSimilarities` with defined starting- and end-indices in both sequences (see 4.3.1). Subsequently, the `BlastResult` object is the input for the class `PGFCompilator` which assembles the PGFs (class `PGF`) for each genomic feature of a reference model (the class `ReferenceModel` is not shown in Figure 6.3). During this process an object of `GeneFinder` is used to locate putative borders in case a `PGF` represents a protein gene (see 4.3.2.2). Eventually, one of the evaluation algorithms (either backtracking (see 4.4.1), incremental (see 4.4.2), or optimization (see 4.4.3)) of the class BiologicalContext is used to generate a `KnowAlign` object.

6.2 Application of KnowAlign

The central part of the graphical user interface (GUI) of KnowAlign consists of two views: one for the visualization of a reference model (Figure 6.4), and another for displaying and browsing the reference genome and the resulting alignment (Figure 6.6). For all processes like the generation of a reference model or an alignment, a user is supported by separated dialogs (Figure 6.5). In the following, the complete work flow necessary to generate a `KnowAlign` object (i.e., an alignment of two genomes) in KnowAlign will be described briefly.

Before performing an alignment, a reference model (see 4.2.2 and 4.2.3) has to be generated by the user. Several steps are involved in this process. First, the user selects the reference genome to which the query genome is later compared. In addition, further genomes can be selected by the user which together with the reference genome constitute the reference genome set (stored in table `ReferenceGenomeSet`, see 6.1.1) for the discovery of a common genome architecture (see Chapter 5)(Figure 6.5 a). The genomes in this set should be selected with respect to their phylogenetic relationship among each other and to the query genome (if known), since only conserved orders of genomic patterns within this set can later be integrated into the reference model. From our experience, it is better to make this selection not too restrictive, but to incorporate even phylogenetically further distanced species with, e.g., the same ecological lifestyle in order to gain as much information as possible.

In a next step, the user can make presettings for the mapping of frequencies found for a relation between two genomic features to their weights (see 4.2.2.3)(Figure 6.5 b). In order to make this mapping intuitive for a user, weights are displayed as qualitative classes (i.e., *lowest*, *low*, *medium*, *high*, and *highest*) to which intervals of occurrences (e.g., 1 to 2 times $\rightarrow$ *lowest*, 9 to 9 $\rightarrow$ *highest*) are assigned. This mapping, however, can later be adapted by the user if necessary. Second, the discovery of a common genome architecture can be started by the user in a dialog where all necessary parameters (e.g., quorum, see Chapter 5) are inserted (Figure 6.5 c). Different DCGA experiments (i.e., many runs of the

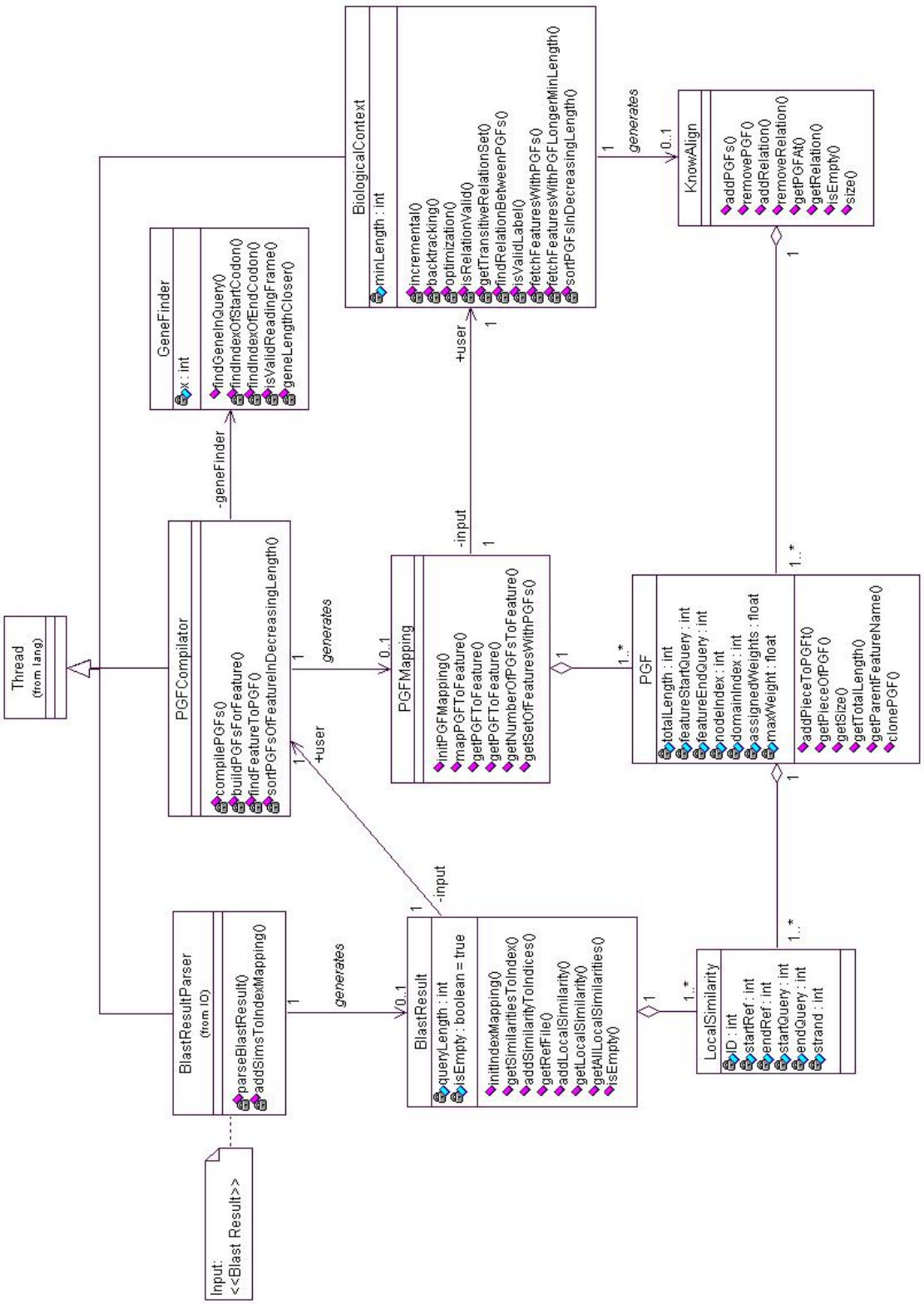


Figure 6.3: UML diagram (in Rational Rose) of the Alignment Process.

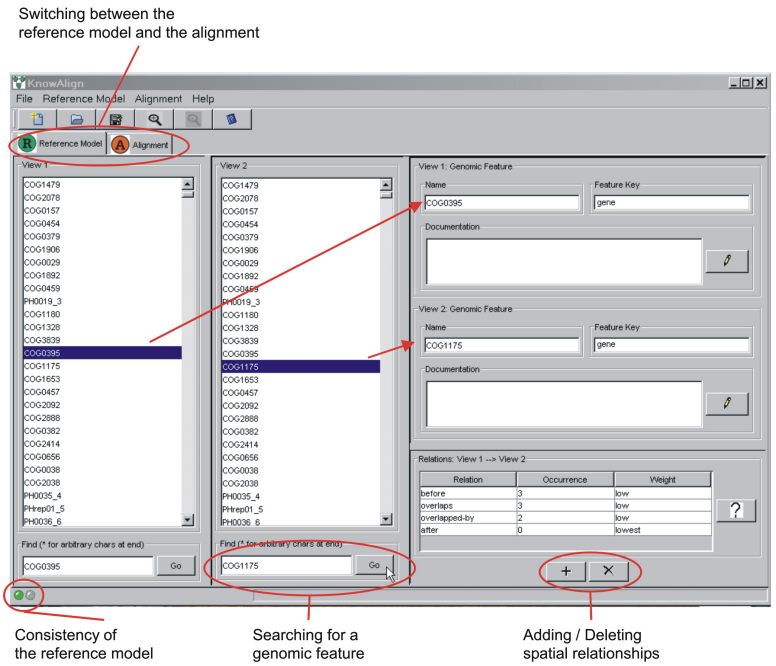


Figure 6.4: GUI of KnowAlign with the tab for editing a reference model.

procedure with different parameters) can be performed which are all stored in the data base. Finally, a user selects the DCGA result(s) for the integration into the reference model (i.e., the storage of the spatial relations and their frequencies in the tables of the data base corresponding to the reference model). In the case that multiple DCGA results are incorporated into a reference model and a spatial relation between genomic features appears in more than one result in different frequencies, only the highest frequency is modeled. In addition, the interface allows the user to add/delete further spatial relations (Figure 6.4). The deletion/addition of spatial relations should be done carefully, since any change can result in the deletion of relations elsewhere in the reference model (see 4.2.2.4).

Subsequently, the user can start the search for local similarities by BLAST in a dialog. Alternatively, existing BLAST results can be imported and then the genetic code (either standard or bacteria, see 2.1) and the allowed search distance for start- and stop-codons (see 4.3.2.2) have to be inserted (Figure 6.5 d). The program will then compute the PGFs and fill the domains of the reference model. Finally, in the 'Find Best Alignment' dialog, the user can choose the method (either backtracking, incremental, or optimization) to evaluate the PGFs (Figure 6.5 e). The result is then displayed in the alignment view (Figure 6.6), where several browsing possibilities (e.g., zoom) enable the user to evaluate the result. All other basic actions can be selected in the menu bar (alternatively in the button bar) of the main GUI.

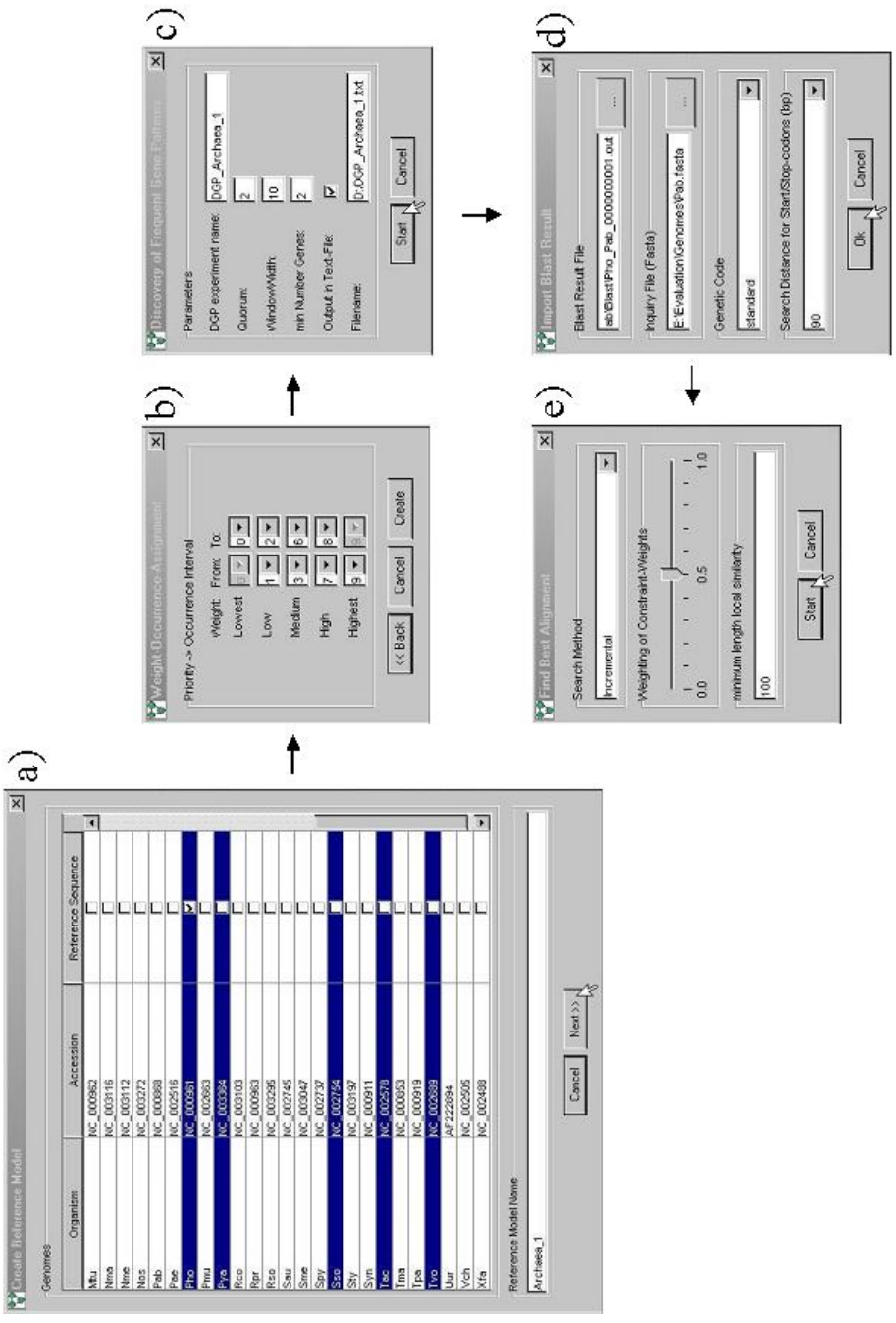


Figure 6.5: The dialogs for a complete workflow to generate an alignment.

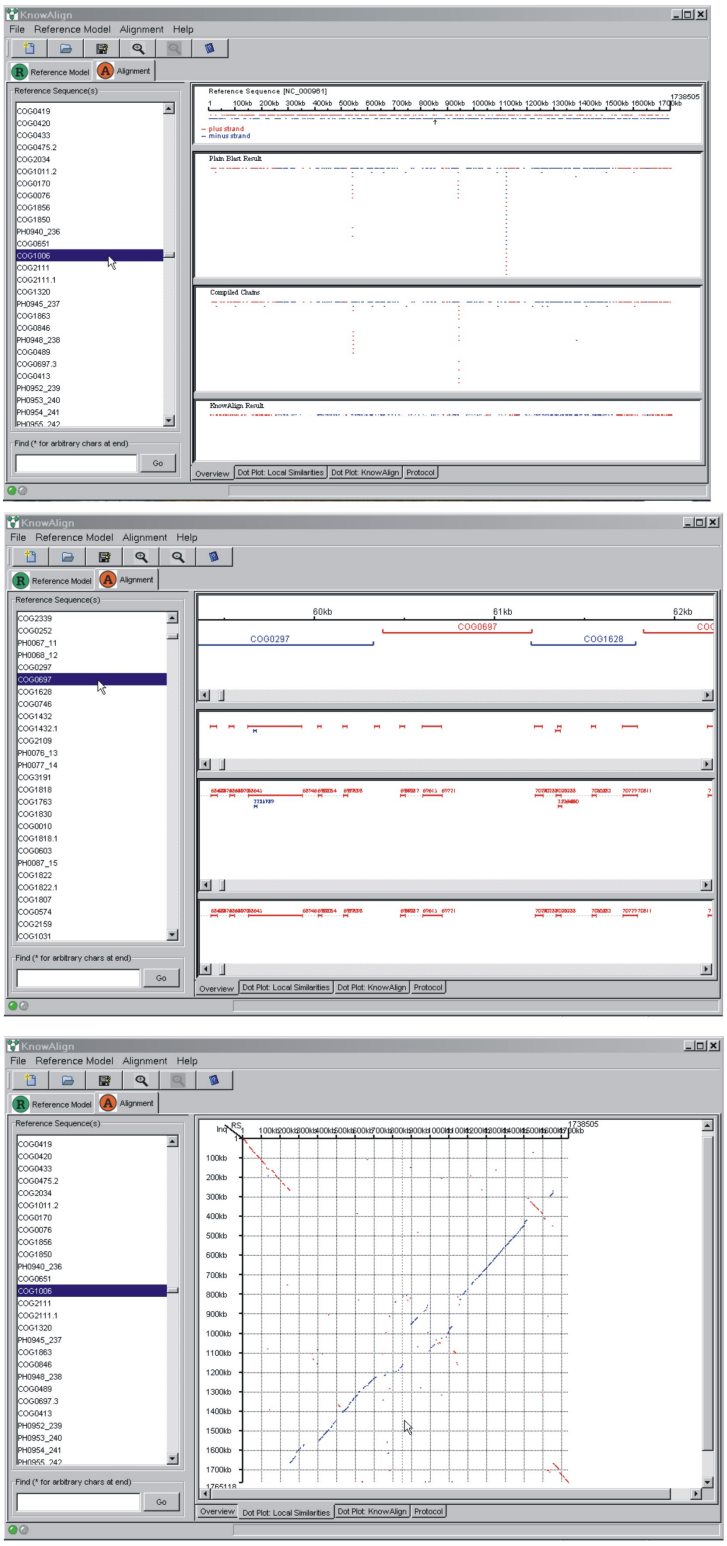


Figure 6.6: GUI KnowAlign with the tab for browsing an alignment.

Chapter 7

Evaluation

In this chapter, alignments generated by KnowAlign *will be evaluated and compared to other genome alignment approaches with respect to the quality of the alignments.*

The comparison of alignment approaches can be performed under different perspectives. For example, Chain *et al.* (Chain et al., 2003) made a comprehensive review of comparative genomic tools with respect to their capabilities and limitations (i.e., time and space complexity of the algorithms, input parameters, and browsing abilities), whereas in Pollard *et al.* (Pollard et al., 2004) the quality (i.e., measured as sensitivity, specificity, and coverage) of alignments computed by tools for the alignment of functional non-coding DNA is compared. Furthermore, Thompson *et al.* (Thompson et al., 1999) compared amino acid alignments to reference alignments validated by experts with respect to their quality. On the one hand, comparing approaches with respect to their input parameters is complicated, since their (combinatorial) effects on the quality of the resulting alignment often cannot be well characterized. On the other hand, no reference alignments for complete prokaryotic genomes are currently available. Here we will concentrate on the evaluation of alignment approaches for complete prokaryotic genomes regarding the quality of the resulting alignments like sensitivity, specificity, and coverage in comparison to other approaches.

7.1 Materials and Methods

In order to compare the results of KnowAlign with existing approaches (compare 3.2), a measurement of the quality of the alignments is required. Here we will use similar measurements as described in e.g., Pollard *et al.* (Pollard et al., 2004) which we adapt for the case of genome alignments.

7.1.1 Quality Measurements of Genome Alignments

If both genomes (the reference sequence R and the query sequence Q) are annotated and functionally equivalent[1] genomic features are known, then there are two distinct classes which a PGF found (see 4.3.2) can be assigned to, either

- **True Positive (TP)** if the PGF is between functionally equivalent genomic features or
- **False Positive (FP)** if the PGF is between functionally inequivalent genomic features (Figure 7.1).

In addition, two more classes can be distinguished based on the absence of a PGF in the alignment, either

- **False Negative (FN)** if there exists no PGF for functionally equivalent genomic features or
- **True Negative (TN)** if there exists no PGF for functionally inequivalent genomic features (Figure 7.1).

Hence, only those PGFs occurring in annotated genomic features can be classified into one of the above classes and thus are used for the following evaluation (see 7.1.2).

As a measurement of the quality of the resulting alignment the concepts sensitivity and specificity can be used. In general, sensitivity is the ability to detect true positive matches whereas specificity is the ability to reject false positive matches. For the purpose of comparing alignment methods we define both terms as:

Definition 7.1.1 *The Sensitivity of a genome alignment is the probability that all true positive PGFs are assigned. Sensitivity (in %) is calculated as*

$$(7.1)\quad sensitivity = \frac{\sum TP}{\sum TP + \sum FN} \cdot 100.$$

Definition 7.1.2 *The Specificity of a genome alignment is the probability that true negative PGFs are not assigned. Specificity (in %) is calculated as*

$$(7.2)\quad specificity = \frac{\sum TN}{\sum TN + \sum FP} \cdot 100.$$

Another objective of the knowledge-based alignment approach is to maximize the number of letters aligned between the two genomes by increasing the sensitivity without losing specificity (see Chapter 1). As a measurement of the number of letters aligned, we will use the coverage of the *true positive* PGFs.

[1] In the following we will concentrate on orthologous and paralogous protein genes (see 7.1.3).

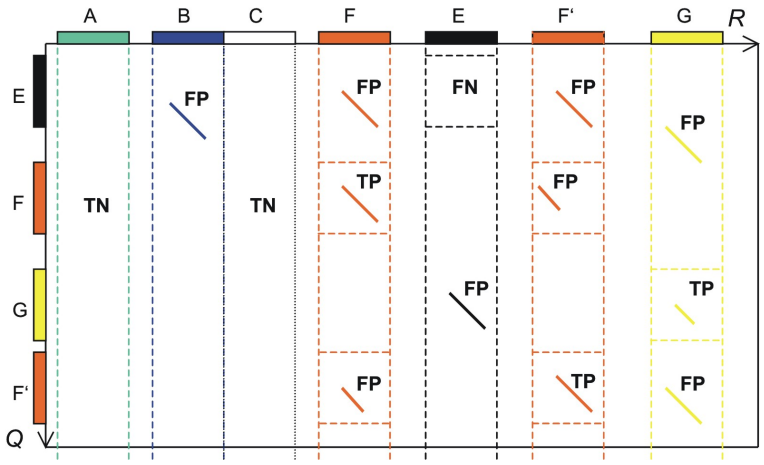


Figure 7.1: Dot-plot scheme of the alignment of two genomes R and Q. The identified PGFs (diagonal lines) are assigned to different classes (TP = true positive; FP = false positive; FN = false negative; TN = true negative; see text for explanation). Identical letters indicate duplications of genomic features, e.g., F' marks the duplication of this feature in one of the genomes.

Definition 7.1.3 *The Coverage of the genome alignment is the fraction of letters of the query sequence aligned to the reference sequence of true positive PGFs. Coverage (in %) is calculated as*

$$(7.3)\quad coverage = \frac{\sum length(TP)}{|Q|} \cdot 100,$$

where $length(TP)$ is a function calculating the total length of a (true positive) putative genomic feature by summing up the lengths of its local similarities.

7.1.2 Evaluation Procedure

The procedure of calculating the quality measurements of a genome alignment is as follows:

1. According to the description of the alignment tool under evaluation, the alignment of the two genomes is performed (the tools and their parameters can be found in Table 7.1).

2. The textual result file (containing the starting and end indices of a local similarity in both sequences) is parsed. Local similarities (either a Blast hit, a MUM, etc.) found are assembled to PGFs as described in Section 4.3.2 in order

(a) to determine the parts of the local similarities that correspond to the annotated genomic features and

(b) to avoid multiple counting of hits as either *true positives* or *false positives* in case they represent conserved parts which belong together but are separated by less conserved regions (see 4.3).

3. *True positives*, *true negatives*, *false positives*, and *false negatives* are determined and the quality measurements are calculated as described above.

Table 7.1: Summary of the settings and parameters used for genome alignment tools in the following evaluation. For a detailed description of the parameters please refer to the tool specific user manual.

Approach	**Version**	**Settings**	**Parameters**
ASSIRC	1.4	default	default
BLAST	2.2.9	blastn	E-values = 100, 10, 1, 0.1, $1 \cdot 10^{-3}, 1 \cdot 10^{-6}, 1 \cdot 10^{-8}$
BLASTZ	01/27/2004	default	B = 1, K = 2400, T = 0, W = 6
KnowAlign	$0.\bar{9}$	1) backtracking, 2) incremental, 3) optimization	for 1) $len = 0$, $x = 90$ for 2) $x = 0, 60, 90, 120$ for 2) and 3) $len = 0, 100, 200, 500$ and for 3) $x = 90$, $max\_tries = 100, \alpha = 1$
DIALIGN	2.2.1	-lgs	default
MUMmer	3.15	1) unique matches 2) maxmatch	for 1) and 2) default
OWEN	1.4a	Procedure: Align, Expand, (2×) Merge, Reconcile, Greedy, Kill	default
Pipmaker	06/11/2004	1) single coverage 2) chaining	for 1) and 2) both strands, high sensitivity
Vmatch	03/10/2004	vmatch	l = 18, evalue = 100, identity = 25
WABA	10/26/2001	all	default

7.1.3 Sequence Data

Genomes used for the comparison were extracted from the Entrez Genomes Division of the NCBI (`ftp://ftp.ncbi.nlm.nih.gov/genomes/Bacteria/`). In order to determine the quality of a hit, we require that the genomes in the comparison are annotated and, furthermore, that functional equivalent genomic features are labelled identically. Therefore, we will use the update version of the database of clusters of orthologous groups of proteins (COGs) (see 3.3.1). However, since the COG database does not necessarily reflect the true number of orthologous genes between the genomes, the resulting genomic patterns (in the following called gene patterns, since only genes are used for the pattern discovery) can have multiple appearances in each of the associated genomes. Furthermore, for the discovery of gene patterns, COGs contained in multidomain proteins (see 3.3.1) were not annotated with interval relationships in our genome data set but labelled as a single gene as annotated in their corresponding GenBank entries. In the following we will compare the DNA sequences of genomes. In case the evolutionary pressure of a protein is not on the DNA sequence but on the amino acid sequence (i.e., many silent or neutral mutations, see 2.3.1), there may not exist enough similarity on the DNA level for putative orthologous genes to be found by the local similarity search approaches. However, since the data set and the evaluation procedure remains the same for all approaches (i.e., equal conditions) the quality measurements described are equitable for this evaluation.

In the following evaluation, we will use 23 genomes of the 63 prokaryotic genomes available in the updated COG database. The selected genomes belong to two taxonomic groups, 13 genomes of the Archaea and another 10 genomes of the γ-Proteobacteria (Table 7.2). Consequently, we will describe and compare alignment qualities for the two phylogenetic groups of prokaryotes, namely Archaea and (Eu)bacteria (see Chapter 2), in order to demonstrate the scope of our approach. Within the (Eu)bacteria we have decided for the γ-Proteobacteria, since this group contains the species *Escherichia coli* (Eco) which is assumably the best studied bacterium so far and, thus, results are easily comparable. In addition, this group contains the species *Salmonella typhimurium* (Sty) which is closely related to *E. coli*. Together, they were the first prokaryotes species compared for gene order and it was observed that between them gene order is highly conserved (see 2.2.2.4), i.e., their genomes are collinear. On the other hand, the genome lengths, the number of protein genes, and the number of genes labeled as COGs are heterogeneous within these taxonomic groups, ranging e.g., in gene number for the Archaea between 1,481 for *Thermoplasma acidophilum* (Tac) to 4,540 for *Methanosarcina acetivorans* (Mac) and for the γ-Proteobacteria between 564 for *Buchnera aphidicola* (Buc) to 5,567 for *Pseudomonas aeruginosa* (Pae) (Table 7.2). Thus, data sets reflect the actual situation for such comparisons including a wide phylogenetic variety of species (Figure 7.2).

Altogether, five pairwise alignments of prokaryotic genomes were performed.

Combinations of the genomes to be aligned were selected to be either collinearity or non-collinearity, and to have a different similarity on the DNA level in order to compare alignment approaches on a broad spectrum of sequence variability. In the following, more biological information of the genomes aligned is given. For the abbreviations of the species names please see Table 7.2.

Archaea:

a) **Pho *vs.* Pab:** Both species belong to the same genus[2] *Pyrococcus* (Figure 7.2 a), i.e., are taxonomically closely related (in the COG database even described as twins). 67.2% of the annotated genes of Pho (R) are classified as COGs to annotated genes in Pab (Q). They have nearly the same AT content of 58.1% in Pho and 55.3% in Pab (Ussery and Hallin, 2004) and a similar codon usage[3], i.e., we can expect DNA sequences of the orthologous protein genes to be highly similar. Nevertheless, genomes are not completely collinear, but show beside long collinear stretches some major rearrangements (Lecompte et al., 2001). Ecologically, both species are hyperthermophilic and obligately anaerobic (Figure 7.2a).

b) **Pho *vs.* Afu:** These species are taxonomically further apart, since they only belong to the same phylum Euryarchaeota. There are 46,5% of the annotated genes of Pho classified as COGs to Afu. Their AT content varies about 6.7% with Afu having an AT content of 51.4% (Ussery and Hallin, 2004), however, their codon usage differs and especially the frequency of the third (wobble) position is different (see Appendix A, Figure A.1). These two genomes are non-collinear. However, ecologically both species are hyperthermophilic and obligately anaerobic (Figure 7.2a).

c) **Mka *vs.* Mja:** Both species belong to the same phylum Euryarchaeota. 52.7% of the genes of Mka can be found as well in Mja, however, their AT content (38.8% in Mka and 68.6% in Mja) show major differences as well as their codon usage having a high divergence at the wobble position. Thus, DNA sequences of the orthologous genes between these species are likely to be less similar as, e.g., between species in a) and b). Furthermore, genomes are non-collinear. Ecologically, however, both species are in addition hyperthermophilic, obligately anaerobic, and methanogens (Figure 7.2a).

[2]In general, prokaryotes are taxonomically classified by: kingdom ← phylum ← subphylum ← class ← order ← family ← *genus* ← *species*, where the arrow indicates a generalization.

[3]For all species, codon usages can be found in Appendix A, A.1, Figure A. 1 to A. 8.

γ-Proteobacteria:

d) Eco *vs.* Sty: These species belong to the same family Enterobacteriaceae and are thus closer related than species under e). They have 53.0% of their genes in common (classified as COGs). Their AT content is very similar with 49.2% in Eco and 47.8% in Sty. Furthermore, their codon usage pattern appears to be similar. Thus, DNA sequences can be expected to be highly similar. The genomes of these species were described to be collinear (Kolstø, 1997). Ecologically, again both species are mesophilic, facultative aerobes, and opportunistic pathogens (Figure 7.2b).

e) Eco *vs.* Pmu: Species are taxonomically further distanced, they belong to the same subphylum γ-Proteobacteria. With only 31.1% of the genes of Eco classified as COGs to Pmu, these two species share the lowest number of genes of our examples. Their AT content varies about 10.4% (49.2% in Eco and 59,6% in Pmu). In addition, their codon usage differs and especially the frequency of the third (wobble) position is different. Genomes are non-collinear. Ecologically, both species are mesophilic, facultative aerobes, and opportunistic pathogens (Figure 7.2b).

Table 7.2: The list of species used for the evaluation. Their taxonomical allocation, number of protein genes, and number of COGs were taken from the COG data base (`http://www.ncbi.nlm.nih.gov/COG/new/release/biocats.html`).

Group	Species	Acronym	Accession	Length (bp)	Genes	COGs
Archaea	*Archaeoglobus fulgidus* DSM 4304	Afu	NC_000917	2,178,400	2,420	1,930
	Halobacterium sp. NRC-1	Hbs	NC_002607	2,014,239	2,075	1,540
	Methanosarcina acetivorans str. C2A	Mac	NC_003552	5,751,492	4,540	3,087
	Methanobacterium thermoautotrophicum str. Delta H	Mth	NC_000916	1,751,377	1,873	1,494
	Methanocaldococcus jannaschii DSM 2661	Mja	NC_000909	1,664,970	1,729	1,416
	Methanopyrus kandleri AV19	Mka	NC_003551	1,694,969	1,687	1,243
	Thermoplasma acidophilum DSM 1728	Tac	NC_002578	1,564,906	1,481	1,255
	Thermoplasma volcanium GSS1	Tvo	NC_002689	1,584,804	1,499	1,268
	Pyrococcus horikoshii OT3	Pho	NC_000961	1,738,505	1,955	1,419
	Pyrococcus abyssi GE5	Pab	NC_000868	1,765,118	1,895	1,499
	Pyrobaculum aerophilum str. IM2	Pya	NC_003364	2,222,430	2,605	1,512
	Sulfolobus solfataricus P2	Sso	NC_002754	2,992,245	2,826	2,153
	Aeropyrum pernix K1	Ape	NC_000854	1,669,695	1,841	1,225
γ-Proteobacteria	*Escherichia coli* K12	Eco	NC_000913	4,639,675	4,242	3,587
	Escherichia coli 0157:H7	Ecs	NC_002695	5,498,450	5,253	3,958
	Yersinia pestis CO92	Ype	NC_003143	4,653,728	3,885	3,219
	Salmonella typhimurium LT2	Sty	NC_003197	4,857,432	4,425	3,645
	Buchnera aphidicola str. APS (*Acyrthosiphon pisum*)	Buc	NC_002528	640,681	564	559
	Vibrio cholerae O1 biovar eltor str. N16961	Vch	NC_002505	2,961,149	2,742	2,199
	Pseudomonas aeruginosa PAO1	Pae	NC_002516	6,264,403	5,567	4,625
	Haemophilus influenzae Rd KW20	Hin	NC_000907	1,830,138	1,657	1,591
	Pasteurella multocida Pm70	Pmu	NC_002663	2,257,487	2,015	1,824
	Xylella fastidiosa 9a5c	Xfa	NC_002488	2,679,306	2,766	1,687

7.1.4 Reference Models

The reference models for KnowAlign (see 4.2.2) were generated by using all the genes annotated in the genome entry of the corresponding reference genome R. Gene names were mapped onto COG labels and stored in the local data base (see 6.1.1). Genes without COG label were stored with their annotated label. The qualitative spatial relations between these genes were computed using their annotated locations (starting index, end index, and strand) which were then stored in the data base. In addition, for the set of genomes belonging to the same taxonomic group (either Archaea or γ-proteobacteria) as the R but excluding Q, we performed DCGA experiments (see Chapter 5).

Parameters for all DCGA experiments were adjusted to the following: $w = 10$ and $q = 2$ (see Chapter 5 for explanation), i.e., all k-patterns with $2 \leq k \leq 10$ of COGs occurring in at least two of the 13 Archaea or of the 10 γ-Proteobacteria genomes were generated. The size of w was approximately determined for the Archaea reference models by increasing the size of w for the experiments until no k-patterns with a $k = w$ could be found, i.e., no 11-pattern was found with $w = 11$. For the γ-Proteobacteria the same parameters were used in order to have comparable results.

For example, for the alignment of Pho (R) *vs.* Pab (Q), all interval relations between the genes in Pho were included in the reference model. In addition, we have performed a DCGA experiment using the parameters described above on the set of all Archaea genomes (see Table 7.2) excluding the genome of Pab. Spatial relations found in frequent gene patterns were then integrated into the reference model as described in Section 4.2.2.3. In the following, this reference model is denoted by Archaea\Pab.

7.2 Results and Discussion

The results of the five pairwise genome alignments conducted with different approaches will be described and discussed in the following with respect to the quality of the alignments. Furthermore, the results obtained by the DCGA experiments (see 7.1.4) are described and discussed. However, since the main focus of this thesis is the alignment of genomes, the results are not discussed in full detail nor are they compared to other approaches for the discovery of gene clusters. We assume, however, that the results of the pattern discovery approach (see Chapter 5) bear a biological meaning (e.g., functional coupling of genes or that they reflect a phylogenetic relationship) as has been shown for other approaches (e.g., (Bansal, 1999; Overbeek et al., 1999; Ermolaeva et al., 2001; Mazumder et al., 2001; Wolf et al., 2001; Rogozin et al., 2002a)).

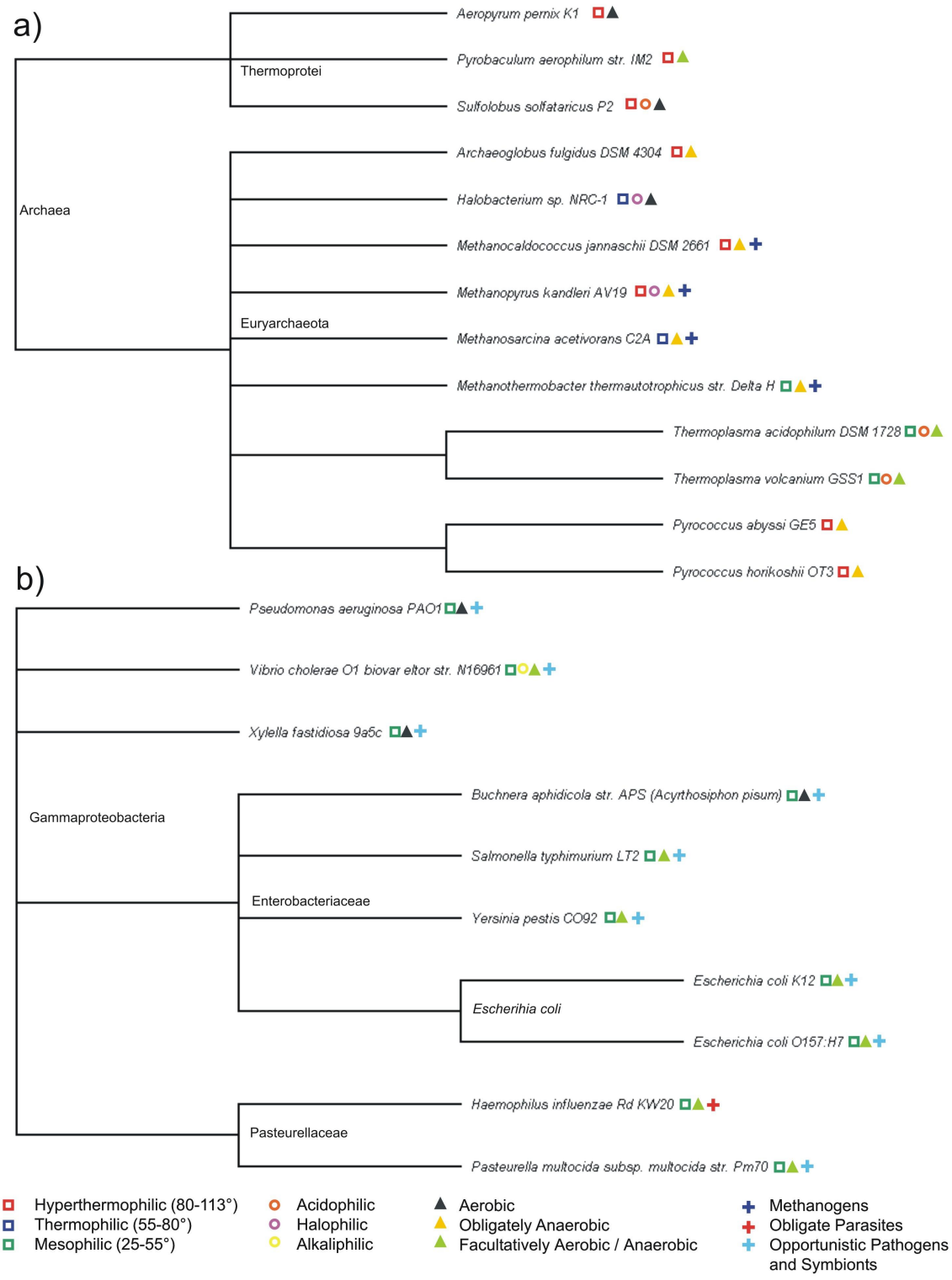


Figure 7.2: Taxonomy (retrieved from the NCBI taxonomy browser) of a) the 13 Archaea species and b) the 10 γ-Proteobacteria species used to generate reference models. In addition, species are labeled with some autecological characteristics (see legend) as specified by the COG database (`http://www.ncbi.nlm.nih.gov/COG/new/release/biocats.html`).

7.2.1 Gene Patterns

The total number of gene patterns and their distribution to k-patterns for all five reference models can be found in Figure 7.4. Some examples of gene patterns discovered for the *trp* operon are given in Figure 7.3. For the three Archaea reference models (Figure 7.4 a, b, and c), the majority ($\sim$93%) of these patterns are present on average up to 64.5% in two genomes, up to 18.6% in three genomes, and up to 9.8% in four genomes. For the two γ-Proteobacteria reference models (Figure 7.4 d and e), the majority ($\sim$95%) of these patterns are present on average up to 59.6% in two genomes, up to 25.4% in three genomes, and up to 9.9% in four genomes. These distributions with a higher percentage of patterns in three genomes for the γ-Proteobacteria reflect the taxonomy of the two groups, since genomes in the group of γ-Proteobacteria are more closely related (e.g., in the family of Enterobacteriaceae) than in the Archaea (Figure 7.2), and closely related species are supposed to have a higher gene order conservation (Tamames, 2001; Mira et al., 2003) (see as well 2.2.2.4). There is a peak of the number of 5-patterns observable for the Archaea genomes while for the γ-Proteobacteria the number of patterns always increases with a decreasing number k. However, we can not give an explanation for this observation without a more detailed analysis of the results which out of the scope of this thesis.

For all DCGA experiments, an increase in the number of 2-patterns (i.e., consisting of two genes) and a decrease of larger k-patterns (three to ten genes) can be observed if a genome of a species closely related to another one of the reference set is removed from it, whereas by removing a further distanced species the number of larger patterns increases. For example, Pho and Pab (Archaea) are two closely related species. By removing Pab (Figure 7.4 a), the number of 2-patterns is 1840 and of 10-patterns is four in comparison to 1746 2-patterns and eight 10-patterns by removing Afu (Figure 7.4 b). In addition to the distributions of the k-patterns, this reflects the phylogenetic relationships of the genomes in the reference sets.

The frequencies of spatial relationships holding between adjacent COGs in these k-patterns show a clear dominance of the relationship *before*, followed by *overlaps* and *meets* (Table 7.3). An up to a four fold higher frequency of the relationship *overlaps* in the Archaea patterns in comparison to patterns found in the γ-Proteobacteria can be observed. The relationship *overlaps* between two genes provides a genetic basis for protein fusions (Zheng et al., 2002) usually generated by the loss of the stop codon (Snel et al., 2000; Yanai et al., 2002). Such genes are transcribed to the same mRNA, i.e., they are polycistronic (see 2.2.2.1). In the case if two genes meet (i.e., they are adjacent without any non-coding base in between, see Figure 4.3), the promotor region of the second gene in order has to be located within its predecessor and the terminator region of the first one has to be located in its successor. This would increase the evolutionary pressure on the DNA sequence in order to conserve these regions (e.g., even if

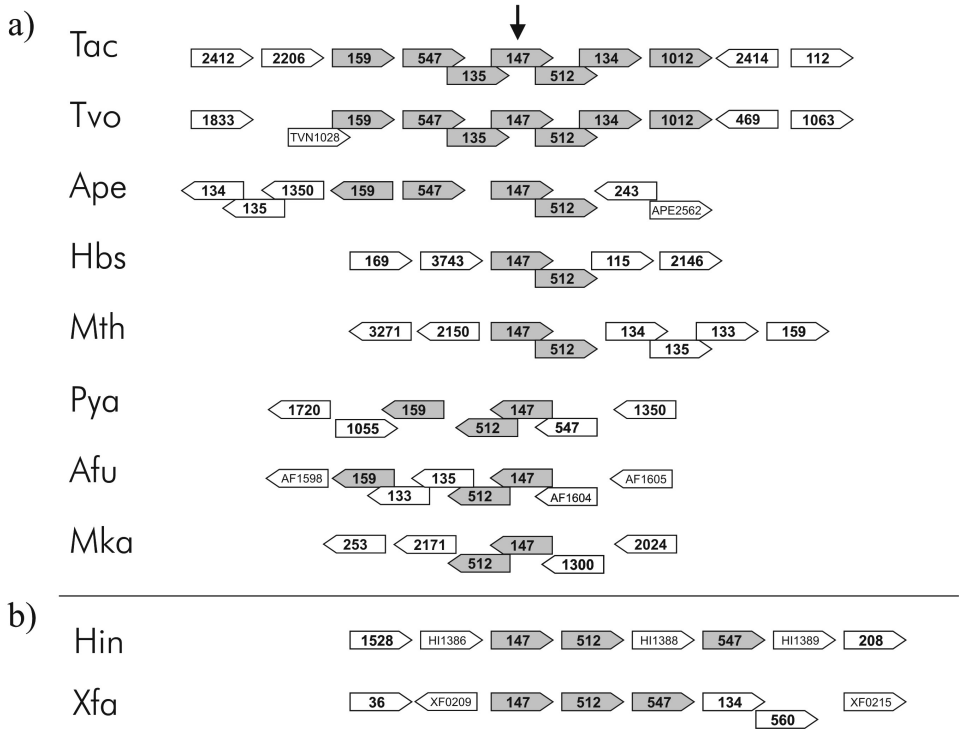


Figure 7.3: Examples of gene patterns found for the *trp* operon in the reference models a) Archaea\Pab and b) γ-Proteobacteria\Sty. Only those gene patterns containing COG0147 (Anthranilate/para-aminobenzoate synthases component I) and COG0512 (Anthranilate/para-aminobenzoate synthases component II) are shown. Genes are indicated by arrows which gives their direction of transcription. The numbers in the genes are their COG labels, i.e., 147 means COG0147.

other encodings for the amino acids would be possible it may be necessary to keep the status quo to ensure the regulatory function). This is probably the reason for the low frequency of the relationship meets between adjacent COGs in this data set.

7.2.2 Alignment Qualities

The following discussion starts with the comparison of the *Incremental Search* strategy of KnowAlign (see 4.4.2) to the other approaches before we discuss the results of the three different approaches implemented in KnowAlign among each other.

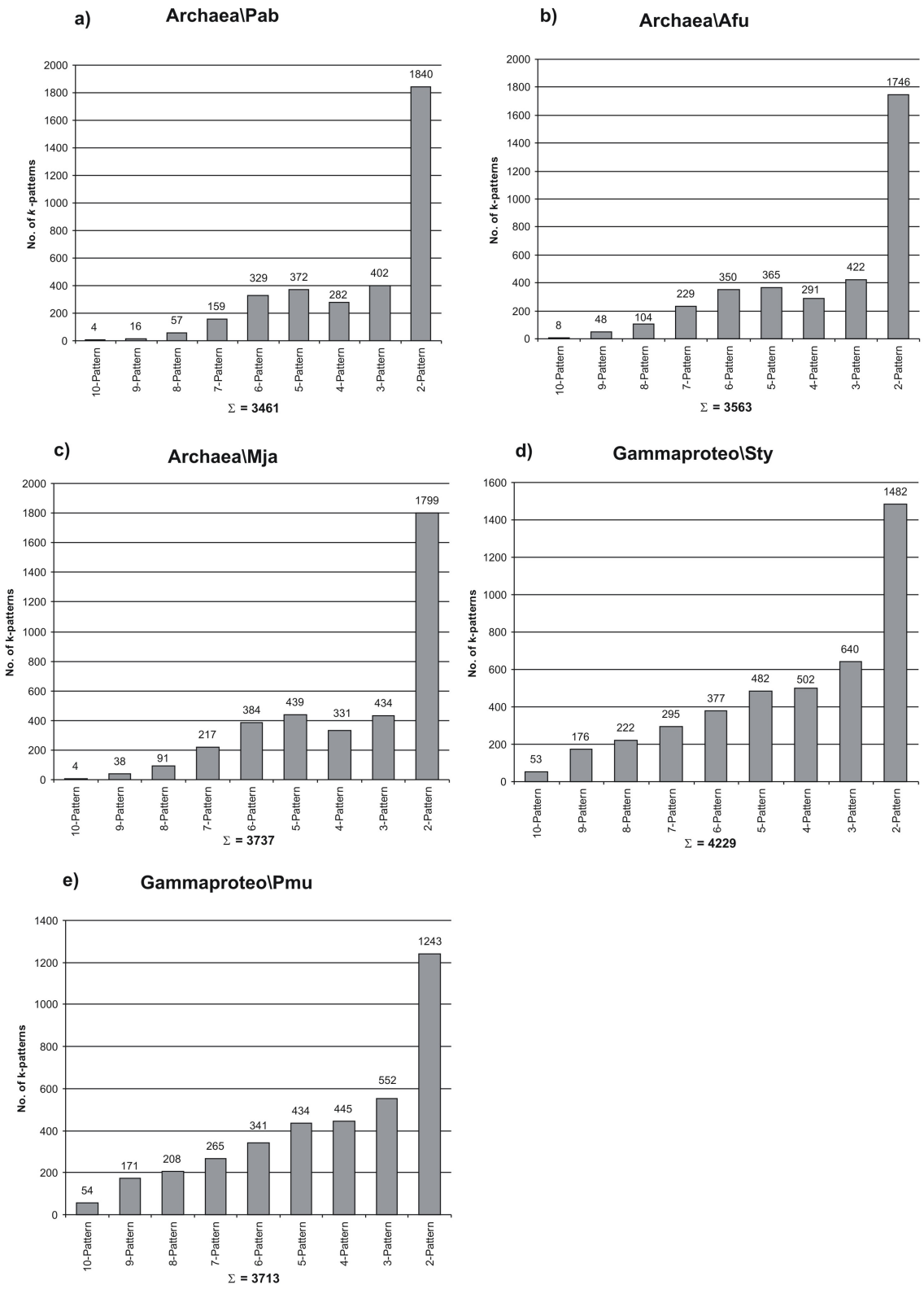


Figure 7.4: Number of k-patterns found by the DCGA experiments ($q = 2$ and $w = 10$ for explanation) in all Archaea species a) without Pab, b) without Afu, and c) without Mja; and in all γ-Proteobacteria d) without Sty and e) without Pmu.

Table 7.3: Frequencies of spatial relations in gene patterns of adjacent COGs (genes). For example, Archaea\Pab denotes the reference model generated using all Archaea genomes without the genome of Pab.

	Reference Model	*before* (%)	*overlaps* (%)	*meets* (%)
a)	Archaea\Pab	88.46	11.41	0.12
b)	Archaea\Afu	86.38	13.41	0.21
c)	Archaea\Mja	86.82	12.98	0.20
d)	γ-Proteobacteria\Sty	96.96	2.98	0.06
e)	γ-Proteobacteria\Pmu	96.60	3.25	0.07

7.2.2.1 Comparing KnowAlign to other Alignment Tools

The coverage for all approaches (compare Table 7.1) of all five genome alignments (compare 7.1.3 for sequence data) can be found in Figure 7.5. In addition, Figure 7.6 shows the sensitivity *vs.* the specificity of these approaches for the five alignments.[4] For the alignment of Eco *vs.* Sty (Figure 7.5d) the result of Pipmaker (single coverage) is lacking since the Pipmaker server cancelled the computation because of a time limit. Moreover, no results for DIALIGN (see 3.2.2.5) are available at all, since none of the alignments performed terminated after two weeks computation time which corresponds to other observations for this approach described in Chain *et al.* (Chain et al., 2003).

In general, the coverage decreases with decreasing similarity of codon usages and decreasing number of COGs, i.e., with decreasing sequence similarity (Figure 7.5). The highest coverage for most approaches of the five genome alignments is observed for Eco *vs.* Sty (Figure 7.5d) and Pho *vs.* Pab (Figure 7.5a). The species of these alignments share the highest percentage of COGs of the five examples, have similar codon usages, and similar gene orders, i.e., their DNA sequences are quite similar in comparison to the pairs used in the remaining three alignments (see 7.1.3). In all five comparisons, the coverages obtained by either BLAST, BLASTZ, Pipmaker (based on BLASTZ, see 3.2.2.3), or WABA were the highest. Simple BLAST had for Pho *vs.* Pab (Figure 7.5a) the highest coverage, WABA for Eco *vs.* Sty (Figure 7.5d) and in the other alignments BLASTZ performed the best. In particular, for further distanced species as Pho *vs.* Afu (Figure 7.5b), Mka *vs.* Mja (Figure 7.5c), and Eco *vs.* Pmu (Figure 7.5e) BLASTZ, Pipmaker, and WABA showed high coverages in comparison to the other approaches. However, the coverages of KnowAlign (BLASTZ, $len = 100$, $x = 90$) using as well BLASTZ as the input are in comparison to the other approaches in these cases always higher. This indicates that the strategies single coverage of Pipmaker (keeping only the highest scoring hits) and chaining (keeping only hits appearing in the

[4]The raw data for all calculations of the alignment qualities can be found in Appendix B.

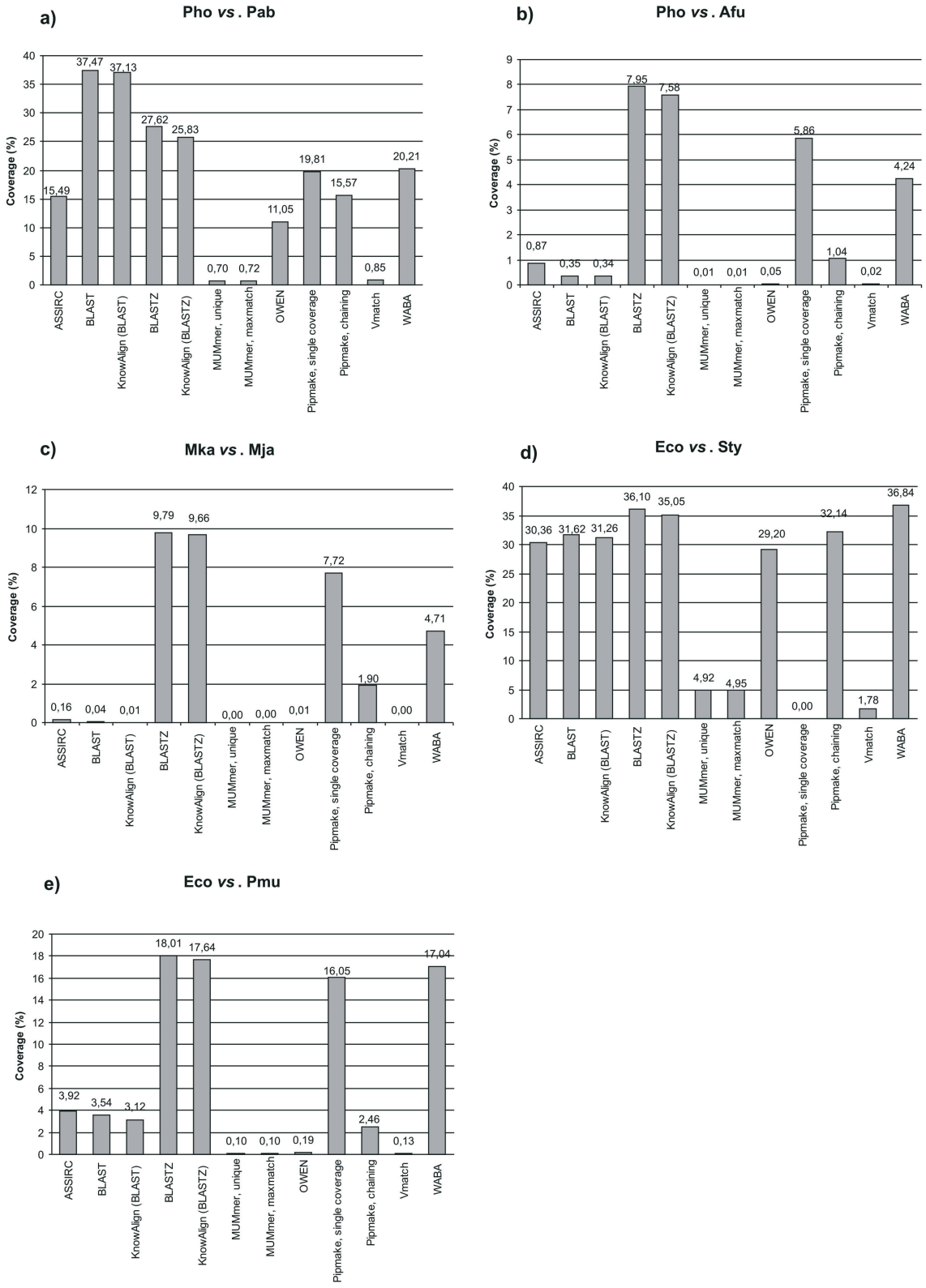


Figure 7.5: Coverage of KnowAlign in comparison to other approaches.

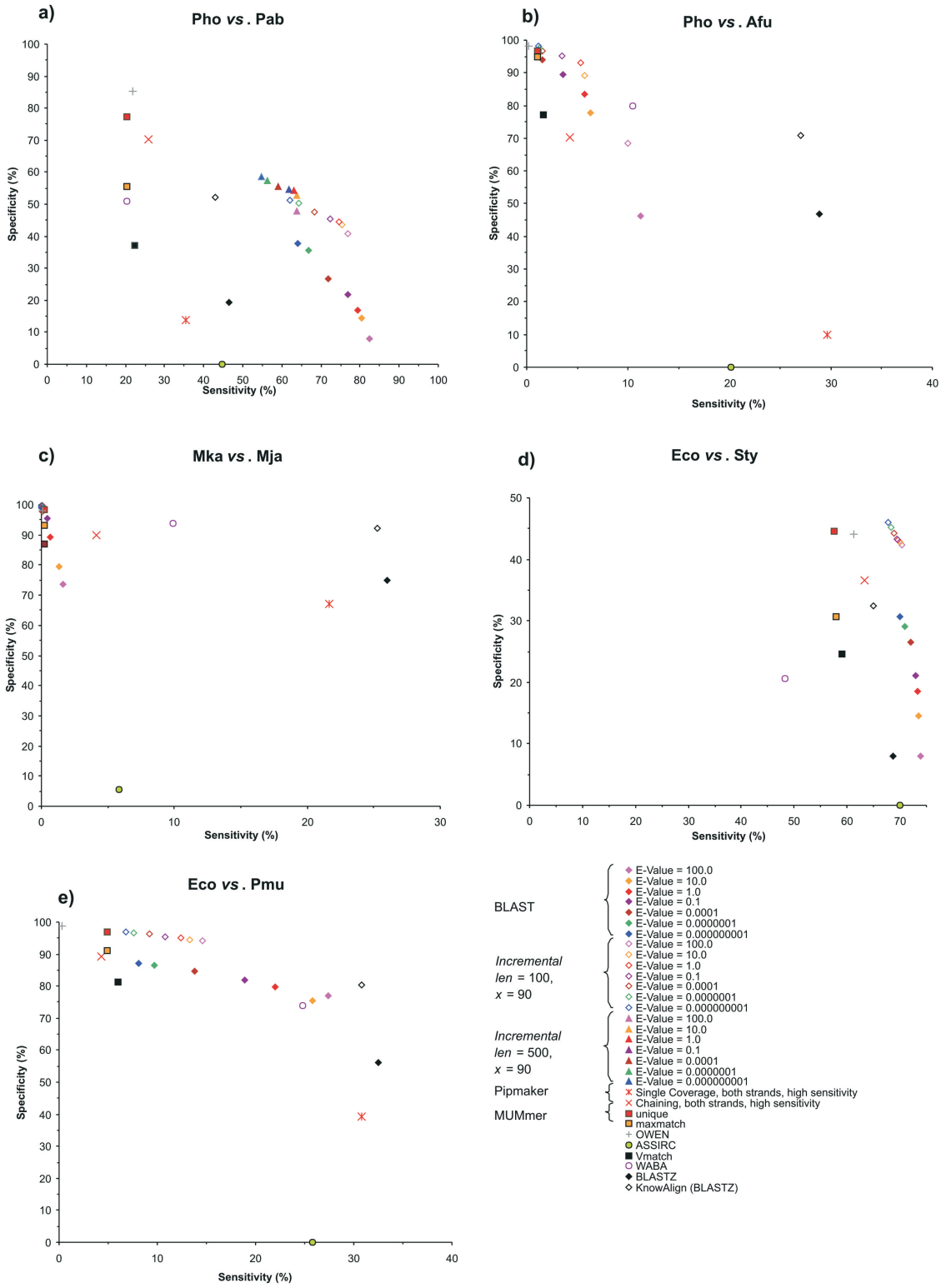


Figure 7.6: Sensitivity *vs.* specificity of KnowAlign in comparison to other approaches.

same relative order) are very strict, since at any region of the alignment the same conditions are assumed. In contrast, in KnowAlign the modeling of genome architectures in the reference model and the comparison formalized as an incremental constraint satisfaction problem enable a flexible evaluation of hits with respect to different regions (e.g., a region representing an operon where genes are clustered in contrast to a patchy region of genes without functional correspondence). In the case of WABA, the consideration of the wobble (third codon) position as a variable position in the alignment enables this approach to get a higher coverage in cases where the codon usages of the genomes are non-similar (see 7.1.3 for a description of the sequence data).

Since the coverage as defined here (see Definition 7.3) is directly connected to the sensitivity via the hits classified as *true positives*, the described decrease of the coverage with decreasing sequence similarity correlates with a decrease of the sensitivity of the alignments. The alignments of the genomes having the highest sequence similarity have the highest sensitivities (Figure 7.6). For each of the five alignments, the sensitivity decreases with increasing requirements on the seeds (e.g., for MUMmer or Vmatch 'anchors' of a min. length of 18 bp, for BLAST w-mers of min. length 11 bp), i.e., increasing length and increasing similarity. In particular, for anchor-based approaches like MUMmer and Vmatch which depend on relatively long seeds as basis for their alignments (i.e., which are very strict with respect to the similarity), the sensitivity decreases to nearly zero (i.e., nearly no *true positive* hits found) with decreasing sequence similarity (e.g., Figure 7.6c). Furthermore, these approaches require that the genomes to be aligned are collinear (see Figure 3.5) (Chain et al., 2003) which is only the case for Eco *vs.* Sty. Here, these approaches perform best, however, still not as well as KnowAlign (BLAST, $len = 100$, $x = 90$) and OWEN (Figure 7.6d).

Overall, KnowAlign based on either BLAST or BLASTZ has the highest sensitivity with simultaneously having the highest specificity which is always as high as most other approaches with a considerably lower sensitivity (Figure 7.6). The sensitivity (and in consequence as well the coverage) which can be achieved by KnowAlign depends on the sensitivity of the underlying local similarity search, i.e., the sensitivity (coverage) of KnowAlign can never exceed the input sensitivity (coverage) of BLAST or BLASTZ, respectively. Indeed, the sensitivity of KnowAlign is for the five cases tested always lower than its original input (Figure 7.7). This may be caused by two reasons, either

1. the reference model is over-constrained (see 4.4.1) or
2. the local similarities classified as *true positive* which are removed by KnowAlign are not real *true positives*, i.e., they do not reflect a true phylogenetic relationship, but are hits occurring only by chance (*false positives*).

On the one hand, the majority of the *true positives* deleted by, e.g., KnowAlign (BLAST, E-value = 100, $len = 100$, $x = 90$) for all five alignments have a length

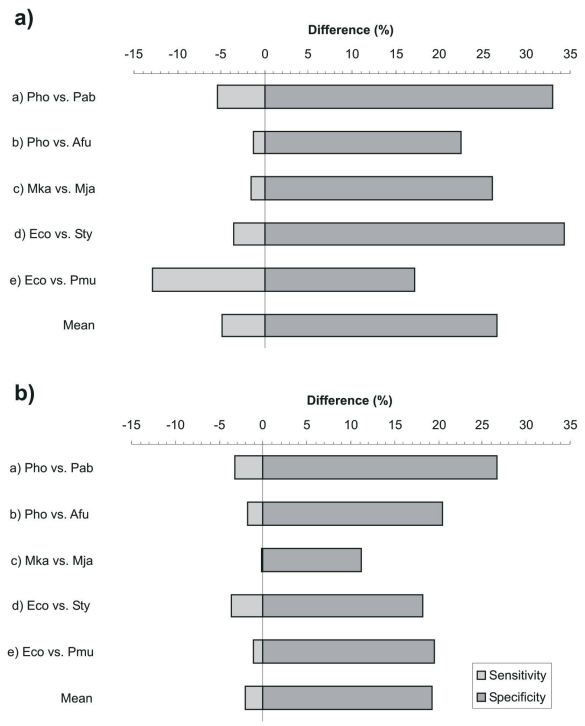


Figure 7.7: Differences of sensitivity and specificity between a) BLAST (E-Value = 100) and KnowAlign(BLAST, E-Value = 100, $len = 100$, $x = 90$) and b) BLASTZ and KnowAlign(BLASTZ, $len = 100$, $x = 90$).

<100 bp which supports the assumption that many of them could be in reality *false positives*. On the other hand, even for KnowAlign (BLAST, E-value = $1{\cdot}10^{-8}$, $len = 100$, $x = 90$) the sensitivity is lower than the original input indicating that the reference model may be over-constraint. However, a further (manual) evaluation of their quality is missing and thus we can only state that the combination of both reasons lead to this reduction in sensitivity. There is, however, always an increase in the specificity between the original input and its usage by KnowAlign (KnowAlign (BLAST, E-value = 100, $len = 100$, $x = 90$): mean 26.60±7.16%, KnowAlign (BLASTZ, $len = 100$, $x = 90$): mean 19.24±5.51%) which is considerably higher than the simultaneous decrease in sensitivity (KnowAlign (BLAST, E-value = 100, $len = 100$, $x = 90$): mean -4.94±4.75%, KnowAlign (BLASTZ, $len = 100$, $x = 90$): mean -2.00±1.44%) (Figure 7.7). Furthermore, KnowAlign (BLAST, E-value = 100, $len = 100$, $x = 90$) has in comparison to BLAST with an E-value = $1 \cdot 10^{-8}$ (a more likely adjustment of this parameter for such comparisons) in all cases a higher sensitivity and coverage, and in three cases even a higher specificity (Figure 7.6). For example, in the case of Pho *vs.* Pab, this increase in sensitivity is 12.9% and of specificity is 3.2%.

Table 7.4: Distribution of the total increase in specificity by KnowAlign (BLAST, E-value = 100, $len = 100$, $x = 90$) in comparison to its input based on 1. singular instantiation of variables, 2. path consistency, and 3. comparison to reference model.

	Reference Model	**1 & 2 (%)**	**1 & 3 (%)**	**Ratio** ($\frac{1\&2}{1\&3}$)	**COGs (%)**
a)	Pho *vs.* Pab	18.66	14.28	1.31	67.20
b)	Pho *vs.* Afu	15.10	7.34	2.06	46.50
c)	Mka *vs.* Mja	18.07	8.01	2.26	52.70
d)	Eco *vs.* Sty	18.43	15.89	1.16	53.00
e)	Eco *vs.* Pmu	3.70	13.53	0.27	31.10

The increase in specificity which can be reached using the *Incremental Search* strategy of KnowAlign is a result of the combination of

1. the singular instantiation of variables (genomic features) with PGFs,
2. the assurance of the interval path consistency of the instantiated PGFs, and
3. the comparison of the spatial configuration of the PGFs with the knowledge of the genome architecture modeled in the reference model.

The distribution of the total increase in specificity by the above described factors for all five alignments can be found in Table 7.4. For the example of Pho *vs.* Pab, the combination of 1. & 2. gives an increase of 18.66% (Figure 7.8, KnowAlign (BLAST, E-value = 100, $len = 0$, $x = 90$))[5] and the application of 1. & 3. gives an additional increase of 14.28% (Figure 7.8, KnowAlign (BLAST, E-value = 100, $len = 100$, $x = 90$)) of a total increase in specificity of 32.94%.

7.2.2.2 Comparing Different Parameter Settings of KnowAlign

The parameters of KnowAlign (see Table 7.1) have only a slight influence on the resulting alignment qualities. In the following, we will discuss their influence on the example of Pho *vs.* Pab, however, in principle their influences are similar for the other alignments (see Appendix B). With an increase of the parameter len (0, 100, 200, or 500 bp) an increase of specificity is observable (Table 7.5), since this in turn increases the number of PGFs to be evaluated by the reference model (see Appendix A, Table A.1). On the other hand, this increase in specificity results always in decreasing sensitivity which is, however, usually lower than the increase in specificity (see above). The same pattern is observable for an increase of the

[5]By adjusting $len = 0$ no comparison to the reference model occurs (see 4.4.2), since $|G^*| = 0$ (see Appendix A, Section A.2, Table A.1).

Table 7.5: Comparison of the alignment qualities (sensitivity = sen., specificity = spec., and coverage = cov.) for the three approaches (see 4.4) of KnowAlign for the evaluation of local similarities. Results are shown for Pho *vs.* Pab with an BLAST input of an E-value = 100.

Approach	**Parameters**	**Sen. (%)**	**Spec. (%)**	**Cov. (%)**
Backtracking Search	$len = 0$, $x = 90$	-	-	-
Incremental Search	$len = 0$, $x = 90$	79.89	26.54	37.16
	$len = 100$, $x = 0$	79.17	33.95	37.14
	$len = 100$, $x = 60$	77.08	39.65	37.14
	$len = 100$, $x = 90$	76.92	40.82	37.13
	$len = 100$, $x = 120$	76.76	40.95	37.13
	$len = 200$, $x = 90$	72.12	43.82	36.62
	$len = 500$, $x = 90$	63.94	47.98	34.61
Optimization	$len = 100$, $x = 90$, $max\_tries = 100, \alpha = 1$	76.68	40.65	37.03

parameter x (the tolerance of bases upstream and downstream which are searched in the query sequence for the start and stop codons)(Table 7.5). With the increase of x (0, 60, 90, or 120 bp), the specificity increases and the sensitivity decreases. The highest increase of the specificity occurs when increasing x from 0 to 60 bp (i.e., a search for start and stop codons is enabled) and decreases with further increasing x (Table 7.5). This indicates that by identifying the start and stop codons for PGFs partially representing a gene, the precision of the determination of the spatial relations between genes in the query sequence is increased which enables a more plausible evaluation by the reference model (see 4.3.2.2).

For all five alignments, the *Backtracking Search* of KnowAlign (see 4.4.1) gave no results, i.e., the search for a solution returned *failure* (see Algorithm 3). The possible reasons for that are: the reference model is over-constrained, or for some genes only *false positive* PGFs exist or both (more details can be found in 4.4.1). The generation of *false positive* PGFs is quite common, since in order to detect less conserved regions when comparing further distanced species we have lowered the similarity threshold for the local alignments. Thus, *Backtracking Search* is not an adequate search strategy (not even for closely related species as Eco and Sty) for the problem of evaluating local similarities with a low similarity value found between prokaryotic genomes as formalized in this thesis. For the *Optimization* strategy (with an $\alpha = 1.0$, see 4.4.3.1), altogether 136 comparisons using the alignments of the *Incremental Search* as the input were produced (i.e., all combinations of BLAST E-values with all adjustments for *len* for all five alignments, see Table 7.1). In the majority (129 of 136) of these comparisons, the *Optimization* had no effect on the alignment qualities and in 7 cases the quality

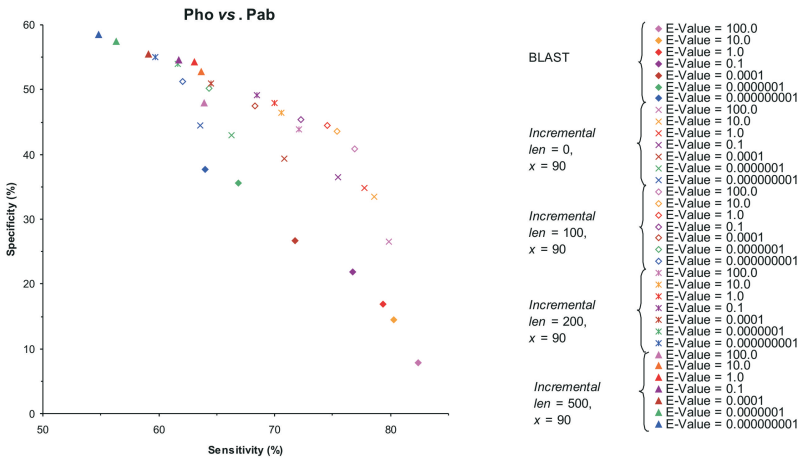


Figure 7.8: Sensitivity *vs.* specificity of BLAST and KnowAlign for Pho *vs.* Pab with different minimum lengths (*len*).

was decreased. These decreases can however be neglected, since the difference of *true positives* to the original input was maximally one with a maximal decrease in coverage of 0.02%. Thus, we can state that the *Optimization* strategy was not able to increase the alignment qualities for the alignments performed. This is probably due to two reasons:

1. in many cases there exists only one PGF for a genomic feature, in particular for further distant species and thus no alternatives exists for the optimization strategy and

2. the initial instantiation of the genomic features with the longest PGF is a good heuristic in order to find a plausible alignment which can later not be optimized.

7.2.2.3 Conclusion

Altogether, only the *Incremental Search* strategy of KnowAlign was able to generate biologically plausible results. We were able to show that alignments at different levels of sequence similarity (taxonomical relationship) are possible with this approach. Furthermore, using KnowAlign alignments for Archaea as well as (Eu)bacteria of different lengths are possible. Results shown indicate an increase in sensitivity by using sensitive inputs (by lowering the threshold of the similarity value) with a simultaneously high specificity. The factors which can be influenced by a user in order to get high quality results are:

1. the original sensitivity of the underlying local similarity search in order to get a high sensitivity and

2. the quality of the reference model in order to get a high specificity.

In order to get a high sensitivity, a user should decide for a very sensitive but not necessarily specific local alignment approach to combine it with KnowAlign. In addition, the threshold for the local similarity value could be lowered (e.g., the E-value for BLAST) to increase its sensitivity. Thus, BLAST has been selected for the prototype described in Chapter 6, since in many cases it is the approach with the highest sensitivity and furthermore, it is well established and accepted within the bioinformatics and biology communities. For an increase in specificity, effort by a user (expert) is required in order to get a reference model of high quality (i.e., selecting an adequate reference sequence and genomes for the reference set). Here we have generated very simple reference models where genomes included were selected only according to their taxonomical relationship (see Figure 7.2). More sophisticated reference models, however, can be generated by selecting genomes (species) using more biological background knowledge, e.g., their ecological lifestyle (see Figure 7.2) which reflects their gene content with respect to the capability of specific metabolic pathways (Huynen et al., 2000). This could help to identify more informative genomic patterns using the DCGA approach described (see Chapter 5), and in turn to overcome the problem of an over-constrained reference model. Remember that over-constrainment is caused by a limited set of spatial relations modeled between genomic features. This possibility will increase in the future with an at present exponentially increasing number of prokaryotic genomes available for such comparisons. However, a considerable amount of work is required to accurately model the structure of the reference model, but once a reference genome is modeled accurately it will prove to be helpful to many analysis. Eventually, a user should adjust parameters of KnowAlign (len and x) according to the expected sequence similarity (DNA and genome architecture) between the reference and the query sequence, where in general len should be lowered and x should be raised with decreasing sequence similarity.

Finally, we could show that reasoning (ensuring the consistency and find a consistent scenario, see 4.2.1.3) on the interval framework in order to model prokaryotic genome architecture is practicable, since all alignments could be accomplished within an acceptable computation time. For example, the generation of the reference model plus the performing of the alignment of Mka *vs.* Mja took approximately two hours, of Pho *vs.* Pab approximately five hours, and of Eco *vs.* Sty approximately eight hours on a PC with 1 GHz and 750 MB memory. Thus, the approach described increases the computational expenses of prokaryotic genome alignments whereas the biological plausibility of the analysis is significantly improved. In summary we can state that thesis 1.1.1 *"Using the biological context*

of local similarities found between genomic sequences can support their biologically feasible assignment for interspecies genome alignments" is supported by the results shown in Figures 7.4 to 7.8 and Tables 7.3 to 7.5.

Chapter 8

Summary and Perspective

The comparison of prokaryotic genomes is an important analysis step in bioinformatics which can help to discover genes and their functions, regulatory mechanisms, and phylogenetic relationships. Alignments generated with tools available for this task commonly show a relative low coverage and sensitivity, since only strong local similarities (i.e., the longest similarities with the highest similarity values) are integrated into the alignment. The reason is the high number of global mutations (e.g., rearrangements and duplications) which frequently occur in prokaryotes which causes ambiguity during the assignment of similar subsequences of the genomes. This thesis describes a new knowledge-based framework which integrates biological knowledge of prokaryotic genome architecture into the alignment process of whole genomes in order to increase its quality (i.e., sensitivity, coverage, and specificity). The approach validates local similarities found between a reference genome for which knowledge exists and a query genome for which knowledge is not necessarily available with respect to the genome architecture of the reference genome. The framework allows for a manual and a automatical integration of more knowledge of common genome architecture into the reference model. The application of the knowledge-based framework on the set of local similarities found with a relaxed similarity threshold can increase the specificity of the resulting alignment while the sensitivity remains nearly as good as with a strict similarity value. Moreover, by using BLAST with an E-value = 100 (a very unlikely adjustment of the E-value parameter because many unspecific hits will be found) the knowledge-based framework had in comparison to BLAST with an E-value = $1 \cdot 10^{-8}$ (a more likely adjustment of this parameter) in all cases a higher sensitivity and in the majority of cases even a higher specificity.

Two major research questions had to be addressed during the development of the knowledge-based framework, namely,

1. how can a common prokaryotic genome architecture be represented to allow for a computation and

2. how can this knowledge be discovered in a set of real multiple genomes in

order to integrate it into the reference model.

The knowledge-based framework developed represents the order of genomic features (genes and regulatory regions) of a prokaryotic genome as linear structures and uses a system of qualitative spatial relations (e.g., *before*, *after*, *meets*, or *overlaps*) known from research on temporal and spatial reasoning (Beek, 1992). In addition, in order to represent the likelihood of a spatial relation between genomic features a weight can be assigned to each relation modelled. In the following, the set of genomic features can be formalized as binary variables on which constraints (the spatial relations) are defined. This allows a formalization of the computation tasks as a *constraint satisfaction problem* (CSP). Thus, consistency of the knowledge contained in the reference model can be ensured by a path consistency algorithm which finds all feasible relations between all pairs of genomic features. An alignment is then a solution to the CSP, i.e., a consistent instantiation of all genomic features with local similarities. The procedure to generate an alignment consists of three steps, namely,

1. the generation of a reference model,
2. the local similarity search with a subsequent step to combine them to constitute *putative* genomic features (PGF) of the query genome, and
3. the evaluation of the PGFs using the reference model.

The generation of a reference model starts with the modeling of the genomic features of the reference sequence. In addition, a method to discover a common genome architecture in a set of real multiple genomes was developed. The method uses the same spatial representation of genome architecture as for the alignment procedure and is based on association rule mining. It finds all patterns of genomic features in a sliding window of a given size which occur in a minimum number of genomes. The benefit of the approach is that not only all frequent genomic features in the window will be identified, but that a pattern comprises also their spatial configuration. Thus, an *overlaps* or a *meets* of e.g., two genes which is a potential gene fusion event can be distinguished from an ordinary succession of genes which is model by the relation *before*. The discovered knowledge can directly be integrated into the reference model of the knowledge-based genome alignment framework.

Prior to the validation of the local similarities, they are combined to constitute *putative* genomic features of the query genome. A PGF of the query genome is a chain of local similarities all having the same relative order in both genomes and appearing in the same genomic feature of the reference genome. Furthermore, for a PGF constituting a protein coding gene its borders if not already covered by a local similarity are detected by searching for start- and stop-codons.

Based on the knowledge representation of the reference model, three algorithms were developed which have different instantiation strategies of the genomic

features (variables). The first approach is a standard backtracking algorithm commonly used in solving CSPs. However, this strategy requires a consistent instantiation of all variables and thus may fail in finding a solution to the CSP, since for some genomic features only short local similarities occurring just by chance may exists and for others no local similarity may exist at all because of the absence of this genomic feature from the query genome. To overcome this problem, the second approach relaxes the demands on a solution and requires only the instantiation of genomic features which have a PGF of a user-defined minimum length. Subsequently, the PGFs not fulfilling this requirement are incrementally introduced to the solution and if their instantiation is consistent with it become part of it. The third approach tries to optimize a solution with respect to its coverage (i.e., the sum of the length of all instantiated PGFs) and its specificity (i.e., the precision for excluding wrong assigned PGFs) of the alignment by taking the weights of spatial relations into account. Here, relations with higher weights are preferred, since they are more likely to reflect a biological meaning.

A prototype (called KnowAlign) of the approach was implemented as a database-based JAVA application which uses as an external source a stand alone BLAST implementation in order to perform local alignments. KnowAlign was compared to eight other publicly available genome alignment tools (MUMmer, Pipmaker, OWEN, ASSIRC, Vmatch, WABA, BLASTZ, and BLAST) in five pairwise genome alignments of species at different evolutionary distances (including Archaea and Bacteria). In all comparisons, the incremental extension strategy of KnowAlign had always a higher sensitivity (on average +19.5%) and coverage while simultaneously having only a slight decrease in specificity (on average -3.7%) in contrast to the other alignment tools (with default parameter settings). From the three strategies implemented in KnowAlign only the incremental extension strategy showed good results while backtracking never found a solution and the optimization strategy did not find a better solution as the one generated by the incremental strategy.

8.0.3 Future Work

Presently, the genomic features annotated in genome entries in e.g., GenBank are mainly protein coding genes, tRNAs, and rRNAs. Less knowledge is available on regulatory units like promotors, terminators, or operators. However, in the future this knowledge will be accessible, e.g., by integrating further data mining approaches into the framework or by integrating the knowledge of other (secondary) databases. We expect that this will increase the plausibility of finding a consistent instantiation. Each variable to be instantiated would have to be validated to more variables which increases the likelihood that a inconsistency emerge. This in turn can increase the precision for excluding wrong assigned PGFs and thus to increase the specificity. Furthermore, such a more comprehensive knowledge-base can be adapted to support a data mining tool to find

regulatory units by validating its predicted regulatory units by checking whether the spatial configuration of the unit is valid with its knowledge.

A further extension is the integration of an enhanced gene-prediction procedure for the query genome into the framework which goes beyond the simple start-codon and stop-codon searches within a reading frame as used here. This could increase the quality of the alignment by increasing the precision of the prediction of the borders of putative genes in the query genome and thus enabling for a better validation of local similarities within these genes. In cases where the gene prediction in the reference genome is not complete, this procedure could as well used on the reference sequence which would increase the number of genomic features for the reference model.

Another interesting task is the extension of the alignment procedure to a multiple genome alignment. Here, however, different strategies for the evaluation phase of the local similarities have to be distinguished depending on:

- the underlying alignment strategy (e.g., MGA, Mauve, or ABA could be used),
- the availability of knowledge for the genomes under evaluation.

The alignment strategy used can either search for local similarities occurring in all genomes (like the multiMEMs used by MGA) or it could pairwise search for local similarities between all combinations of the input genomes (like the multi-MUMs used by Mauve). Using an approach searching for local similarities occurring in all genomes (which is identical to aligning all query genomes pairwise to the reference genome), the knowledge-based framework could be used as is, since any such 'anchor' would have to occur in the (single) reference genome and thus can be mapped to one of its genomic features. However, this kind of alignment is not a real multiple alignment where usually one wants to identify all local similarities between all combinations of genomes. For this purpose, the reference model needs to be extended with genomic features appearring in the query genomes in order to validate local similarities occurring only between query genomes.

The prototype KnowAlign implemented here should be extended to enable a multithreading in order to allow for a highly parallel computation of alignments. Furthermore, the user interface should be adapted to enable a user to interactively adapt resulting alignments, to highlight different genomic features, and with an enhanced explanation component.

The method to discover a common genome architecture can be extended in a way that the window size iteratively increases until no more significant patterns are found. Since the method described is computational demanding, a standard gene clustering approache could be used as a preprocessing step and the clusters found by such an approach serve as the input for the pattern discovery. Accordingly, this could prune the search space.

By modeling the two strands of a genome separately, the pattern discovery can be used to identify operons in a set of prokaryotic genomes. The precision of the operon prediction can then be increase by combining it with other approaches using e.g., intergenic distances between genes, the annotation of genes, and a metabolic mapping.

Bibliography

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and Verkamo, A. I. (1996). Fast discovery of association rules. In Fayyad, U. M., Piatesky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. MIT Press.

Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K., and Watson, J. D. (1994). *Molecular Biology of the Cell*. Garland Publishing, New York, USA, 3. edition.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

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

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402.

Andersson, S. G. E. (2000). The genomic gamble. *Nature Genetics*, 26:134–135.

Andersson, S. G. E. and Eriksson, K. (2000). Dynamics of gene order structures and genomic architectures. In Sankoff, D. and Nadeau, J. H., editors, *Comparative Genomics*, volume 1 of *Computational Biology Series*, pages 267–280. Kluwer Academic Publisher, London.

Bansal, A. K. (1999). An automated comparative analysis of 17 complete microbial genomes. *Bioinformatics*, 15(11):900–908.

Bansal, A. K., Bork, P., and Stuckey, P. J. (1998). Automated pair-wise comparisons of microbial genomes. *Math. Modelling and Sci. Computing*, 9(1):1–23.

Beek, P. v. (1992). Reasoning about qualitative temporal information. *Artificial Intelligence*, 58:297–326.

Beek, P. v. and Manchak, D. W. (1996). The design and experimental analysis of algorithms for temporal reasoning. *Journal of Artificial Intelligence Research*, 4:1–18.

Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., and Wheeler, D. L. (2003). GenBank. *Nucleic Acids Research*, 31(1):23–27.

Borning, A., Freeman-Benson, B., and Wilson, M. (1992). Constraint hierarchies. *LISP and Symbolic Computation*, 5:223–270.

Bray, N., Dubchak, I., and Pachter, L. (2003). AVID: A global alignment program. *Genome Research*, 13(1):97–102.

Bronstein, I. N., Semendjajew, K. A., Musiol, G., and Mühlig, H. (2001). *Taschenbuch der Mathematik*. Harri Deutsch, Frankfurt am Main, 5. edition.

Bruijn, F. J. d., Lupski, J. R., and Weinstock, G. M. (1998). *Bacterial Genomes: Physical Structure and Analysis*. Kluwer Academic Publishers, Boston.

Buckley, M. R. (2004). The global genome question: Microbes as the key to understanding evolution and ecology. Technical report, American Academy of Microbiology.

Buhler, J. (2001). Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, 17(5):419–428.

Casjens, S. (1998). The diverse and dynamic structure of bacterial genomes. *Annu. Rev. Genet.*, 32:339–377.

Chain, P., Kurtz, S., Ohlebusch, E., and Slezak, T. (2003). An applications-focused review of comparative genomics tools: Capabilities, limitations and future challenges. *Briefings in Bioinformatics*, 4(2):105–123.

Cohn, A. G. and Hazarika, S. M. (2001). Qualitative spatial representation and reasoning: An overview. *Fundamenta Informaticae*, 43:2–32.

Darling, A. C., Mau, B., Blattner, F. R., and Perna, N. T. (2004a). Mauve: Multiple alignment of conserved genomic sequence with rearrangements. *Genome Research*, 14(7):1394–1403.

Darling, A. E., Mau, B., Blattner, F. R., and Perna, N. T. (2004b). GRIL: Genome rearrangement and inversion locator. *Bioinformatics*, 20:122–124.

Dechter, R. (2003). *Constraint Processing*. Morgan Kaufmann Publishers, San Francisco, CA, USA, 1. edition.

Dechter, R., Meiri, I., and Pearl, J. (1991). Temporal constraint networks. *Artificial Intelligence*, 49:61–95.

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

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

Dölz, R. (1994). GCG: comparison of sequences. *Meth. Mol. Biol.*, 24:65–82.

Doolittle, W. F. (2000). The nature of the universal ancestor and the evolution of the proteome. *Current Opinion in Structural Biology*, 10:355–358.

Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. (1998). *Biological Sequence Analysis*. Cambridge University Press, Cambridge.

Ermolaeva, M. D., White, O., and Salzberg, S. L. (2001). Prediction of operons in microbial genomes. *Nucleic Acids Research*, 29(5):1216–1221.

Fayyad, U. M., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery: an overview. In Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*, pages 1–34. AAAI Press, Menlo Park, CA.

Frazer, C. M., Eisen, J., Fleischmann, R. D., Ketchum, K. A., and Peterson, S. (2000). Comparative genomics and understanding of microbial biology. *Emerging Infectious Diseases*, 6(5):505–512.

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

Freuder, E. C. and Wallace, R. J. (1992). Partial constraint satisfaction. *Artificial Intelligence*, 58:21–70.

Fukuda, Y., Washio, T., and Tomita, M. (1999). Comparative study of overlapping genes in the genomes of *Mycoplasma genitalium* and *Mycoplasma pneumoniae*. *Nucleic Acids Research*, 27(8):1847–1853.

Golumbic, M. C. and Shamir, R. (1993). Complexity and algorithms for reasoning about time: A graph-theoretic approach. *Journal of the ACM*, 40:1108–1133.

Gusfield, D. (1997). *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, Cambridge.

Hallin, P. F. and Ussery, D. W. (2004). CBS genome atlas database: A dynamic storage of bioinformtic results and sequence data. *Bioinformatics*, Bioinformatics Advance Access(doi:10.1093/bioinformatics/bth423):1–8.

Hand, D., Mannila, H., and Smyth, P. (2001). *Principles of Data Mining*. MIT Press, London.

Harris, J. K., Kelley, S. T., Spielman, G. B., and Pace, N. R. (2003). The genetic core of the universal ancestor. *Genome Research*, 13:407–412.

Heber, S. and Stoye, J. (2001). Algorithms for finding gene clusters. In Gascuel, O. and Moret, B. M. E., editors, *WABI 2001*, volume 2149, pages 252–263, Aarhus, Denmark. Springer Verlag, Berlin.

Henz, M., Fong, L. Y., Chong, L. S., Ping, S. X., Walser, J. P., and Yap, R. H. C. (2000). Solving hierarchical constraints over finite domains. In *Proceedings of the Sixth International Symposium on Artificial Intelligence and Mathematics*, pages 1–8, Fort Lauderdale, Florida, USA.

Höhl, M., Kurtz, S., and Ohlebusch, E. (2002). Efficient multiple genome alignment. *Bioinformatics*, 18(Suppl. 1):S312–S320.

Höppner, F. (2001). Discovery of temporal patterns. In *Proceedings of the 5th European Conference on Priciples and Practices of Knowledge Discovery in Databases*, volume 2168 of *Lecture Notes in Artificial Intelligence*, pages 192–203. Springer Verlag, Berlin.

Horimoto, K., Fuckuchi, S., and Mori, K. (2001). Comprehensive comparison between locations of orthologous genes on archaeal and bacterial genomes. *Bioinformatics*, 17(9):791–802.

Huynen, M., Snel, B., Lathe III, W., and Bork, P. (2000). Predicting protein function by genomic context: Quantitative evaluation and qualitative inferences. *Genome Research*, 10(8):1204–1210.

Jain, R., Rivera, M. C., Moore, J. E., and Lake, J. A. (2002). Horizontal gene transfer in microbial genome evolution. *Theoretical Population Biology*, 61:489–495.

Jáurequi, R., Abreu-Goodger, C., Moreno-Hagelsieb, G., Collado-Vides, J., and Morino, E. (2003). Conservation of DNA curvature signals in regulatory regions of prokaryotic genes. *Nucleic Acids Research*, 31(23):6770–6777.

Jordan, K. I., Makarova, K. S., Spouge, J. L., Wolf, Y. I., and Koonin, E. V. (2001). Lineage-specific gene expansions in bacterial and archaeal genomes. *Genome Research*, 11:555–565.

Jordan, K. I., Rogozin, I. B., Wolf, Y. I., and Koonin, E. V. (2002). Essential genes are more evolutionarily conserved than are nonessential genes in bacteria. *Genome Research*, 12:962–968.

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

Kolstø, A.-B. (1997). Dynamic bacterial genome organization. *Molecular Microbiology*, 24(2):241–248.

Kondrashov, F. A., Rogozin, I. B., Wolf, Y. I., and Koonin, E. V. (2002). Selection in the evolution of gene duplications. *Genome Biology*, 3(2):research0008.1–research0008.9.

Kunin, V. and Ouzounis, C. A. (2003). The balance of driving forces during genome evolution in prokaryotes. *Genome Research*, 13(7):1589–1594.

Kurtz, S. (1999). Reducing the space requirement of suffix trees. *Software-Practice and Experience*, 29(13):1149–1171.

Kurtz, S. (2003). A time and space efficient algorithm for the substring matching problem. Technical report, Zentrum für Bioinformatik, Universität Hamburg.

Kurtz, S., Phillippy, A., Delcher, A. L., Smoot, M., Shumway, M., Antonescu, C., and Salzberg, S. L. (2004). Versatile and open software for comparing large genomes. *Genome Biology*, 5(2):R12.1–R12.9.

Ladkin, P. B. and Reinefeld, A. (1992). Effective solution of qualitative interval constraint problems. *Artificial Intelligence*, 57:105–124.

Lathe III, W. C., Snel, B., and Bork, P. (2000). Gene context conservation of a higher order than operons. *Trends in Biochemical Science*, 25:474–479.

Lawrence, J. G. and Hendrickson, H. (2003). Lateral gene transfer: when will adolescence end? *Molecular Microbiology*, 50:739–749.

Lawrence, J. G. and Roth, J. R. (1996). Selfish operons: Horizontal gene transfer may drive the evolution of gene clusters. *Genetics*, 143:1843–1860.

Lecompte, O., Ripp, R., Puzos-Barbe, V., Duprat, S., Heilig, R., Dietrich, J., Thierry, J.-C., and Poch, O. (2001). Genome evolution at the genus level: Comparison of three complete genomes of hyperthermophilic archaea. *Genome Research*, 11:981–993.

Lillo, F., Basile, S., and Mantegna, R. N. (2002). Comparative genomics study of inverted repeats in bacteria. *Bioinformatics*, 18(7):971–979.

Liu, Y., Harrison, P. M., Kunin, V., and Gerstein, M. (2004). Comprehensive analysis of pseudogenes in prokaryotes: Widespread gene decay and failure of putative horizontally transfer genes. *Genome Biology*, 5(9):R64.1–R64.11.

Mazumder, R., Kolaskar, A., and Seto, D. (2001). GeneOrder: Comparing the oder of genes in small genomes. *Bioinformatics*, 17(2):162–166.

Miller, W. (2001). Comparison of genomic DNA sequences: Solved and unsolved problems. *Bioinformatics*, 17(5):391–397.

Mira, A., Klasson, L., and Andersson, S. G. E. (2003). Microbial genome evolution: Sources of variability. *Current Opinion in Microbiology*, 5(5):506–512.

Mira, A., Ochman, H., and Moran, N. A. (2001). Deletional bias and the evolution of bacterial genomes. *Trends in Genetics*, 17(10):589–596.

Moreno-Hagelsieb, G. and Collado-Vides, J. (2002). A powerful non-homology method for the prediction of operons in prokaryotes. *Bioinformatics*, 18(Suppl. 1):S329–S336.

Morgenstern, B., Dress, A., and Werner, T. (1996). Multiple DNA and protein sequence alignmnet based on segment-to-segment comparison. *Proc. Natl. Acad. Sci.*, 93:12098–12103.

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

Morgenstern, B., Stoye, J., and Dress, A. (1999). Consistent equivalence relations: A set-theoretical framework for multiple sequence alignment. Technical Report Materialien/Preprints 133, Universität Bielefeld, Forschungsschwerpunkt Mathematisierung - Strukturbildungsprozesse.

Mount, D. W. (2001). *Bioinformatics : sequence and genome analysis.* Cold Spring Harbor Laboratory Press, Cold Spring Harbor, N.Y.

Mushegian, A. R. and Koonin, E. V. (1996). A minimal gene set for cellular life derived by comparison of complete bacterial genomes. *Proc. Natl. Acad. Sci.*, 93:10268–10273.

Needlemann, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453.

Nelson, K. E., Clayton, R. A., Gill, S. R., Gwinn, M. L., Dodson, R. J., Haft, D. H., Hickey, E. K., Peterson, J. D., Nelson, W. C., Ketchum, K. A., McDonald, L., Utterback, T. R., Malek, J. A., Linher, K. D., Garrett, M. M., Stewart, A. M., Cotton, M. D., Pratt, M. S., Phillips, C. A., Richardson, D., Heidelberg, J., Sutton, G. G., Fleischmann, R. D., Eisen, J. A., White, O., Salzberg, S. L., Smith, H. O., Venter, J. C., and Fraser, C. M. (1999). Evidence for lateral gene transfer between archaea and bacteria from genome sequence of *Thermotoga maritima*. *Nature*, 399:323–329.

Notredame, C. and Higgins, D. G. (1996). SAGA: sequence alignment by genetic algorithm. *Nucleic Acids Research*, 24(8):1515–1524.

Ochman, H., Lawrence, J. G., and Groisman, E. A. (2000). Lateral gene transfer and the nature of bacterial innovation. *Nature*, 405:299–304.

Ogurtsov, A. Y., Roytberg, M. A., Shabalina, S. A., and Kondrashov, A. S. (2002). OWEN: aligning long collinear regions of genomes. *Bioinformatics*, 18(12):1703–1704.

Overbeek, J., Woese, C. R., and Overbeek, R. (1994). The winds of evolutionary change: Breathing new life into microbiology. *Journal of Bacteriology*, 176(1):1–6.

Overbeek, R., Fonstein, M., D'Souza, M., Maltsev, N., and Pusch, G. D. (1999). The use of gene clusters to infer functional coupling. *Proc. Natl. Acad. Sci.*, 96:2896–2901.

Overbeek, R., Fonstein, M., D'Souza, M., Pusch, G. D., and Maltsev, N. (1998). Use of contiguity on the chromosome to predict functional coupling. *In Silico Biology*, 1.

Pearson, R. (1990). Rapid and sensitive sequence comparison with FASTP and FASTA. *Meth. Enzymol.*, 183:63–98.

Pearson, W. R., Robins, G., and Zhang, T. (1999). Generalized neighbor-joining: More reliable phylogenetic tree construction. *Mol. Biol. Evol.*, 4:406–425.

Pollard, D. A., Bergman, C. M., Stoye, J., Celniker, S. E., and Eisen, M. B. (2004). Benchmarking tools for the alignment of functional noncoding DNA. *BMC Bioinformatics*, 5(6):1–17.

Pritsker, M., Liu, Y.-C., Beer, M. A., and Tavazoie, S. (2004). Whole-genome discovery of transcription factor binding sites by network-level conservation. *Genome Research*, 14:99–108.

Pushker, R., Mira, A., and Rodriguez-Valera, F. (2004). Comparative genomics of gene-family size in closely related bacteria. *Genome Biology*, 5(4):R27.1–R27.15.

Raphael, B., Zhi, D., Tang, H., and Pevzner, P. (2004). A novel method for multiple alignment of sequences with repeated and shuffled elements. *Genome Res.*, 14(11):2336–2346.

Reeck, G. R., Haen, C. d., Teller, D. C., Doolittle, R. F., Fitch, W. M., Dickerson, R., Cahmbon, P., McLachlan, A. D., Margoliash, E., and Jukes, T. H. (1987). 'Homology' in proteins and nucleic acids: a terminology muddle and a way out of it. *Cell*, 50:667–667.

Rocha, E. P. C. (2003). An appraisal of the potential for illegitimate recombination in bacteria genomes and its consequences: From duplications to genome reduction. *Genome Research*, 13:1123–1132.

Rogozin, I. B., Makarova, K. S., Murvai, J., Czabarka, E., Wolf, Y. I., Tatusov, R. L., Szekely, L. A., and Koonin, E. V. (2002a). Connected gene neighborhoods in prokaryotic genomes. *Nucleic Acids Research*, 30(10):2212–2223.

Rogozin, I. B., Makarova, K. S., Natale, D. A., Spiridonov, A. N., Tatusov, R. L., Wolf, Y. I., Yin, J., and Koonin, E. V. (2002b). Congruent evolution of different classes of non-coding DNA in prokaryotic genomes. *Nucleic Acids Research*, 30(19):4264–4271.

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

Russell, S. and Norvig, P. (2003). *Artificial Intelligence - A Modern Approach*. Prentice Hall, Upper Saddle River, New Jersey, 2. edition.

Salgado, H., Gama-Castro, S., Martinez-Antonio, A., Diaz-Peredo, E., Sanchez-Solano, F., Peralta-Gil, M., Garcia-Alonso, D., Jimenez-Jacinto, V., Santos-Zavaleta, A., Bonavides-Martinez, C., and Collado-Vides, J. RegulonDB (version 4.0): transcriptional regulation, operon organization and growth conditions in *escherichia coli* k-12, journal = Nucleic Acids Research, volume = 1, number = 32, pages = D303-D306, year = 2004.

Salgado, H., Moreno-Hagelsieb, G., Smith, T. F., and Collado-Vides, J. (2000). Operons in *Escherichia coli*: Genomic analyses and predictions. *PNAS*, 97(12):6652–6657.

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

Schwartz, S., Zhang, Z., Frazer, K. A., Smit, A., Riemer, C., Bouck, J., Gibbs, R., Hardison, R., and Miller, W. (2000). PipMaker - a web server for aligning two genomic DNA sequences. *Genome Research*, 10:577–586.

Shimkets, L. J. (1998). Structure and size of genomes of the Archaea and Bacteria. In Bruijn, F. J. d., Lupski, J. R., and Weinstock, G. M., editors, *Bacterial Genomes: Physical Structure and Analysis*, pages 5–11. Kluwer Academic Publishers, Boston.

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

Snel, B., Bork, P., and Huynen, M. (2000). Genome evolution: gene fusion versus gene fission. *Trends in Genetics*, 16(1):9–11.

Snel, B., Bork, P., and Huynen, M. A. (2002). Genomes in flux: The evolution of archeal and protobacterial gene content. *Genome Research*, 12:17–25.

Sowa, J. F. (2000). *Knowledge Representation*. Brooks/Cole, Pacific Grove.

Stoesser, G., Baker, W., Broek, A. v. d., Garcia-Pastor, M., Kanz, C., Kulikova, T., Leinonen, R., Lin, Q., Lombard, V., Lopez, R., Mancuso, R., Nardone, F., Stoehr, P., Tuli, M. A., Tzouvara, K., and Vaughan, R. (2003). The EMBL nucleotide sequence database: major new developments. *Nucleic Acids Research*, 31(1):17–22.

Suyama, M. and Bork, P. (2001). Evolution of prokaryotic gene order: Genome rearrangements in closely related species. *Trends in Genetics*, 17(1):10–13.

Tamames, J. (2001). Evolution of gene order conservation in prokaryotes. *Genome Biology*, 2(6):research0020.1–research0020.11.

Tatusov, R. L., Koonin, E. V., and Lipman, D. J. (1997). A genomic perspective on protein families. *Science*, 278(1):631–637.

Thompson, J. D., Plewniak, F., and Poch, O. (1999). A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Research*, 27(13):2682–2690.

Thomson, J. D., Gibson, T. J., Plewniak, F., Jeanmougin, F., and Higgins, D. G. (1997). The CLUSTALX windos interface: Flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Research*, 25(24):4876–4882.

Tillier, E. R. M. and Collins, R. A. (2000). Genome rearrangements by replication-directed translocation. *Nature Genetics*, 26:195–197.

Ussery, D. W. and Hallin, P. F. (2004). Genome update: AT content in sequenced prokaryotic genomes. *Microbiology*, 150:749–752.

Vilain, M. and Kautz, H. (1986). Constraint propagation algorithms for temporal reasoning. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 377–382, Philadelphia, PA, USA.

Vincens, P., Badel-Chagnon, A., André, C., and Hazout, S. (2002). D-ASSIRC: distributed program for finding sequence similarities in genomes. *Bioinformatics*, 18(3):446–451.

Vincens, P., Buffat, L., André, C., Chevrolat, J.-P., Boisvieux, J.-F., and Hazout, S. (1998). A strategy for finding regions of similarity in complete genome sequences. *Bioinformatics*, 14(8):715–725.

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

Waterman, M. S. (1995). *Introduction to Computational Biology*. Chapman and Hall, London.

Watson, J. D. and Crick, F. H. C. (1953). Molecular structure of nucleic acid. a structure for deoxyribose acid. *Nature*, 171:737–738.

Weinstock, G. M. and Lupski, J. R. (1998). Chromosomal rearrangements. In Bruijn, F. J. d., Lupski, J. R., and Weinstock, G. M., editors, *Bacterial Genomes: Physical Structure and Analysis*, pages 112–118. Kluwer Academic Publishers, Boston.

Wetjen, T. H. (2002). Discovery of frequent gene patterns in microbial genomes. Technical Report TZI-Report 27, Technologie Zentrum Informatik, Universität Bremen.

Wetjen, T. H. (2003). A concept to use spatial knowledge of genomic structures to support the alignment of bacterial genomic DNA sequences. In Guesge, H. W., Mitra, D., and Renz, J., editors, *AAAI Spring Symposium Series*, Palo Alto, California, USA. AAAI Press.

Wixon, J. (2001). Reductive evolution in bacteria: *Buchnera* sp., *Rickettsia prowazekii* and *Mycobacterium leprea*. *Comparative and Functional Genomics*, 2:44–48.

Woese, C. (1998). The universal ancestor. *Proc. Natl. Acad. Sci.*, 95:6854–6859.

Wolf, Y. I., Rogozin, I. B., Grishin, N. V., and Koonin, E. V. (2002). Genome trees and the tree of life. *Trends in Genetics*, 18(9):472–479.

Wolf, Y. I., Rogozin, I. B., Kondrashov, A. S., and Koonin, E. V. (2001). Genome alignment, evolution of prokaryotic genome organization, and prediction of gene function using genomic context. *Genome Research*, 11:356–372.

Wong, W. H., Lam, T. W., Lu, N., Ting, H. F., and Yiu, S. M. (2004). An efficient algorithm for optimizing whole genome alignment with noise. *Bioinformatics*, 20(16):2676–2684.

Yanai, I., Derti, A., and DeLisi, C. (2001). Genes linked by fusion events are generally of the same functional category: A systematic analysis of 30 microbial genomes. *PNAS*, 98(14):7940–7945.

Yanai, I., Wolf, Y. I., and Koonin, E. V. (2002). Evolution of gene fusions: Horizontal transfer versus independent events. *Genome Biology*, 3(5):research0024.1–0024.13.

Zheng, Y., Szustakowski, J. D., Fortnow, L., Roberts, R. J., and Kasif, S. (2002). Computational identification of operons in microbial genomes. *Genome Research*, 12:1221–1230.

Zouine, M., Sculo, Q., and Labedan, B. (2002). Correct assignment of homology is crucial when genomics meets molecula evolution. *Comparative and Functional Genomics*, 3:488–493.

Appendix A

A.1 Codon Usages

Codon usages for all species have been taken from `http://www.cbs.dtu.dk/services/GenomeAtlas/` (Hallin and Ussery, 2004).

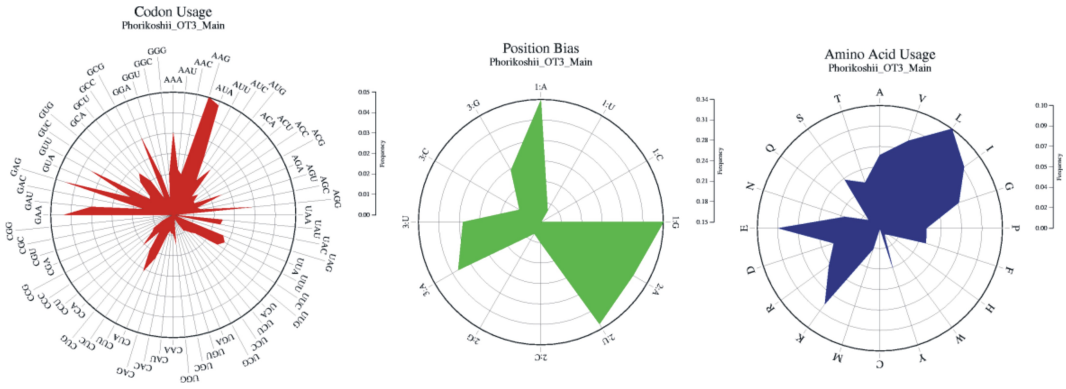


Figure A.1: Frequencies of the codon usage of Pho.

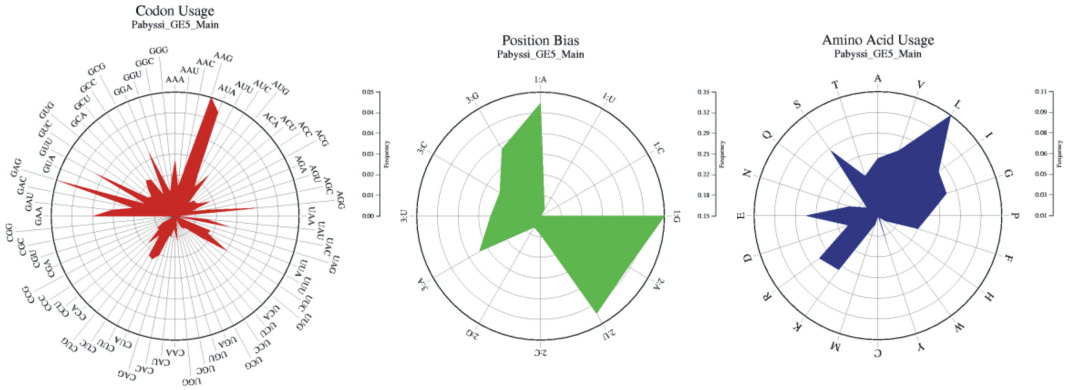


Figure A.2: Frequencies of the codon usage of Pab.

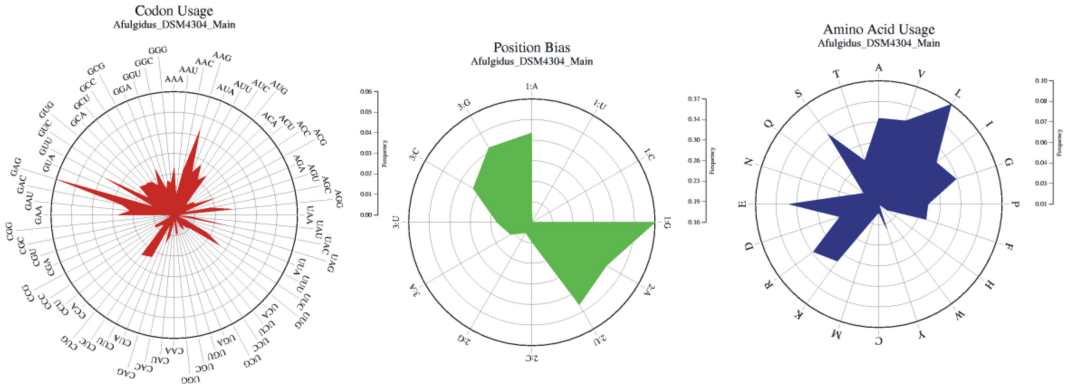


Figure A.3: Frequencies of the codon usage of Afu.

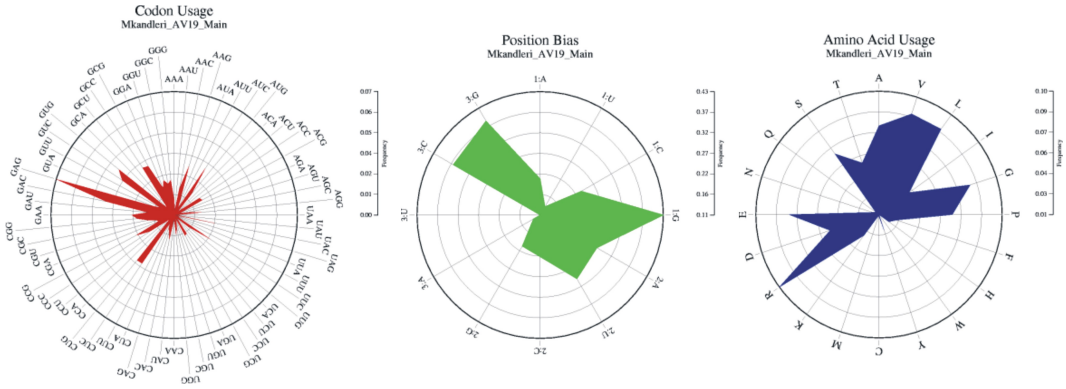


Figure A.4: Frequencies of the codon usage of Mka.

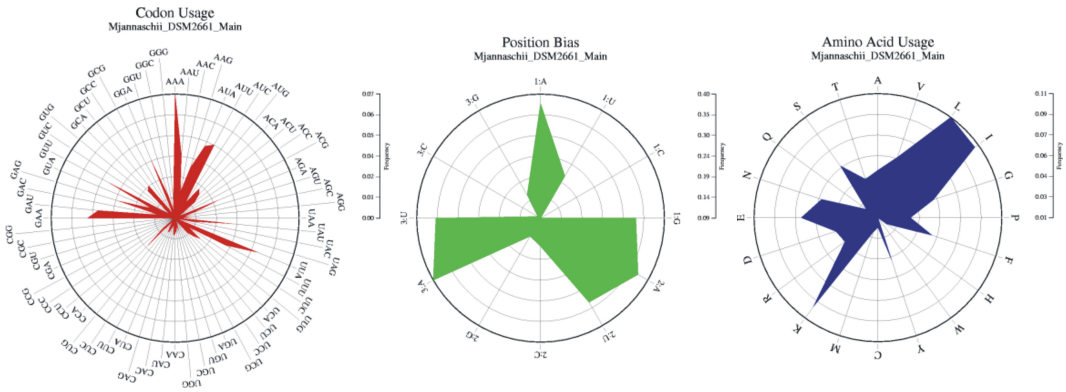


Figure A.5: Frequencies of the codon usage of Mja.

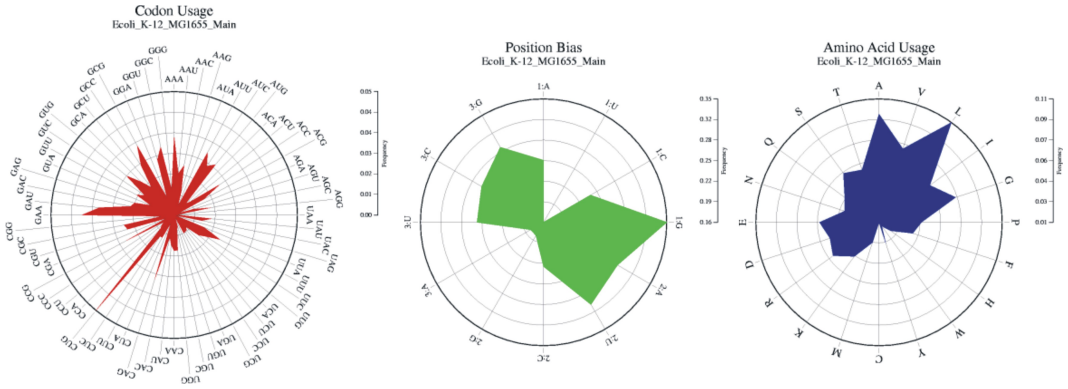


Figure A.6: Frequencies of the codon usage of Eco.

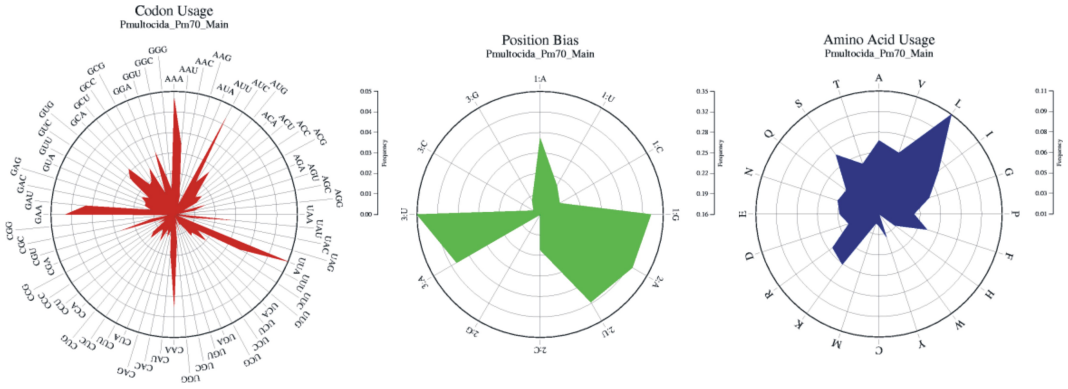


Figure A.7: Frequencies of the codon usage of Pmu.

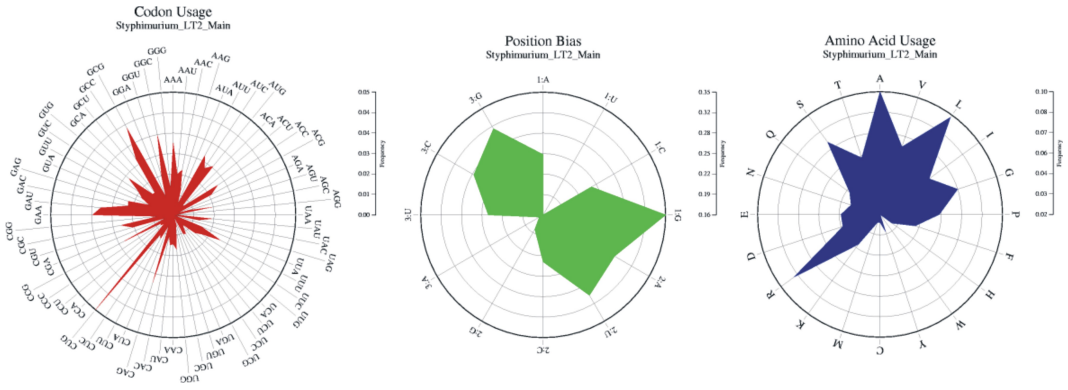


Figure A.8: Frequencies of the codon usage of Sty.

A.2 Number of PGFs at Different Minimum Lengths

Table A.1: Distribution of numbers of PGFs to different minimum lengths *len*.

	a)		**b)**		**c)**		**d)**		**e)**	
len	Pho *vs.* Pab		Pho *vs.* Afu		Mka *vs.* Mja		Eco *vs.* Pmu		Eco *vs.* Sty	
(bp)	$\|G''\|$	$\|G^*\|$	$\|G''\|$	$\|G^*\|$	$\|G''\|$	$\|G^*\|$	$\|G''\|$	$\|G^*\|$	$\|G''\|$	$\|G^*\|$
0	1560	0	535	0	84	0	976	0	3050	0
100	1257	303	34	501	3	81	390	586	2846	204
200	1072	488	15	520	2	82	228	748		
500	602	958	4	531	2	82	68	908		

A.3 Domain Dependent Functions

The domain dependent function $f : F \rightarrow W$, with F being the set of all possible frequencies fr (see Equation 5.1, Section 5.3) and W being the set of all possible weights, maps any spatial relation r to a weight $w \in W$ according to its frequency $fr(r)$. In the following, the functions for all five reference models used for the five genome alignments produced are given. For the sake of clarity, here we present this functions as a mapping of the occurrence $occ(r)$ of a spatial relation r to a weight class. With α, β we denote occurrences and by $[\alpha, \beta] : \alpha, \beta \in \mathbb{N}_0 \wedge 0 \leq \alpha \leq \beta \leq n$ we denote an interval of such occurrences. This allows for a direct comparison of the number of genomes in which a relation r between two genomic features g_i and g_j was found. The corresponding frequency $fr(r)$ of any occurrence can be calculated from the number of occurrences by Equation 5.1 (see Section 5.3).

Archaea For the three reference models Archaea\Pab, Archaea\Afu, and Archaea\Mja the following function has been used:

(A.1) $f : occ \rightarrow w,$

occ	$[0,0]$	$[1,3]$	$[4,6]$	$[7,10]$	$[11,12]$
w	0	1	2	3	4

γ-Proteobacteria For the two reference models γ-Proteobacteria\Pmu and γ-Proteobacteria\Sty the following function has been used:

(A.2) $f : occ \rightarrow w,$

occ	$[0,0]$	$[1,3]$	$[4,6]$	$[7,8]$	$[9,9]$
w	0	1	2	3	4