Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

Detection of Orthologs in Large-Scale Analysis

Diplomarbeit

Leipzig, December,2009

vorgelegt von:
Marcus Lechner
Studiengang: Informatik

Danksagung

Zunächst möchte ich allen danken, die zum Gelingen dieser Arbeit beigetragen haben. Sonja, für die Diskussionen und Denkanstöße. Sven, für die überaus nützlichen Hinweise und das Bewahren der Übersicht sowie Lydia, für's immer Zuhören, Mitdenken und nicht zuletzt das ausdauernde Verbessern meines Ausdrucks. Dank geht auch an die Jungs aus Zimmer 329. Christian, Axel, Sebastian, Jan(e) - es war lustig mit euch.

Unser Administrator Jens hat seinem Berufsbild alle Ehre gemacht und stand bei technischen Fragen immer mit Rat und Tat zur Seite. Selbiges gilt für Petra in Bezug auf Formalien, Büromaterial und selbst Bastelbedarf. Thank you Martin, für die Unterstützung bei der Wahl eines vernünftigen Titels, Lizzie für die sprachlichen Tipps und Dominic, den ich sicher immer mal aus der Arbeit gerissen habe, für die spontane Hilfestellung bei technischen Problemen. Danke Peter und Sven, dass ihr es mir ermöglicht habt an einigen sehr interessanten Konferenzen teilzunehmen.

Lydia, Maria, Friedi - ohne euch wäre das Studium nicht was es war. Danke für die tolle Zeit, das gemeinsame Lernen und eure Freundschaft. Schließlich möchte ich auch meiner Familie für die Unterstützung und Begeisterung danken, die sie mir in all der Zeit entgegengebracht hat. Ein aufrichtiger Dank geht ebenfalls an Familie Rudolph, bei denen ich mittlerweile ein zweites Zuhause gefunden habe. Diana, Du hast mich immer wieder zum Weiterkommen motiviert und mir darüber hinaus vor Augen geführt, was in meinem Leben wirklich wichtig ist.

Schließlich möchte ich Peter und Sonja dafür danken, dass sie es möglich gemacht haben diese Arbeit so kurzfristig zu bewerten. Die Zeit in der Bioinformatik Leipzig hat Spaß gemacht, mich aber gleichzeitig auch sehr voran gebracht.

Zusammenfassung

Orthologe Gene in verschiedenen Spezies stammen von einem einzelnen Gen in ihrem gemeinsamen Vorfahren ab, wobei Sie Ihre ursprüngliche Funktion in der Regel beibehalten. Daher ist ihr Auffinden ein wesentlicher Bestandteil der funktionellen Annotation von Genomen. Informationen über Orthologie tragen zudem zur Identifikation von konservierten oder abweichenden biochemischen Vorgängen zwischen mehreren Spezies bei. Ihre Detektion bietet darüber hinaus nützliche Daten für evolutionäre Analysen, wie zum Beispiel die Rekonstruktion von phylogenetischen Bäumen oder Studien der Genom Reorganisation. Orthologie Analysen erlauben die Identifikation von Proteinen, die nur in bestimmten taxonomischen Gruppen vorkommen und können somit auch die Entwicklung von Wirkstoffen gegen bestimmte Pathogene unterstützen. Das ist besonders hervorzuheben, da mikrobielle Medikamentenresistenzen ein immer größer werdendes Problem darstellen.

Mittlerweile wurden verschiedene Methoden entwickelt um orthologe Proteine genomweit vorherzusagen. Allerdings sind diese meist zu zeitintensiv um in größerem Maßstab eingesetzt werden zu können, oder weisen andere Limitierungen auf. Sie sind nicht in der Lage die Leistung aktueller Computer effizient zu nutzen. Das betrifft insbesondere die ständig wachsende Anzahl verfügbarer CPU-Kerne. In dieser Diplomarbeit wird das Programm `Proteinortho` vorgestellt. Ein Ansatz, der in der Lage ist Orthologie Vorhersagen sowohl im großen, als auch im kleinen Maßstab zu treffen. Darüber hinaus wird das Programm in einer domänenweiten Untersuchung auf eine Vielzahl von Mikroorganismen angewendet. Dafür wurden die Daten sämtlicher vollständig sequenzierter Bakterien des National Center for Biotechnology genutzt. Zusätzlich wurde eine Pipeline erstellt, welche automatisch die taxonomische Klassifikation als auch die Annotation neu sequenzierter bakterieller Genome übernimmt. Sie basiert auf den Informationen zu orthologen Proteinen in verwandten Arten.

Abstract

Orthologous genes in different species have originated from a single gene in their common ancestor. These genes have often retained identical functions. Thus, their detection is an essential part of functional annotation to approach the raising number of newly sequenced genomes. Additionally, the information aids in identification of conserved as well as divergent biochemical pathways between several species. Furthermore, orthology detection provides useful data for evolutionary analysis such as phylogenetic tree reconstruction and genome rearrangement studies. It allows the identification of taxonomically restricted sequences which can facilitate the development of medical treatments against certain pathogens. This is especially important as microbial drug resistance becomes more and more problematic.

Several methods were developed to predict orthologs on a genomic scale. However, these are often to complex for larger applications and suffer from several limitations. They are not capable of using today's computational capacities efficiently. This concerns the growing number of available CPU cores in particular. In this thesis the program `Proteinortho` is presented. An approach which can handle orthology prediction in small- as well as in large-scale analysis. Moreover, the tool is applied on a domain-wide scale to all completely sequenced bacterial genomes which were provided by the National Center for Biotechnology Information. Additionally, a pipeline for non-supervised taxonomic classification and genomic annotation of newly sequenced bacterial genomes is introduced. It is based on information about orthologous proteins within related species.

Contents

Chapter 1

Introduction

The genome-wide identification of orthologs and paralogs is a central problem of comparative genomics. It can yield clues to functional labeling and thus, spare a great amount of wet lab labor. This is especially important as the number of sequenced species increases from day to day [1]. Furthermore, phylogenetic and genomic content analysis as well as studies on protein evolution and target prediction largely depend on orthology [2, 3]. In turn, a lot of databases exist which permit to search by gene name or sequence comparison. However, this results in a limitation to species included in the certain projects. These are mostly restricted to groups of special interest or model organisms. Therefore, the species of interest have to be related to these groups. Otherwise, an assignment can solely cover the amount of genes that occurs in larger taxonomic groups. This should be a small part compared to the capabilities a stand-alone tool can offer. Therefore, several prediction programs were developed to address this problem. They allow orthology analysis within a self defined group of species. However, these approaches are often limited to a comparison of only two species. Whereas programs for identification of orthologous and paralogous proteins within multiple genomes exist, they acquire a huge amount of space, suffer from time expensive calculations or are restricted to closely related species. Additionally, these programs often require supercomputers with an enormous amount of memory even for medium sized approaches. The design of their algorithms is not intended to facilitate multiple CPU cores or even distribute calculations over several machines such as cluster computers. These facts make them not suitable for large-scale analysis in a reasonable period of time.

The aim of this thesis is to develop a tool which allows detection of putative orthologs and paralogs in pairwise species comparison as well as in large-scale analysis with several hundreds of bacterial genomes. It scales well with the increasing

number of available CPU cores in today's computers and offers optimizations for a distributed application on cluster computers. Details of the implementation and the underlying theory will be presented. Furthermore, the tool is applied to an appropriate large set in a domain-wide scale. The objective is to gather a set of proteins common in most of the available fully sequenced bacteria. Subsequently, this set is used to facilitate an approach for fully automated taxonomic classification and annotation of protein coding genes within newly sequenced bacterial genomes of unknown origin. Moreover, this method provides useful information about genomic properties, conserved transcriptional and translational elements which can aid further investigations. Introductory, information about the domain of interest (bacteria), formal definitions of orthology and paralogy as well as existing approaches and orthology databases will be given.

Chapter 2

Background

2.1 Bacteria

Bacteria form a large domain of unicellular organisms. Typically, they are very small and live in nearly every imaginable habitat including bodies of animals and plants [4, 5, 6]. Together with archaea, they form the group of prokaryotes. However, both domains are very different. They have evolved independently from an ancient common ancestor [7]. Prokaryotes differ in many properties from eukaryotes. The most important difference is the missing cell nucleus which is separated by a hull of two membranes in eukaryotes. It contains the genetic material, organized in linear, helix shaped DNA molecules combined with multiple proteins such as histones. These units are called chromosomes [8]. On the contrary, prokaryotes do not form such a barrier. Moreover, they rarely contain organelles divided by membranes in general. Their genetic material consists of circular molecules within the cell. Together with multiple proteins and a small amount of RNA, they form the nucleoid, a term which implies that it is like the nucleus of eukaryotes [9, 10]. Most bacteria have exactly one circular chromosome within this nucleoid. However, linear chromosomes appear in some species, too [11]. In fact, the term chromosome for bacterial DNA is misleading as the configuration and composition differs largely from the eukaryotic eponym. Still, it is commonly used for prokaryotes as well. Additional to the chromosome, bacteria can contain small DNA molecules which are called plasmids. They are frequently used for horizontal gene transfer between different organisms and often contain antibiotic resistance genes [12, 13, 14].

Bacteria are asexual clonal organisms, thus contain identical copies of their parents' genetic material. Evolution is basically driven by mutation and selection. Additionally, species can transfer DNA between cells by horizontal gene transfer.

Three fundamental methods are known. Transformation names the uptake of exogenous DNA from the environment. Transduction instead refers to phage mediated import of foreign DNA. Finally, two organisms can transfer genetic material directly through cell-cell contacts as well. This is called conjugation [9, 14].

Fundamental insights in the function of genes and metabolic pathways can be derived from these microorganisms and applied to other, more complex species [8]. Additionally, many bacteria are pathogens which cause diseases in human, animals and plants. In this respect, the spread of microbial antibiotic resistance becomes more and more critical [15]. Understanding bacteria can aid the biotechnological production of recombinant therapeutic proteins such as insulin, growth factors or antibodies [16, 17, 18].

2.2 Homology of genes

The original definition of homology was stated in 1843 as 'the same organ under every variety of form and function' [19]. At that time, a common ancestry was not mentioned, which is not surprising as the theory of evolutionary biology arose later with Darwin and Mendel. Nowadays, the feature of similarity is still a good indication but neither essential nor sufficient [20]. It is important to mention that homology of two proteins for instance, is neither equivalent with a common function nor sequence nor structure [21]. Actually, they can have diverged in a way making them not appear related even if they arose from same protein in a common ancestor.

Additionally, reverse effects can be expected. Very similar proteins are not necessarily homologs. They may have evolved in a convergent way from non-related ancestors. Their similarity is due to adaption to similar functions. This phenomenon is called homoplasy [22]. Without further investigation, it is hard to tell from sequence data only, whether the proteins are actually homologs or evolved analogously [21]. Furthermore, the rate of homoplasy is expected to be very low regarding homology.

However, official definitions have been established for protein homology and are described and discussed in some detail below.

Official definitions

Homologous genes are derived from a common ancestor. They are divided into two main groups, depending on the way they arose. Orthologous genes have evolved by speciation. They are thought to have the same or at least similar functions due to the common ancestor [21]. These are about the same genes in different species. However, it is clear that they will further diverge while adapting to the environmental conditions and habits of the certain taxa [23]. Paralogs mark the second group. They are further divided into out-paralogs and in-paralogs. Out-paralogs arose from a duplication preceding a speciation whereas in-paralogs evolved more recently by duplication subsequent to speciation [24, 25]. If two paralogous genes have an ortholog in another species, they are called co-orthologs [21]. If two genes have arisen directly (without further duplication) in a set of (co-)orthologous genes, they are furthermore called main-orthologs [26].

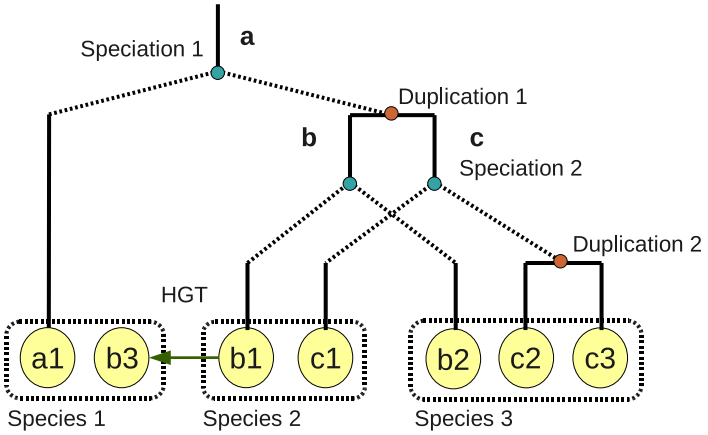


Figure 2.1: Illustration of relationships: The image depicts the situation of three species after events of speciation, gene duplication and horizontal gene transfer (HGT). Genes $c2$ and $c3$ are in-paralogs while $b1$ and $c1$ are out-paralogs with respect to speciation 2. $c1$ is orthologous to $c2$ as to $c3$. Genes $b1$ and $b2$ are called main-orthologous as they arose from the same ancestor (b). $b3$ evolved by horizontal gene transfer and is thus xenologous to $b1$. All genes are co-orthologous to $a1$. The enumeration can be continued. Adapted from [21].

Due to horizontal (interspecies) gene transfer, which is very common in prokaryotes, a special phenomenon occurs [27]. Orthologous genes which arose from such gene transfer, anciently belong to another species and are called xenologous [28]. All

mentioned relationships except this are reflexive. An overview is given in Figure 2.1. As this thesis is focused on protein coding genes, the terms ortholog and paralog will be used in context of proteins for simplicity, while referring to their genes as well.

Problems

Inconsistency One problem addresses the inconsistency of the definition for paralogous groups. Assuming the truth is known, there is no problem. But in practical experience the truth has to be estimated from data. This is especially critical if the data is not complete. To exemplify this, have a look back at Figure 2.1. Genes $b1$ and $c1$ are out-paralogs by definition because there was an event of speciation after duplication. However, if species 3 were absent or unknown, they will be assigned as in-paralogs because the event of speciation cannot be distinguished anymore. For that reason some interpretations use the definition only in subjection to a certain species. They would tell that $b1$ and $c1$ are out-paralogs with respect to species 3 but in-paralogs with respect to species 1 [26]. Therefore, any clustering to those subtypes will be conditioned by the species which are involved, their relationships and the intended point of view. The same problem concerns the concept of main-orthologs.

Information benefit The definition of homology which was introduced above is an evolutionary concept. It raises the question up to which point homology remains interesting for practical experience. Supposing both copies of an occasionally duplicated gene remained in the genome, two evolutionary processes are likely. First is neofunctionalization. As one copy is free of evolutionary pressure, it can adapt to a new function. Second is subfunctionalization. In this case, both genes specialize and adapt to a subset of the previous functions [29]. To go further, multiple copies of a gene should, over a long period of time, result in various (supposably related) functions. Some copies could have become very different, both in function and sequence. As mentioned above, two proteins are homologs per definition whenever they have a common ancestor, irrespective of their actual similarity. However, gene duplications are known to be a major source of innovation in evolution. At this point, the resulting information can become blurry, especially if large sets of paralogs are included. For most purposes, homology is only interesting up to a certain level. Smaller sets

with functional similarity should be favored for most subsequent analysis instead of large anciently related sets which no longer reflect to this day similar functions. Certainly, other scenarios are imaginable where this data is still useful. However, choosing the boundaries of similarity to broad will falsely assign many proteins as paralogs. Introducing more stringent limits on the other hand would miss homologs that have diverged. Again, the limits depend on the intended point of view.

Conclusion

Summing up, there are many problems regarding the actual definition of orthology and homology. The concept was raised in a pre-Darwinan, pre-Mendelian time and adapted to upcoming knowledge. By now, it is inconsistent regarding the paralogous subgroups and main-orthologs. It provides no absolute view but raises fundamental questions to practical situations. Additionally, the use of an exact discrimination, especially of the two groups of in- and out-paralogs, remains questionable if the interest is not any protein's history in particular.

The primary objective is to view these concepts in a consistent and handier way. This concerns both, classification and practical benefit for subsequent analysis. Within the scope of this thesis, a simplified view will be used. The main assumption is that proteins, arisen from a common ancestor and still fulfilling related functions, are conserved regarding their sequence. This property is measurable. Depending on a similarity measure (which will be given through an E-value threshold for `blast`), proteins are treated as orthologs if there is no other candidate within the same species that would match significantly better. In succession, this can become a many-to-many relation. Furthermore, it is symmetric as well as transitive under certain conditions but not reflexive. If protein a is orthologous to b, this holds vice versa as well. If protein c is additionally orthologous to protein b, then it is orthologous with protein a likewise, no matter if the similarity measure between a and c is sufficient. Nevertheless, this property is constrained to proteins within different organisms. Any protein cannot be orthologous to itself or another protein encoded within the same species' genome. Paralogs detection is scaled-down to the case that orthology occurs within two or more proteins in one species. In this case, they are paralogous instead of orthologous to each other while still remaining

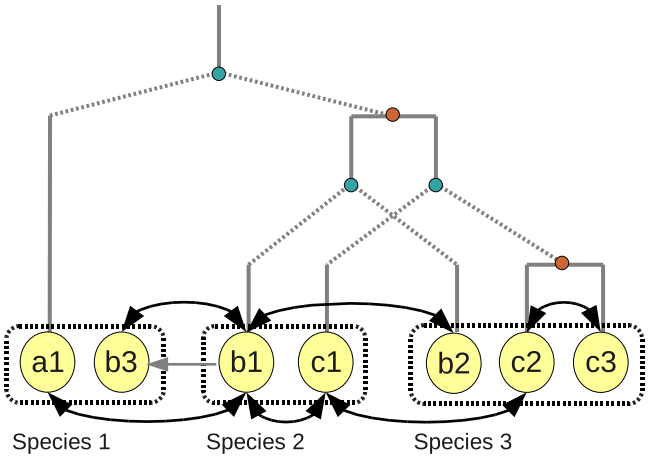


Figure 2.2: Illustration of relationship detection: Based on the situation presented in Figure 2.1, this picture shows how the relationships can be detected. Black arrows represent similarity between genes based on a certain method and threshold. Regarding symmetric and transitive property of the definition, all proteins except $b3$ are orthologous to $a1$. $a1$ and $b3$ are paralogous as $b1$ and $c1$ are. Same holds for $b2$, $c2$ and $c3$. Thus, the seven proteins form an orthoset. A lower threshold could assign additional similarities for example between $b2$ and $c2$. The classification in categories would however stay the same. Closer division into in- and out-paralogs is not intended.

orthologous to proteins from other species. From now, any group of orthologous and/or paralogous proteins will be called orthoset. An example of this relationship is shown in Figure 2.2. It reveals an application of the similarity method.

2.3 Existing approaches

Pairwise methods

Several methods have been developed to detect and distinguish orthologs and paralogs. Two basic approaches are `RBH` (Reciprocal Best `blast` Hit) and the extension `RSD` (Reciprocal Smallest Distance) [30, 3, 31, 32]. The underlying definition of orthology is about the same as described above, except for the fact, that they are only defined for a pair of species. Additionally, `RBH` does not take paralogy into account. It is based on two sets of protein sequences and applies `blast` for both species against

each other. Proteins that were returned as each others best matching partner are assumed to be orthologs. A major drawback is that the highest scoring protein reported by `blast` is often not the nearest phylogenetic neighbor [33]. This can result in deviating hits. If paralogs of certain proteins exist in both genomes, their best hit was often reported to be not equal to the vice versa best hit. To illustrate this, assume proteins a and a' to be paralogs in species 1 and b and b' as paralogs in species 2. The best hit of a is b but a's best hit is b'. In turn, the set will be rejected as no reciprocal best hit is achieved. `RSD` tries to avoid this. It is based on global sequence alignment and maximum likelihood estimation of evolutionary distances to detect orthologs [30]. Again, `blast` is utilized first. However, not the best hit is taken into account solely but all. They are aligned to the queried sequence using `clustalw` [34, 35]. Depending on a certain threshold, the best alignable sequences are evaluated. Using a substitution rate matrix, the evolutionary distances are approximated and the most likely candidate is chosen from this set. If two proteins are each others smallest distance partners, they are assumed to be orthologs.

Another long-serving example is the publicly available tool `InParanoid` [36, 25]. Starting from pairwise best `blast` hits between two sets of proteins, it determines reciprocal hits. Additionally, a third set can be included as outgroup to approximate and remove potential out-paralogs. Out-paralogs are defined as sequences which have a lower pairwise score within the set of both species than against the outgroup. This is reasonable but makes the choice of the outgroup a delicate proposition as it largely influences the amount of putative out-paralogs. Finally, overlapping groups of orthologs are resolved and confidence values are calculated based on a group's internal similarity.

An extended approach is `MSOAR` [26, 37]. The program combines sequence similarity as done in `InParanoid` with heuristics for rearrangement and tandem duplication distances in order to distinguish putative main-orthologs from co-orthologs. However, the approach is more complex as it incorporates the probability and graph flow theory based Markov Cluster algorithm [38]. The algorithm simulates random walks on the graph and determines transition probabilities within a similarity space. These probabilities can be used to split large groups into smaller sets depending on their similarities. In this way, putative false positives can be filtered efficiently. Furthermore, `MSOAR` uses `clustalw` alignments from putative groups to assign gene families

and according gene trees. However, an important requirement of this approach is that both genomes are closely related. Otherwise, the distances will largely mislead the process of assignment.

Multi-species methods

`MultiParanoid` combines several outputs from `InParanoid` to determine groups which cover multiple species [39]. An analog tool, `MultiMSOAR`, exists for `MSOAR` [40, 41]. The basic algorithms have to be applied pairwise to all species of interest. Orthologous groups will be merged subsequently as their relationships are assumed to be transitive.

The approach of `OrthoMCL` is similar to `InParanoid` but allows including multiple species directly [42]. Furthermore, paralogs do not need to be more similar to each other than to a third species to be reported within orthologous groups. In order to resolve the relationships between multiple species, `OrthoMCL` applies the Markov Cluster algorithm globally to all protein similarities within and in between the species as well [38].

Databases

Several orthology databases haven been established in the past. One of the first was the `COG` database (Clusters of Orthologous Groups) [43, 44]. It consists of data from several genomes of prokaryotes and unicellular eukaryotes. Since the last update in 2003, these are 66 species in total. As this date points out, the database is not recently updated. A possible reason is the required manual part of the database creation. In the first step, putative sets of orthologs are detected automatically. This is achieved by taking triangular best `blast` hits into account. Thus, a putative ortholog has to be encoded in at least three species. These triangular hits are merged afterwards with overlapping groups in order to cluster proteins from multiple species. Figure 2.3 shows an example. In the second step, all multi-domain proteins are divided manually into the component domains, in order to split clusters of unrelated groups. Finally, manual annotation is required.

Unfortunately, a program to detect putative orthologs in this way is not available. Instead, the authors offer the `blast`-based tool `COGNITOR` [43, 45]. It assigns

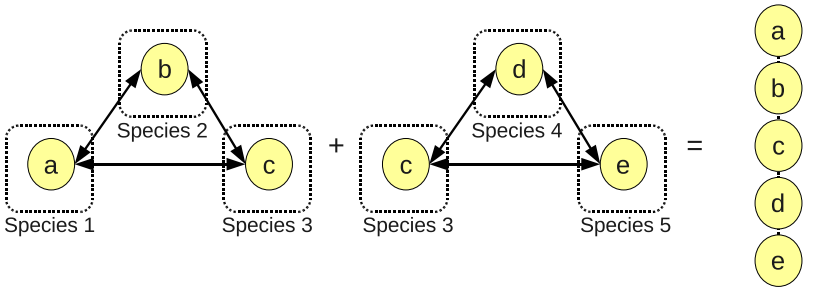


Figure 2.3: Triangular best hit: The proteins a, b and c are encoded in three different genomes. As the arrows point out, they all were returned as each others best hit. Same holds for proteins d, e and c. Only triangular hits like this will be regarded further. Thereby, overlapping triangles are merged which results in clustering of all five proteins.

genomic data to pre-existing groups of the `COG` database. In addition to `COG`, a second database, `KOG` (euKaryotic Orthologous Groups), was established in a similar way [44]. It contains eukaryotic proteins from 7 organisms including *Homo sapiens*.

The database `eggNOG` (evolutionary genealogy of genes: Non-supervised Orthologous Groups) is based on the `COG/KOG` database but avoids the drawback of required manual curation by incorporating a more complex algorithm [46]. To increase accuracy, proteins are initially assigned to the respective sets within `COG/KOG` if this is possible. Afterwards, all against all Smith-Waterman similarities are calculated. The proteins are divided into orthologs and in-paralogs depending on these similarities. Both groups will be treated separately. The clustering is done in about the same way as introduced for the `COG` database. Overlapping triangular best `blast` hits are merged. However, this criterion is relaxed to bidirectional best hits if not all proteins could be assigned. Finally, functional annotation is added automatically based on the majority of protein annotations within the set. In November 2009, data for 373 eukaryotic as well as prokaryotic species were available.

`Ensemble Compara` uses a very different approach [47]. The method makes use of gene and species trees which are inferred and reconciled. This makes the database especially reasonable for evolutionary interests but further increases complexity. 51 eukaryotic species were available in November 2009 (release 56). Again, a tool for applying this method to separate genomes was not published. Putative orthologs within a genome of interest have to be detected by `blast`, based on orthologous

groups annotated within the databases. A tool which is not limited to a single orthology database but can access many sources of this kind is `BLASTO` (Blast on Orthologous groups) [1].

Additionally, most authors offer a publicly available database derived from their tools, nevertheless with a limited number of included species. For example, the `OrthoMCL DB`, one of the biggest compilations, contained 128 genomes in November 2009 [48]. However, more than a thousand fully sequenced genomes were available at this time [49].

Chapter 3

Proteinortho

3.1 Background

As introduced, orthologous and paralogous proteins can be approximated by analysis of their sequence similarity. If two proteins from different species are similar, they are considered to be related. In order to gather such related proteins from two sets, a local alignment search can be accomplished. The program `blast` (Basic Local Alignment Search Tool), a keystone of bioinformatics, is adequate even for large-scale approaches [50, 51]. It locates subsequences within a database superior to a given E-value threshold. The E-value is a measure for the alignment score with respect to its likelihood regarding the given database and query sequence. Thus, the most similar sequences can be found.

In order to increase the accuracy of this approach, a reciprocal search is performed. All proteins of species 1 are matched against those of species 2 and vice versa. If two of them are closely related, they should both return their counterpart as best hit. Whenever protein a has protein b as best hit and reverse, it can be supposed that they are related. If one direction is missing, there might be an ambiguous relation which nevertheless can not be clarified without further investigation. Therefore, only reciprocal `blast` hits will be considered in this work as used within the presented `RBH` method. However, this is not limited to best hits alone.

Extending this approach to multiple species raises two basic problems. One is the runtime which increases quadratically with the number of species and proteins respectively. The other is the enormous amount of data which needs to be examined in order to derive groups of orthologs. The first problem can be addressed by massive parallelization. The second one by converting the data efficiently to a graph and detecting its connected components. An example is shown in Figure 3.1.

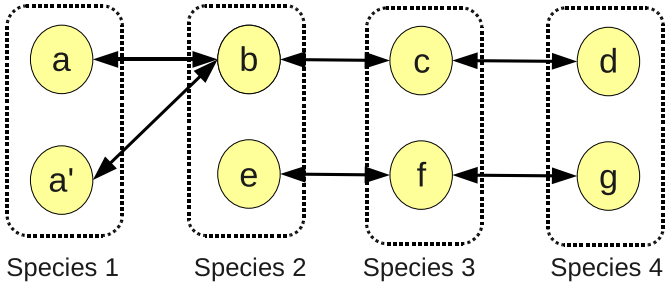


Figure 3.1: Example for a graph representation: Proteins shown as circles connected to their reciprocal best `blast` hit. Finding connected components in such graphs allows to reconstruct groups of related proteins. The strategy is based on the assumption that unrelated groups would not have best `blast` hits within each other. Here proteins a, a', b, c, d and e, f, g form two groups of orthologous proteins. a and a' are located within the same species and are therefore paralogs.

A major part of this thesis is the reimplementation and improvement of a program called `Proteinortho`. It was initially written by Sonja J. Prohaska. This tool is designed to detect orthologous proteins in a given set of species by using reciprocal best `blast` hits and a graph representation as introduced above. The basic assumption is that homologous proteins have similar sequences as well, a perspective which is followed up in this thesis. However, this is not equivalent with the determination of functional homologs which would require experimental validation for thousands of proteins. Since this is not possible for every species, the method is a compromise that yields reasonable candidates. `Proteinortho` worked sufficiently with small datasets but was not capable to handle larger amounts as it leaked scalability and exhausted memory resources too quickly. Furthermore, only perfect orthosets were reported in the normal output to spare further investigation of blurry data. As soon as a single paralog was determined, the data had to be processed from log-files separately. This behavior is reasonable for phylogenetic approaches but turns out to be inconvenient in practical experience with differing aims. `Proteinortho` is based on sequence similarity, hence all similar proteins should be treated equally, no matter to which species they belong. This includes paralogs, where a reasonable discrimination cannot be achieved. From this point of view, they are as important for subsequent analysis as orthologs even though they represent ambiguous data. While the basic procedure was kept, major changes were made in the accomplish-

ment of the single steps. This mainly concerns the application of `blast`, filtering of hits and determination of orthologous groups. These steps were reimplemented and optimized in terms of speed, usage of threads and memory efficiency. Furthermore, a more sophisticated choice of `blast` hits was established as tests revealed opportunities for improvement. Details are presented within the next sections.

3.2 Methods

3.2.1 Workflow

`Proteinortho` requires a `fasta` file for each of at least two species containing the sequences of all encoded proteins or genes. Therefore, either amino acid or nucleotide representation is possible. Initially, for each of those files a database for `blast` is created using `formatdb`. A reciprocal `blast` search (either using `blastp` if amino acid or `blastn` if nucleotide representation of proteins was chosen) is performed for all `fasta` files against each other. This is the most time-consuming part of the program. Afterwards, the results are filtered. Only hits with more than 25% sequence identity and at least 50% overlap will be regarded further. Both limits were chosen to increase the chance of actually detecting a functional conservation in contrast to fusion genes, random similarity or far evolved homologs. Above the level of 25% sequence identity, functional conservation can be supposed [52, 53]. Additionally, using the default parameters a reciprocal hit within the adaptive best hits is required. Their formation is described later among the features in Subsection 3.2.2. Only pairs with `blast` matches in both directions will remain. Meaning, for each protein a with a `blast` match to protein b, an additional match from b to a is required. Otherwise, the match will not be considered. This requirement of reciprocal hits can substantially improve the results compared to allowing even unidirectional hits [54]. Hence, it is recommended. However, `Proteinortho` allows to turn this requirement off if unwanted.

The protein and species identifiers are converted to unique numbers in order to make them addressable in a memory efficient way. Using these new identifiers, a mapping from species to proteins and an edge list is created to gather a graph representation of the proteins' orthologies. Orthologous and paralogous proteins are

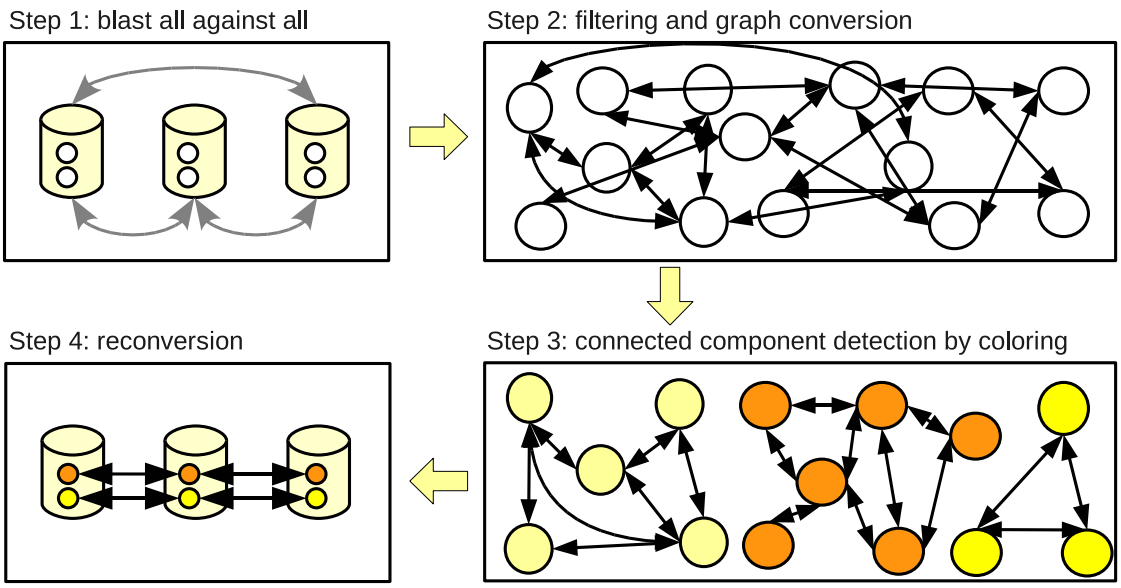


Figure 3.2: `Proteinortho` workflow: Step 1: Reciprocal `blast` runs (gray arrows) are done with all species (light gray 'databases') against all other. Step 2: The retrieved information about `blast` hits (black arrows) between all proteins (circles) was filtered, compressed and transformed to a graph representation. Step 3: The graph's connected components were detected via coloring. Nodes are the same as in step 3 but the components are now clearly recognizable. Step 4: Finally, it is reconverted and remapped to the species with encoded proteins.

supposed to be within a connected component as they share common features and thus have similar sequences. These components are tagged by graph coloring. The detection of connected components is shown in Algorithm 1.

After the connected components were tagged, they are remapped to the original proteins and species. The result is a list of orthologous and paralogous proteins between all investigated species. An illustration of this workflow is presented in Figure 3.2.

3.2.2 Features

Adaptive hit-inclusion

Instead of using only the best `blast` hit or the best n `blast` hits for each protein in other species, `Proteinortho` applies an adaptive approach. Regarding the bit-

Algorithm 1 `Proteinortho`'s graph decomposition by coloring

```
1: for all uncolored nodes a do
2:    choose a new color ψ
3:    push a to stack S
4:    for all nodes b on stack S do
5:       pop b
6:       color b with ψ
7:       for all uncolored neighbors c of b do
8:          push c on stack S
9:       end for
10:   end for
11: end for
12: for all colors ψ do
13:   for all nodes a and b of color ψ do
14:      if species(a) = species(b) then
15:         mark ψ as 'contains paralogs'
16:      end if
17:   end for
18: end for
```

scores of the best hit (*best*) and any another considerable hit, the scoring function $s(candidate) = \frac{best+candidate}{best} - 1$ returns a relative similarity value between 0 (not similar at all) and 1 (very similar). In this manner, closely related paralogs can be detected and included without the requirement of a fixed quantity of considered best candidates which would introduce new problems as Figure 3.3 shows.

Multi-core usage

While `blastall` offers a switch for multi-threaded usage, the present version (2.2.17) is not able to comprise all CPUs efficiently when applied with example protein sets during tests. Although eight CPUs were available on the computer, it used not even half of them, fluctuating between only 25% and 40%. Due to this, the internal threading was disabled and deferred to `Proteinortho` itself. The script partitions the tasks and coordinates several `blast` jobs simultaneously to achieve full efficiency. The same optimization method is used for creation of `fasta` databases, file-parsing and assignment of `blast` hits.

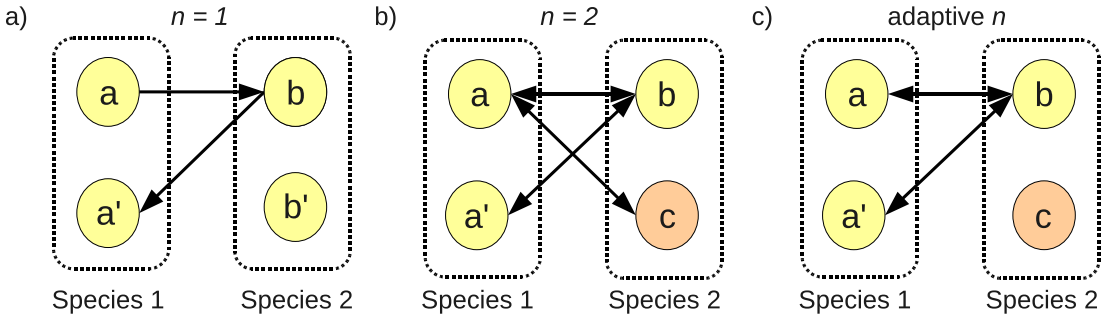


Figure 3.3: Adaptive hit-inclusion: Exemplification of problems with the n reciprocal best `blast`-hit criterion using a fixed n. Proteins are shown in circles, `blast`-hits as arrows. a) $n = 1$: a hits b correctly but b's best hit points to the paralog a' of a. Because a reciprocal (best) hit is required, orthologous proteins a and b were not assigned. b) $n = 2$: By raising n, proteins a and b are included. Furthermore, a' was detected as paralog of a. On the other hand, a protein c not belonging to the orthologous group may falsely be added to it as two hits are enforced. c) Adaptive n: The number of hits is not fixed but corresponds to hits which are very similar to the best scoring one. Default is at least 0.95 regarding the scoring function presented before. The orthoset is identified correctly.

Optional distributed computing

The most time-intensive part of `Proteinortho` concerns the `blast` jobs. As already mentioned in the introduction, their number increases quadratically with the number of species and their duration with the number of proteins. Therefore, a complete run with a larger number of species could take more than half a day. Thus, the tool's application is critical for a bigger amount of species and proteins respectively as Figure 3.7 shows. In contrast, the multi-core feature mentioned above reduces runtime only linearly.

To overcome this problem, `Proteinortho` offers the possibility to distribute the `blast` calculations over multiple machines. This requires a shared data storage for example a network-file-system (NFS). Different instances of the script are synchronized on file level. New instances can be added and aborted on the fly, making an application on multiple machines very flexible to changing workloads. Details are shown in Figure 3.4.

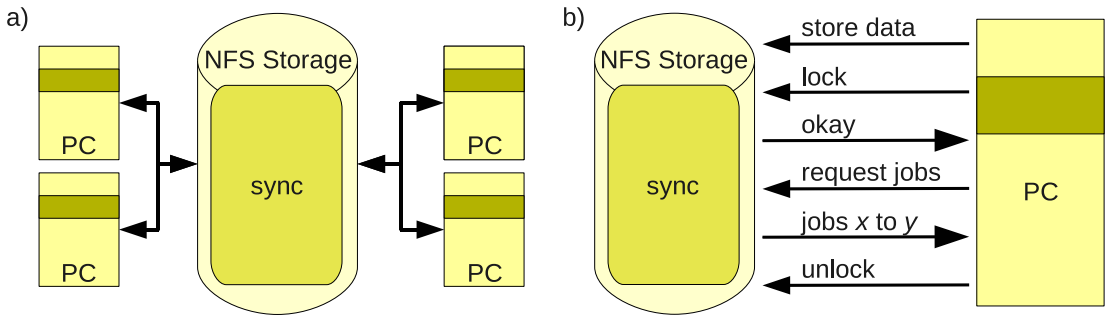


Figure 3.4: Distributed computing: a) Multiple computers running `Proteinortho`. They cooperate dynamically using an N-way technique. All `blast` jobs are distributed over the machines. The results are stored on a shared storage (e.g. an NFS Storage). There is no master. The coordination of jobs is done via a synchronization file (sync) instead. b) Communication of a single computer to the storage. Whenever the computer has done a job, it stores the results. The synchronization file is locked and new jobs are chosen according to it. This information is stored for the next computers requesting new jobs. The file gets unlocked afterwards. All possible jobs are numbered for identification and systematic processing. If a computer has multiple threads finishing at the same time, it acquires multiple jobs at once to reduce the overhead for locking.

Memory efficiency

Besides the runtime, the new version of `Proteinortho` optimizes memory-usage as well. The complete graph-representation and decomposition was reimplemented in `C` as this turned out to be considerably more memory-efficient in comparison to the original implementation in `Perl`. Instead of storing the full protein-ids and species-names, they were mapped to integers and remapped afterwards. This allowed to use simple arrays instead of hashes for addressing proteins. The graph itself is represented as an edge-list. Other ortholog detection programs use matrices instead. This in turn results in a space requirement of n^2, no matter how closely connected the graph is. The edge-list can use memory more efficient as only existing connections require space. Figure 3.5 points this out.

The mapping of protein-ids and species-names to integers could be done in `Perl` using hash-maps. However, this turned out to be very inefficient if huge amounts of data were handled. A benchmark comparing the behavior of `C++` maps and `Perl`'s hash-maps is shown in Figure 3.6. A `C++` map requires about half of the memory of a

a)

	a	b	c	...
a	1	1	0	
b	1	0	1	
c	0	1	0	
⋮				⋱

b)

1 -> 2
2 -> 1,3
3 -> 2

Figure 3.5: Edge-list vs. matrix: a) Most programs for orthology analysis use similarity matrices for graph representation. The required memory is thus always in $O(n^2)$, no matter how closely connected the graph is. b) `Proteinortho` instead uses an edge-list. This allows to use memory more efficiently. Furthermore, it maps the ids to integers and does not use fully qualified names. While the worst case scenario requires n^2 in memory for n proteins as well, real world data will omit most possible edges.

`Perl` hash-map for $500,000$ pairs. This significantly increases by raising the number of pairs which will be about 1.5 million in the later studied case of 'Domain-wide commons' in Chapter 4. Thus, the use of a `C++` map instead of its `Perl` counterpart is reasonable. The implementation was done using `SWIG` [55]. `SWIG` is a wrapper which makes `C` and `C++` functions available within other programming- and scripting-languages such as `Perl`.

Tree builder

This additional tool creates common protein trees based on `Proteinortho`'s output. It clusters species by shared proteins (by means of having orthologs or paralogs) using an adapted `UPGMA` algorithm. A min-operation is used instead of average for scoring of clusters. In this way, relations of similar characteristics, habitats, phylogeny and horizontal gene transfer are reflected. It allows viewing species relationship from the perspective of shared proteins and might give an insight into evolutionary regards.

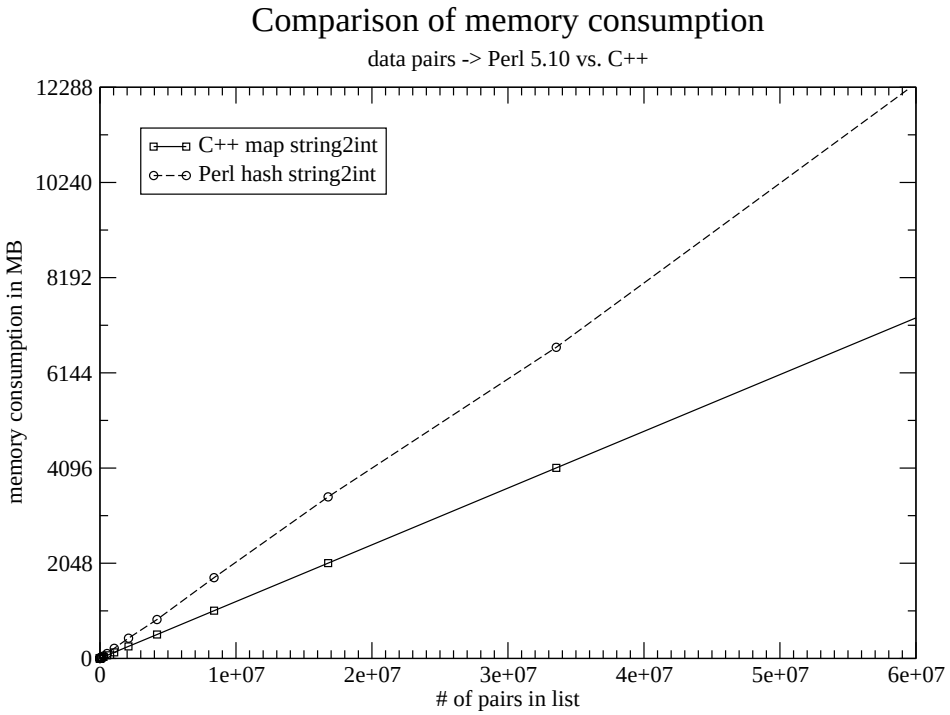


Figure 3.6: Pair storing benchmark: As integer a counter starting with zero was used, the strings was unique with 30 chars each. `C++` maps clearly use memory more efficient. Starting from 50,000 pairs this data structure is to prefer from `Perl`'s correspondent hash. However in dimensions under 25,000 pairs the overhead of `SWIG` overbids the gain of efficiency.

3.3 Results

3.3.1 Program

`Proteinortho` is the first publicly available tool for large-scale orthology analysis without the need of supercomputers and enormous amounts of memory. Later parts of this thesis will reveal that today's standard computers have sufficient memory to handle analysis of over 700 bacterial genomes, an amount of data that was not analyzed all at once before. However, the complexity for `blast` is still the biggest weakness. To advance this, the tool provides efficient threading and dynamic distributed computation. Thereby, the time factor can be reduced by utilizing multiple

machines. `Proteinortho` additionally bypasses problems of most reciprocal best `blast` hit methods as it is not fixed to a certain reciprocal best hit but can utilize virtually similar hits as well.

Behavior

The tool does not distinguish between putative paralogs and orthologs. It rather clusters similar proteins, no matter to which species they belong. This decision differs largely from common approaches like `InParanoid` or `OrthoMCL` which try to unambiguously distinguish in-paralogs and out-paralogs in order to detect main-orthologs. However, the concepts can be regarded as inconsistent. This was explained in Subsection 2.2. Additionally, the conclusion was drawn that an unambiguously distinction cannot be achieved in a reliable way solely based on sequence data. The concept of main-orthologs is not traced at all. For this reason only unambiguous putative orthologs (orthosets with not more than one protein per species) should be used for phylogenetic analysis and estimation of relative evolutionary rates. On the contrary, this behavior enhances questions of target prediction. If paralogs are allowed, the reported orthosets represent all similar sequences within the regarded species which could interact in the same way.

Computational effort

To evaluate the improvement regarding the computational effort, an example species (*Corynebacterium glutamicum ATCC*) with 2026 proteins was chosen, renamed and copied to multiple files in order to simulate up to 16 species with a complete orthologous protein pool. Concluding from this runtime evaluation, no significant improvement was achieved in comparison to the original version. Figure 3.7 illustrates this when utilizing one CPU only. In this case, only the graph decomposition works faster but still remains a negligible small part compared to the time for `blast`. The effect scales slightly as the number of species grows. A clear improvement can be observed when threads are used. The speedup again increases with the number of species and proteins respectively. Markedly, the new version is even faster using four threads than the old version using eight. A slope is noticeable which becomes smother the more threads are used. This is especially interesting as the process

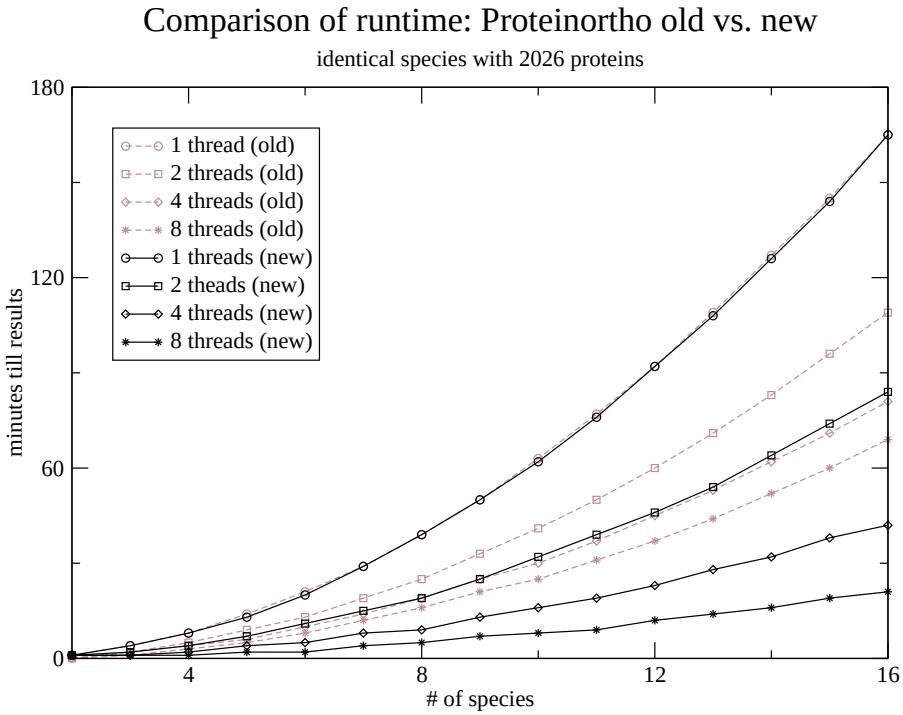


Figure 3.7: Proteinortho benchmark: Comparison of old version and the improved version 3.0 (new) with an E-value of $1e^{-10}$ utilizing different numbers of CPUs (Intel® Xeon® CPU at 2.33GHz). Competitive I/Os with other processes may lead to fluctuations. However the trend can clearly be determined. The reimplemented version largely benefits from threads.

can be distributed over an arbitrarily number of machines and thus, many CPUs. However, it is presumably that the slope catches up when applying the program to a corresponding quantity of species.

$(\sum_{i=1}^{n} i) - n = \frac{n(n+1)}{2} - n = \frac{n(n-1)}{2}$

runs have to be done, in order to `blast` each of the n analyzed species against all others. The computational effort for each `blast` lies in $O(l_D l_Q)$ whereas l_D is the length of the database and l_Q the query's length [56]. Since `blast` will be used for all species against each other, one can assume both variables as well as their product to be equal on average ($l_D = l_Q = l$), what results in $\frac{n(n-1)}{2} l^2$. As l^2 will be equal using the average for approximation, it can be dismissed as constant. The

conclusion is a runtime complexity of $O(n^2)$ adding up to the previously mentioned result and thus staying overall in $O(n^2)$. However, the constant l^2 grows quadratic. Therefore, the length of sequences has a great influence in the over all runtime and should be considered as well. Doubling the sequence lengths will take about the same extra time as doubling the number of species. Pre- and post-processing - namely `formatdb`, building up the graph representation and decomposition in its connected components - come along with a resulting complexity of $O(n) + O(n) + O(n^2)$. However the most time intensive part is to perform the required `blast` runs.

Memory requirements

Moreover, significant improvements were made regarding the memory consumption. The original version needed about 160 GB RAM for a test set of 710 of completely sequences bacteria. As there was no appropriate machine available, the calculations could not be done. Using the improved version, these calculations could be done on a machine with 2.5 GB of available memory. Therefore, memory should not limit the application to larger sets anymore.

3.3.2 Benchmarks

Accomplishment

Basis for the evaluation of `Proteinortho` is the supervised `COG` database and the fully automatic approach `OrthoMCL` which applies reciprocal best `blast` hits as well but uses the Markov Cluster algorithm for clustering. As the available data in `COG` is limited to certain organisms, 16 of them were chosen. The species are listed in Table B.2. Six Gram-positive bacilli, six gamma and four alpha proteobacteria were selected. The underlying protein sets were obtained directly from `COG`. `OrthoMCL` and `Proteinortho` were applied to this sets with an E-value of $1e^{-10}$, a thread number of eight and default parameters. However, `Proteinortho` was switched to output orthosets with paralogs to match the behavior of `COG` and `OrthoMCL`. Moreover, the E-values are not directly comparable. While both tools use `blast`, their way of application is rather different. `OrthoMCL` collects all protein data in one `fasta` file and uses it as query and database at the same time. Therefore, every sequence will

be found at least once. Proteinortho on the other hand, splits the blast jobs each species against each other. Therefore, a query sequence may not be included within the database since both are considerably smaller. As the E-value depends on query and database, the pairwise E-values will differ between both tools.

Performance

Performance data is shown in Table 3.1. Proteinortho took 16% less CPU time when compared to OrthoMCL as it does not apply a self blast (species x against species x) and clusters the proteins differently. Furthermore, it clearly outperforms OrthoMCL in thread efficiency and completes the task in less than a third of the time. Finally, memory efficiency behaves significantly better as well.

Program	Wall clock time	CPU time	Kernel time	Max. memory
OrthoMCL	117m46s	292m32s	3m30s	273 MB
Proteinortho	34m24s	245m58s	1m16s	108 MB

Table 3.1: Performance benchmark: Applied to the same dataset and eight available CPUs, Proteinortho clearly outperforms OrthoMCL by means of memory and multi-core efficiency.

Outcome

The outcome shows that Proteinortho's predictions can be settled between those of OrthoMCL and the COG database. As Figure 3.8 points out, Proteinortho reports more groups that cover a large amount of species than OrthoMCL but less than COG. With decreasing coverage OrthoMCL reports more sets than all others while the biased COG falls back significantly. The database was designed to gather groups which cover preferably large sets of species while the focus of manual curation was not to find protein occurring in small subsets. On the other hand, OrthoMCL applies a clustering strategy which splits large sets into smaller separate groups. Both strategies are reflected well in the outcome.

Figure 3.9 yields a similar picture. Proteinortho lies between OrthoMCL and the COG database regarding the number of proteins which were included. However, it reports some sets which are very large in comparison and even exceed the COG

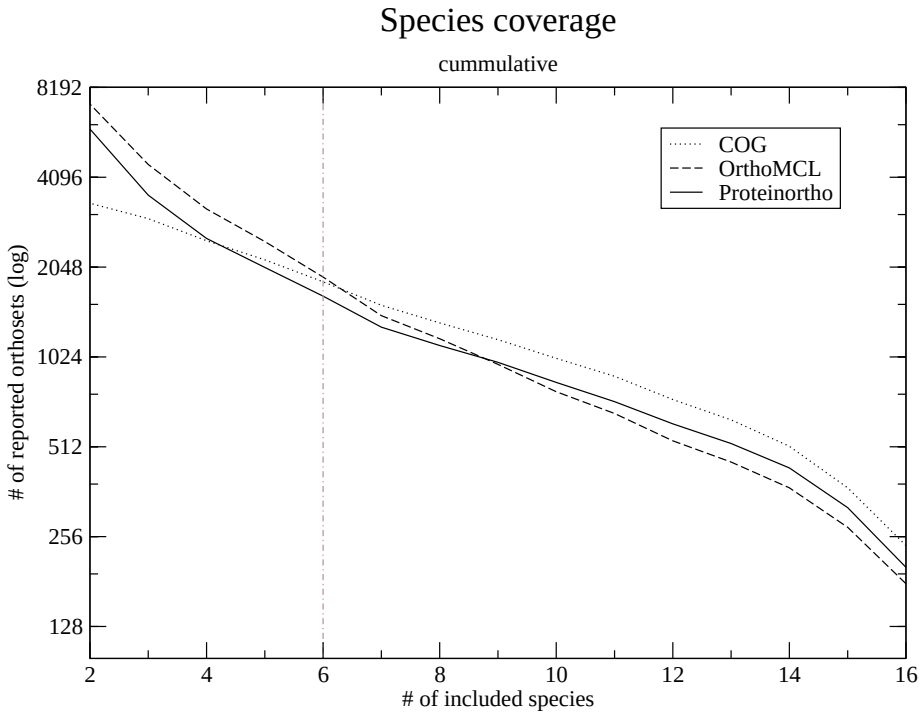


Figure 3.8: Species coverage: The diagram illustrates the number of orthosets which cover at least n species. `OrthoMCL` returns many small sets but the fewest sets which cover all species. The two classes with six species can be recognized by the increased slope at this position for `Proteinortho` and `OrthoMCL` whereas `COG` does not show any significant changes here.

predictions. These are groups which should probably be split. On the other hand, `OrthoMCL` yields less large sets. It obviously tends to report smaller groups. The `COG` database relies on triangular best hits. Thus, it requires a protein to be present in at least three species. Despite the fact that the chosen species were extracted from a larger set, it is certain that most solely pairwise occurring proteins will not be present in a third species within the whole set as well. This explains the small gain of groups from three to two species regarding the pairwise working approaches. Noteworthy, the curve of `Proteinortho` is more similar to scale-free data than the two other approaches. Assuming the data is unbiased, this indicates a higher consistency with the random distribution of groups where small groups occur more

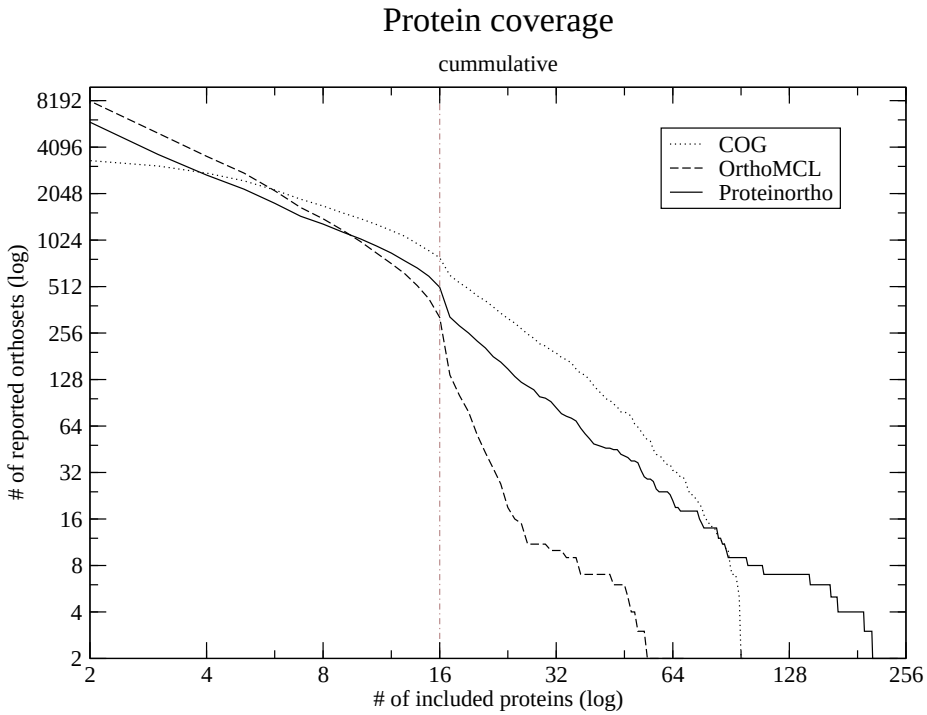


Figure 3.9: Protein coverage: The diagram illustrates the number of orthosets which cover at least n proteins. Again, `OrthoMCL` reports the most small groups but less large groups than the competitors. On the other hand, `Proteinortho` reports some sets even larger than `COG` does. These are probably candidates which should be split into smaller groups. An increasing slope is recognizable at position 16 which matches the number of species.

often than large groups. The leap in all three curves at the number of 16 species matches the number of overall species included. It can be ascribed to the fact that there are less orthosets which contain paralogs than which do not.

The number of orthosets is illustrated in Figure 3.10. `OrthoMCL` returns the most of them. This is due to the large amount of small groups as Figure 3.9 reveals. On the other hand, `COG` defines the smallest number of sets as it focuses on large groups. `Proteinortho` again, is in between. The approach prefers large groups as well but does not miss small groups in turn. Furthermore, `Proteinortho` includes the least number of proteins in comparison to `OrthoMCL` and `COG`. A reason is the avoided

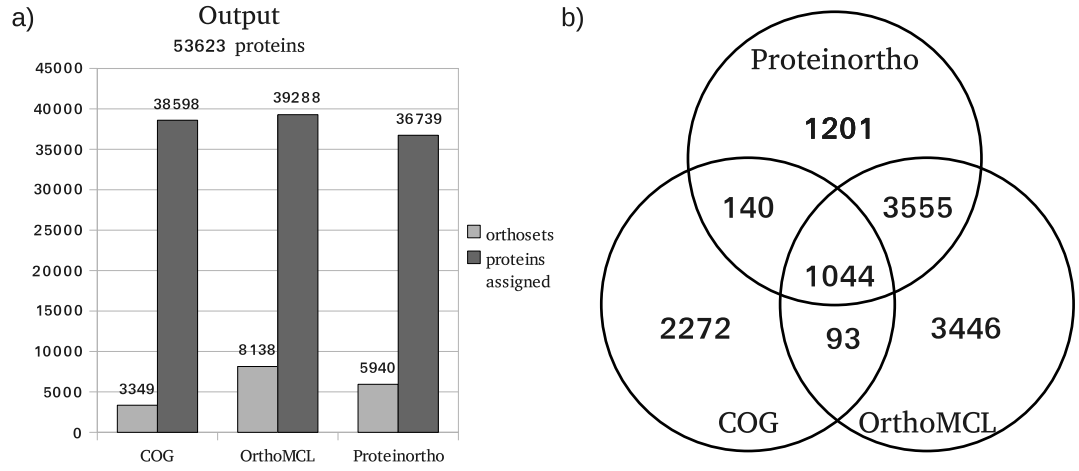


Figure 3.10: Methods comparison: a) Direct comparison of the sets. `OrthoMCL` reports more than twice as many groups as `COG`. `Proteinortho` reports more orthosets than `COG` but fewer than `OrthoMCL`. b) Illustration of equal sets. The relationship of `Proteinortho` and `OrthoMCL` is evidently shown by their huge overlap.

self `blast` for each organism. Paralogous proteins which are not similar enough to proteins in other species are not included within the prediction. Thus, `Proteinortho` focuses on well conserved sets while `OrthoMCL` and `COG` try to include every possibly related protein. Figure 3.10 illustrates the number of overlapping predictions. As `Proteinortho` and `OrthoMCL` are based on the same approach (reciprocal best `blast` hit), it is not surprising that they return many equal groups. 4599 equal orthosets are reported by both programs. `Proteinortho` reports additional 1341 sets which are not included or differ from the `OrthoMCL` output. These are primarily larger sets while `OrthoMCL` reports multiple small sets in addition (3539). About one third of the predicted sets is consistent with the `COG` database for both tools.

Evaluation

In order to evaluate `Proteinortho` and `OrthoMCL`, their predicted orthosets regarding the chosen 16 species were compared to the `COG` database. The sets are partitioned in four groups as shown in Figure 3.11. They can either be identical to the `COG` sets, be subsets, supersets or overlap partially with them. An overview of the results is shown in Figure 3.12.

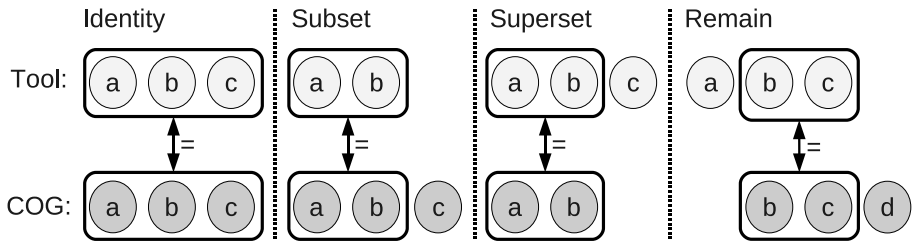


Figure 3.11: Orthoset relationships: Four mutually exclusive groups can be derived when comparing the output of both tools to the COG database. If a predicted orthoset is not equal to a set in the database, it can be a contained in a group (subset) or contain a group of the database itself (superset). The remaining sets overlap only partially with groups in the database or are not contained at all.

When expressed in percents, Proteinortho behaves best. It reports more equal sets as well as subsets and supersets of COG annotations. Furthermore, the amount of not assignable groups is considerably smaller. However, if the absolute values are measured, Proteinortho's advance drops. Still, it reports a few more equal sets but some fewer sub- and supersets. On the other hand, OrthoMCL reports about 25% more orthosets than Proteinortho which are mostly small groups. Thus, the chance of generating multiple subsets from the COG data raises as well. Assume a set of proteins a, b, c, d and e within the COG database. Proteinortho may report a group a, b, c, d. One subset group is counted. OrthoMCL on the contrary, tends to report smaller sets. Therefore, it may yield for instance a, b and c, d as two separate groups which would both count as subset. However, the average of size of subset groups did not vary much between both tools. For Proteinortho it was 6.57 while OrthoMCL's average size was 6.49 proteins. There is no indication for an overbalance of small groups within the group of subsets reported by OrthoMCL. Summing up, the differences between both tools regarding COG are not outstanding. They rather emphasize the beforehand determined trends. OrthoMCL tends to report a huge amount of small groups whereas large groups are avoided. Proteinortho on the other hand, matches most of the large groups reported by COG but is less prone to small sets in comparison to OrthoMCL.

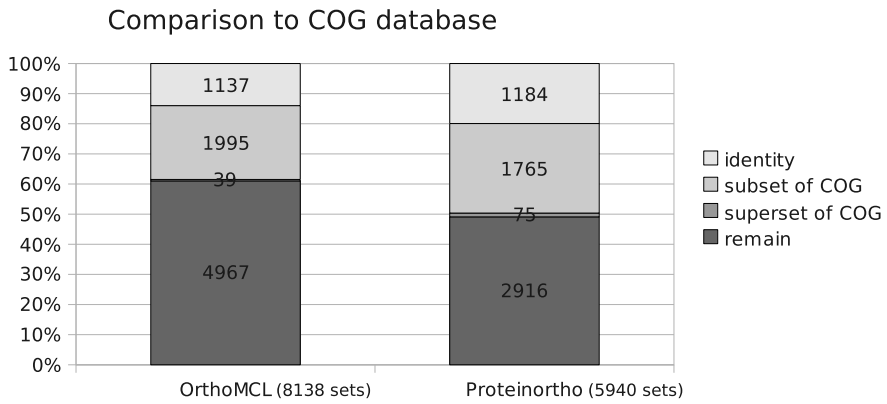


Figure 3.12: Comparison to COG database: The putative orthosets of both tools were compared to the annotations within the `COG` database. `Proteinortho` reports slightly more groups which are equal to this annotation while `OrthoMCL` reports more subset groups. The number of superset groups is negligible as the sets are rarely larger than `COG` sets. However, the overall amount of remaining groups is considerably bigger for `OrthoMCL`.

Speed comparison

To compare the runtime of `OrthoMCL` and `Proteinortho`, the same method was used as in Subsection 3.3.1 (see Figure 3.7). Both programs were applied to a set of up to 16 equal species (only the protein identifier were renamed) with eight available CPU cores. The absolute time till results (real time) as well as the the combined time on all CPUs (CPU time) was measured. Figure 3.13 reveals that `Proteinortho` requires significantly less CPU time. While the overall time for `blast` should be about the same, `Proteinortho` does not apply a self `blast`. In this way, minus n runs are needed for n species compared to `OrthoMCL`'s strategy. Details will be discussed in Subsection 3.4.1. Moreover, the clustering algorithm is faster.

An outstanding observation is the overall duration. `Proteinortho` clearly performs better if multiple CPU-cores are available. In the example, it finishes the annotation within one hour while `OrthoMCL` took more than five hours. Furthermore, this effect scales with the number of species and CPU cores. The benchmark depicts that `Proteinortho` was optimized for these situations and thus, is more appropriate for large-scale analysis.

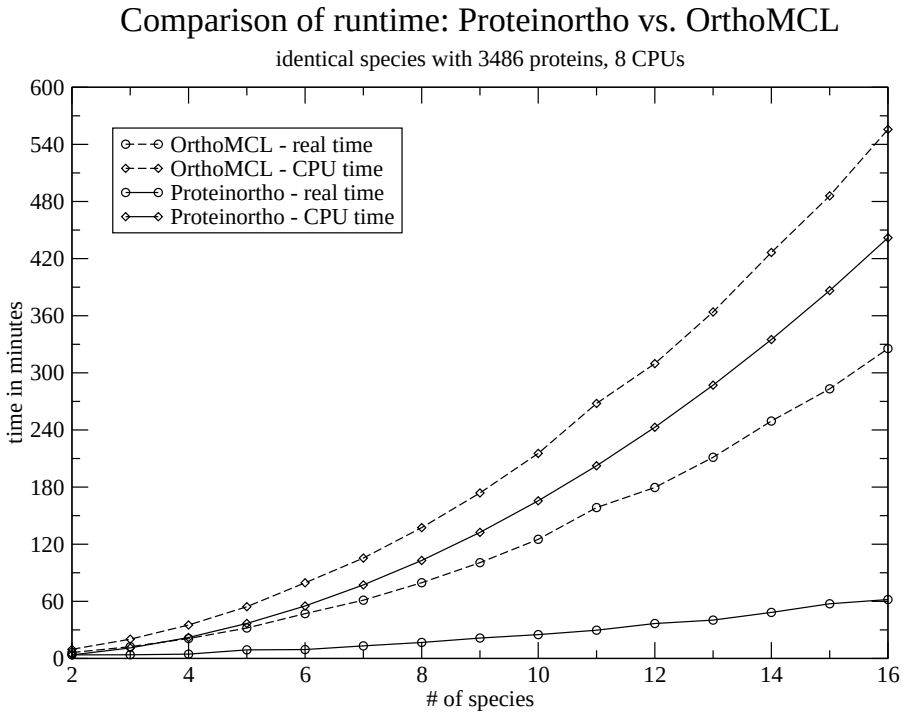


Figure 3.13: Comparison of runtime: `OrthoMCL` and `Proteinortho` were applied to a set of up to 16 identical species (*Escherichia coli K12*) with an E-value of $1e^{-10}$ and eight available CPUs (Intel® Xeon® CPU at 2.33GHz).

3.4 Discussion

3.4.1 Self `blast`

In contrast to the previously presented `blast` based methods, `Proteinortho` does not apply an additional `blast` against the own protein database of every regarded species by default. However, this should be performed to be aware of paralogs and avoid the necessity of larger species sets for detection of these properties. Otherwise, `Proteinortho` is not supposed to report the same data for any subset of species in the regarded group. The set of species is important for assignment as depicted in Figure 3.14. Removing one species from the set can lead to different annotation regarding paralogs in its subsets. A representative example is shown in Figure 3.15.

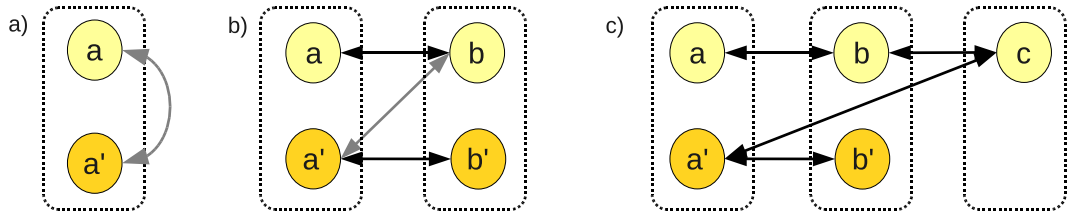


Figure 3.14: Finding paralogs: Different sets of proteins a to c. Primed letters are paralogs of the non-primed versions. Black arrows show `blast` hits, gray arrows not detected relationships. a) A paralogous protein will not be found directly if a search within one species is not applied. b) If only the reciprocal best `blast` hit is taken into account, paralogs will not be detected between only two species. c) At least three species are needed which have each a protein within a connected component to detect paralogs.

Actually, the presented effects are largely reduced by applying the adaptive hit inclusion introduced in Subsection 3.2.2. By using this technique, `Proteinortho` can detect paralogs even for sets of only two species if they are similar enough. This makes the n further `blast` runs for a set of n analyzed species broadly unnecessary and saves runtime. Hence, this is way of application recommended. Still, this is no guarantee for stable results regarding the mentioned effects. Therefore, a self `blast` can be applied optionally if stability and paralogs detection is important in particular.

3.4.2 Large putative orthosets

Using `Proteinortho`, all proteins of species 1 are applied in a `blast` run against the set of proteins derived from species 2 for all regarded n species. Only the best group of reciprocal `blast` hits is taken into account. Proteins are assumed to be orthologs if they are - representing the reciprocal matches as a graph - within a connected component. That implies there has to be a path from protein $a1$ to protein $a2$, either direct or indirect over other proteins.

Misassigned orthosets of increasing size were observed during the tests if the number of species was raised. Especially, if `Proteinortho` is applied to huge sets, they grow enormously. This observation is reported in literature as 'mega clusters'

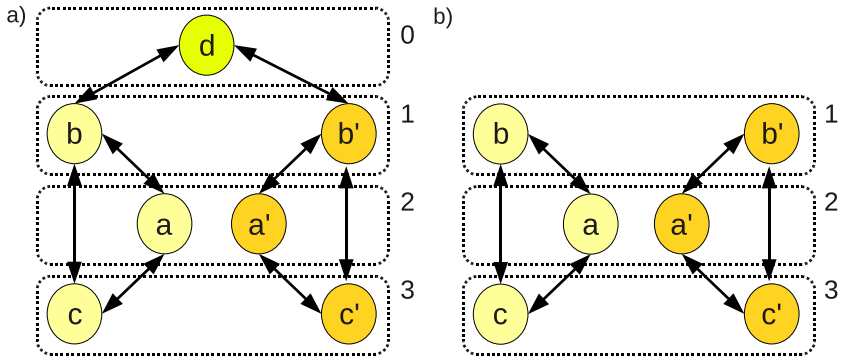


Figure 3.15: Paralogs are not detected: Black arrows represent `blast` hits between species 0 to 3. Primed letters are paralogs again. This illustration emphasize the issue shown in Figure 3.14b to c. a) Two groups of paralogs (light and dark gray) are grouped together through a protein in species 0. b) This does not occur when species 0 is not included to the set. Therefore, different groups are reported for the subset. Assuming `blast` finds paralogs by direct application, `Proteinortho` is supposed to return more stable results when applying self `blast` runs.

for reciprocal best `blast` hit approaches as well [57, 58, 42, 43]. Within the biggest test, the domain wide commons, nearly half of the proteins were reported to be part of a single putative orthoset. Due to size and heterogeneity, this result is not useful and determines a drawback of the presented method. It conveys the impression that putative orthosets were joined together and reported as a big cluster.

It is reliable to imply that two or more proteins form one orthoset if all are connected to each other. However, this is questionable for a protein that is linked to only one protein of such a (nearly) complete connected group. Particularly, this may be the case if large groups are assumed. One might suppose more links to other elements of the same origin. A single one or a few connections regarding the rest could indicate a false positive assignment or imply that the protein is very different and thus may have another but possibly related function. In both cases, it would be preferable to dismiss this protein, since the aim is to find candidates that are supposed to have the same function. Especially, this holds for chains of proteins where only the ends are actually connected to bigger components. These cases become more critical if two big connected groups are joined through such chains. Misassignments are the consequence.

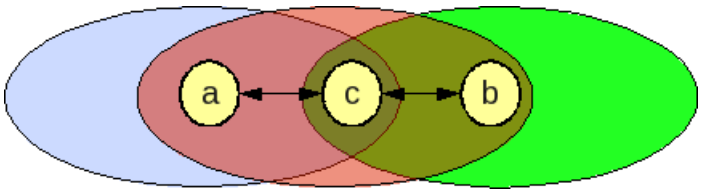


Figure 3.16: Bridging effect: Two proteins a and b are not connected by reciprocal best `blast` hits (black arrows), because they are not within each others similarity range (gray ellipses). However, a third protein c has a and b in its similarity range and vice versa. In this way, a and b get connected despite the lack of a reciprocal best `blast` hit between them.

A main reason for the anomaly of large putative orthosets due to joined groups are increased distances between indirectly connected proteins. The purpose of the initial E-value was to define a similarity threshold up to which the proteins are supposed to be isofunctional. If a protein a was not found by a `blast` through another protein b using this threshold, they are supposed to be not similar and therefore not related. However, both can be connected (and as a result be declared as related) by another protein c located 'between' them. Protein a as well as protein b are within this similarity threshold of c (and vice versa to fulfill the reciprocal `blast` hit requirement). An illustration can be found in Figure 3.16.

As mentioned above, the effect of bridging is intended as it allows to find homologs more flexible than any fixed threshold. In this way, homologs between distant species can be detected. The rejection of indirectly connected proteins would restrict orthosets to fully connected components. Doubtless, they would return reliable sets. On the other hand, a large amount of homologs would be missed or partitioned into smaller groups. This again would countermine a reasonable large-scale application. Nevertheless, the effect introduces a new problem especially for those large-scale applications. To exemplify the problem, we assume a set of protein sequences P_m of fixed length m and for simplification an additive metric $d(a, b)$ with $a, b \in P_m$. Regarding a threshold t, both protein sequences a and b are similar if $d(a, b) \leq t$. However, they can be connected by a third sequence $c \in P$ if $d(a, c) \leq t$ and $d(c, b) \leq t$ as well. In this case, protein a is similar to c which is similar to b. Therefore, a and b are connected regarding this metric, even if $d(a, b) > t$. In fact $d(a, b) \leq 2t$ is sufficient. Supposing multiple sequences $c_1 ... c_{n-1}$ with $n \in \mathbb{N}$ where

$d(c_{a-1}, c_a) \leq t$ and $d(c_a, c_{a+1}) \leq t$ but all other $d(c_a, c_b) > t$ with $a, b \in \mathbb{N}$, it turns out that $d(c_0, c_n) \leq nt$ is sufficient to connect both proteins due to chained bridging.

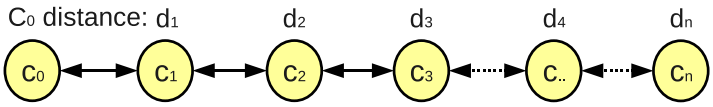


Figure 3.17: Chained bridging: If bridging occurs over $n + 1$ ($n \in \mathbb{N}$) proteins ($c_0..c_n$), the maximum possible distances between the chains start c_0 and its end c_n increases: $d_a < d_{a+1} | a \in [1..n]$

Transferred to the representation of reciprocal best `blast` hits as graph, indirect connected proteins can actually have a much larger distance than the given threshold. It depends on the number of proteins on the shortest path between both. The more proteins are taken into account, the higher a possible dissimilarity of elements in orthosets returned by `Proteinortho` can be. Essentially, the E-value is not additive since it takes the database context into account [50]. The exemplified effect of raising distances between the first and last element of a chain however, is still present as Figure 3.17 illustrates.

An additional reason for large groups are fusion genes. These are hybrids formed from two previously separate genes which can connect two unrelated connected components as they gain high similarity to both groups. To avoid this, `blast` hits have to cover at least 50% of the protein coding region for both, query and database sequence. Otherwise, the hit will be rejected. Figure 3.18 illustrates an example. Due to this requirement, a fusion gene might still be clustered to a group of non-fusion genes. However, this should be one group solely. It is unlikely (but not impossible) that hits from two groups cover more than 50% of a combined sequence at once. Unfortunately, this feature has been added subsequently to `Proteinortho`. In order to globally incorporate the information of protein lengths for all threads, massive restructuring of the source code and data structures would be necessary. Otherwise, a global allocation of the additional data would increase the required amount of memory considerably. For that reason, a lower limit is used instead. It is given by the end of a certain `blast` hit which might not reflect the actual sequence length. However, this feature allowed to split parts of the large group

and recover about 6% more distinct connected components in the 'Domain-wide commons' set. An exact consideration of sequence lengths will be added in later versions of `Proteinortho`.

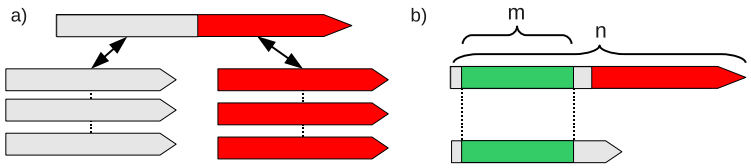


Figure 3.18: Fusion gene: a) Two groups of orthologous genes (gray and black) are connected by a fusion gene containing parts of both groups. This effect can result in large non-related clusters. b) To avoid this, the matching region reported by `blast` (darker gray area) of length m has to be at least half as long as the according protein's length n. Otherwise, for $2m < n$ the hit is rejected as shown in the illustration.

Shadow E-value

To address the issue of large connected components due to increasing distances, a second similarity threshold, the Shadow E-value, could be introduced. It has to be less restrictive than the actual E-value and represents a fixed lower boundary for protein similarity. Figure 3.19 shows an illustration. The maximum distance between every protein of a putative orthoset will be restricted to this value. Thus, the similarity within any reported orthoset will be within the range of the lower bound. This counteracts the effect of excessive protein accumulation in a single set. Thus, `Proteinortho`'s prediction should become more reliable and accurate while still remaining flexible with respect to more distant relatives.

Unfortunately, this solution would lead to another problem. Once an unreliable assignment was detected, there are several possibilities to handle it. An easy way would be to dismiss the whole set. This in turn may lead to massive loss of data. It would not be more than a filter which is automatically dismissing questionable sets. The intended improvement could not take place. An approach for partitioning large sets is desirable.

Intuitively, one could immediately remove any protein which is outside the shadow range (more distant than the Shadow E-value allows) of any so far colored

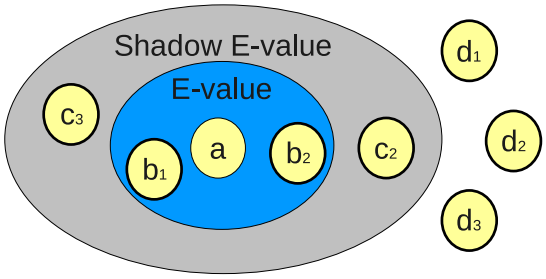


Figure 3.19: Shadow E-value: Starting from protein a two distance boundaries, the E-value and the Shadow E-value are shown. Proteins b_x are inside the given E-value distance (dark area) and will be found via `blast`. In contrast, the c_x-proteins can only be found using `blast` with a less restrictive (Shadow) E-value, indicated by the gray ellipse. They can be added by bridging. In contrast no protein of the d_x group, which is outside the lower bound, will be grouped with a as putative orthoset.

node regarding the introduced decomposition Algorithm 1. However, this would not be stable and results in different outcomes as it strongly depends on the order of coloring. In order to obtain stability, it is possible to wait until coloring of the complete graph is done, apply the deletion rule afterwards and color again. This must be repeated until all proteins within a putative orthoset are within each others shadow. In turn, the method might remove many proteins which actually belong to the group solely because another protein was misassigned, sweeping the 'witness(es)' with it.

Algorithm 2 Graph partitioning depending on a second threshold

```
1:  for all connected components C do
2:    while C has nodes outside threshold shadow do
3:      for all nodes a in C do
4:        n_c ← # of nodes within C and threshold shadow
5:      end for
6:      for all nodes a in C do
7:        if n_a = min n_m|m ∈ ℕ then
8:          remove a
9:        end if
10:       recalculate C
11:     end for
12:   end while
13: end for
```

A promising approach based on the Shadow E-value is presented in Algorithm 2. It searches for nodes within a connected component that have the fewest nodes within their shadow range. They can not be assigned confidently to it and are therefore removed. This concept is proposed to remove bridging nodes which connect conserved groups. The certain groups should fall into connected components instead of accumulating within a 'mega cluster'. Figure 3.20 illustrates the concept on a small example. At the moment, this method is not implemented. However, it might be applied to `Proteinortho` in later versions.

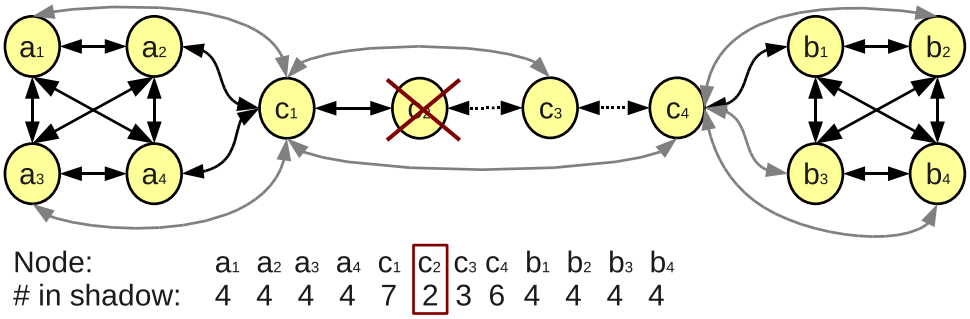


Figure 3.20: Shadow E-value utilization: This example illustrates the problem of connected groups (a_x and b_x) via chained bridging (c_x nodes). Black arrows are `blast` hits below the E-value threshold, gray below the (less restrictive) Shadow E-value threshold only. Counting of nodes within the shadow range for each node allows to determine probably misleading ones and disconnects the connected components into its conserved parts following Algorithm 2.

Besides considerably improving accuracy, the presented method has drawbacks, too. Especially, the memory requirements will increase notably both in RAM and hard disk. Runtime is effected as well since the coloring step for several components has to be done over and over again. However, regarding the time consumption of the `blast` runs themselves, this should take less than 10% of runtime.

3.4.3 Functional conservation

`Proteinortho` detects groups of proteins with high sequence similarity. It is widely accepted to approximate functional conservation from this data [36, 59, 1, 26, 25]. However, it can be argued that sequence identity is a very important factor apart

from the `blast` hit under a certain E-value [60]. Literature points out that 40% sequence identity should be used as a confident threshold for enzymes [61]. However, the detection of functional conservation is not limited to enzymes. The accuracy should be improved by taking functional parts into account instead of bare similarity. Homologous proteins within distant species, sharing similar functions are not necessarily similar by sequence as well. The contrary is the case as Figure 3.21 points out. The likelihood to find a similar sequence drops, the further speciation has progressed. Even if its function remains the same.

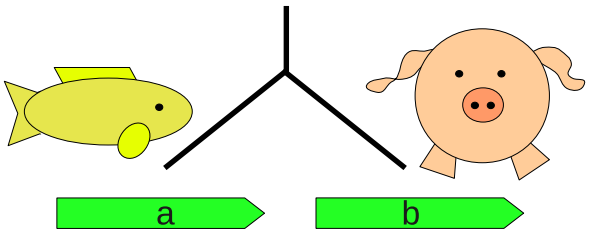


Figure 3.21: Similarity dilemma: Proteins a and b are derived from the same ancestor and are found in two different species. The likelihood that both functions are similar is apparently bigger, than the likelihood that both have a similar sequence. This is due to the fact that protein sequences within a species use to share common features which have adapted to its environment and biochemicals [62, 63, 64]. The preferred usage of certain codons is an example.

An interesting approach is to take, beside `blast` hits, functional domains into account as well. A possible source could be the `SUPERFAMILY` project which was already used for a reliable assignment of homology in certain species [65]. `SUPERFAMILY` provides a library of hidden Markov models based on the `SCOP` database (Structural Classification of Proteins) [66, 67]. Thus, proteins on superfamily level were used solely. Their structure and, in a lot of cases, functional features suggest the existence of a common evolutionary origin [68] . These can be utilized for detecting functional domains within putative orthologs. In addition, `PFAM` (Protein Families) offers hidden Markov models for detection of functional domains that are based on multiple sequence alignments without the restriction to structural or functional conserved proteins [69, 70, 71]. Thus, the amount of data is larger. However, due to the way of creation it is presumably not as reliable as the data from `SUPERFAMILY` [72].

Additional to `blast`, domains could be annotated for each protein within a puta-

tive orthoset using both hidden Markov models. The same domains at each protein can be expected if similar function is supposed. Therefore, this information could be incorporated to refine `Proteinortho`'s output as presented in Figure 3.22. This is nevertheless not a trivial approach and will increase runtime notably. Furthermore, it would limit the homologs detection largely to confident isofunctional groups and therefore to orthologs. As introduced in Section 2.2, paralogs are not supposed to be isofunctional. Thus, an optional application is suggested. This method would be convenient as quality evaluation for putative orthosets as well.

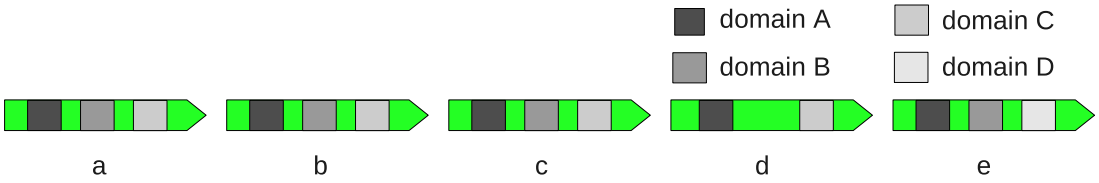


Figure 3.22: Domain based approach: Assume the arbitrary proteins a, b, c, d and e to be detected as putative orthologs and are thus similar in sequence. Additionally, the equality of a, b and c indicates a functional conservation. Nevertheless, domain annotation points out that protein d misses a domain present in the majority and protein e differs in one. However, a functional similarity is not precluded by the fact that they differ in a domain. For instance, they might bind different sites on DNA but cause the same effect.

3.4.4 Methods for evaluation

Unfortunately, there is no universal applicable method to evaluate predicted orthologs and paralogs. The required similarity between two proteins in terms of sequence identity, regulation of chemical activities, interaction partners often varies across studies [57]. In some wet lab studies two genes are considered to be orthologs if they have the ability to complement each others functions [73, 74]. On the other hand, genome rearrangement studies refer to orthology from an evolutionary point of view as the original sequence within its genomic context regarding duplication events [2]. Methods for computational evaluation can be based for example on gene ontology, enzyme classification numbers, correlation in expression profiles, functional genomics data, gene neighborhood conservation or phylogenetic

information [75, 76, 77, 78]. However, the quality of methods often differs largely according to the way of evaluation [57, 58]. A general assessment is hard to draw as methods behave differently, regarding the type and aggregation of data. However, the accomplished comparative benchmark in Subsection 3.3.2 can be considered as orientation.

Chapter 4

Domain-wide commons

4.1 Background

It is widely accepted that all bacteria arose from a common ancestor [79, 80, 7]. Following this assumption, most of them should have a common set of proteins (and RNA) to cover basic functions of life. These essential proteins are more evolutionary conserved than non-essential proteins [31]. `Proteinortho` should thus be a reasonable approach to detect them. A set of essential proteins outlines targets that should not be aimed for drug design as this would influence beneficial bacteria as well. However, it represents potential targets for chemical disinfection. Furthermore, this information can be used to give a more detailed insight into bacterial evolution and can aid in taxonomic classification based on multiple reference proteins. The objective of this chapter is to identify such domain-wide commons for bacteria. To improve confidence about this set, it should be isofunctional and contain no or at least as few as possible paralogs.

4.2 Methods

4.2.1 Data source

To find proteins occurring in most bacterial species, all 710 available genomes were downloaded from `NCBI` (National Center for Biotechnology Information) at 2008/11/27 [49]. The amino acid sequences of the proteins on all chromosomes and plasmids were collected in `fasta` files, one for each species. Entries containing the description 'hypothetical' or 'predicted' were dismissed in order to use verified data exclusively. Finally, about 1.5 million proteins were used for subsequent analysis.

The protein distribution within the used set of bacteria can be viewed in Figure 4.1.

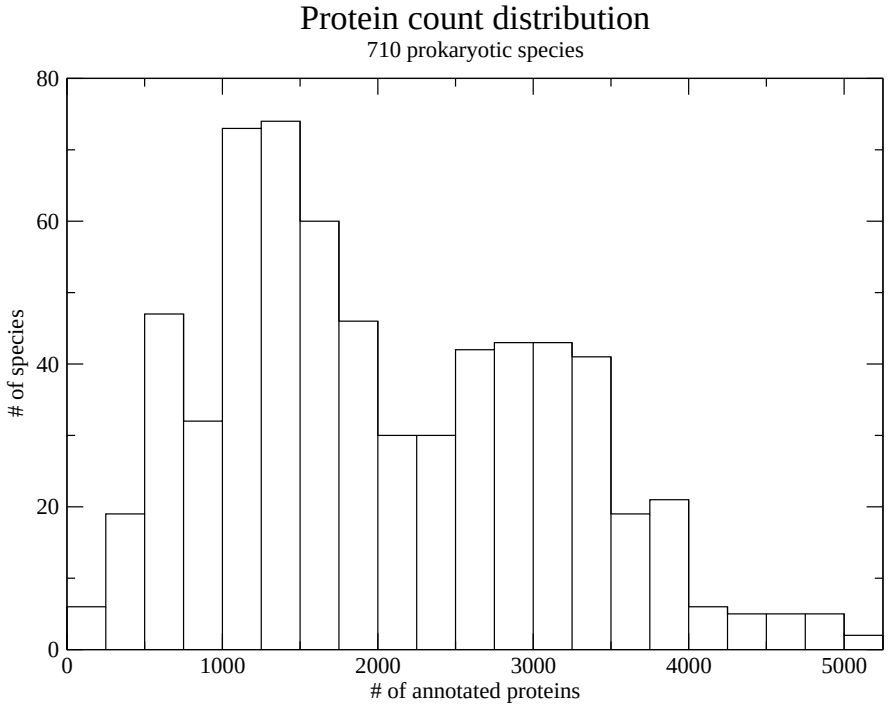


Figure 4.1: Protein distribution: Protein quantity of the considered species after filtering 'putative' and 'hypothetical' entries. The 710 analyzed bacteria had 2026 proteins on average.

4.2.2 Processing

The analysis was done with `Proteinortho` distributed over multiple machines. The used version differs from the previously presented one. Neither the 25% pairwise identity nor 50% coverage for `blast` hits were implemented at this time. Paralogs were allowed in this initial step. `Proteinortho`'s output returned not a single protein type that occurred in all species. One reason could be missing annotations. At the first glance, surely not every protein is annotated in every bacterial species. At the second, all hypothetical and predicted proteins were dismissed from the given

data. To pass over these problems, the output of `Proteinortho` was filtered. Solely, connected components which covered at least 50% of all species were left. Knowing that over half of all species contain a certain protein, it is justified to investigate species where this protein was missed. Therefore, these proteins were used to restock the output as described below.

Restocking

Reduction of sequence number The amino acid sequences of all proteins of the remaining connected components were stored in `fasta` files. A separate file was created for each component. To reduce the complexity of the following `blast` runs, highly similar sequences were replaced by their consensus sequence. The process of truncating is presented in Algorithm 3.

Algorithm 3 Consensus shortening

```
1: for all files f do
2:    align f using clustalw
3:    for all thresholds t = 1...0.8; t-=0.05 do
4:       for all aligned sequences s do
5:          if pairwise_identity(s_n, s_{n+1}) ≥ t then
6:             replace s_n by consensus(s_n, s_{n+1})
7:             remove s_{n+1}
8:          end if
9:       end for
10:      if blast misses hits then
11:         LAST
12:      end if
13:   end for
14: end for
```

After each step, `blast` was applied against the original file with an E-value of 1. This E-value was required because database and query were very short and similar. Every protein had to be found at least once within the reduced data. The set which fulfilled this condition with the lowest threshold was used.

Blast For each connected component the truncated set was used to `blast` against the genomes of all species where no orthologs were found. An E-value of $1e^{-10}$ was

applied. The hits were enlarged to the next surrounding open reading frame. For details see Section 5.2.2. If this was not possible, they were dismissed to avoid false positives. The information was used to restock the orthosets in order to give a more accurate overview in how many species orthologous proteins occur.

Filtering

The resulting sets were filtered. Only connected components having up to 5% paralogs were allowed. This step resulted in 16 proteins present in all species which were regarded. The set of proteins revealed a conspicuous issue. As the coverage overview in Figure 4.2 shows, there are significantly more proteins that occur in 99% of all species than in 100%. It conveys the impression that some widely essential proteins are not necessary to all species or have scattered replacements in some of them. In order to gather a preferably large set, these species (about 1% of the set) were removed. This resulted in 27 domain-wide common proteins.

Some species had duplications within the chosen proteins. For some, it turned out that they even had all of them duplicated. This was consistent with literature where massive duplications in general or duplications of ribosomal elements were reported [81, 82]. As the proteins should be used as reference, duplications would make an assignment more complicated and ambiguous. Up to 25% (= 6) duplicated proteins were allowed per species. If this was not the case, they were dismissed. Again about 1% of the investigated set was removed. Finally, 688 species remained.

Generation of the reference set

The amino acid sequences from all common proteins were fetched. For species that had duplications of individual proteins included, one was chosen randomly. Finally, a reference database for each dedicated group of proteins was created. It contains the equivalent domain-wide proteins for each species. These files were aligned using `clustalw` [83]. The guide trees were used to generate a consensus tree with `consens`, a part of the `PHYLIP`-Package [84].

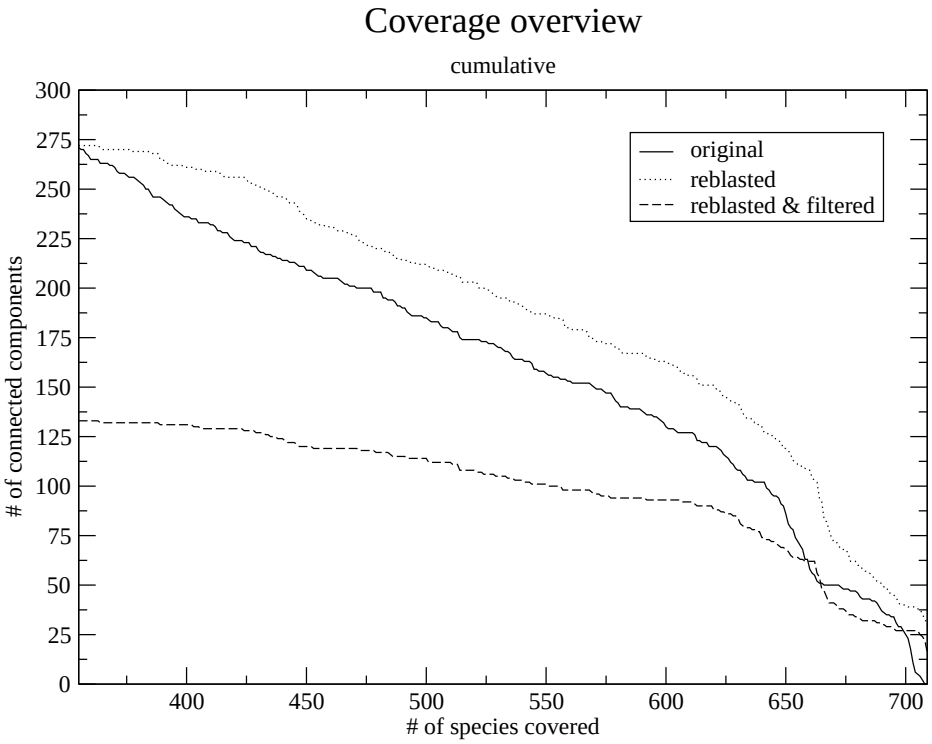


Figure 4.2: Coverage overview: Results of the `Proteinortho` run with 710 bacteria. Using `blast` additional appearances of proteins could be discovered that actually covered all species. Sets with over 5% paralogs were filtered.

4.3 Results

A pool of 27 shared proteins was determined for all 688 species. These can be used to have an insight into conserved pathways and essential elements which are potential drug targets. They allowed to generate a protein based phylogenetic tree which can give an overview from the protein perspective instead of the prevalent $16S$ based classification [85]. Therefore, classification and protein annotation of new species can be facilitated using these sets as additional molecular markers. An according

application will be presented in the next chapter.

Common proteins

The following proteins were found to be orthologous in at least 688 bacterial species:

- $30S$ ribosomal proteins S2, S3, S4, S5, S7, S8, S10, S11, S12, S13, S17, S19
- $50S$ ribosomal proteins L1, L2, L3, L5, L6, L11, L14, L22, L23
- tRNA synthetases for seryl, arginyl, phenylalanyl (alpha chain)
- preprotein translocase, $SecY$ subunit
- peptidase $M22$, O-sialoglycoprotein endopeptidase
- transcription elongation/termination factor $NusA$

Ribosomal proteins The ribosomes are complexes of RNA and proteins responsible for protein synthesis. They translate mRNA into chains of amino acids and assemble proteins. The prokaryotic complex consists of a small $30S$ and a large $50S$ subunit. Both insisting of multiple proteins themselves [8]. For $30S$, 12 of 21 proteins were found, for $50S$, 9 of 34. Many of the not included subunits are known to be homologs [86]. They were filtered for that reason as described in Subsection 4.2.2.

Aminoacyl-tRNA synthetases These proteins charge tRNAs with their corresponding amino acids. Apart from some exceptions, most bacteria have 20 of them [87, 88]. They arose early in evolution and are supposed to be a very ancient group of proteins necessary for the translational machinery. A large number of homologs could be confirmed in the aminoacyl-tRNA synthetases database [89]. Even more of the 20 synthetases should be expected to occur as common homologs. Due to their ancientness, they might have many paralogs and were thus filtered.

Preprotein translocase SecY This protein is involved as subunit in the $secEGY$ protein secretion complex [90]. It is responsible for transporting unfolded proteins to the cell membrane. The subunit can be regarded as essential, since mutations in

the secY gene were shown to be mutual [91]. However, its exact role is unclear [92]. Further investigation might be of interest as this protein is present in most bacteria.

Peptidase M22 *M*22 is a poorly characterized endopeptidase [93]. Following the `MEROPS` peptidase database, homologs can be found in all kingdoms of life. Up to some exceptions, bacteria usually have at least one homolog, but usually more [94, 95, 96]. As activity could only be proven in *Pasteurella haemolytica* up to day, it is suggested that the gene has either been misassigned or a special co-factor is required [97]. For this analysis, only orthosets with less than 5% of paralogous were allowed. Therefore, the data of `MEROPS`, implying multiple copies in most bacterial genomes, could not be verified. Instead, it supports the theory of misassignment. Otherwise, these homologs seem to have diverged in a way that makes them not appear as homologs to `Proteinortho`.

NusA This transcription factor is reported to be highly conserved and essential for transcription elongation. It forms a complex with the RNA Polymerase and prevents premature termination. Thereby, it regulates the rate of transcription [98, 99]. It is required for rho-dependent termination as well [100, 101].

Reference database

The set of common proteins can be used to approximate related species. For each of the detected common proteins, a reference database was created. This database allows estimating the most similar species within the reference set regarding its sequence in a genome of interest. Furthermore, similar species can be detected using the consensus tree of the protein alignments with respect to their protein conservation.

4.4 Discussion

The number of common proteins appears quite small. Certain processes like transport of amino acids, iron, phosphate, synthesis of arginine, biotine, ribose, and the SOS response are supposed to be conserved. This should hold for the contributing

proteins as well. However, the regulatory networks of bacteria are known to be extremely flexible including the covered proteins [102]. Especially, transcription factors are poorly conserved. This fact actually makes similarity based orthology detection unfavorable to detect them. Additionally, problems are introduced with subfunctionalization of paralogs. Even if a pathway is conserved, details for participating proteins may have change during evolution.

Additionally, about half of the proteins were dismissed during filtering of 'hypotheticals' and 'putatives' in order to generate a reliable set. In context of domain-wide analysis, this might have been a disadvantageous decision. It obviously reduces the chance of finding homologs solely because they were not validated yet. Sets which basically consist of not validated proteins could have been dismissed within post processing.

A not negligible part of proteins was assigned to a 'mega cluster' which could not be partitioned further. Reasons and possible solutions were discussed in Subsection 3.4.2. This insight gave the impact to add the 25% pairwise identity and the 50% overlapping requirements to the defaults of `Proteinortho`. Repeating the study with the improved version and including predicted protein annotations would be favorable.

Chapter 5

Annotation pipeline

In this chapter, a more complex application for `Proteinortho` will be presented. The concerned problem is annotation of genes and transcriptional elements for newly sequenced genomes. An additional challenge is that no further knowledge is required as genomic sequence data from will be used and compared to previous `Proteinortho` output. In detail, taxonomic classification is not necessary and the genome does not need to be complete. The only condition is that the species belongs to bacteria.

5.1 Background

The way from gene to protein is equal in all bacteria. It is basically divided into two steps. The first one is transcription. The information from DNA is copied to a RNA molecule. To achieve this, an enzyme called RNA Polymerase initiates binding to the template strand of the DNA. This occurs on certain sites, called promoters. The enzyme moves along while copying the information to RNA. The process is called elongation. Finally, the RNA Polymerase terminates transcription due to a special site it has reached (terminator) and dissociates from the template [9, 103].

The prepared RNA is subsequently bound by a special protein complex, the ribosome which initiates the process of translation. During this procedure, the information encoded on the RNA molecule is translated to a protein. Unlike than in eukaryotes, transcription and translation are not individual processes separated by a physical barrier. Translation can start as soon as the emerging RNA exposes a binding site for the ribosome [9, 103]. The following part will describe the elements and mechanisms in more detail.

5.1.1 Transcription

Promoter

Promoters are specific sites on the DNA where the RNA Polymerase binds and initiates the transcription. They are usually placed proximal upstream to the transcription start site and consist of two important elements which are assumed to be highly conserved within all prokaryotes. The first one is the −35-box and second the −10-box which is called Pribnow-box as well. As their names indicate, they are located (centered) around 35 and 10 base pairs in front of (upstream) the transcription start site. Both are separated by 17 ± 1 base pairs [8]. An extension of the -10 box, called extended −10 box (TGn) can substitute the -35 box [103, 104]. The most significant −10 consensus sequence in *Escherichia coli* and most other bacteria is `TATAAT`. An example sequence logo for *Bacillus subtilis* is shown in Figure 5.1. The according consensus sequence of the −35-box is `TTGACA` [105]. An UP-element is located upstream to the previously mentioned regions. It supports the recruitment of the RNA Polymerase which is mainly mediated by DNA-binding proteins [9, 103, 106].

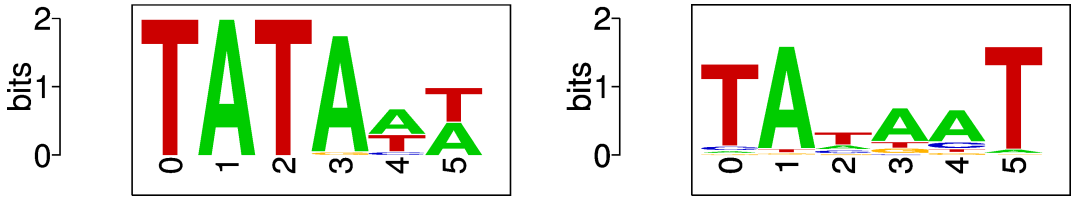


Figure 5.1: -10-box sequence logo: On the left, the logo of predicted TATA-boxes generated during tests with *Bacillus subtilis* is shown. It is based on 692 sequences. The logo of 329 experimentally validated σ^A TATA-boxes is shown on the right [107]. σ^A is the major housekeeping σ-factor in this species [108]. The consensus sequence found in literature is well represented. However, the pipeline's predictions focus on the most conserved sequences and thus, miss varying sequences.

The mentioned characteristic regions are fundamental for the initial binding. Their relative contribution differs. They can compensate each others imperfections. A perfect promoter by means of consensus sequence is actually never found [103]. Probably, such a promoter would bind the polymerase with a strength that prevents it from moving further for elongation.

Terminator

Two basic types of transcription termination are known, directly through secondary structure of the emerging mRNA and indirectly by interaction between mRNA and special factors. The mechanisms are described in more detail below.

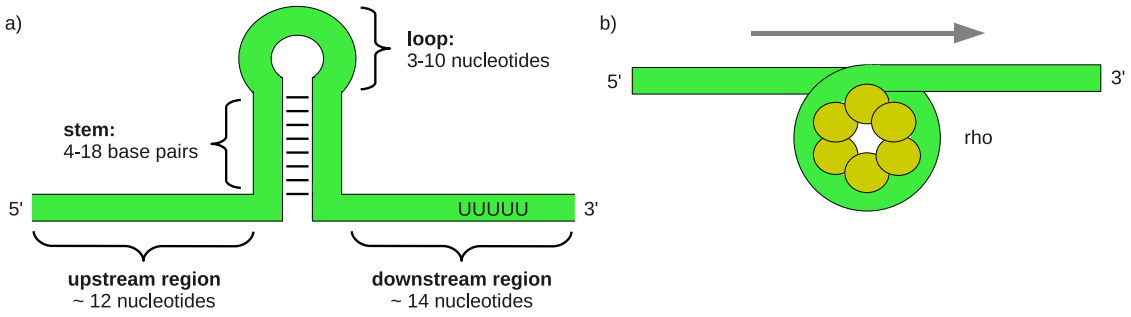


Figure 5.2: Terminator models: a) The basic model of an intrinsic rho-independent terminator on mRNA level. Most important is the stem loop structure. Internal bulges and small loops may occur. The downstream region contains a poly-uracil trail. b) Ring-shaped protein *Rho* (light gray) winding the emerging RNA, leading to rho-dependent termination.

Rho-independent In most cases, termination is achieved directly without the requirement of any factors. The (intrinsic) terminator is a characteristic about 40 nucleotides long and GC-rich sequence of the emerging mRNA. It is responsible for the termination effect. It contains an (not necessarily perfect) inverted repeat which is separated by a spacer of three to ten nucleotides (see Figure 5.2a). This sequence forms a hairpin loop which pauses the polymerase. The following poly-uracil tail leads to dissociation of the RNA Polymerase and thus to the termination of transcription [109, 8, 110, 111].

Rho-dependent This mechanism of termination is actually a subgroup of factor-dependent termination. However, rho-dependent termination is the most common and investigated one. Other factor dependent mechanisms are supposed to act in a similar way [112, 113, 114].

In contrast to direct termination as described above, the termination sequence does not rely on hairpin loop formation or a poly-uracil tail. Instead, a binding site

for the protein *Rho* the so called rho-site is required. The protein is ring-shaped and has ATPase and helicase activities. Once bound to the ribosome-free mRNA, it moves along heading to the DNA-RNA hybrid at the RNA polymerase. If this location is reached, the hybrid gets unwound. This in turn leads to transcription termination [115]. *Rho* releases the transcript afterwards. Additional factors such as $NusA$, $NusB$ and $NusG$ participate [9, 112, 103]. An illustration of *Rho* winding the mRNA is shown in Figure 5.2b.

Experiments point out that the binding site is very sensitive to cytosine deletions whereas uracil, adenine and especially guanine deletions have a lower influence on successful termination. Furthermore, a stop codon seems to be required for termination. As the ribosome attaches in front of the *Rho*-factor, it needs to fall off before the factor can pass it. Otherwise, it can not reach the hybrid [9].

Antitermination As described above, terminator structures - rho-dependent or rho-independent - assess the end of a transcript. It can be much longer if the termination signal is elided. This in general results in a larger number of products encoded to mRNA [9, 8]. Rho-dependent termination for example, is disabled by inactivating the *Rho* protein or hindering it to bind due to formation of secondary structure or previous binding of other proteins. In the rho-independent case, the responsible hairpin loop is blocked by proteins. Numerous variations of these basic models are known [113, 114].

Regulation

The regulation within cells is very complex and usually based on multiple layers. Transcriptional regulation is only one of them. In this field, the initiation of transcription is an important controlling factor. For prokaryotes, three different types of transcription activation are known which result in different promoters. Additional transcription factors, activating or repressing proteins and transcription termination play a role in regulation [9, 103]. In the first instance, it is important to introduce the RNA Polymerase enzyme in more detail. It is the key element of transcription and thus, important for regulation.

RNA Polymerase This enzyme has a central role for the transcription. Structural as well as functional similarities to the DNA dependent RNA Polymerase II, found in yeast and other eukaryotes, are present [116, 117, 118, 119]. Its core consists of several subunits which are namely β, β', ω and two α units. The subunits are important for the transcription [103]. β and β' form the active site of the polymerase. It binds both, template DNA and the RNA transcript. ω on the other hand, is supposed to assist folding the β' subunit [120]. Both α subunits have an amino-terminal-domain (αNTD). They are necessary to assemble β and β'. Furthermore, a carboxyl-terminal-domain (αCTD) is present. It mediates DNA-binding proteins located on the up-element and thus, is important for initiation at most promoters [106, 121].

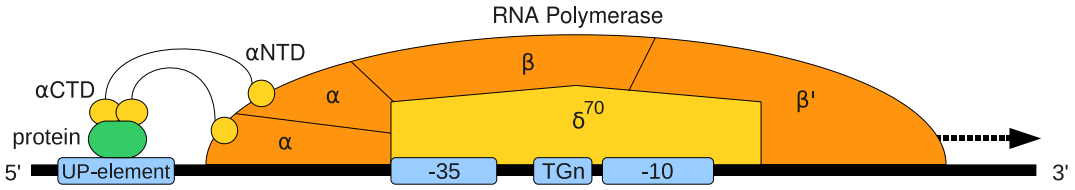


Figure 5.3: RNA Polymerase and promoter: 2D-illustration of interactions between prokaryotic RNA Polymerase with an attached σ^{70} subunit and promoter elements. The UP-element is bound by a DNA-binding protein which is bound in turn to the αCTD of the polymerase. Adapted from [103].

To initiate the transcription at a particular promoter, the formation of a holoenzyme is necessary. This takes place by integration of a σ-family subunit [103]. This subunits effect the enzyme in two important ways: Firstly, it is needed to recognize specific promoter sequences and positions the holoenzyme at the right location. Secondly, it helps unwinding the DNA next to the transcription start site [122].

Most bacteria have different σ-factors (σ subunits) which share common features. However, an exception in this case is the σ^{54} family. It does not share any sequence homology with the other families and uses a distinct pathway of open complex formation [123, 124]. The σ-factors have four different conserved domains that are involved in binding to the core enzyme and DNA melting. Furthermore, they recognize the promoter's -10- and -35-box, as well as an extended -10-region [103, 123, 125]. Whereas σ^{70} is the major subunit, different σ-factors enable the RNA

Polymerase to recognize different sets of promoters with different efficiency. This makes the cell able to alter transcription in response to stresses [123].

σ competition An important fact regarding gene expression is the competition between promoters for RNA Polymerase and σ-factors which both are normally short in supply [126, 127]. Due to this fact, it is not always possible to utilize effective promoters in different species. In subjection to the availability of σ-factors other promoters could be favored. The number of promoters utilizing a certain σ-factor is important for regulation as well. If all have an affinity to the same factor, they need to share the available ones. Reduced transcription ratio is the result.

Transcription factors Transcription factors are proteins that up- or down-regulate the transcription of genes [128, 129]. They affect the expression of genes based on environmental signals. The majority has specific binding sites next to certain promoters [130]. These can be UP-elements or so called operator sites which suit as binding site for proteins. Some factors control large sets of genes, others only act on a single promoter [131, 132, 133]. However, their functions can be different according to the type of promoter, gene location and other proteins they interact with. Depending on presented or hidden regions, some can act as activator in one but as repressor in another case [129]. For example, the *Escherichia coli* infecting phage λ encodes a protein that activates its own gene while repressing the anti-sense gene for *Cro* [134].

Regulated recruitment (Class I) Regulated recruitment is a mechanism bacterias have in common with eukaryotes. If no repressor inhibits it, the genes' transcription always works on a basal level because polymerase will bind the promoter from time to time. This happens more often if the concentration of polymerase in the cytoplasm increases [9].

While the promoter itself does not need to be very strong, other regions around (mostly upstream) are responsible for the recruitment of polymerase. This happens indirect by binding an activator protein which has an affinity to the polymerase. Furthermore, there can be more of these activators cooperating synergically. They

might bind the polymerase in different ways for example. This allows a combinatorial control of gene expression [8, 103].

Summing up, the rate of polymerase binding is influenced. Without activators the gene expression is not off but works in a very inefficient manner and therefore, on a basal level solely.

Polymerase activation (Class II) In this case, the polymerase binds the promoter but stays inactive. There will be no transcription until it is activated by a protein which induces a change of polymerase conformation. Furthermore, ATP is required for this reaction [103].

Promoter activation (Conformation change) Normally, the distance between the -10- and the -35-box is 17 ± 1 base pairs. Promoters of this kind instead, hold a distance of 19 base pairs between these boxes. The increased distance prevents the polymerase from adjusting correctly. A special protein is required which reshapes the promoter. It does not need to operate with the polymerase itself since it can bind on a different side. This reshaping of the promoter makes the polymerase able to start transcription. Otherwise, the holoenzyme will be stuck on the promoter what makes it work like a repressor [103].

Operons

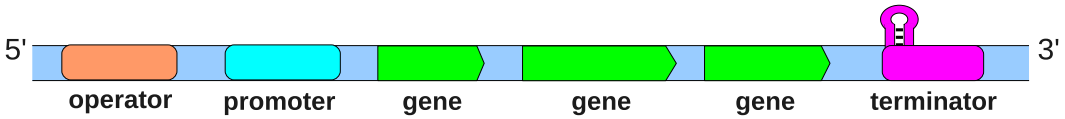


Figure 5.4: Operons: Multiple (protein coding) genes share operator, promoter and terminator and are transcribed together. In this way related proteins can be regulated mutually. Further transcriptional elements can occur within the unit and allow a more exact regulation.

Operons are units of clustered genes in bacterial genomes. They are commonly subject to gene expression and regulation as they share transcriptional elements such as the promoter and terminator. For that reason, they tend to have related functions [9, 135]. The position of co-transcribed genes is often conserved among

related species whereas it is shown that single genes tend to be rearranged in the absence of evolutionary pressure [136].

5.1.2 Translation

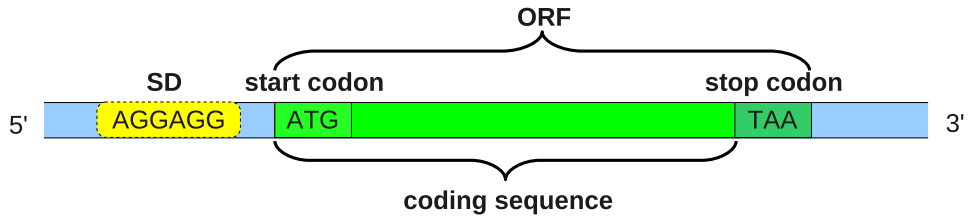


Figure 5.5: Translational elements: Overview of translational elements on DNA-level. In most cases, the coding sequence is led by a Shine-Dalgarno sequence (SD). The protein coding sequence itself always starts with a start codon and ends immediately in front of a stop codon while the open reading frame (ORF) additionally contains the stop codon.

Coding sequence

The coding sequence codes for the actual protein. It consists of codons which are tri-nucleotide sequences. Each is coding for a certain amino acid [9]. An illustration is presented in Figure 5.5. Every such sequence begins with a start codon which initiates the transcription start. The most frequent occurring start codon is `ATG`. It codes for the amino acid methionine. `GTG` (valine) and `TTG` (leucine) occur in some cases as well [137, 138, 139, 140, 141, 142, 143, 144]. Furthermore, `ATT` (isoleucine) and `CTG` are found to act as start codon in rare cases [145, 146, 147, 141, 148].

In comparison to eukaryotes, introns are negligible in bacteria as they occur in rare cases only [149, 9, 150]. This fact makes it possible to gather the amino acid sequence from the coding sequence and backwards by using a mapping between both. It is known as the genetic code. Some organisms show slight variations in this code [151]. For example, *Mycoplasma capricolum* is translating `TGA` to tryptophan instead of treading it like a stop codon following the universal genetic code [152].

The stop codon is located directly after each coding sequence (downstream). It is recognized by release factors. They make the ribosome release the peptide and

thus, finish translation, no matter which nucleotides follow after. In contrast to all other codons, it does not code for an amino acid additionally to its function. Common stop codons are `TAA`, `TAG` and `TGA` [151, 9, 8]. Under special circumstances, these regions pause the ribosome only but do not make it fall apart. The complex can then continue translation [153].

Areas between a possible start and stop codon on DNA level are called open reading frame (ORF). The existence of an open reading frame is however no evidence for an encoded protein at this location but it provides a good indication.

Shine-Dalgarno sequence

The Shine-Dalgarno sequence is located around four to seven nucleotides upstream of the start codon. It consists of an about ten nucleotides long conserved motif with the consensus sequence `AAGGAGGTGA` [154]. However, the `AGGAGG` containing area is the most conserved and will be focused in this thesis. Figure 5.6 shows an example. The Shine-Dalgarno sequence allows the ribosome to attach the mRNA and furthermore, detect the right initial start codon since it is usually complementary to a sequence near the 3'-end of the 16S-rRNA which is part of the ribosome complex. This detection is not trivial as their might be several codons which would suit as initial start codon [8, 155].

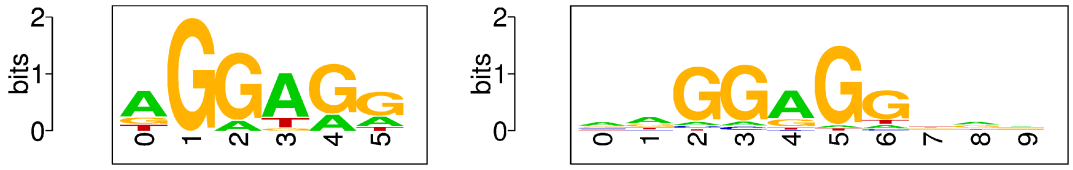


Figure 5.6: Shine-Dalgarno sequence logo: The predicted ribosome binding site generated during tests with *Moorella thermoacetica ATCC* is shown on the left. It is based on 821 sequences. The logo of 867 Shine-Dalgarno sequences annotated in the `ProTISA` database is shown on the right [156]. They were predicted using the two reliable methods. The first is based on comparisons to the Conserved Domain Database (`CDD`) [157, 158]. The second uses alignments of orthologous genes [159]. The consensus sequences match well (when trimmed). However, the pipeline predicts shorter sequences as it focuses on the most conserved part.

Even if this ribosome binding site belongs to the basic model of a gene, it is not compulsory [160]. In fast-growing bacteria for example, the amount of genes that

hold a Shine-Dalgarno sequence in front (upstream of an ORF) is around 90% while some parasites and cyanobacteria are reported to have only 20% genes with such a strong dedicated sequence [154].

One reason is the phenomenon of coupled transcription which occurs in operons. In these cases, a stop codon overlaps with the start codon of the following coding sequence. Such a sequence could be `ATGA` for example. The ribosome is evidently not falling of there but goes on generating the next protein [153]. Apparently, there are additional other mechanisms that may not acquire an overlap. The overall probability of a present Shine-Dalgarno sequence in front of genes organized in operons drops, the further downstream they are located from the initial start [154].

In conclusion, the existence of these sequences upstream of open reading frames is a good indicator for active translation. However, the absence cannot be used as criterion for exclusion.

5.2 Methods

As stated before, the intention is to construct an annotation pipeline for a given (newly sequenced) genome which is based on the `Proteinortho` results from the 'Domain-wide commons' in Chapter 4. An overview is illustrated in Figure 5.7.

Due to the lack of taxonomic information for an unannotated genome, a classification has to be done to determine already annotated and closely related species. For this, the 27 reference proteins from Chapter 4 are utilized. Every protein from the determined related species is then used as seed if they are included in at least 25% of them. This results in a trustworthy initial set. Common proteins within putative relatives are likely to be present in the species of interest as well. They are used to approximate the properties of protein coding regions within the unannotated genome. Motifs for transcriptional and translational elements are of interest especially. These are used to predict further proteins and are finally refined by the newly predicted data for subsequent analysis.

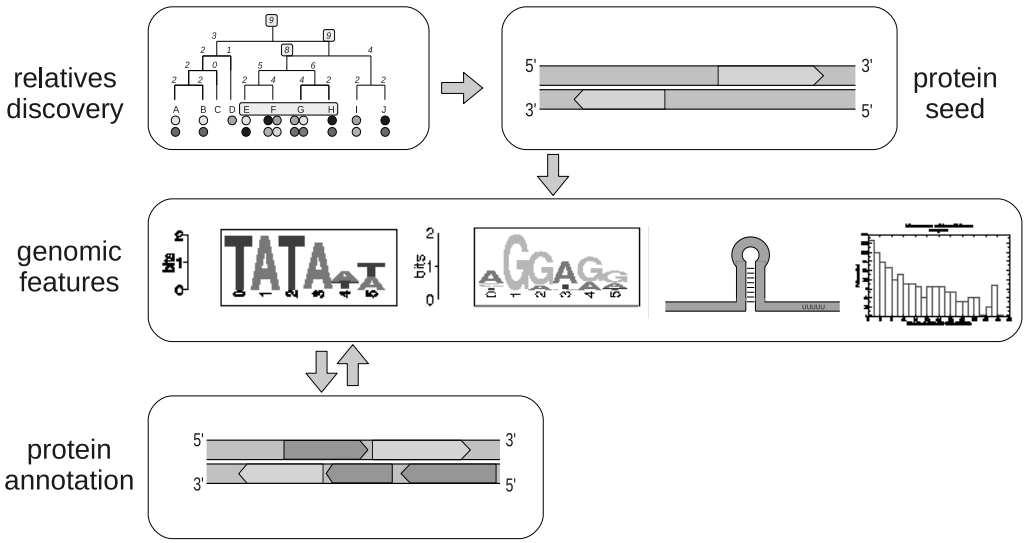


Figure 5.7: Pipeline overview: Initial step of the pipeline is the discovery of related species. Concluding from this, common proteins are used as seed for trustworthy candidates (light gray). These can be used to derive sequence and genomic features which aid in further annotation (dark gray). Finally, these features can be refined based on the gained annotation.

5.2.1 Relatives discovery

Related species have to be identified in the first step. This information will be the starting point for the choice of proteins in the next homology based annotation step. To achieve good protein similarity, the reference proteins and the resulting consensus tree, established in Chapter 4, are used. The proteins serve as similarity database to estimate related species. Each protein set is used as query to search by `blast` in the new unannotated genome. The best hit regarding its bitscore is used as a reference point and thus, will be marked within the reference tree. This is done for each protein set resulting in multiple markers within the reference tree.

The markers are derived from 27 different protein sets. In turn, this results in 27 markers normally. However, it can be more or less, depending on the number of equally good proteins within the reference set and the hits within the genome. If the genome is not complete or certain common proteins are lost, less markers could be found. Additionally, some protein sequences within the reference database are similar. The reference set was shrunken to a quasi-non-redundant database

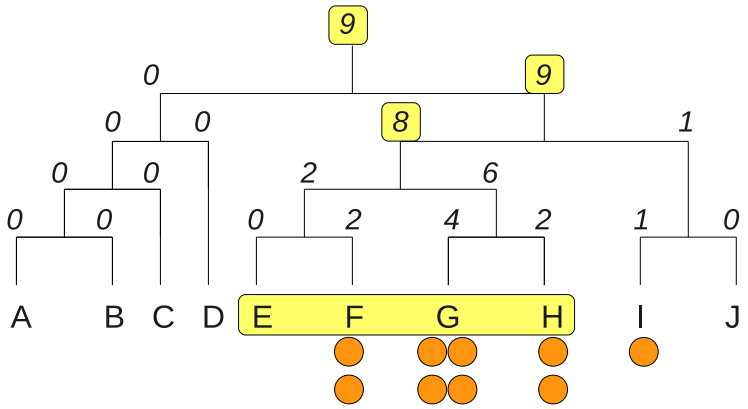


Figure 5.8: Subtree assignment graph example: A subtree is searched that includes at least eight of nine `blast` hits (reference points, marked as circles). Starting from each leaf with a marker, a counter within the parent nodes is incremented recursively. Therefore, this counter tells how many hits can be found within the subtree of its descendants. Species E, F, G and H will be chosen as they are in the smallest subtree containing at least 75% of all markers.

using `nrdb` [161]. Identical sequences were merged in order to avoid unnecessary `blast` effort. In these cases multiple, less weighted markers will be used to represent these hits. Thereby, equally good hits are included as well.

The smallest subtree is chosen that includes at least 75% of these markers. This value was applied to add some robustness against separate misleading hits. As there are more of these subtrees possible in general, the one with the smallest number of descendants is chosen (including leafs and inner nodes). For more details, see Figure 5.8. Once a subtree was determined, the species within can be regarded as related by means of their protein conservation and are used for further homology based annotation.

Handling similar sequences

As noted above, weighted markers representing more than one species are possible. This occurs especially for species that have many closely related organisms within reference tree. Applying the relatives discovery to *Escherichia coli* for example, will result in a lot of markers over multiple species. For this purpose, colors are used. A marker only counts if none of the same color was counted before as Figure 5.9

illustrates. In this way, a reliable relatives detection is achievable despite widespread hits caused by very similar sequences in different reference species.

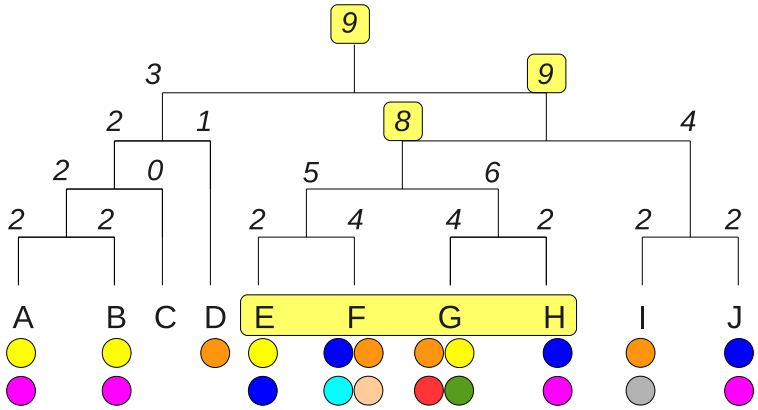


Figure 5.9: Subtree assignment using colors: The basic method acts as described in Figure 5.8, but in this case some reference points split into multiple hits. The weighted markers are marked as colored circles. Every color (here shown in gray-scale) represents hits for one of the reference proteins. Again the counters within parent nodes are incremented recursively but in contrast, the counter is only incremented if a node of the same color was not counted already. In this way, the best fitting subtree for multiple hits can be chosen, even if there is not a single unambiguous hit.

5.2.2 Protein annotation based on homology

Choice of sample proteins

Based on the number of assigned relatives, different methods are utilized to increase specificity. Proteins present in at least 25% of the relatives will be determined in order to gather a trustworthy set. Figure 5.10 points out details. This is the most important factor at this stage as the resulting annotations will be used as training sets for all subsequent steps.

Direct `blast` method This method is used if one to four related species are located in the assigned subtree. If a protein is found in at least one of these species, the probability to find it in a related one is high. Assuming the chance to find it would be $\frac{m}{n+1}$. Where n is the number of species in the subtree and m the number

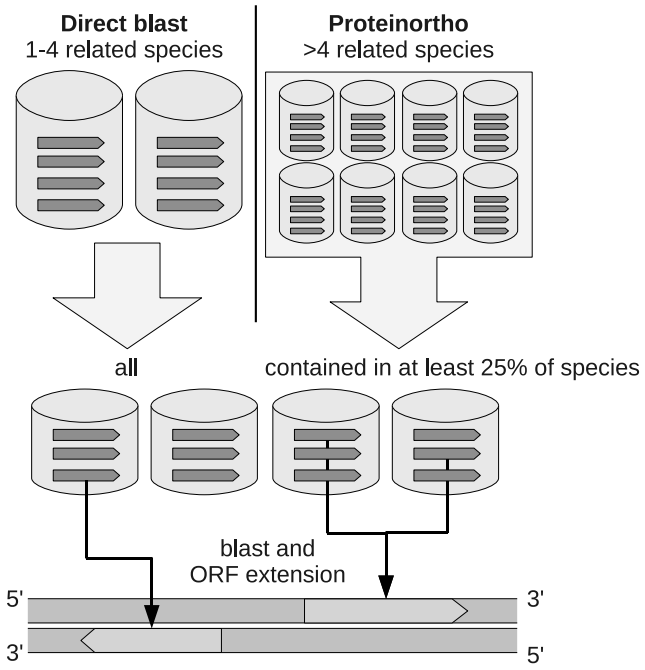


Figure 5.10: Reference choosing methods: Depending on the number of species in the assigned subtree, two different methods are used to generate a set of reference proteins. The direct `blast` method is used if not more than four species were determined as relatives. Otherwise, the considerable proteins (dark gray) are filtered using `Proteinortho`. The remaining candidates will be used subsequently to annotate homologs (light gray) in the genome of interest.

of species containing the protein (as homologs). The worst case ($n = 4, m = 1$) would be a probability of 20%. This is good since nothing else is known about the new genome. Furthermore, the chance of false positives will be decreased as more attributes such as the presence of an open reading frame are checked. The according protein sets will be downloaded from `NCBI` and used directly for a `blast` run against the unannotated genome.

Proteinortho method With a raising number of species in the subtree, the chance to find the proteins in the genome of interest decreases. The according protein sets of all related species are downloaded from `NCBI` and filtered using `Proteinortho` [49]. A reciprocal `blast` search with an E-value of $1e^{-10}$ is done with these sets. Only proteins detected in at least 25% of all regarded species will

be kept in order to use frequent proteins only. This leads to a worst case of $\frac{\lceil 0.25n \rceil}{n+1} \geq \frac{2}{9}$ for $n \geq 5$ and thus stays over 20% as well. Using the remaining proteins, a quasi-non-redundant database is generated with `nrdb`. This database is used subsequently to annotated the investigated genome.

Choice of `blast` hits

The `blast` runs are performed using `tblastn` with an E-value of $1e^{-10}$. Hits which are shorter than 75% of the respective protein in the query are not considered further. If a query results in more than one hit, the best hits are taken into account solely. This is calculated as before. Regarding their relative bitscore, $\frac{best+candidate}{best} - 1 > 0.75$ has to be fulfilled for each candidate. In this way, conserved paralogs should be detected as well.

Hit validation

An according open reading frame is assigned to every chosen `blast` hit to approve that it matches a possible protein coding region. In order to locate it, each `blast` hit is extended up- and downstream:

- + the number of amino acids which are missing to achieve the respective protein's full length regarding the hit's relative location
- +10% of the respective protein length of the query
- +10 amino acids, to embrace very short proteins

The search for a start codon in the extended sequence begins at −10% of the position of the original hit's $3'$ end. It heads towards the upstream extension of the query's hit. The most upstream candidate is chosen under the condition that no stop was encountered before. An illustration is presented in Figure 5.11. If none of the regular start codons (`ATG`, `GTG`, `TTG`) is found, this is repeated with the alternative codons (`CTG` and `ATT`) introduced in Subsection 5.1.2. A stop codon (`TAA`, `TGA` or `TAG`) is searched downstream the located start codon. The very first one is chosen as the translation would not continue afterwards a stop signal.

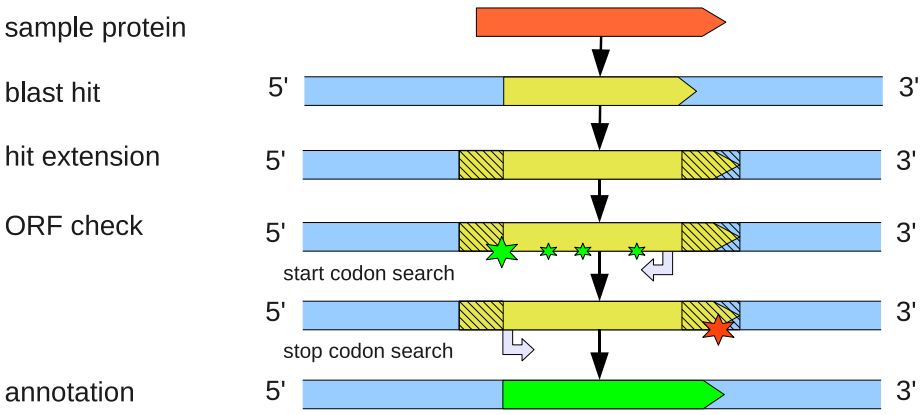


Figure 5.11: ORF check: After the hit was extended, a start codon is searched from the 3′-end to the 5′-end. The very last one is supposed to be the right. Starting from this position back to the 5′-end, a stop codon is searched. Here, the very first one is taken. A protein will only be annotated if both are found.

Finally, the length of the located open reading frame has to be at least 75% of the respective protein from the query. Shorter sequences are supposed to be pseudogenes or false positives and therefore, will be rejected. Otherwise, the located open reading frame will be annotated as protein. The maximum length is limited by the way the extension was done. Located open reading frames cannot exceed the length of the query protein to more than 20% + 20 amino acids (+10% of query length +10 amino acids up- and downstream).

Feature extraction

The validated hits will serve as seeds for subsequent analysis. Typical features as the derivation for start and stop codons, protein lengths and amino acid frequencies are extracted. These features can enhance further predictions with respect to the characteristics of the present genome.

5.2.3 Transcriptional elements

Data about transcriptional elements is derived from the set of beforehand predicted homologs. These are trustworthy as they appear in closely related species. Promoter and Shine-Dalgarno sequence can provide important clues to further predict protein

coding genes. Their sequence motifs are analyzed by `meme`, an iterative expectation maximization approach [162]. The genomes nucleotide frequencies are identified and used as background distribution to increase accuracy. The tool can derive this information itself regarding the given sequences. However, this data will be biased as most of them should contain transcriptional elements. Additionally, two programs, `aln2pattern` and `fragrep`, were used to generate the sequence patterns and detect them within the genome [163, 164]. Their source code had to be changed slightly to handle larger sequence sets as well.

Shine-Dalgarno sequence

The ribosome-binding site should be found in front (upstream) of every protein coding gene. To detect them, `meme` was used. Given a set of sequences, it determines the most common motif. Thereby, the amount of time increases dramatically with the number of sequences. Additionally, operons can be assumed to appear frequently in bacteria. Genes grouped in these transcriptional units may not need a separate ribosome-binding site as derived in Subsection 5.1.2. In order to keep the assignment within a usable time-scale and furthermore enhance accuracy, all protein coding genes that have another one close in front are removed from the set. The concerning distance will be introduced later. For `meme`, a number of up to 1300 sequences has shown to be reasonable. As a representative choice of sequences can be assumed, it would be of less use to analyze more sequences. The required time becomes much larger while the result improves only marginal.

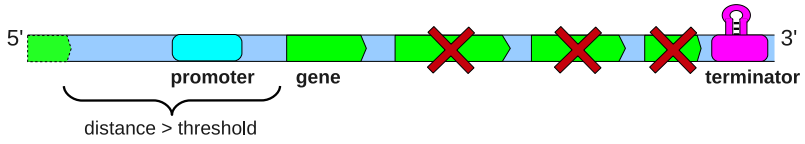


Figure 5.12: Simple operon model: Not all protein coding genes need a Shine-Dalgarno sequence immediately in front. Especially for genes within operons, such a sequence in front of the very first gene may be sufficient. Therefore, these leading genes are used preferably for distinguishing the element's pattern. As profound operon prediction can hardly be done at this stage, it will be approximated by a forced minimum distance to the preceding gene. If this requirement is not fulfilled, the gene will not be considered further (cross).

Concerning these circumstances, an adaptive choice of sample sequences was introduced. A threshold starting from zero base pairs is the initial allowed distance to an upstream protein coding gene. Solely, genes with distances below will not be considered for Shine-Dalgarno motif analysis. If the number of considered genes is above the limit of 1300, the threshold is raised by additional 50 base pairs. The procedure is repeated until the number of genes is within the limit. In this way, genes within possible operon units can be filtered preferentially while the leading genes with the highest chance for presence of a Shine-Dalgarno sequence should remain for analysis. See Figure 5.12 for details.

All areas from 1 to 25 base pairs upstream the remaining sequences are analyzed for a conserved motif. The iteration starts from the common consensus sequence `AGGAGG`. This pattern is supposed to belong to the typical Shine-Dalgarno sequences for that certain species. The resulting motif is then generated using `aln2pattern` and searched within the remaining sequences using `fragrep`. All hits are marked as putative Shine-Dalgarno sequence.

Promoter

The subset of protein coding genes lead by a (putative) Shine-Dalgarno sequence is used to gather the promoter pattern. Again, if their number exceeds 1300, they will be filtered preferring operon leading genes. It is very likely to find promoters next to these locations. 250 base pairs in front of each marked Shine-Dalgarno sequence are analyzed using `meme`. The iteration starts from the consensus sequence `TATAAT`. These hits are supposed to be the -10-box. In this way, the most utilized promoter sequence within the investigated genome should be found. It has to be noted that different σ-factors are not considered.

Terminator

For prediction of rho-independent terminator structures, `TransTermHP` was used [165]. It detects stem loops within a given sequence and calculates their stability. Furthermore, it considers the typical terminator characteristics introduced in Subsection 5.1.1. The presence and quality of all features are evaluated, resulting in a score

between 40 (uncertain) and 100 (confident). For some reason, scores below 40 are not reported by `TransTermHP`.

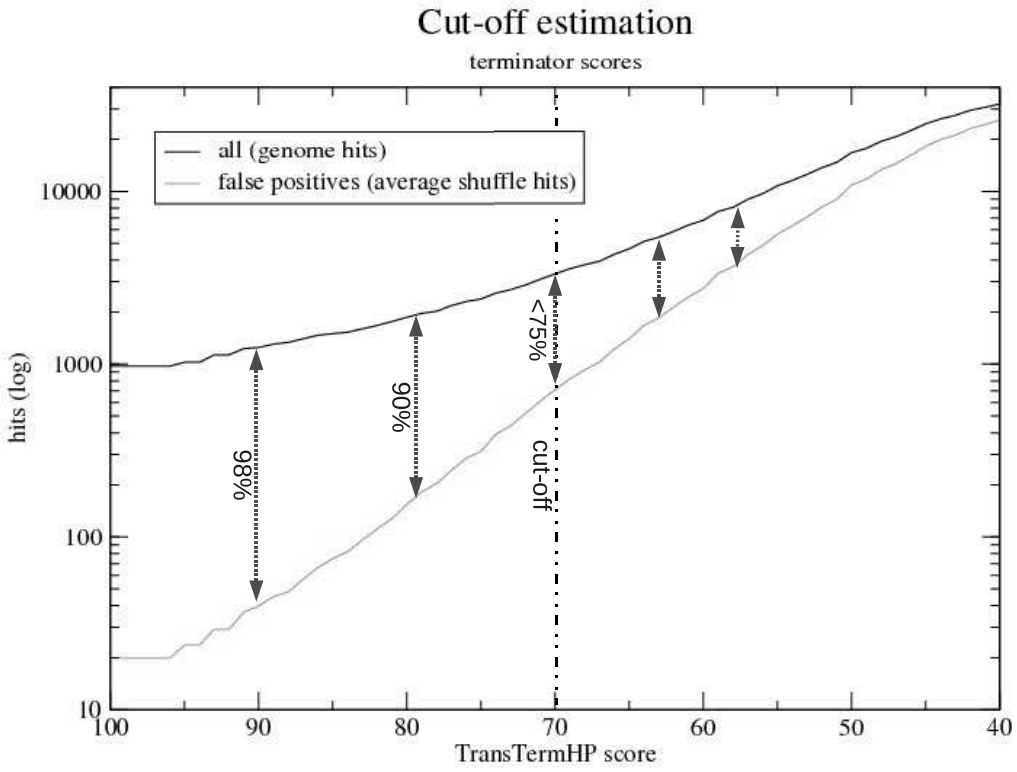


Figure 5.13: Terminator cut-off: To estimate a reasonable cut-off for `TransTermHP` predictions, the lowest score is chosen that approximately provides 75% true positive. The figure shows an illustration for application with the *Bacillus subtilis* genome.

Using a fixed schema, the approach is inflexible to adapt to terminators in different species. Extraordinary terminators should not be detected especially. However, it allows annotating well known prokaryotic terminator structures very accurate. The basic task is to estimate a threshold regarding the annotated scores for a certain unknown species. This is done as follows.

Initially, the genome is shuffled ten times while remaining the mononucleotide frequencies. `TransTermHP` is then applied to these shuffled genomes and the number of hits per score are averaged. All hits are supposed to be false positives as no certain

terminator structure should be present within a shuffled genome. Subsequently, `TransTermHP` is applied to the original genome. The difference of hits between this run and the averaged shuffled runs is considered to be the actual true positives rate. The first score between 100 and 40 where less than 75% of the predicted terminators are supposed to be true positives is chosen for subsequent prediction. Figure 5.13 illustrates this with an example.

The prediction is then applied to all downstream areas of herefore annotated protein coding genes. 250 base pairs are considered. Unfortunately, `TransTermHP` requires an annotation file to work properly. Otherwise, it will not search for terminators at all. The algorithm requires at least two genes to be present. It takes the background GC-content into account to compute the scores. This content normally differs from intergenic to intragenic regions. Furthermore, data about genes is used to tag putative terminators as 'inside genes' or 'intergenic' which is not important in this case [165]. There is nothing known about genes within the unannotated genome anyway. To avoid this problem, the author advises to introduce fake genes of length one at the start and the beginning of the examined sequence which were set to positions 1,2 and $L-1$,L where L is the length of the sequence. In this way, `TransTermHP` could be used without a former annotation file. A side effect of the described workaround for `TransTermHP` is that the scores are less reliable. They basically depend on the GC-content in comparison to protein coding sequences. If genes occur that are only one base pair in length, this clearly harms the point of origin for score calculation. However, the derived threshold for scores should be sufficient to resist this effect.

5.2.4 Protein coding genes

The annotation of protein coding genes is enhanced by the derived Shine-Dalgarno sequence combined with statistical information derived from the homology based annotation using `glimmer`. This is a very common gene prediction tool for bacteria [166, 167]. It is based on a hidden Markov model which has to be trained. An appropriate training set can be derived from large open reading frames located within the genome. Instead, the trustworthy and more numerous previously annotated protein coding genes are used. The Shine-Dalgarno motif is incorporated to

aid as important signal for genes. Additionally, the derivations of start and stop codons are used for a probability scoring. Subsequent to the annotation, the promoter, terminator and Shine-Dalgarno motif recovery is repeated and if required, refined based on the more complete annotation.

5.3 Example run

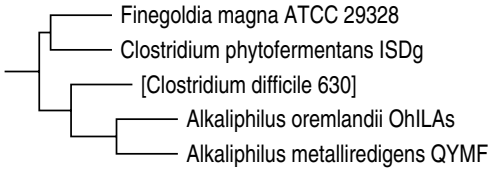
In order to present an example of the pipelines output, the newly sequenced genome of *Clostridium difficile CD196* was downloaded from `NCBI` at 2009/11/20 and applied to the annotation pipeline. The genome contained 3464 putative genes according to the annotation file which was enclosed. In the following, boxes mark the results of the pipeline. Each box is succeeded by the description of the result. Due to the recent sequencing, no reliable data for comparison was available. However, the derived sequence motifs match the widely assumed consensus sequences for Shine-Dalgarno sequence as well as the -10-box. Additionally, they were located upstream of most predicted protein coding sequences. Thus, their approximation can be assumed to be senseful. Additionally, an annotation file and `fasta` files are provided by the pipeline which contain the putative promoters, Shine-Dalgarno sequences and terminators. A summarization regarding the gene prediction is given in Figure 5.14. The whole results of this run can be found in the web at *http://www.bioinf.uni-leipzig.de/∼marcus/*.

Reference protein `blast`

Reference protein	Blast hits
30S ribosomal protein *S*17:	Alkaliphilus metalliredigens QYMF
30S ribosomal protein *S*4:	Clostridium difficile 630
⋮	⋮
50S ribosomal protein *L*23:	Clostridium difficile 630

The reference proteins were used to specify related species. In the majority of cases *Clostridium difficile 630* was assigned as most related. For one protein, *Alkaliphilus metalliredigens QYMF* gained the best hit.

Neighborhood



[] marks nearest species used for classification

Based on the previous results, the group of related species regarding the reference tree is determined. *Alkaliphilus metalliredigens QYMF* is within close range. The marked species will be used for taxonomic classification as well as for the initial annotation.

Taxonomic classification

cellular organisms, Bacteria, Firmicutes, Clostridia, Clostridiales, Clostridiaceae, Clostridium

The taxonomic assignment is based on one the maximum level of accordance regarding one or more related species. The classification is consistent to the `UniProt` lineage classification [168].

Derived features

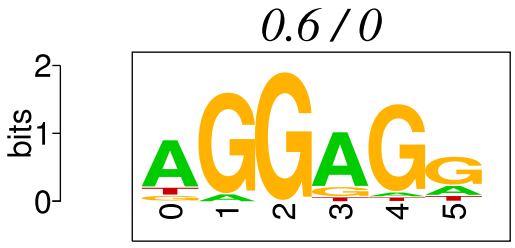
Direct annotated proteins	3084
Predicted ORFs	2145
Min. protein length	39 amino acids
Max. protein length	2710 amino acids

Based on the determined reference species, 3084 proteins could be annotated directly by homology search. The maximum length is used to remove unlikely long ORFs from the additional open reading frame predictions of `glimmer`. In total, 5229 putative genes were annotated.

	TTG	15.56%			
	ATT	0.19%		TAA	69.46%
Start-Codons:	CTG	0.03%	**Stop-Codons:**	TAG	25.23%
	ATG	71.69%		TGA	5.32%
	GTG	12.52%			

The derived start and stop codons from the directly annotated proteins were used to facilitate the `glimmer` predictions.

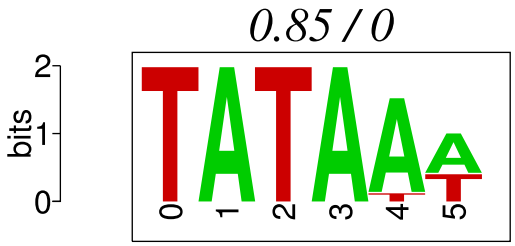
Estimation of Shine-Dalgarno profile



Motif found in 905 of 5229 sequences
Min./Avg./Max.-Distance to start-codon: 1/13.76/20 bp

Additionally, a putative Shine-Dalgarno sequence motif is derived from the directly annotated proteins and subsequently refined by adding the open reading frames. It is used to facilitate the `glimmer` predictions as well. In this case, it was located in front of about one fifth of all putative genes. This is reasonable as the SD sequence is not mandatory in front of all protein coding genes. Especially within operons, this motif is suspensible. Details were discussed in Subsection 5.1.2. However, the amount of sequences containing this motif is reasonable high. The average distance to the start codon was about 14 base pairs.

Estimation of promoter profile (−10-box)



Motif found in 4876 of 5228 sequences
Min./Avg./Max.-Distance to SD: 1/74.52/195 bp

The −10-box was determined from the upstream regions of directly annotated and predicted protein coding genes. It was located in the majority of sequences and shows a high conservation. The average distance to the Shine-Dalgarno sequence was 75 base pairs. One upstream sequence was not considered because the referring gene was to close to the beginning of the genome. The implementation does not comply with the circular nature of bacterial genomes at the moment. However, only a small amount of genes can be effected.

Terminator data

Estimated minimal score for `TransTermHP`: 73
Found: 914
Average score: 93.15

A reasonable score for terminators was gathered by using the estimation explained in Subsection 5.2.3. 914 of the 5229 putative protein coding genes had a terminator of this kind within the downstream region. Their average TransTermHP score was 93.

The pipeline was able to recover nearly all official annotations for protein coding sequences automatically. Figure 5.14 summarizes the annotations in comparison with the given data. Most genes were predicted identically, some differ in the assigned start codon. Due to the fact that the start codon has a dual function, this is reasonable. On the one hand, it acts as translation start introduced in Subsection 5.1.2. On the other, it codes for an amino acid as well. Furthermore, several different codons can act as start codon. Thus, it is likely to encounter a codon of this kind multiple times. This makes the determination of the exact translation start ambiguous. Only 53 coding sequences could not be determined. This includes 10 spliced genes. The pipeline is not designed to detect this type of coding sequences. In addition, overlapping regions are often missed. An example is presented in Figure 5.15a. However, it was possible to detect 166 additional coding sequences. In many cases possibly related transcriptional or translational elements were predicted that indicate the reliability of certain annotations. An example is presented in Figure 5.15b.

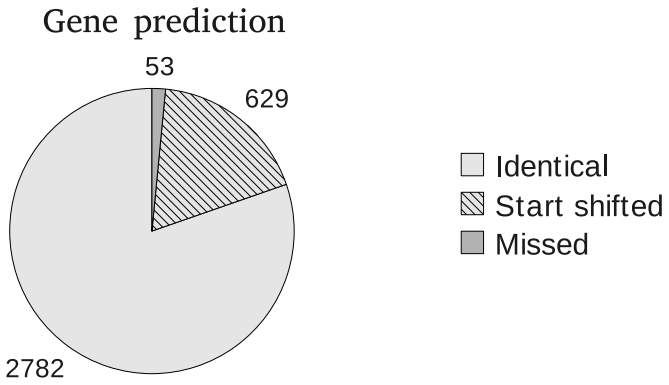


Figure 5.14: Comparison of annotation: The genome of *Clostridium difficile CD196* had 3464 putative protein coding sequences according to the enclosed annotation file. The predictions are consistent with the given annotation. The majority was annotated identically, less than a quarter with a different start codon. 53 genes were not found at all. However, the pipeline revealed 166 additional genes that do not match the given predictions.

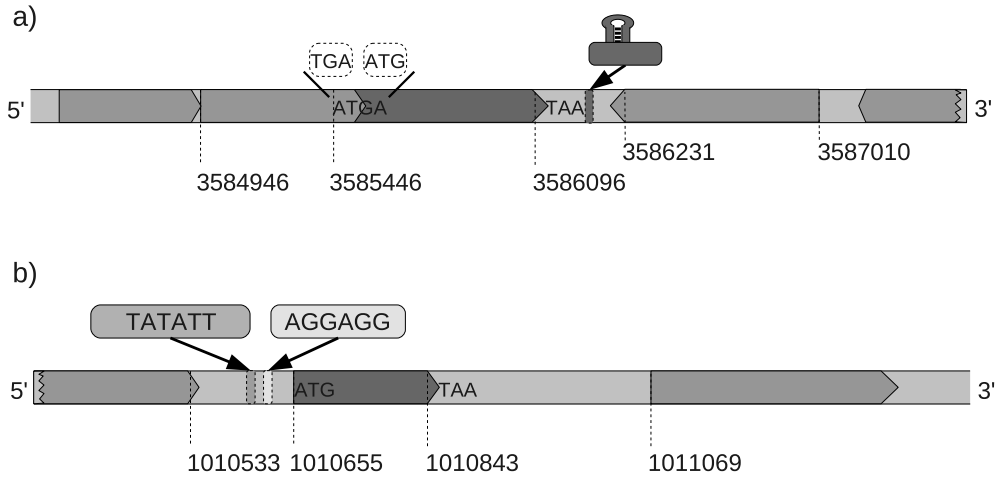


Figure 5.15: Annotation examples: a) Example for missed annotation: The dark gray coding sequence was included in the given annotation but was not detected by the pipeline. Neither promoter nor Shine-Dalgarno motif could be determined. The coding sequences convey the impression that coupled transcription occurs. This phenomenon was introduced in Subsection 5.1.2. Additionally, a high scoring terminator was detected between the missing gene and the upstream anti-sense gene. Thus, the annotation can be regarded as plausible. b) Example for additional annotation: The dark gray coding sequence was not included in the given annotation but located by the pipeline. Besides utilizing the most frequent start and stop codons (`ATG` and `TAA`) the presence of promoter and Shine-Dalgarno sequence indicates the plausibility of this additional prediction.

5.4 Results and discussion

Relatives discovery

The reference proteins and the derived tree, introduced in Chapter 4, were utilized in order to detect putative related species. Combined with the available representative set of 688 bacterial species, this allows to evaluate the assignment's quality. To achieve this, the relatives discovery part of the pipeline was applied to each species. Thereby, the species itself was removed from the reference tree as well as its proteins set from the reference proteins database in order to simulate a new species.

In the first step, best matching proteins for each reference orthoset are detected by `blast`. This can result in multiple hits as some may have equally good bitscores or are even identical in sequence. Figure 5.16 points out the results. About 75%

had 16 or less markers in the subtree. However, species which had many related subspecies within the references gained more hits. Some examples are *Salmonella enterica*, *Streptococcus pyogenes*, *Staphylococcus aureus* and *Escherichia coli*. Hence, their hits spread within this groups whereas species with less related subspecies accumulate hits in one or only a few related organisms.

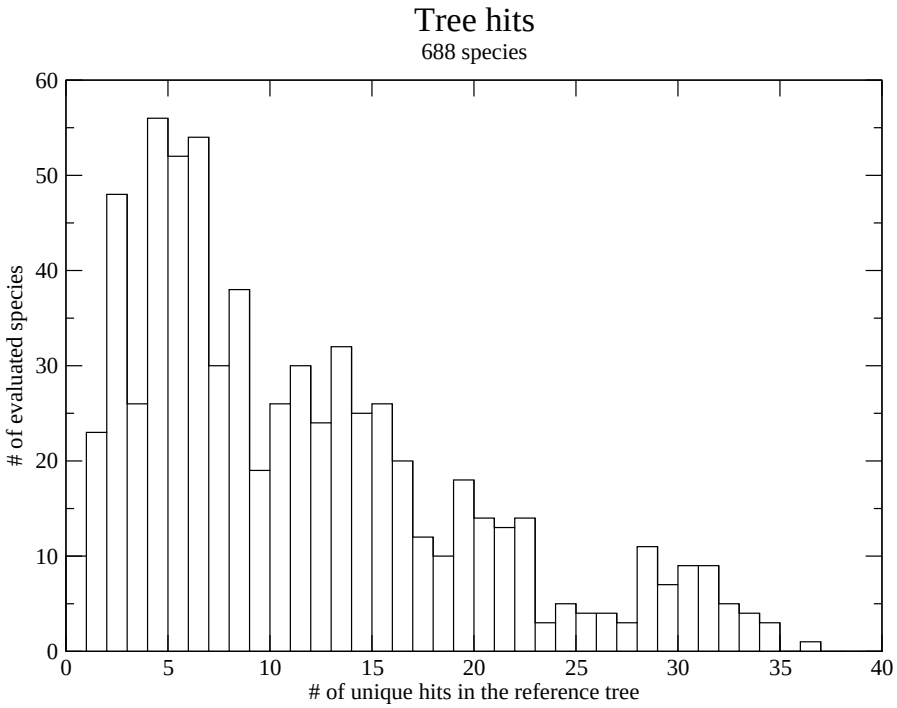


Figure 5.16: Hits in the reference tree: The relatives prediction step was applied to all 688 species within the reference tree. The number of derived markers within the reference tree was counted. Fewer hits indicate higher accuracy of the method.

Quality of discrimination Besides the number of related organisms within the tree, the degree of protein conservation within the database is important for the number of resulting hits. Proteins which are very conserved among the species are poor discriminators for relationships. For this purpose, rather different proteins are preferable. Table 5.1 shows results for the discrimination capabilities of the 27 reference proteins. Outstanding is the strong conservation for ribosomal proteins when

compared with the remaining common proteins. In order to make the discrimination more effective, the derived markers should not be treated equally. This can be achieved by attaching weight to unambiguous hits.

Number of detected relatives The second step is to gather a subtree of related species. Depending on the locations of the before derived markers, a group of putatively related species is determined. This group should be as close as possible to the original position of the investigated organism and be preferably small in order to retrieve a representative set of proteins. As Figure 5.17 shows, the majority of species result in a subtree of four or less species. This indicates that closely related species were found and the direct `blast` method would be applied to use their encoded proteins as seed for annotation. However, about 30% resulted in larger subtrees. This often occurs if the reference contains many related species. `Proteinortho` would be utilized in these cases. It can be argued that the method can take much time if the set of determined relatives is to large. Referring to the benchmark in Figure 3.7, this should be achievable in reasonable time up to a size of ten. Still about 18% remain with larger sets. Over 300 species were returned for some exceptions. However, this is not due to overrepresentation of related species. The contrary is the case. No closely related organisms are included for these species. As a result, their markers spread within the tree and return a rather large partition of it. Some examples are *uncultured Termite group 1 Bacterium phylotype RsD17*, *Thermodesulfovibrio yellowstonii DSM 11347*, *Pirellula sp.*, *Fusobacterium nucleatum*, *Bdellovibrio bacteriovorus* and *Magnetococcus MC-1*. These species are outgroups for large subtrees and cannot be assigned unambiguously. For both cases, to many related species within the tree and no closely related species, it is reasonable to randomly chose ten species for the `Proteinortho` step. If all species are related, a subset is representative as well. If on the other hand most species are distant from the analyzed organism, the number of in the majority shared proteins should not decrease significantly with a reduced number of species.

Accuracy of subtree assignment Besides the size of the subtree, it should contain the original species' position or be located next to it. To evaluate this, the internal nodes between the contained species and the original location were counted

Protein	Not distinguishable species
arginyl-tRNA synthetase	14.1%
seryl-tRNA synthetase	20.6%
phenylalanyl-tRNA synthetase alpha chain	23.1%
peptidase $M22$, O-sialoglycoprotein endopeptidase	23.8%
transcription elongation/termination factor $NusA$	24.3%
preprotein translocase, $SecY$ subunit	28.0%
30S ribosomal protein $S2$	29.8%
50S ribosomal protein $L1$	29.9%
50S ribosomal protein $L6$	29.9%
50S ribosomal protein $L3$	32.1%
50S ribosomal protein $L5$	32.8%
30S ribosomal protein $S4$	34.2%
30S ribosomal protein $S3$	34.8%
30S ribosomal protein $S8$	35.2%
50S ribosomal protein $L2$	34.5%
50S ribosomal protein $L23$	36.2%
30S ribosomal protein $S7$	36.3%
30S ribosomal protein $S5$	36.3%
50S ribosomal protein $L11$	36.7%
50S ribosomal protein $L22$	37.4%
30S ribosomal protein $S17$	37.7%
30S ribosomal protein $S13$	38.4%
30S ribosomal protein $S11$	40.7%
30S ribosomal protein $S12$	44.0%
30S ribosomal protein $S19$	44.5%
50S ribosomal protein $L14$	45.2%
30S ribosomal protein $S10$	53.2%

Table 5.1: Discrimination quality: The reference proteins are differently conserved among the species and thus have different discrimination properties. $S10$ for example was not able to report a unique hit for over half of all investigated species while the arginyl-tRNA synthetase gave inconclusive hits for only 14.1% of the 688 species.

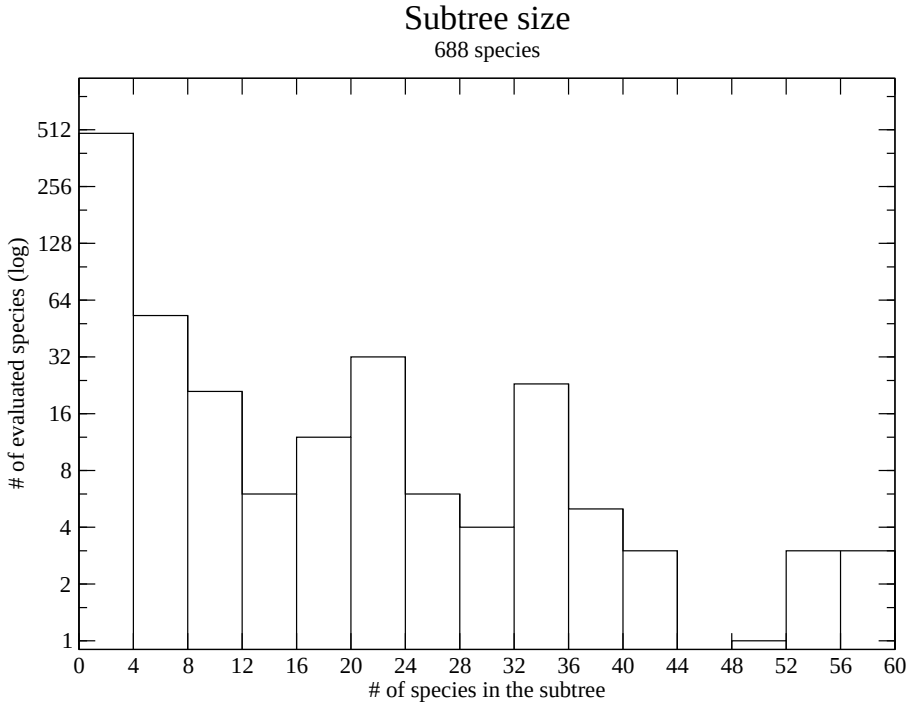


Figure 5.17: Subtree size: Based on the markers from `blast`, subtrees were determined which should include the most related species within the reference tree. The bar chart points out the number of species within these subtrees. Smaller sets are preferable. This is the case in the majority. However, in particular cases the size even became considerably larger than 60 (data not shown).

and averaged. Figure 5.18 shows the results. The majority of species resulted in a small distance of less than eight inner nodes. Larger distances basically result from larger subtrees as discussed above. Overall, no unexpectedly large distances were observed.

Prediction quality

During tests with several species the annotation pipeline worked well. Annotations of protein coding genes were nearly complete regarding the available data from `NCBI`. Especially, overlapping and very short coding regions were often missed. However, the filtering of manifold predicted areas is not very sophisticated and can be im-

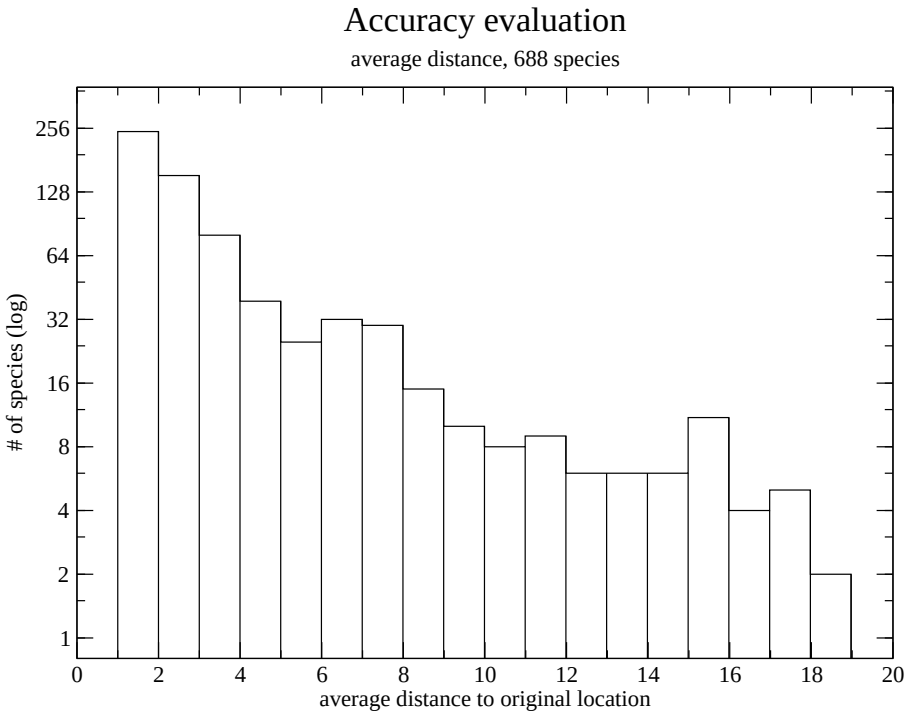


Figure 5.18: Subtree distances: Average distance between the located subtree and the species' original location. Measure were the nodes between the subtree leafs and the original location of the investigated species. Closer is better.

proved. The motifs estimated for Shine-Dalgarno sequence and −10-box largely differ in plausibility and conservation. In *Escherichia coli K12 substr. DH10B* for example the −10-box was reported to have the putative consensus sequence `GATAAA` which is obviously wrong. The reason is easily traceable. Between coding region and −10-box are often stretches of `A`. These mislead the expectation maximization approach. Manually removing them results in properly annotated motifs. However, the method is fully automated and does not claim to replace individual prediction approaches. Results have to be regarded as quick overview which can aid further investigations.

Conclusion

The annotation pipeline allows achieving a quick and compact overview to newly sequenced bacterial genomes. This includes a preliminary taxonomic classification based on species with similar reference proteins, annotation of protein coding genes and conservation of transcriptional and translational elements (namely promoter, Shine-Dalgarno sequence and terminator). However, derived sequence motifs are not satisfying in every case. A special benefit is the improved accuracy (in comparison to the gene prediction method `glimmer3` as stand-alone program) with respect to genomic features such as typical start and stop codons. Their relative frequencies are estimated from the initial prediction based on homologous genes in related species. The information allows preferring putative open reading frames with typical start and stop codons. In this way, predictions are enhanced especially for 'exotic' genomes where these features differ largely from model organisms. Furthermore, it provides putative functional annotation for many protein coding genes which is reasonable as it is based on the conserved counterparts within related genomes. The method is fully automated and capable to handle even partially sequenced genomes. Normally, results are returned in less than 120 minutes. Small genomes can be finished considerably faster. The tool was written to benefit from multiple CPU cores by using treads. Thus, it scales well with their number.

Chapter 6

Conclusion and outlook

The tool for orthology prediction can be used for multiple types of analysis in small ad-hoc as well as in large-scale projects. However, the amount of proteins which are clustered in overstated large groups is not satisfactory. This problem will be addressed in the future, for instance by additional filtering steps. This would probably yield more proteins for the investigation of domain-wide commons. Possible approaches have been discussed in Section 3.4.

An application for comparative operon detection is planed. These units can help to identify the function of genes and thus, are a desirable part of genome annotation [169]. By now, this detection is achieved by comparing positions of anciently related genes in related species [135, 170]. A conserved arrangement indicates an operon unit. However, the definition of related genes between species is done by `blast` or by concluding homologous groups based on their textual annotation. Both approaches are reasonable but error-prone with respect to missed groups and overestimated relationships. For this purpose orthosets reported by `Proteinortho` should represent a more reliable basis. Figure 6.1 presents a simplified sketch. Furthermore, an integration to the annotation pipeline is imaginable. It already makes use of conserved groups within related species for an initial annotation of putative proteins. This approach can be extended to conserved locations as well.

Additionally, the concept of orthology could be transferred to non-coding RNAs as well. Whereas it is difficult to assume appropriate relationships from sequence data of these genes themselves, it is possible to use assignments from flanking proteins for this purpose. In this case, orthology can refer RNAs which are consistently located next to a protein of a certain orthoset through multiple species. It is reasonable to assume related functions from this correlation. Thus, RNA relationships can derived from protein relationships and synteny information.

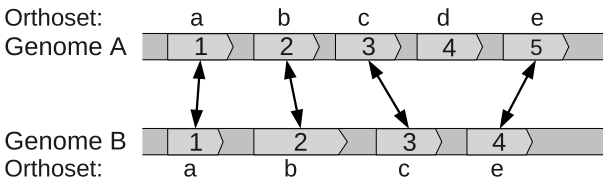


Figure 6.1: Simplified operon prediction approach: The order of all protein coding genes in two or more genomes is determined first. Based on orthology predictions from `Proteinortho` and their genomic locations, conserved groups can be estimated. In this example, genes of the orthosets a, b, c and e are located in a conserved order. Even if d is not present in species B, a putative operon unit can be derived [135, 170].

Clusters of interesting proteins and non-coding RNAs could be analyzed in more detail. Especially, orthologs provide useful information on the rate of evolution within different lineages. Gene trees can be derived which may improve phylogenetic classification for sophisticated situations. The information of orthologous and paralogous genes can be used to reduce the amount of data in interaction partner analysis for target prediction as well. Assume the unknown interaction partner of a protein a is of interest which also interacts in some other organisms in the same way. Normally, all proteins of the certain species would come into consideration. `Proteinortho` on the other hand, allows to detect consistent genes within all species where homologs of protein a are known likewise within short time. Even though these species are not known, the tool is able to detect candidates itself as these are orthologous to a. Finally, the resulting set of common proteins is reasonably reduced. More complex subsequent analysis are enhanced as they can operate on a smaller set. The same method would be feasible for encoded RNAs if the previously mentioned orthology approach on non-coding RNAs is implemented.

Domain-wide common proteins were searched within 710 fully sequenced species in Chapter 4. Figure 4.2 revealed that a large amount of common proteins was not annotated within many species. The number of proteins occurring in at least 50% of all analyzed species could be doubled by reblasting them against species where they were not found beforehand. They were missed in many 'old' genome annotations. Thus, a replenishment based on common proteins or even using the annotation pipeline's homology based approach to initially predict proteins, would gain additional candidates for genes with high confidence. In the set of 710 species,

one third of all protein coding genes were marked as 'putative' or 'hypothetical' already. Hence, a semi-automated replenishment could improve the predicted annotations without much effort while it would not significantly increase the amount of putative genes.

The annotation pipeline reveals a far-reaching application of `Proteinortho`. However, it still offers many capabilities. Besides the -10, the -35-box could be considered as well. The -35 box is located in the region $18-26$ bp upstream of the -10-box. For this purpose the expectation maximization approach which is used to estimate the -10 sequence motif, can be applied once more to this region. Additionally, promoters from different σ-factors could be considered. In Subsection 5.2.3 upstream regions of putative protein coding genes are used to find promoters which match the most utilized σ-factor. A possible approach is to remove all sequences where a putative promoter was detected and repeat the expectation maximization approach. In this way, sequence motifs for less frequent but conserved elements could be detected in addition.

In order to enhance the predictions further, putative promoters and terminators could be used as additional descriptors for protein coding genes. By now, this is done for the estimated Shine-Dalgarno sequence motif solely. However, open reading frames which are lead by a promoter or followed by a terminator are likely candidates for additional protein coding genes as well. Likewise, this should aid in detection of very small proteins. The problems with overlapping proteins should be solvable by applying a more sophisticated filter for overlapping predictions. On the other hand, the presence of promoter and terminator could indicate the existence of non-coding RNAs in between. These sequences represent good candidates for subsequent prediction approaches for genes of this kind. An example approach is presented in literature [171].

Furthermore, the reliability of protein coding gene predictions can be estimated by using a scoring schema based on the presence of certain properties. Genes which are surrounded by promoter, Shine-Dalgarno sequence and terminator would gain the highest score while genes without any of this features would receive the lowest score. This can be refined by annotating putative operon structures. Therefore, a promoter in front of the first operon gene may be sufficient to increase the score of the contained genes. The same holds for terminator and Shine-Dalgarno sequence.

Moreover, this score can be refined by the estimated quality of the predicted sequence motifs. As revealed in Section 5.4, the expectation maximization approach which is used for promoter and Shine-Dalgarno estimation, can easily be mislead by additional conserved sequences or a fuzzy dataset. In this case the estimated sequence motifs can become misguided. However, their quality can be approximated like it is done for quality of terminator scores. By comparing the frequency of their presence within shuffled genomes and the real genome, an estimation of their relevance can be achieved. Conserved motifs should be present more often within the real genome than in the shuffled versions. Thus, the quotient $\frac{\#\text{ of matches in real genome}}{\#\text{ of matches in shuffled genome}}$ can be used as automatically retrievable information for the quality of sequence motif estimation. Summing up, a scoring function of this kind would allow to distinguish reliable from questionable gene predictions. However, the scores would highly depend on the certain organisms and thence, not be comparable over different species.

Another logical step would be to annotate 5'- and 3'-UTRs (untranslated regions). This is feasible as the position of the coding sequence as well as putative promoter and terminator is known for many genes. Even though this approximations could be inaccurate if the estimation of sequence motifs was mislead, it still yields reasonable candidates for target prediction approaches and detection of binding sites on RNA level.

Appendix A

Manuals

A.1 Proteinortho

Name

proteinortho - Orthologous proteins finder

Syntax

proteinortho.pl [*OPTION*]... <*FILES*>... >*OUTPUT*

proteinortho.pl [*OPTION*]... <*FILE*> >*OUTPUT*

Description

This program finds orthologous proteins within different species.

It can either be started giving the intended files in fasta-format (at least two) after the *OPTIONS* or just one file containing the paths to these. This is especially useful if their number grows. Each file should represent all proteins (or the part of it that should be investigated) of one species.

In a first step all files are blasted against each other. The hits will be evaluated according to the given *OPTIONS* and transformed into a graph, where each protein is represented by a node. This graph will be fragmented into its connected components, thus proteins which are connected to each other.

`Proteinortho` was designed to deal with large data sets and also behave nicely regarding the memory consumption.

To have an example: Investigating about 700 bacterial species took two weeks using 50 CPU-cores, 300 GB hard disk space and less than 2.5 GB of RAM per

contributing workstation. Small sets are done within minutes. Only megabytes of hard disk and RAM are needed.

Important: Protein ids must be globally different! You should also consider that blast may cut the ids on a whitespace using the first part only.

Output Format

The *OUTPUT* is a tab separated matrix.

First line starts with # followed by the file names. Second line starts with # followed by the corresponding number of proteins in the files.

From here each line represents a connected component and therefore the ids of determined orthologous proteins.

Options

-e=<*E-VALUE*> <*E-VALUE*> for blasts

[default: 1e-10]

-a=<*THREADS*> number of <*THREADS*> to make use of dual- and multi-core CPUs

[default: 1]

-p=*blastp*|*blastn*; defines the blast program

[default: blastp]

-r=*0*|*1*; enables or disables reciprocal the blast condition

[default: 1 (enabled)]

-m=*(0..1)*; minimum similarity of best blast hits allowed are doubles within the interval (0..1) all hits with $(bestscore + new)/bestscore - 1 < m$ are included 0 takes all hits into account, 1 only the best (maybe more with equal bitscore) useful to handle paralogous better

[default: 0.95 (nearly equal)]

-selfblast applies an additional blast for every species against itself
this may increase the detection of paralogs, but is normally not necessary a similar hits are found if **-m** is not set to 1

-f force blastall (even if blast output is found)

-ff force formatdb (even if databases are found)

-remove removes blast outputs after use

-verbose gives information about what happens, including a progress report and a lasting time approximation

-dir[=<*DIRECTORY*>] defines the <*DIRECTORY*> for the blast outputs

[default directory: working directory]

-cmat includes putative paralogous proteins to the output
sets which contain such proteins are not reported otherwise

paralogous protein ids are separated by ","

[default file: cc.matrix]

-debug keeps temporary files for debugging

-plog[=<*FILE*>] logfile for pairwise blast hits

[default file: pb.log]

-plog[=<*FILE*>] logfile for connected components

[default file: cc.log]

-ulog[=<*FILE*>] ultimate logfile, this is actually a post-process of plog and clog

the creation is very time intensive and not recommended if more than 10,000 proteins are involved

[default file: ultimate.log]

Multiple Machine Options

The main part of `Proteinortho` consists of blasting each species against each other. This can take several hours up to days if hundreds of species are involved - even on multi-core machines. For this purpose a mechanism has been implemented which allows to distribute that workload over multiple machines.

Every option aside from **-a=**<*THREADS*> needs to be the same. This is especially important for the directory in which the blasts are stored. A file named **sync** will be created their and used to synchronize the processes. As flock is not capable for network file systems a temporary directory named **lock/** is used for locking. Both may need to be removed if `Proteinortho` was interrupted or crashed and a restart is intended.

Run all scripts using the option

-blastonly

As the scripts synchronize themselves the order or time you start it on different machines does not matter. You can even stop certain processes if needed. See *SIGNALS* for more details to that topic. After the blasts are done, all started scripts will be terminated.

If that happened, you can grab the results and finish the calculations. Start the script again on one machine using the same options as before. Instead of **-blastonly** use the option

-blastdone

this will lead to skip database creation and blasts and thus speed up the beginning of the connected component calculation.

Signals

Sending signal **INT** or **TERM** to a `Proteinortho` process will lead to a clean stop which allows a later continuation at this point. If used on **MULTIPLE MACHINES** this allows to stop certain processes without interference with the on going calculation. As going blast jobs need to be finished first, the termination may take a while.

However, sending the signal twice (or using **KILL**) will lead to an immediate stop and may result in corrupted data. It is advisable to remove all files from the blast out directory and not use the data any further. This is also the case if the the blasts were distributed over multiple machines.

Furthermore, if a full stop of all processes on **MULTIPLE MACHINES** is intended, a file named stop can be placed in the blast out directory. This will lead to clean stop as described above for all running scripts.

Examples

To run this program the standard way comparing two or more species type:

proteinortho.pl speciesA.faa speciesB.faa >orthologs.out

If you want to define the number of threads, have live progress report and store blast files in a separate folder, type:

mkdir blastout/ proteinortho.pl -a=4 -verbose -dir=blastout/ files.list >orthologs.out

Copyright

Reporting Bugs

Marcus Lechner <marcus@bioinf.uni-leipzig.de>

Authors

Written by Marcus Lechner, Lydia Steiner and Sonja J. Prohaska

Bioinformatics Group, Department of Computer Science, and Interdisciplinary Center for Bioinformatics, University of Leipzig

See Also

orthomatrix2tree.pl a tool which allows to generate trees based on the shared proteins

A.2 Treebuilder

Name

orthomatrix2tree - treebuilder based on shared proteins

Syntax

orthomatrix2tree.pl *ORTHOMATRIX* > *OUTTREE*

Description

This program generates pseudo phylogenetic trees based on shared proteins.

It requires an output file from `Proteinortho`. Clustering works like `UPGMA`. However, a min-operation instead of average for scoring of clusters is used. In this way, relations of similar characteristics, habitats, phylogeny and, as the case may be, horizontal gene transfer are reflected. It allows to view species relationship from the perspective of shared proteins and might give an insights to evolutionary regards.

Important: Protein ids must be globally different! You should also consider that blast may cut the ids on a whitespace using the first part only.

Output Format

OUTTREE is a tree in Newick format. Lengths represent the number of shared proteins within the group. Comments show the number of additional proteins with respect to the superior group.

Comments

The program is optimized for `SSE2` (Streaming SIMD Extensions 2) capable CPUs. Even large calculation should take less than two minutes on these machines.

Examples

To run this program type: orthomatrix2tree groupA.mat >groupA.tree

Copyright

Reporting Bugs

Marcus Lechner <marcus@bioinf.uni-leipzig.de>

Authors

Written by Marcus Lechner, Lydia Steiner

Bioinformatics Group, Department of Computer Science, and Interdisciplinary Center for Bioinformatics, University of Leipzig

Appendix B

Data

Genomes

Species	Accession number(s)	Last updated
Acaryochloris marina MBIC11017	NC_009925	2007/11/20
Acholeplasma laidlawii PG 8A	NC_010163	2008/03/18
Acidiphilium cryptum JF-5	NC_009484	2007/05/23
Acidithiobacillus ferrooxidans ATCC 53993	NC_011206	2008/09/08
Acidobacteria bacterium Ellin345	NC_008009	2007/01/23
Acidothermus cellulolyticus 11B	NC_008578	2007/01/23
Acidovorax avenae citrulli AAC00-1	NC_008752	2007/01/05
Acidovorax JS42	NC_008782	2007/01/11
Acinetobacter baumannii AB0057	NC_011586	2008/11/18
Acinetobacter baumannii ACICU	NC_010611	2008/06/11
Acinetobacter baumannii ATCC 17978	NC_009085	2007/03/07
Acinetobacter baumannii AYE	NC_010410	2008/03/19
Acinetobacter baumannii SDF	NC_010400	2008/03/14
Acinetobacter sp ADP1	NC_005966	2007/01/23
Actinobacillus pleuropneumoniae L20	NC_009053	2007/02/26
Actinobacillus pleuropneumoniae serovar 3 JL03	NC_010278	2008/01/24
Actinobacillus pleuropneumoniae serovar 7 AP76	NC_010939	2008/06/13
Actinobacillus succinogenes 130Z	NC_009655	2007/07/25
Aeromonas hydrophila ATCC 7966	NC_008570	2007/01/23
Aeromonas salmonicida A449	NC_009348	2007/04/18
Aeropyrum pernix	NC_000854	2007/01/23
Akkermansia muciniphila ATCC BAA 835	NC_010655	2008/07/27
Alcanivorax borkumensis SK2	NC_008260	2008/01/07
Aliivibrio salmonicida LFI1238	NC_011312, NC_011313	2008/10/21
Alkalilimnicola ehrlichei MLHE-1	NC_008340	2007/01/23
Alkaliphilus metalliredigens QYMF	NC_009633	2007/07/03
Alkaliphilus oremlandii OhILAs	NC_009922	2007/10/17
Alteromonas macleodii Deep ecotype	NC_011138	2008/08/15
Anabaena variabilis ATCC 29413	NC_007413	2007/01/23
Anaeromyxobacter dehalogenans 2CP-C	NC_007760	2007/01/23
Anaeromyxobacter Fw109-5	NC_009675	2007/07/25
Anaeromyxobacter K	NC_011145	2008/08/23
Anaplasma marginale St Maries	NC_004842	2007/04/25
Anaplasma phagocytophilum HZ	NC_007797	2007/01/23
Anoxybacillus flavithermus WK1	NC_011567	2008/11/14
Aquifex aeolicus	NC_000918	2005/12/04
Archaeoglobus fulgidus	NC_000917	2007/01/23
Arcobacter butzleri RM4018	NC_009850	2007/09/29
Arthrobacter aurescens TC1	NC_008711	2006/12/28
Aster yellows witches-broom phytoplasma AYWB	NC_007716	2006/01/19
Azoarcus BH72	NC_008702	2008/01/07
Azoarcus sp EbN1	NC_006513	2007/01/23
Azorhizobium caulinodans ORS 571	NC_009937	2007/10/19
Bacillus amyloliquefaciens FZB42	NC_009725	2007/08/04
Bacillus anthracis Ames 0581	NC_007530	2007/01/23
Bacillus anthracis Ames	NC_003997	2007/01/23
Bacillus anthracis str Sterne	NC_005945	2005/12/04
Bacillus cereus ATCC 10987	NC_003909	2007/01/23

Bacillus cereus ATCC14579	NC_004722	2005/12/04
Bacillus cereus cytotoxis NVH 391-98	NC_009674	2007/07/25
Bacillus cereus ZK	NC_006274	2007/01/23
Bacillus clausii KSM-K16	NC_006582	2007/01/23
Bacillus halodurans	NC_002570	2007/01/23
Bacillus licheniformis ATCC 14580	NC_006270	2007/12/26
Bacillus licheniformis DSM 13	NC_006322	2007/12/26
Bacillus pumilus SAFR-032	NC_009848	2007/09/27
Bacillus subtilis	NC_000964	2008/02/19
Bacillus thuringiensis Al Hakam	NC_008600	2007/01/24
Bacillus thuringiensis konkukian	NC_005957	2005/12/04
Bacillus weihenstephanensis KBAB4	NC_010184	2008/03/18
Bacteroides fragilis NCTC 9434	NC_003228	2007/01/23
Bacteroides fragilis YCH46	NC_006347	2005/12/04
Bacteroides thetaiotaomicron VPI-5482	NC_004663	2005/12/04
Bacteroides vulgatus ATCC 8482	NC_009614	2007/06/29
Bartonella bacilliformis KC583	NC_008783	2007/01/11
Bartonella henselae Houston-1	NC_005956	2005/12/04
Bartonella quintana Toulouse	NC_005955	2005/12/04
Bartonella tribocorum CIP 105476	NC_010161	2007/12/26
Baumannia cicadellinicola Homalodisca coagulata	NC_007984	2006/05/08
Bdellovibrio bacteriovorus	NC_005363	2007/01/23
Beijerinckia indica ATCC 9039	NC_010581	2008/04/12
Bifidobacterium adolescentis ATCC 15703	NC_008618	2006/12/08
Bifidobacterium longum DJO10A	NC_010816	2008/06/06
Bifidobacterium longum infantis ATCC 15697	NC_011593	2008/11/22
Bifidobacterium longum	NC_004307	2007/01/23
Bordetella avium 197N	NC_010645	2008/05/07
Bordetella bronchiseptica	NC_002927	2007/01/23
Bordetella parapertussis	NC_002928	2007/01/23
Bordetella pertussis	NC_002929	2007/01/23
Bordetella petrii	NC_010170	2008/02/08
Borrelia afzelii PKo	NC_008277	2007/01/24
Borrelia burgdorferi	NC_001318	2007/06/13
Borrelia duttonii Ly	NC_011229	2008/09/18
Borrelia hermsii DAH	NC_010673	2008/09/11
Borrelia recurrentis A1	NC_011244	2008/09/18
Borrelia turicatae 91E135	NC_008710	2008/07/25
Bradyrhizobium BTAi1	NC_009485	2007/05/23
Bradyrhizobium japonicum	NC_004463	2005/12/04
Bradyrhizobium ORS278	NC_009445	2008/01/07
Buchnera aphidicola Cc Cinara cedri	NC_008513	2007/01/23
Buchnera aphidicola	NC_004545	2005/12/04
Buchnera aphidicola Sg	NC_004061	2005/12/04
Buchnera sp	NC_002528	2007/01/23
Burkholderia cenocepacia J2315	NC_011000, NC_011001, NC_011002	2008/09/24
Burkholderia xenovorans LB400	NC_007952, NC_007953	2008/09/10
Caldicellulosiruptor saccharolyticus DSM 8903	NC_009437	2007/05/08
Caldivirga maquilingensis IC-167	NC_009954	2007/11/08
Campylobacter concisus 13826	NC_009802	2007/09/14
Campylobacter curvus 525 92	NC_009715	2007/07/31
Campylobacter fetus 82-40	NC_008599	2007/01/24
Campylobacter hominis ATCC BAA-381	NC_009714	2007/07/31
Campylobacter jejuni 81116	NC_009839	2007/09/22
Campylobacter jejuni 81-176	NC_008787	2007/01/11
Campylobacter jejuni doylei 269 97	NC_009707	2007/07/27
Campylobacter jejuni	NC_002163	2005/12/04
Campylobacter jejuni RM1221	NC_003912	2007/01/23
Candidatus Amoebophilus asiaticus 5a2	NC_010830	2008/07/15
Candidatus Azobacteroides pseudotrichonymphae genomovar CFP2	NC_011565	2008/11/15
Candidatus Blochmannia floridanus	NC_005061	2005/12/04
Candidatus Blochmannia pennsylvanicus BPEN	NC_007292	2007/01/23
Candidatus Carsonella ruddii PV	NC_008512	2007/01/23
Candidatus Desulfococcus oleovorans Hxd3	NC_009943	2007/10/23
Candidatus Desulforudis audaxviator MP104C	NC_010424	2008/03/18
Candidatus Korarchaeum cryptofilum OPF8	NC_010482	2008/03/19

Candidatus Methanoregula boonei 6A8	NC_009712	2007/07/31
Candidatus Pelagibacter ubique HTCC1062	NC_007205	2005/12/04
Candidatus Phytoplasma australiense	NC_010544	2008/08/27
Candidatus Phytoplasma mali	NC_011047	2008/07/15
Candidatus Ruthia magnifica Cm Calyptogena magnifica	NC_008610	2007/01/23
Candidatus Sulcia muelleri GWSS	NC_010118	2007/12/10
Candidatus Vesicomyosocius okutanii HA	NC_009465	2007/05/23
Carboxydothermus hydrogenoformans Z-2901	NC_007503	2007/01/23
Caulobacter crescentus	NC_002696	2005/12/04
Caulobacter K31	NC_010338	2008/03/18
Cellvibrio japonicus Ueda107	NC_010995	2008/06/24
Chlamydia muridarum	NC_002620	2005/12/04
Chlamydia trachomatis 434 Bu	NC_010287	2008/01/26
Chlamydia trachomatis A HAR-13	NC_007429	2007/01/23
Chlamydia trachomatis L2b UCH 1 proctitis	NC_010280	2008/01/26
Chlamydia trachomatis	NC_000117	2007/01/23
Chlamydophila abortus S26 3	NC_004552	2007/01/23
Chlamydophila caviae	NC_003361	2005/12/04
Chlamydophila felis Fe C-56	NC_007899	2006/03/16
Chlamydophila pneumoniae AR39	NC_002179	2005/12/04
Chlamydophila pneumoniae CWL029	NC_000922	2007/01/23
Chlamydophila pneumoniae J138	NC_002491	2005/12/04
Chlamydophila pneumoniae TW 183	NC_005043	2005/12/04
Chlorobaculum parvum NCIB 8327	NC_011027	2008/07/01
Chlorobium chlorochromatii CaD3	NC_007514	2007/04/25
Chlorobium limicola DSM 245	NC_010803	2008/09/11
Chlorobium phaeobacteroides BS1	NC_010831	2008/06/10
Chlorobium phaeobacteroides DSM 266	NC_008639	2007/07/31
Chlorobium tepidum TLS	NC_002932	2005/12/04
Chloroflexus aurantiacus J 10 fl	NC_010175	2008/03/18
Chloroherpeton thalassium ATCC 35110	NC_011026	2008/07/01
Chromobacterium violaceum	NC_005085	2005/12/04
Chromohalobacter salexigens DSM 3043	NC_007963	2007/01/23
Citrobacter koseri ATCC BAA-895	NC_009792	2007/09/14
Clavibacter michiganensis NCPPB 382	NC_009480	2007/05/23
Clavibacter michiganensis sepedonicus	NC_010407	2008/03/26
Clostridium acetobutylicum	NC_003030	2007/01/23
Clostridium beijerinckii NCIMB 8052	NC_009617	2007/06/29
Clostridium botulinum A3 Loch Maree	NC_010520	2008/03/24
Clostridium botulinum A ATCC 19397	NC_009697	2007/07/27
Clostridium botulinum A Hall	NC_009698	2007/07/27
Clostridium botulinum A	NC_009495	2008/01/07
Clostridium botulinum B1 Okra	NC_010516	2008/03/24
Clostridium botulinum B Eklund 17B	NC_010674	2008/05/10
Clostridium botulinum E3 Alaska E43	NC_010723	2008/09/11
Clostridium botulinum F Langeland	NC_009699	2007/07/27
Clostridium difficile 630	NC_009089	2008/01/07
Clostridium kluyveri DSM 555	NC_009706	2007/07/27
Clostridium novyi NT	NC_008593	2007/01/24
Clostridium perfringens ATCC 13124	NC_008261	2007/01/23
Clostridium perfringens	NC_003366	2005/12/04
Clostridium perfringens SM101	NC_008262, NC_008265	2007/04/25
Clostridium phytofermentans ISDg	NC_010001	2008/03/18
Clostridium tetani E88	NC_004557	2005/12/04
Clostridium thermocellum ATCC 27405	NC_009012	2007/02/17
Colwellia psychrerythraea 34H	NC_003910	2007/01/23
Coprothermobacter proteolyticus DSM 5265	NC_011295	2008/09/27
Corynebacterium diphtheriae	NC_002935	2007/01/23
Corynebacterium efficiens YS-314	NC_004369	2005/12/04
Corynebacterium glutamicum ATCC 13032 Bielefeld	NC_006958	2007/04/30
Corynebacterium glutamicum ATCC 13032 Kitasato	NC_003450	2007/01/23
Corynebacterium glutamicum R	NC_009342	2007/04/18
Corynebacterium jeikeium K411	NC_007164	2007/01/23
Corynebacterium urealyticum DSM 7109	NC_010545	2008/04/04
Coxiella burnetii CbuG Q212	NC_011527	2008/11/07
Coxiella burnetii CbuK Q154	NC_011528	2008/11/07
Coxiella burnetii Dugway 7E9-12	NC_009727	2007/12/14

Coxiella burnetii	NC_002971	2007/01/23
Coxiella burnetii RSA 331	NC_010117	2008/03/18
Cupriavidus taiwanensis	NC_010528, NC_010530	2008/07/17
Cyanobacteria bacterium Yellowstone A-Prime	NC_007775	2006/03/23
Cyanobacteria bacterium Yellowstone B-Prime	NC_007776	2006/03/23
Cytophaga hutchinsonii ATCC 33406	NC_008255	2006/08/30
Dechloromonas aromatica RCB	NC_007298	2007/01/23
Dehalococcoides BAV1	NC_009455	2007/05/18
Dehalococcoides CBDB1	NC_007356	2005/12/04
Dehalococcoides ethenogenes 195	NC_002936	2007/01/23
Deinococcus geothermalis DSM 11300	NC_008025	2006/05/09
Delftia acidovorans SPH-1	NC_010002	2008/03/18
Desulfitobacterium hafniense Y51	NC_007907	2006/03/16
Desulfotalea psychrophila LSv54	NC_006138	2005/12/04
Desulfotomaculum reducens MI-1	NC_009253	2007/03/30
Desulfovibrio desulfuricans G20	NC_007519	2007/01/23
Desulfovibrio vulgaris DP4	NC_008751	2007/01/05
Desulfovibrio vulgaris Hildenborough	NC_002937	2007/01/23
Dichelobacter nodosus VCS1703A	NC_009446	2007/05/09
Dictyoglomus thermophilum H 6 12	NC_011297	2008/09/27
Dinoroseobacter shibae DFL 12	NC_009952	2007/11/08
Ehrlichia canis Jake	NC_007354	2005/12/04
Ehrlichia chaffeensis Arkansas	NC_007799	2007/01/23
Ehrlichia ruminantium Gardel	NC_006831	2005/12/04
Ehrlichia ruminantium str. Welgevonden CIRAD	NC_006832	2005/12/04
Ehrlichia ruminantium Welgevonden UPSA	NC_005295	2007/01/23
Elusimicrobium minutum Pei191	NC_010644	2008/07/30
Enterobacter 638	NC_009436	2007/05/08
Enterobacter sakazakii ATCC BAA-894	NC_009778	2007/09/07
Enterococcus faecalis V583	NC_004668	2007/01/23
Erwinia carotovora atroseptica SCRI1043	NC_004547	2007/01/23
Erwinia tasmaniensis	NC_010694	2008/11/20
Erythrobacter litoralis HTCC2594	NC_007722	2006/01/20
Escherichia coli 536	NC_008253	2006/07/24
Escherichia coli APEC O1	NC_008563	2007/01/24
Escherichia coli C ATCC 8739	NC_010468	2008/05/09
Escherichia coli CFT073	NC_004431	2007/01/23
Escherichia coli E24377A	NC_009801	2007/09/14
Escherichia coli HS	NC_009800	2007/09/14
Escherichia coli K 12 substr DH10B	NC_010473	2008/04/22
Escherichia coli K12 substr MG1655	NC_000913	2008/05/19
Escherichia coli O157 H7 EC4115	NC_011353	2008/10/11
Escherichia coli O157H7 EDL933	NC_002655	2007/01/23
Escherichia coli O157H7	NC_002695	2007/01/23
Escherichia coli SE11	NC_011415	2008/10/24
Escherichia coli SMS 3 5	NC_010498	2008/07/30
Escherichia coli UTI89	NC_007946	2007/01/23
Escherichia coli W3110	AC_000091	2006/03/02
Exiguobacterium sibiricum 255 15	NC_010556	2008/04/06
Fervidobacterium nodosum Rt17-B1	NC_009718	2007/07/31
Finegoldia magna ATCC 29328	NC_010376	2008/03/14
Flavobacterium johnsoniae UW101	NC_009441	2007/05/08
Flavobacterium psychrophilum JIP02 86	NC_009613	2008/01/07
Francisella philomiragia ATCC 25017	NC_010336	2008/02/13
Francisella tularensis FSC 198	NC_008245	2008/02/07
Francisella tularensis holarctica FTA	NC_009749	2008/04/17
Francisella tularensis holarctica	NC_007880	2008/01/07
Francisella tularensis holarctica OSU18	NC_008369	2007/01/23
Francisella tularensis mediasiatica FSC147	NC_010677	2008/05/10
Francisella tularensis novicida U112	NC_008601	2007/01/23
Francisella tularensis tularensis	NC_006570	2007/01/23
Francisella tularensis WY96-3418	NC_009257	2007/03/30
Frankia alni ACN14a	NC_008278	2008/01/07
Frankia CcI3	NC_007777	2007/01/23
Frankia EAN1pec	NC_009921	2007/10/17
Fusobacterium nucleatum	NC_003454	2005/12/04
Geobacillus kaustophilus HTA426	NC_006510	2005/12/04

Geobacillus thermodenitrificans NG80-2	NC_009328	2007/04/03
Geobacter bemidjiensis Bem	NC_011146	2008/08/23
Geobacter lovleyi SZ	NC_010814	2008/06/07
Geobacter metallireducens GS-15	NC_007517	2007/01/23
Geobacter sulfurreducens	NC_002939	2007/01/23
Geobacter uraniumreducens Rf4	NC_009483	2008/05/08
Gloeobacter violaceus	NC_005125	2007/01/23
Gluconacetobacter diazotrophicus PAl 5	NC_010125	2007/12/10
Gluconobacter oxydans 621H	NC_006677	2005/12/04
Gramella forsetii KT0803	NC_008571	2006/12/30
Granulobacter bethesdensis CGDNIH1	NC_008343	2007/01/24
Haemophilus ducreyi 35000HP	NC_002940	2005/12/04
Haemophilus influenzae 86 028NP	NC_007146	2007/12/26
Haemophilus influenzae	NC_000907	2008/02/19
Haemophilus influenzae PittEE	NC_009566	2007/06/14
Haemophilus influenzae PittGG	NC_009567	2007/06/14
Haemophilus somnus 129PT	NC_008309	2007/01/24
Haemophilus somnus 2336	NC_010519	2008/03/25
Hahella chejuensis KCTC 2396	NC_007645	2005/12/15
Halobacterium salinarum R1	NC_010364	2008/03/14
Halobacterium sp	NC_002607	2005/12/04
Haloquadratum walsbyi	NC_008212	2006/08/25
Halorhodospira halophila SL1	NC_008789	2007/01/13
Helicobacter acinonychis Sheeba	NC_008229	2006/07/03
Helicobacter hepaticus	NC_004917	2005/12/04
Helicobacter pylori 26695	NC_000915	2007/01/23
Helicobacter pylori G27	NC_011333	2008/10/04
Helicobacter pylori HPAG1	NC_008086	2006/06/07
Helicobacter pylori J99	NC_000921	2007/01/23
Helicobacter pylori P12	NC_011498	2008/10/31
Helicobacter pylori Shi470	NC_010698	2008/10/12
Heliobacterium modesticaldum Ice1	NC_010337	2008/02/13
Herminiimonas arsenicoxydans	NC_009138	2008/01/07
Herpetosiphon aurantiacus ATCC 23779	NC_009972	2008/03/18
Hydrogenobaculum Y04AAS1	NC_011126	2008/08/07
Hyperthermus butylicus	NC_008818	2007/01/25
Hyphomonas neptunium ATCC 15444	NC_008358	2007/01/23
Idiomarina loihiensis L2TR	NC_006512	2005/12/04
Ignicoccus hospitalis KIN4 I	NC_009776	2007/09/07
Jannaschia CCS1	NC_007802	2007/01/23
Janthinobacterium Marseille	NC_009659	2007/07/25
Kineococcus radiotolerans SRS30216	NC_009664	2007/07/25
Klebsiella pneumoniae 342	NC_011283	2008/09/24
Klebsiella pneumoniae MGH 78578	NC_009648	2007/07/25
Kocuria rhizophila DC2201	NC_010617	2008/07/29
Lactobacillus acidophilus NCFM	NC_006814	2007/11/08
Lactobacillus brevis ATCC 367	NC_008497	2006/10/23
Lactobacillus casei ATCC 334	NC_008526	2006/10/23
Lactobacillus casei	NC_010999	2008/06/25
Lactobacillus delbrueckii bulgaricus ATCC BAA-365	NC_008529	2006/10/24
Lactobacillus delbrueckii bulgaricus	NC_008054	2006/06/14
Lactobacillus fermentum IFO 3956	NC_010610	2008/05/08
Lactobacillus gasseri ATCC 33323	NC_008530	2006/10/24
Lactobacillus helveticus DPC 4571	NC_010080	2007/12/04
Lactobacillus johnsonii NCC 533	NC_005362	2005/12/04
Lactobacillus plantarum	NC_004567	2005/12/04
Lactobacillus reuteri F275 JGI	NC_009513	2007/06/04
Lactobacillus reuteri F275 Kitasato	NC_010609	2008/11/15
Lactobacillus sakei 23K	NC_007576	2007/01/23
Lactobacillus salivarius UCC118	NC_007929	2008/01/07
Lactococcus lactis cremoris MG1363	NC_009004	2007/02/14
Lactococcus lactis cremoris SK11	NC_008527	2006/10/24
Lactococcus lactis	NC_002662	2007/01/23
Lawsonia intracellularis PHE MN1-00	NC_008011	2006/05/09
Legionella pneumophila Corby	NC_009494	2007/05/29
Legionella pneumophila Lens	NC_006369	2007/01/23
Legionella pneumophila Paris	NC_006368	2007/01/23

Legionella pneumophila Philadelphia 1	NC_002942		2007/01/24
Leifsonia xyli xyli CTCB0	NC_006087		2005/12/04
Leptothrix cholodnii SP 6	NC_010524		2008/03/28
Leuconostoc citreum KM20	NC_010471		2008/03/18
Leuconostoc mesenteroides ATCC 8293	NC_008531		2006/10/24
Listeria innocua	NC_003212		2007/01/23
Listeria monocytogenes 4b F2365	NC_002973		2007/01/23
Listeria monocytogenes	NC_003210		2007/01/23
Listeria welshimeri serovar 6b SLCC5334	NC_008555		2008/01/07
Lysinibacillus sphaericus C3 41	NC_010382		2008/03/13
Magnetococcus MC-1	NC_008576		2007/01/23
Magnetospirillum magneticum AMB-1	NC_007626		2005/12/07
Mannheimia succiniciproducens MBEL55E	NC_006300		2005/12/04
Maricaulis maris MCS10	NC_008347		2007/01/23
Marinobacter aquaeolei VT8	NC_008740		2007/01/10
Marinomonas MWYL1	NC_009654		2007/07/25
Mesoplasma florum L1	NC_006055		2005/12/04
Mesorhizobium BNC1	NC_008254		2007/01/23
Mesorhizobium loti	NC_002678		2005/12/04
Metallosphaera sedula DSM 5348	NC_009440		2007/05/08
Methanobacterium thermoautotrophicum	NC_000916		2007/05/09
Methanobrevibacter smithii ATCC 35061	NC_009515		2007/06/06
Methanococcoides burtonii DSM 6242	NC_007955		2007/01/23
Methanococcus aeolicus Nankai-3	NC_009635		2007/07/03
Methanococcus jannaschii	NC_000909,	NC_001732,	2008/02/19
	NC_001733		
Methanococcus maripaludis C5	NC_009135		2007/03/26
Methanococcus maripaludis C6	NC_009975		2008/03/18
Methanococcus maripaludis C7	NC_009637		2007/07/03
Methanococcus maripaludis S2	NC_005791		2007/01/23
Methanococcus vannielii SB	NC_009634		2007/07/03
Methanocorpusculum labreanum Z	NC_008942		2007/02/07
Methanoculleus marisnigri JR1	NC_009051		2007/02/26
Methanopyrus kandleri	NC_003551		2007/01/23
Methanosaeta thermophila PT	NC_008553		2007/01/23
Methanosarcina acetivorans	NC_003552		2005/12/04
Methanosarcina mazei	NC_003901		2007/01/23
Methanosphaera stadtmanae	NC_007681		2007/01/23
Methanospirillum hungatei JF-1	NC_007796		2007/01/23
Methylacidiphilum infernorum V4	NC_010794		2008/07/10
Methylibium petroleiphilum PM1	NC_008825		2007/01/30
Methylobacillus flagellatus KT	NC_007947		2006/04/11
Methylobacterium 4 46	NC_010511		2008/03/26
Methylobacterium extorquens PA1	NC_010172		2008/03/18
Methylobacterium populi BJ001	NC_010725		2008/09/11
Methylobacterium radiotolerans JCM 2831	NC_010505		2008/03/24
Methylococcus capsulatus Bath	NC_002977		2007/01/23
Microcystis aeruginosa NIES 843	NC_010296		2008/03/18
Moorella thermoacetica ATCC 39073	NC_007644		2007/01/23
Mycobacterium avium 104	NC_008595		2006/11/30
Mycobacterium avium paratuberculosis	NC_002944		2005/12/04
Mycobacterium bovis BCG Pasteur 1173P2	NC_008769		2007/01/11
Mycobacterium bovis	NC_002945		2007/01/24
Mycobacterium gilvum PYR-GCK	NC_009338		2007/04/16
Mycobacterium JLS	NC_009077		2007/03/01
Mycobacterium KMS	NC_008705		2006/12/23
Mycobacterium leprae	NC_002677		2007/01/23
Mycobacterium marinum M	NC_010612		2008/04/22
Mycobacterium MCS	NC_008146		2007/01/23
Mycobacterium smegmatis MC2 155	NC_008596		2006/11/30
Mycobacterium tuberculosis CDC1551	NC_002755		2005/12/04
Mycobacterium tuberculosis F11	NC_009565		2007/06/15
Mycobacterium tuberculosis H37Ra	NC_009525		2007/06/06
Mycobacterium tuberculosis H37Rv	NC_000962		2007/01/23
Mycobacterium ulcerans Agy99	NC_008611		2007/01/23
Mycobacterium vanbaalenii PYR-1	NC_008726		2006/12/29
Mycoplasma agalactiae PG2	NC_009497		2008/03/18

Mycoplasma arthritidis 158L3 1	NC_011025	2008/06/28
Mycoplasma capricolum ATCC 27343	NC_007633	2007/01/23
Mycoplasma gallisepticum	NC_004829	2005/12/04
Mycoplasma genitalium	NC_000908	2008/02/19
Mycoplasma hyopneumoniae 232	NC_006360	2005/12/04
Mycoplasma hyopneumoniae 7448	NC_007332	2005/12/04
Mycoplasma hyopneumoniae J	NC_007295	2005/12/04
Mycoplasma mobile 163K	NC_006908	2005/12/04
Mycoplasma mycoides	NC_005364	2007/03/09
Mycoplasma penetrans	NC_004432	2005/12/04
Mycoplasma pneumoniae	NC_000912	2005/12/04
Mycoplasma pulmonis	NC_002771	2005/12/04
Mycoplasma synoviae 53	NC_007294	2005/12/04
Myxococcus xanthus DK 1622	NC_008095	2007/01/23
Nanoarchaeum equitans	NC_005213	2007/01/23
Natranaerobius thermophilus JW NM WN LF	NC_010718	2008/09/11
Natronomonas pharaonis	NC_007426	2007/01/23
Neisseria gonorrhoeae FA 1090	NC_002946	2005/12/04
Neisseria gonorrhoeae NCCP11945	NC_011035	2008/07/12
Neisseria meningitidis 053442	NC_010120	2007/12/13
Neisseria meningitidis FAM18	NC_008767	2007/01/11
Neisseria meningitidis MC58	NC_003112	2005/12/04
Neisseria meningitidis Z2491	NC_003116	2005/12/04
Neorickettsia sennetsu Miyayama	NC_007798	2007/01/23
Nitratiruptor SB155-2	NC_009662	2007/07/26
Nitrobacter hamburgensis X14	NC_007964	2007/01/23
Nitrobacter winogradskyi Nb-255	NC_007406	2007/01/23
Nitrosococcus oceani ATCC 19707	NC_007484	2007/01/23
Nitrosomonas europaea	NC_004757	2007/01/23
Nitrosomonas eutropha C71	NC_008344	2007/01/23
Nitrosopumilus maritimus SCM1	NC_010085	2008/03/18
Nocardia farcinica IFM10152	NC_006361	2005/12/04
Nocardioides JS614	NC_008699	2006/12/22
Nostoc punctiforme PCC 73102	NC_010628	2008/06/11
Nostoc sp	NC_003272	2008/03/18
Novosphingobium aromaticivorans DSM 12444	NC_007794	2007/01/23
Oceanobacillus iheyensis	NC_004193	2008/03/18
Oenococcus oeni PSU-1	NC_008528	2006/10/23
Oligotropha carboxidovorans OM5	NC_011386	2008/10/23
Onion yellows phytoplasma	NC_005303	2005/12/04
Opitutus terrae PB90 1	NC_010571	2008/04/12
Orientia tsutsugamushi Boryong	NC_009488	2007/05/24
Orientia tsutsugamushi Ikeda	NC_010793	2008/06/02
Parabacteroides distasonis ATCC 8503	NC_009615	2007/06/29
Parachlamydia sp UWE25	NC_005861	2005/12/19
Parvibaculum lavamentivorans DS-1	NC_009719	2007/07/31
Pasteurella multocida	NC_002663	2005/12/04
Pediococcus pentosaceus ATCC 25745	NC_008525	2006/10/23
Pelobacter carbinolicus	NC_007498	2006/04/01
Pelobacter propionicus DSM 2379	NC_008609	2007/01/23
Pelodictyon luteolum DSM 273	NC_007512	2007/01/23
Pelodictyon phaeoclathratiforme BU 1	NC_011060	2008/07/21
Pelotomaculum thermopropionicum SI	NC_009454	2007/05/18
Petrotoga mobilis SJ95	NC_010003	2008/03/18
Phenylobacterium zucineum HLK1	NC_011144	2008/08/22
Photorhabdus luminescens	NC_005126	2007/01/23
Picrophilus torridus DSM 9790	NC_005877	2007/01/24
Pirellula sp	NC_005027	2007/01/23
Polaromonas JS666	NC_007948	2007/01/23
Polaromonas naphthalenivorans CJ2	NC_008781	2007/01/11
Polynucleobacter necessarius STIR1	NC_010531	2008/04/01
Polynucleobacter QLW-P1DMWA-1	NC_009379	2007/04/25
Porphyromonas gingivalis ATCC 33277	NC_010729	2008/06/12
Porphyromonas gingivalis W83	NC_002950	2005/12/04
Prochlorococcus marinus AS9601	NC_008816	2007/01/23
Prochlorococcus marinus CCMP1375	NC_005042	2005/12/04
Prochlorococcus marinus MED4	NC_005072	2007/01/23

Prochlorococcus marinus MIT 9211	NC_009976	2008/03/18
Prochlorococcus marinus MIT 9215	NC_009840	2007/09/22
Prochlorococcus marinus MIT 9301	NC_009091	2007/03/07
Prochlorococcus marinus MIT 9303	NC_008820	2007/01/24
Prochlorococcus marinus MIT 9312	NC_007577	2007/01/23
Prochlorococcus marinus MIT9313	NC_005071	2007/01/23
Prochlorococcus marinus MIT 9515	NC_008817	2007/01/23
Prochlorococcus marinus NATL1A	NC_008819	2007/01/24
Prochlorococcus marinus NATL2A	NC_007335	2007/12/26
Propionibacterium acnes KPA171202	NC_006085	2005/12/04
Prosthecochloris aestuarii DSM 271	NC_011059	2008/07/21
Prosthecochloris vibrioformis DSM 265	NC_009337	2007/04/16
Proteus mirabilis	NC_010554	2008/08/27
Pseudoalteromonas atlantica T6c	NC_008228	2007/01/23
Pseudoalteromonas haloplanktis TAC125	NC_007481	2008/10/01
Pseudomonas aeruginosa	NC_002516	2006/07/24
Pseudomonas aeruginosa PA7	NC_009656	2007/07/25
Pseudomonas aeruginosa UCBPP-PA14	NC_008463	2007/01/24
Pseudomonas entomophila L48	NC_008027	2008/01/07
Pseudomonas fluorescens Pf0 1	NC_007492	2008/09/19
Pseudomonas fluorescens Pf-5	NC_004129	2007/01/23
Pseudomonas mendocina ymp	NC_009439	2007/05/08
Pseudomonas putida F1	NC_009512	2007/06/04
Pseudomonas putida GB 1	NC_010322	2008/03/18
Pseudomonas putida KT2440	NC_002947	2007/01/23
Pseudomonas putida W619	NC_010501	2008/03/25
Pseudomonas stutzeri A1501	NC_009434	2007/05/08
Pseudomonas syringae phaseolicola 1448A	NC_005773	2005/12/04
Pseudomonas syringae pv B728a	NC_007005	2005/12/04
Pseudomonas syringae tomato DC3000	NC_004578	2005/12/04
Psychrobacter arcticum 273-4	NC_007204	2005/12/04
Psychrobacter cryohalolentis K5	NC_007969	2007/01/23
Psychrobacter PRwf-1	NC_009524	2007/06/06
Psychromonas ingrahamii 37	NC_008709	2006/12/27
Pyrobaculum aerophilum	NC_003364	2007/01/23
Pyrobaculum arsenaticum DSM 13514	NC_009376	2007/04/25
Pyrobaculum calidifontis JCM 11548	NC_009073	2007/03/01
Pyrobaculum islandicum DSM 4184	NC_008701	2006/12/23
Pyrococcus abyssi	NC_000868	2005/12/04
Pyrococcus furiosus	NC_003413	2007/01/23
Pyrococcus horikoshii	NC_000961	2007/01/23
Ralstonia solanacearum	NC_003295	2007/01/23
Renibacterium salmoninarum ATCC 33209	NC_010168	2007/12/26
Rhizobium etli CFN 42	NC_007761	2007/01/23
Rhizobium etli CIAT 652	NC_010994	2008/06/21
Rhizobium leguminosarum bv trifolii WSM2304	NC_011369	2008/10/17
Rhizobium leguminosarum bv viciae 3841	NC_008380	2008/01/07
Rhodobacter sphaeroides ATCC 17025	NC_009428	2007/05/08
Rhodococcus RHA1	NC_008268	2006/07/31
Rhodoferax ferrireducens T118	NC_007908	2007/01/23
Rhodopseudomonas palustris BisA53	NC_008435	2007/01/23
Rhodopseudomonas palustris BisB18	NC_007925	2007/01/23
Rhodopseudomonas palustris BisB5	NC_007958	2007/01/23
Rhodopseudomonas palustris CGA009	NC_005296	2007/01/23
Rhodopseudomonas palustris HaA2	NC_007778	2007/01/24
Rhodopseudomonas palustris TIE 1	NC_011004	2008/06/24
Rhodospirillum centenum SW	NC_011420	2008/10/25
Rhodospirillum rubrum ATCC 11170	NC_007643	2007/01/24
Rickettsia akari Hartford	NC_009881	2007/10/04
Rickettsia bellii OSU 85-389	NC_009883	2007/10/04
Rickettsia bellii RML369-C	NC_007940	2007/01/24
Rickettsia canadensis McKiel	NC_009879	2007/10/03
Rickettsia conorii	NC_003103	2007/01/24
Rickettsia felis URRWXCal2	NC_007109	2007/01/24
Rickettsia massiliae MTU5	NC_009900	2007/10/10
Rickettsia prowazekii	NC_000963	2007/01/24
Rickettsia rickettsii Iowa	NC_010263	2008/01/19

Rickettsia rickettsii Sheila Smith	NC_009882	2007/10/04
Rickettsia typhi wilmington	NC_006142	2007/01/24
Roseiflexus castenholzii DSM 13941	NC_009767	2007/09/05
Roseiflexus RS-1	NC_009523	2007/06/06
Roseobacter denitrificans OCh 114	NC_008209	2006/07/25
Rubrobacter xylanophilus DSM 9941	NC_008148	2007/01/24
Saccharophagus degradans 2-40	NC_007912	2007/01/24
Saccharopolyspora erythraea NRRL 2338	NC_009142	2007/03/26
Salinibacter ruber DSM 13855	NC_007677	2007/01/24
Salinispora arenicola CNS-205	NC_009953	2007/11/08
Salinispora tropica CNB-440	NC_009380	2007/04/25
Salmonella enterica arizonae serovar 62 z4 z23	NC_010067	2008/03/18
Salmonella enterica Choleraesuis	NC_006905	2007/01/24
Salmonella enterica Paratypi ATCC 9150	NC_006511	2007/01/24
Salmonella enterica serovar Agona SL483	NC_011149	2008/08/26
Salmonella enterica serovar Dublin CT 02021853	NC_011205	2008/09/06
Salmonella enterica serovar Enteritidis P125109	NC_011294	2008/10/02
Salmonella enterica serovar Gallinarum 287 91	NC_011274	2008/09/20
Salmonella enterica serovar Heidelberg SL476	NC_011083	2008/07/25
Salmonella enterica serovar Newport SL254	NC_011080	2008/07/25
Salmonella enterica serovar Paratyphi A AKU 12601	NC_011147	2008/09/04
Salmonella enterica serovar Paratyphi B SPB7	NC_010102	2008/03/18
Salmonella enterica serovar Schwarzengrund CVM19633	NC_011094	2008/07/31
Salmonella enterica serovar Typhi Ty2	NC_004631	2007/01/23
Salmonella typhimurium LT2	NC_003197	2005/12/04
Salmonella typhi	NC_003198	2005/12/04
Serratia proteamaculans 568	NC_009832	2007/09/20
Shewanella amazonensis SB2B	NC_008700	2006/12/22
Shewanella baltica OS155	NC_009052	2007/02/26
Shewanella baltica OS185	NC_009665	2007/07/25
Shewanella baltica OS195	NC_009997	2008/03/18
Shewanella denitrificans OS217	NC_007954	2007/01/24
Shewanella frigidimarina NCIMB 400	NC_008345	2007/01/24
Shewanella halifaxensis HAW EB4	NC_010334	2008/03/18
Shewanella loihica PV-4	NC_009092	2007/03/09
Shewanella MR-4	NC_008321	2007/01/24
Shewanella MR-7	NC_008322	2007/01/24
Shewanella oneidensis	NC_004347	2005/12/04
Shewanella pealeana ATCC 700345	NC_009901	2007/10/10
Shewanella piezotolerans WP3	NC_011566	2008/11/14
Shewanella putrefaciens CN-32	NC_009438	2007/05/08
Shewanella sediminis HAW-EB3	NC_009831	2007/09/20
Shewanella W3-18-1	NC_008750	2007/01/05
Shewanella woodyi ATCC 51908	NC_010506	2008/03/25
Shigella boydii CDC 3083 94	NC_010658	2008/05/09
Shigella boydii Sb227	NC_007613	2005/12/04
Shigella dysenteriae	NC_007606	2005/12/07
Shigella flexneri 2a 2457T	NC_004741	2007/01/24
Shigella flexneri 2a	NC_004337	2007/01/24
Shigella flexneri 5 8401	NC_008258	2006/07/28
Shigella sonnei Ss046	NC_007384	2005/12/04
Silicibacter pomeroyi DSS-3	NC_003911	2005/12/04
Silicibacter TM1040	NC_008044	2007/01/24
Sinorhizobium medicae WSM419	NC_009636	2007/07/03
Sinorhizobium meliloti	NC_003047	2007/01/24
Sodalis glossinidius morsitans	NC_007712	2006/01/18
Solibacter usitatus Ellin6076	NC_008536	2007/01/24
Sorangium cellulosum So ce 56	NC_010162	2007/12/14
Sphingomonas wittichii RW1	NC_009511	2007/06/04
Sphingopyxis alaskensis RB2256	NC_008048	2007/01/24
Staphylococcus aureus aureus MRSA252	NC_002952	2007/01/24
Staphylococcus aureus aureus MSSA476	NC_002953	2007/01/24
Staphylococcus aureus COL	NC_002951	2007/01/24
Staphylococcus aureus JH1	NC_009632	2007/07/03
Staphylococcus aureus JH9	NC_009487	2007/05/23
Staphylococcus aureus Mu3	NC_009782	2007/09/07
Staphylococcus aureus Mu50	NC_002758	2007/01/24

Staphylococcus aureus MW2	NC_003923	2005/12/04
Staphylococcus aureus N315	NC_002745	2007/01/24
Staphylococcus aureus NCTC 8325	NC_007795	2006/02/18
Staphylococcus aureus Newman	NC_009641	2007/07/07
Staphylococcus aureus RF122	NC_007622	2007/01/24
Staphylococcus aureus USA300	NC_007793	2007/01/24
Staphylococcus aureus USA300 TCH1516	NC_010079	2008/03/18
Staphylococcus epidermidis ATCC 12228	NC_004461	2005/12/04
Staphylococcus epidermidis RP62A	NC_002976	2007/01/24
Staphylococcus haemolyticus	NC_007168	2005/12/04
Staphylococcus saprophyticus	NC_007350	2006/03/02
Stenotrophomonas maltophilia K279a	NC_010943	2008/06/18
Stenotrophomonas maltophilia R551 3	NC_011071	2008/07/23
Streptococcus agalactiae 2603	NC_004116	2007/01/24
Streptococcus agalactiae A909	NC_007432	2007/01/24
Streptococcus agalactiae NEM316	NC_004368	2007/01/24
Streptococcus equi zooepidemicus MGCS10565	NC_011134	2008/10/01
Streptococcus gordonii Challis substr CH1	NC_009785	2007/09/14
Streptococcus mutans	NC_004350	2005/12/04
Streptococcus pneumoniae CGSP14	NC_010582	2008/04/12
Streptococcus pneumoniae D39	NC_008533	2006/10/24
Streptococcus pneumoniae G54	NC_011072	2008/07/24
Streptococcus pneumoniae Hungary19A 6	NC_010380	2008/03/18
Streptococcus pneumoniae R6	NC_003098	2005/12/04
Streptococcus pneumoniae TIGR4	NC_003028	2008/07/12
Streptococcus pyogenes M1 GAS	NC_002737	2007/01/24
Streptococcus pyogenes Manfredo	NC_009332	2008/01/07
Streptococcus pyogenes MGAS10270	NC_008022	2006/05/09
Streptococcus pyogenes MGAS10394	NC_006086	2005/12/04
Streptococcus pyogenes MGAS10750	NC_008024	2006/05/09
Streptococcus pyogenes MGAS2096	NC_008023	2006/05/09
Streptococcus pyogenes MGAS315	NC_004070	2007/01/24
Streptococcus pyogenes MGAS5005	NC_007297	2007/01/24
Streptococcus pyogenes MGAS6180	NC_007296	2005/12/04
Streptococcus pyogenes MGAS8232	NC_003485	2007/01/24
Streptococcus pyogenes MGAS9429	NC_008021	2006/05/09
Streptococcus pyogenes NZ131	NC_011375	2008/10/17
Streptococcus pyogenes SSI-1	NC_004606	2007/01/24
Streptococcus sanguinis SK36	NC_009009	2007/02/16
Streptococcus suis 05ZYH33	NC_009442	2007/05/08
Streptococcus suis 98HAH33	NC_009443	2007/05/08
Streptococcus thermophilus CNRZ1066	NC_006449	2005/12/04
Streptococcus thermophilus LMD-9	NC_008532	2006/10/24
Streptococcus thermophilus LMG 18311	NC_006448	2005/12/04
Streptomyces avermitilis	NC_003155	2007/12/26
Streptomyces coelicolor	NC_003888	2007/01/24
Streptomyces griseus NBRC 13350	NC_010572	2008/04/12
Sulfurihydrogenibium YO3AOP1	NC_010730	2008/06/12
Sulfurovum NBC37-1	NC_009663	2007/07/26
Symbiobacterium thermophilum IAM14863	NC_006177	2005/12/04
Synechococcus CC9311	NC_008319	2007/01/24
Synechococcus CC9605	NC_007516	2007/01/24
Synechococcus CC9902	NC_007513	2005/12/04
Synechococcus elongatus PCC 6301	NC_006576	2005/12/04
Synechococcus elongatus PCC 7942	NC_007604	2007/01/24
Synechococcus PCC 7002	NC_010475	2008/03/18
Synechococcus RCC307	NC_009482	2007/05/23
Synechococcus sp WH8102	NC_005070	2007/01/24
Synechococcus WH 7803	NC_009481	2007/05/23
Synechocystis PCC6803	NC_000911	2007/01/24
Syntrophobacter fumaroxidans MPOB	NC_008554	2007/01/24
Syntrophomonas wolfei Goettingen	NC_008346	2007/01/24
Syntrophus aciditrophicus SB	NC_007759	2006/04/19
Thermoanaerobacter pseudethanolicus ATCC 33223	NC_010321	2008/03/18
Thermoanaerobacter tengcongensis	NC_003869	2007/01/24
Thermoanaerobacter X514	NC_010320	2008/03/18
Thermobifida fusca YX	NC_007333	2007/01/24

Thermococcus kodakaraensis KOD1	NC_006624	2005/12/04
Thermococcus onnurineus NA1	NC_011529	2008/11/07
Thermodesulfovibrio yellowstonii DSM 11347	NC_011296	2008/09/27
Thermofilum pendens Hrk 5	NC_008698	2006/12/22
Thermoplasma acidophilum	NC_002578	2007/01/24
Thermoplasma volcanium	NC_002689	2006/12/21
Thermosipho melanesiensis BI429	NC_009616	2007/06/29
Thermosynechococcus elongatus	NC_004113	2007/01/24
Thermotoga lettingae TMO	NC_009828	2007/11/20
Thermotoga maritima	NC_000853	2007/01/24
Thermotoga petrophila RKU-1	NC_009486	2007/05/23
Thermotoga RQ2	NC_010483	2008/06/02
Thermus thermophilus HB27	NC_005835	2005/12/04
Thermus thermophilus HB8	NC_006461	2007/01/24
Thiobacillus denitrificans ATCC 25259	NC_007404	2007/01/24
Thiomicrospira crunogena XCL-2	NC_007520	2007/01/24
Thiomicrospira denitrificans ATCC 33889	NC_007575	2008/01/26
Treponema denticola ATCC 35405	NC_002967	2005/12/04
Treponema pallidum	NC_000919	2007/01/24
Treponema pallidum SS14	NC_010741	2008/06/12
Trichodesmium erythraeum IMS101	NC_008312	2007/01/24
Tropheryma whipplei TW08 27	NC_004551	2007/01/24
Tropheryma whipplei Twist	NC_004572	2005/12/04
uncultured Termite group 1 bacterium phylotype Rs D17	NS_000191	2008/11/22
Ureaplasma parvum serovar 3 ATCC 27815	NC_010503	2008/03/26
Ureaplasma urealyticum	NC_002162	2005/12/04
Ureaplasma urealyticum serovar 10 ATCC 33699	NC_011374	2008/10/17
Verminephrobacter eiseniae EF01-2	NC_008786	2007/01/11
Vibrio harveyi ATCC BAA-1116	NC_009783	2008/09/26
Vibrio vulnificus CMCP6	NC_004459	2008/09/25
Wigglesworthia brevipalpis	NC_004344	2007/01/24
Wolbachia endosymbiont of Brugia malayi TRS	NC_006833	2005/12/04
Wolbachia endosymbiont of Culex quinquefasciatus Pel	NC_010981	2008/11/20
Wolbachia endosymbiont of Drosophila melanogaster	NC_002978	2007/01/24
Wolinella succinogenes	NC_005090	2005/12/04
Xanthobacter autotrophicus Py2	NC_009720	2007/07/31
Xanthomonas campestris 8004	NC_007086	2007/01/24
Xanthomonas campestris ATCC 33913	NC_003902	2007/01/24
Xanthomonas campestris B100	NC_010688	2008/09/08
Xanthomonas campestris vesicatoria 85-10	NC_007508	2007/01/24
Xanthomonas citri	NC_003919	2007/01/24
Xanthomonas oryzae KACC10331	NC_006834	2005/12/04
Xanthomonas oryzae MAFF 311018	NC_007705	2007/01/24
Xanthomonas oryzae PXO99A	NC_010717	2008/09/11
Xylella fastidiosa M12	NC_010513	2008/03/24
Xylella fastidiosa M23	NC_010577	2008/04/12
Xylella fastidiosa	NC_002488	2007/01/24
Xylella fastidiosa Temecula1	NC_004556	2005/12/03
Yersinia enterocolitica 8081	NC_008800	2008/01/07
Yersinia pestis Angola	NC_010159	2008/03/18
Yersinia pestis Antiqua	NC_008150	2007/01/24
Yersinia pestis biovar Microtus 91001	NC_005810	2008/09/11
Yersinia pestis CO92	NC_003143	2005/12/03
Yersinia pestis KIM	NC_004088	2005/12/03
Yersinia pestis Nepal516	NC_008149	2007/01/24
Yersinia pestis Pestoides F	NC_009381	2007/04/25
Yersinia pseudotuberculosis IP 31758	NC_009708	2007/07/27
Yersinia pseudotuberculosis IP32953	NC_006155	2007/01/24
Yersinia pseudotuberculosis PB1	NC_010634	2008/07/31
Yersinia pseudotuberculosis YPIII	NC_010465	2008/03/18

Table B.1: Data sources: The protein sets of these species were downloaded from `ftp://ftp.ncbi.nih.gov/genomes/Bacteria/` and used within the domain-wide common approach in Chapter 4.

Benchmark

Species	Class
Bacillus halodurans	Bacilli (Gram-positive)
Bacillus subtilis	Bacilli (Gram-positive)
Lactococcus lactis	Bacilli (Gram-positive)
Listeria innocua	Bacilli (Gram-positive)
Streptococcus pneumoniae TIGR4	Bacilli (Gram-positive)
Streptococcus pyogenes M1 GAS	Bacilli (Gram-positive)
Buchnera sp. APS	Gamma proteobacteria
Escherichia coli K12	Gamma proteobacteria
Pasteurella multocida	Gamma proteobacteria
Salmonella typhimurium LT2	Gamma proteobacteria
Vibrio cholerae	Gamma proteobacteria
Yersinia pestis	Gamma proteobacteria
Brucella melitensis	Alpha proteobacteria
Caulobacter vibrioides	Alpha proteobacteria
Mesorhizobium loti	Alpha proteobacteria
Rickettsia prowazekii	Alpha proteobacteria

Table B.2: Species for benchmark: The protein sets of these 16 species were obtained from `COG` (`ftp://ftp.ncbi.nih.gov/pub/COG/COG/`) at 2009/10/15 for the benchmark of `Proteinortho` in comparison to `OrthoMCL` in Subsection 3.3.2.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] Y Zhou and L F Landweber. Blasto: a tool for searching orthologous groups. *Nucleic Acids Res*, 35(Web Server issue):W678–82, Jul 2007.

[2] M. Marron, K. M. Swenson, and B. M. E. Moret. Genomic distances under deletions and insertions. *Theor. Comput. Sci.*, 325(3):347–360, 2004. Special issue on best papers from COCOON'03.

[3] D P Wall, H B Fraser, and A E Hirsh. Detecting putative orthologs. *Bioinformatics*, 19(13):1710–1, Sep 2003.

[4] C B Hao, G C Wang, J N Dong, Q Zhang, and W T Cai. [bacterial biodiversity in the groundwater contaminated by oil]. *Huan Jing Ke Xue*, 30(8):2464–72, Aug 2009.

[5] J K Fredrickson, J M Zachara, D L Balkwill, D Kennedy, S M Li, H M Kostandarithes, M J Daly, M F Romine, and F J Brockman. Geomicrobiology of high-level nuclear waste-contaminated vadose sediments at the hanford site, washington state. *Appl Environ Microbiol*, 70(7):4230–41, Jul 2004.

[6] W B Whitman, D C Coleman, and W J Wiebe. Prokaryotes: the unseen majority. *Proc Natl Acad Sci U S A*, 95(12):6578–83, Jun 1998.

[7] C R Woese, O Kandler, and M L Wheelis. Towards a natural system of organisms: proposal for the domains archaea, bacteria, and eucarya. *Proc Natl Acad Sci U S A*, 87(12):4576–9, Jun 1990.

[8] Wilfried Janning and Elisabeth Knust. *Genetik*. Georg Thieme Verlag, Stuttgart, 2004.

[9] Benjamin Lewin. *Genes IX*. Jones and Bartlett Publishers, Inc, Sudbury, 2008.

[10] M Thanbichler, S C Wang, and L Shapiro. The bacterial nucleoid: a highly organized and dynamic structure. *J Cell Biochem*, 96(3):506–21, Oct 2005.

[11] J Hinnebusch and K Tilly. Linear plasmids and chromosomes in bacteria. *Mol Microbiol*, 10(5):917–22, Dec 1993.

[12] A Robicsek, G A Jacoby, and D C Hooper. The worldwide emergence of plasmid-mediated quinolone resistance. *Lancet Infect Dis*, 6(10):629–40, Oct 2006.

[13] P J Hastings, S M Rosenberg, and A Slack. Antibiotic-induced lateral transfer of antibiotic resistance. *Trends Microbiol*, 12(9):401–4, Sep 2004.

[14] J Davison. Genetic exchange between bacteria in the environment. *Plasmid*, 42(2):73–91, Sep 1999.

[15] D Mazel and J Davies. Antibiotic resistance in microbes. *Cell Mol Life Sci*, 56(9-10):742–54, Nov 1999.

[16] K Graumann and A Premstaller. Manufacturing of recombinant therapeutic proteins in microbial systems. *Biotechnol J*, 1(2):164–86, Feb 2006.

[17] G Walsh. Therapeutic insulins and their large-scale manufacture. *Appl Microbiol Biotechnol*, 67(2):151–9, Apr 2005.

[18] A Liese and M V Filho. Production of fine chemicals using biocatalysis. *Curr Opin Biotechnol*, 10(6):595–603, Dec 1999.

[19] Richard Owen, Cooper, and William White. *Lectures on the comparative anatomy and physiology of the invertebrate animals*. London :Longman, Brown, Green, and Longmans, 1843. http://www.biodiversitylibrary.org/bibliography/6788.

[20] W M Fitch. Distinguishing homologous from analogous proteins. *Syst Zool*, 19(2):99–113, Jun 1970.

[21] W M Fitch. Homology a personal view on some of the problems. *Trends Genet*, 16(5):227–31, May 2000.

[22] S E Massey, A Churbanov, S Rastogi, and D A Liberles. Characterizing positive and negative selection and their phylogenetic effects. *Gene*, 418(1-2):22–6, Jul 2008.

[23] G Theissen. Secret life of genes. *Nature*, 415(6873):741, Feb 2002.

[24] E V Koonin. Orthologs, paralogs, and evolutionary genomics. *Annu Rev Genet*, 39:309–38, 2005.

[25] M Remm, C E Storm, and E L Sonnhammer. Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *J Mol Biol*, 314(5):1041–52, Dec 2001.

[26] Z Fu, X Chen, V Vacic, P Nan, Y Zhong, and T Jiang. Msoar: a high-throughput ortholog assignment system based on genome rearrangement. *J Comput Biol*, 14(9):1160–75, Nov 2007.

[27] E Bapteste, M A O'Malley, R G Beiko, M Ereshefsky, J P Gogarten, L Franklin-Hall, F J Lapointe, J Dupre, T Dagan, Y Boucher, and W Martin. Prokaryotic evolution and the tree of life are two different things. *Biol Direct*, 4(1):34, Sep 2009.

[28] E V Koonin, K S Makarova, and L Aravind. Horizontal gene transfer in prokaryotes: quantification and classification. *Annu Rev Microbiol*, 55:709–42, 2001.

[29] T A Gibson and D S Goldberg. Questioning the ubiquity of neofunctionalization. *PLoS Comput Biol*, 5(1):e1000252, Jan 2009.

[30] D P Wall and T Deluca. Ortholog detection using the reciprocal smallest distance algorithm. *Methods Mol Biol*, 396:95–110, 2007.

[31] I K Jordan, I B Rogozin, Y I Wolf, and E V Koonin. Essential genes are more evolutionarily conserved than are nonessential genes in bacteria. *Genome Res*, 12(6):962–8, Jun 2002.

[32] A E Hirsh and H B Fraser. Protein dispensability and rate of evolution. *Nature*, 411(6841):1046–9, Jun 2001.

[33] L B Koski and G B Golding. The closest blast hit is often not the nearest neighbor. *J Mol Evol*, 52(6):540–2, Jun 2001.

[34] J D Thompson, F Plewniak, J Thierry, and O Poch. Dbclustal: rapid and reliable global multiple alignments of protein sequences detected by database searches. *Nucleic Acids Res*, 28(15):2919–26, Aug 2000.

[35] J D Thompson, D G Higgins, and T J Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res*, 22(22):4673–80, Nov 1994.

[36] A C Berglund, E Sjölund, G Ostlund, and E L Sonnhammer. Inparanoid 6: eukaryotic ortholog clusters with inparalogs. *Nucleic Acids Res*, 36(Database issue):D263–6, Jan 2008.

[37] X Chen, J Zheng, Z Fu, P Nan, Y Zhong, S Lonardi, and T Jiang. Assignment of orthologous genes via genome rearrangement. *IEEE/ACM Trans Comput Biol Bioinform*, 2(4):302–15, 2005.

[38] A J Enright, S Van Dongen, and C A Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res*, 30(7):1575–84, Apr 2002.

[39] A Alexeyenko, I Tamas, G Liu, and E L Sonnhammer. Automatic clustering of orthologs and inparalogs shared by multiple proteomes. *Bioinformatics*, 22(14):e9–15, Jul 2006.

[40] Z Fu and T Jiang. Clustering of main orthologs for multiple genomes. *J Bioinform Comput Biol*, 6(3):573–84, Jun 2008.

[41] Z Fu and T Jiang. Clustering of main orthologs for multiple genomes. *Comput Syst Bioinformatics Conf*, 6:195–201, 2007.

[42] L Li, C J Stoeckert, Jr, and D S Roos. Orthomcl: identification of ortholog groups for eukaryotic genomes. *Genome Res*, 13(9):2178–89, Sep 2003.

[43] R L Tatusov, M Y Galperin, D A Natale, and E V Koonin. The cog database: a tool for genome-scale analysis of protein functions and evolution. *Nucleic Acids Res*, 28(1):33–6, Jan 2000.

[44] R L Tatusov, N D Fedorova, J D Jackson, A R Jacobs, B Kiryutin, E V Koonin, D M Krylov, R Mazumder, S L Mekhedov, A N Nikolskaya, B S Rao, S Smirnov, A V Sverdlov, S Vasudevan, Y I Wolf, J J Yin, and D A Natale. The cog database: an updated version includes eukaryotes. *BMC Bioinformatics*, 4:41, Sep 2003.

[45] R L Tatusov, D A Natale, I V Garkavtsev, T A Tatusova, U T Shankavaram, B S Rao, B Kiryutin, M Y Galperin, N D Fedorova, and E V Koonin. The cog database:

new developments in phylogenetic classification of proteins from complete genomes. *Nucleic Acids Res*, 29(1):22–8, Jan 2001.

[46] L J Jensen, P Julien, M Kuhn, C von Mering, J Muller, T Doerks, and P Bork. eggnog: automated construction and annotation of orthologous groups of genes. *Nucleic Acids Res*, 36(Database issue):D250–4, Jan 2008.

[47] T J Hubbard, B L Aken, K Beal, B Ballester, M Caccamo, Y Chen, L Clarke, G Coates, F Cunningham, T Cutts, T Down, S C Dyer, S Fitzgerald, J Fernandez-Banet, S Graf, S Haider, M Hammond, J Herrero, R Holland, K Howe, K Howe, N Johnson, A Kahari, D Keefe, F Kokocinski, E Kulesha, D Lawson, I Longden, C Melsopp, K Megy, P Meidl, B Ouverdin, A Parker, A Prlic, S Rice, D Rios, M Schuster, I Sealy, J Severin, G Slater, D Smedley, G Spudich, S Trevanion, A Vilella, J Vogel, S White, M Wood, T Cox, V Curwen, R Durbin, X M Fernandez-Suarez, P Flicek, A Kasprzyk, G Proctor, S Searle, J Smith, A Ureta-Vidal, and E Birney. Ensembl 2007. *Nucleic Acids Res*, 35(Database issue):D610–7, Jan 2007.

[48] F Chen, A J Mackey, C J Stoeckert, Jr, and D S Roos. Orthomcl-db: querying a comprehensive multi-species collection of ortholog groups. *Nucleic Acids Res*, 34(Database issue):D363–8, Jan 2006.

[49] US National Library of Medicine National Center for Biotechnology Information. Ncbi genome assemblies and resources, 2007. `http://www.ncbi.nlm.nih.gov/projects/genome/`.

[50] J Ye, S McGinnis, and T L Madden. Blast: improvements for better sequence analysis. *Nucleic Acids Res*, 34(Web Server issue):W6–9, Jul 2006.

[51] S F Altschul, W Gish, W Miller, E W Myers, and D J Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–10, Oct 1990.

[52] C A Wilson, J Kreychman, and M Gerstein. Assessing annotation transfer for genomics: quantifying the relations between protein sequence, structure and function through traditional and probabilistic scores. *J Mol Biol*, 297(1):233–49, Mar 2000.

[53] R A Abagyan and S Batalov. Do aligned sequences share the same fold? *J Mol Biol*, 273(1):355–68, Oct 1997.

[54] A R Mushegian, J R Garey, J Martin, and L X Liu. Large-scale taxonomic profiling of eukaryotic model organisms: a comparison of orthologous proteins encoded by the human, fly, nematode, and yeast genomes. *Genome Res*, 8(6):590–8, Jun 1998.

[55] D. M. Beazley. Automated scientific software scripting with swig. *Future Gener. Comput. Syst.*, 19(5):599–609, 2003.

[56] Uwe Röhm and Thanh-Mai Diep. How to blast your database - a study of stored procedures for blast searches. In Mong-Li Lee, Kian-Lee Tan, and Vilas Wuwongse, editors, *DASFAA*, volume 3882 of *Lecture Notes in Computer Science*, pages 807–816. Springer, 2006.

[57] A M Altenhoff and C Dessimoz. Phylogenetic and functional assessment of orthologs inference projects and methods. *PLoS Comput Biol*, 5(1):e1000262, Jan 2009.

[58] T Hulsen, M A Huynen, J de Vlieg, and P M Groenen. Benchmarking ortholog identification methods using functional genomics data. *Genome Biol*, 7(4):R31, 2006.

[59] K Dolinski and D Botstein. Orthology and functional conservation in eukaryotes. *Annu Rev Genet*, 41:465–507, 2007.

[60] B Rost. Enzyme function less conserved than anticipated. *J Mol Biol*, 318(2):595–608, Apr 2002.

[61] W Tian and J Skolnick. How well is enzyme function conserved as a function of pairwise sequence identity? *J Mol Biol*, 333(4):863–82, Oct 2003.

[62] H S Najafabadi, H Goodarzi, and R Salavati. Universal function-specificity of codon usage. *Nucleic Acids Res*, Sep 2009.

[63] E De Vendittis, I Castellano, R Cotugno, M R Ruocco, G Raimo, and M Masullo. Adaptation of model proteins from cold to hot environments involves continuous and small adjustments of average parameters related to amino acid composition. *J Theor Biol*, 250(1):156–71, Jan 2008.

[64] J A Killian and G von Heijne. How proteins adapt to a membrane-water interface. *Trends Biochem Sci*, 25(9):429–34, Sep 2000.

[65] J Gough, K Karplus, R Hughey, and C Chothia. Assignment of homology to genome sequences using a library of hidden markov models that represent all proteins of known structure. *J Mol Biol*, 313(4):903–19, Nov 2001.

[66] D Wilson, R Pethica, Y Zhou, C Talbot, C Vogel, M Madera, C Chothia, and J Gough. Superfamily–sophisticated comparative genomics, data mining, visualization and phylogeny. *Nucleic Acids Res*, 37(Database issue):D380–6, Jan 2009.

[67] D Wilson, M Madera, C Vogel, C Chothia, and J Gough. The superfamily database in 2007: families and functions. *Nucleic Acids Res*, 35(Database issue):D308–13, Jan 2007.

[68] A G Murzin, S E Brenner, T Hubbard, and C Chothia. Scop: a structural classification of proteins database for the investigation of sequences and structures. *J Mol Biol*, 247(4):536–40, Apr 1995.

[69] P Coggill, R D Finn, and A Bateman. Identifying protein domains with the pfam database. *Curr Protoc Bioinformatics*, Chapter 2:Unit 2.5, Sep 2008.

[70] R D Finn, J Tate, J Mistry, P C Coggill, S J Sammut, H R Hotz, G Ceric, K Forslund, S R Eddy, E L Sonnhammer, and A Bateman. The pfam protein families database. *Nucleic Acids Res*, 36(Database issue):D281–8, Jan 2008.

[71] R Finn, S Griffiths-Jones, and A Bateman. Identifying protein domains with the pfam database. *Curr Protoc Bioinformatics*, Chapter 2:Unit 2.5, May 2003.

[72] S J Sammut, R D Finn, and A Bateman. Pfam 10 years on: 10,000 families and still growing. *Brief Bioinform*, 9(3):210–9, May 2008.

[73] C Elliott, F Zhou, W Spielmeyer, R Panstruga, and P Schulze-Lefert. Functional conservation of wheat and rice mlo orthologs in defense modulation to the powdery mildew fungus. *Mol Plant Microbe Interact*, 15(10):1069–77, Oct 2002.

[74] C Azevedo, A Sadanandom, K Kitagawa, A Freialdenhoven, K Shirasu, and P Schulze-Lefert. The rar1 interactor sgt1, an essential component of r gene-triggered disease resistance. *Science*, 295(5562):2073–6, Mar 2002.

[75] M Ashburner, C A Ball, J A Blake, D Botstein, H Butler, J M Cherry, A P Davis, K Dolinski, S S Dwight, J T Eppig, M A Harris, D P Hill, L Issel-Tarver, A Kasarskis, S Lewis, J C Matese, J E Richardson, M Ringwald, G M Rubin, and G Sherlock.

Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet*, 25(1):25–9, May 2000.

[76] A Bairoch. The enzyme database in 2000. *Nucleic Acids Res*, 28(1):304–5, Jan 2000.

[77] B Y Liao and J Zhang. Evolutionary conservation of expression profiles between human and mouse orthologous genes. *Mol Biol Evol*, 23(3):530–40, Mar 2006.

[78] R S Datta, C Meacham, B Samad, C Neyer, and K Sjölander. Berkeley phog: Phylofacts orthology group prediction web server. *Nucleic Acids Res*, 37(Web Server issue):W84–9, Jul 2009.

[79] D. R. Maddison and K. S. Schulz. The tree of life web project, 2004.

[80] D. R. Maddison and W.P. Maddison. The tree of life project, 1996.

[81] W D Swingley, M Chen, P C Cheung, A L Conrad, L C Dejesa, J Hao, B M Honchak, L E Karbach, A Kurdoglu, S Lahiri, S D Mastrian, H Miyashita, L Page, P Ramakrishna, S Satoh, W M Sattley, Y Shimada, H L Taylor, T Tomo, T Tsuchiya, Z T Wang, J Raymond, M Mimuro, R E Blankenship, and J W Touchman. Niche adaptation and genome expansion in the chlorophyll d-producing cyanobacterium acaryochloris marina. *Proc Natl Acad Sci U S A*, 105(6):2005–10, Feb 2008.

[82] P S Chain, V J Denef, K T Konstantinidis, L M Vergez, L Agulló, V L Reyes, L Hauser, M Córdova, L Gómez, M González, M Land, V Lao, F Larimer, J J LiPuma, E Mahenthiralingam, S A Malfatti, C J Marx, J J Parnell, A Ramette, P Richardson, M Seeger, D Smith, T Spilker, W J Sul, T V Tsoi, L E Ulrich, I B Zhulin, and J M Tiedje. Burkholderia xenovorans lb400 harbors a multi-replicon, 9.73-mbp genome shaped for versatility. *Proc Natl Acad Sci U S A*, 103(42):15280–7, Oct 2006.

[83] M A Larkin, G Blackshields, N P Brown, R Chenna, P A McGettigan, H McWilliam, F Valentin, I M Wallace, A Wilm, R Lopez, J D Thompson, T J Gibson, and D G Higgins. Clustal w and clustal x version 2.0. *Bioinformatics*, 23(21):2947–8, Nov 2007.

[84] Joseph Felsenstein. Phylip - phylogeny inference package (version 3.2). *Cladistics*, 5:164–166, 1989.

[85] E Pruesse, C Quast, K Knittel, B M Fuchs, W Ludwig, J Peplies, and F O Glöckner. Silva: a comprehensive online resource for quality checked and aligned ribosomal rna sequence data compatible with arb. *Nucleic Acids Res*, 35(21):7188–96, 2007.

[86] R Szklarczyk, M A Huynen, and B Snel. Complex fate of paralogs. *BMC Evol Biol*, 8:337, 2008.

[87] M Ibba, H D Becker, C Stathopoulos, D L Tumbula, and D Söll. The adaptor hypothesis revisited. *Trends Biochem Sci*, 25(7):311–6, Jul 2000.

[88] F H CRICK. On protein synthesis. *Symp Soc Exp Biol*, 12:138–63, 1958.

[89] M Szymanski, M A Deniziak, and J Barciszewski. Aminoacyl-trna synthetases database. *Nucleic Acids Res*, 29(1):288–90, Jan 2001.

[90] I M Keseler, C Bonavides-Martínez, J Collado-Vides, S Gama-Castro, R P Gunsalus, D A Johnson, M Krummenacker, L M Nolan, S Paley, I T Paulsen, M Peralta-Gil, A Santos-Zavaleta, A G Shearer, and P D Karp. Ecocyc: a comprehensive view of escherichia coli biology. *Nucleic Acids Res*, 37(Database issue):D464–70, Jan 2009.

[91] K Ito, M Wittekind, M Nomura, K Shiba, T Yura, A Miura, and H Nashimoto. A temperature-sensitive mutant of e. coli exhibiting slow processing of exported proteins. *Cell*, 32(3):789–97, Mar 1983.

[92] L Wang, A Miller, S L Rusch, and D A Kendall. Demonstration of a specific escherichia coli secy-signal peptide interaction. *Biochemistry*, 43(41):13185–92, Oct 2004.

[93] C E Nichols, C Johnson, M Lockyer, I G Charles, H K Lamb, A R Hawkins, and D K Stammers. Structural characterization of salmonella typhimurium yeaz, an m22 o-sialoglycoprotein endopeptidase homolog. *Proteins*, 64(1):111–23, Jul 2006.

[94] N D Rawlings, D P Tolle, and A J Barrett. Merops: the peptidase database. *Nucleic Acids Res*, 32(Database issue):D160–4, Jan 2004.

[95] A J Barrett. Bioinformatics of proteases in the merops database. *Curr Opin Drug Discov Devel*, 7(3):334–41, May 2004.

[96] N D Rawlings, F R Morton, C Y Kok, J Kong, and A J Barrett. Merops: the peptidase database. *Nucleic Acids Res*, 36(Database issue):D320–5, Jan 2008.

[97] Mellors A Jiang P. *Handbook of Proteolytic Enzymes*. Barrett AJ, Rawlings ND, Woessner JF, eds., London: Elsevier, 2004.

[98] X Yang, S Molimau, G P Doherty, E B Johnston, J Marles-Wright, R Rothnagel, B Hankamer, R J Lewis, and P J Lewis. The structure of bacterial rna polymerase in complex with the essential transcription elongation factor nusa. *EMBO Rep*, 10(9):997–1002, Sep 2009.

[99] S Borukhov, J Lee, and O Laptenko. Bacterial transcription elongation factors: new insights into molecular mechanism of action. *Mol Microbiol*, 55(5):1315–24, Mar 2005.

[100] C J Cardinale, R S Washburn, V R Tadigotla, L M Brown, M E Gottesman, and E Nudler. Termination factor rho and its cofactors nusa and nusg silence foreign dna in e. coli. *Science*, 320(5878):935–8, May 2008.

[101] J Greenblatt, J Li, S Adhya, D I Friedman, L S Baron, B Redfield, H F Kung, and H Weissbach. L factor that is required for beta-galactosidase synthesis is the nusa gene product involved in transcription termination. *Proc Natl Acad Sci U S A*, 77(4):1991–4, Apr 1980.

[102] I Lozada-Chávez, S C Janga, and J Collado-Vides. Bacterial regulatory networks are extremely flexible in evolution. *Nucleic Acids Res*, 34(12):3434–45, 2006.

[103] Douglas F. Browning and Stephen J. Busby. The regulation of bacterial transcription initiation. *Nat Rev Micro*, 2(1):57–65, January 2004.

[104] A G Sabelnikov, B Greenberg, and S A Lacks. An extended -10 promoter alone directs transcription of the dpnii operon of streptococcus pneumoniae. *J Mol Biol*, 250(2):144–55, Jul 1995.

[105] A Vanet, L Marsan, A Labigne, and M F Sagot. Inferring regulatory elements from a whole genome. an analysis of helicobacter pylori sigma(80) family of promoter signals. *J Mol Biol*, 297(2):335–53, Mar 2000.

[106] W Ross, D A Schneider, B J Paul, A Mertens, and R L Gourse. An intersubunit contact stimulating transcription initiation by e coli rna polymerase: interaction of the alpha c-terminal domain and sigma region 4. *Genes Dev*, 17(10):1293–307, May 2003.

[107] T Ishii, K Yoshida, G Terai, Y Fujita, and K Nakai. Dbtbs: a database of bacillus subtilis promoters and transcription factors. *Nucleic Acids Res*, 29(1):278–80, Jan 2001.

[108] J M Scott, T Mitchell, and W G Haldenwang. Stress triggers a process that limits activation of the bacillus subtilis stress transcription factor sigma(b). *J Bacteriol*, 182(5):1452–6, Mar 2000.

[109] M J de Hoon, Y Makita, K Nakai, and S Miyano. Prediction of transcriptional terminators in bacillus subtilis and related species. *PLoS Comput Biol*, 1(3):e25, Aug 2005.

[110] S Unniraman, R Prakash, and V Nagaraja. Conserved economics of transcription termination in eubacteria. *Nucleic Acids Res*, 30(3):675–84, Feb 2002.

[111] T Platt. Transcription termination and the regulation of gene expression. *Annu Rev Biochem*, 55:339–72, 1986.

[112] M S Ciampi. Rho-dependent terminators and transcription termination. *Microbiology*, 152(Pt 9):2515–28, Sep 2006.

[113] T M Henkin and C Yanofsky. Regulation by transcription attenuation in bacteria: how rna provides instructions for transcription termination/antitermination decisions. *Bioessays*, 24(8):700–7, Aug 2002.

[114] T M Henkin. Control of transcription termination in prokaryotes. *Annu Rev Genet*, 30:35–57, 1996.

[115] B R Burgess and J P Richardson. Rna passes through the hole of the protein hexamer in the complex with the escherichia coli rho factor. *J Biol Chem*, 276(6):4182–9, Feb 2001.

[116] R H Ebright. Rna polymerase: structural similarities between bacterial rna polymerase and eukaryotic rna polymerase ii. *J Mol Biol*, 304(5):687–98, Dec 2000.

[117] G Zhang, E A Campbell, L Minakhin, C Richter, K Severinov, and S A Darst. Crystal structure of thermus aquaticus core rna polymerase at 3.3 a resolution. *Cell*, 98(6):811–24, Sep 1999.

[118] J Fu, A L Gnatt, D A Bushnell, G J Jensen, N E Thompson, R R Burgess, P R David, and R D Kornberg. Yeast rna polymerase ii at 5 a resolution. *Cell*, 98(6):799–810, Sep 1999.

[119] S A Darst, A M Edwards, E W Kubalek, and R D Kornberg. Three-dimensional structure of yeast rna polymerase ii at 16 a resolution. *Cell*, 66(1):121–8, Jul 1991.

[120] Michael Hampsey. Omega meets its match. *Trends in Genetics*, 17(4):190 – 191, 2001.

[121] R L Gourse, W Ross, and T Gaal. Ups and downs in bacterial transcription initiation: the role of the alpha subunit of rna polymerase in promoter recognition. *Mol Microbiol*, 37(4):687–95, Aug 2000.

[122] M M Wösten. Eubacterial sigma-factors. *FEMS Microbiol Rev*, 22(3):127–50, Sep 1998.

[123] T M Gruber and C A Gross. Multiple sigma subunits and the partitioning of bacterial transcription space. *Annu Rev Microbiol*, 57:441–66, 2003.

[124] M J Merrick. In a class of its own–the rna polymerase sigma factor sigma 54 (sigma n). *Mol Microbiol*, 10(5):903–9, Dec 1993.

[125] M Lonetto, M Gribskov, and C A Gross. The sigma 70 family: sequence conservation and evolutionary relationships. *J Bacteriol*, 174(12):3843–9, Jun 1992.

[126] A Ishihama. Functional modulation of escherichia coli rna polymerase. *Annu Rev Microbiol*, 54:499–518, 2000.

[127] H Maeda, N Fujita, and A Ishihama. Competition among seven escherichia coli sigma subunits: relative binding affinities to the core rna polymerase. *Nucleic Acids Res*, 28(18):3497–503, Sep 2000.

[128] A Martínez-Antonio, H Salgado, S Gama-Castro, R M Gutiérrez-Ríos, V Jiménez-Jacinto, and J Collado-Vides. Environmental conditions and transcriptional regulation in escherichia coli: a physiological integrative approach. *Biotechnol Bioeng*, 84(7):743–9, Dec 2003.

[129] E Pérez-Rueda and J Collado-Vides. The repertoire of dna-binding transcriptional regulators in escherichia coli k-12. *Nucleic Acids Res*, 28(8):1838–47, Apr 2000.

[130] A Martínez-Antonio and J Collado-Vides. Identifying global regulators in transcriptional regulatory networks in bacteria. *Curr Opin Microbiol*, 6(5):482–9, Oct 2003.

[131] S C Janga, H Salgado, and A Martínez-Antonio. Transcriptional regulation shapes the organization of genes on bacterial chromosomes. *Nucleic Acids Res*, 37(11):3680–8, Jun 2009.

[132] M M Babu, N M Luscombe, L Aravind, M Gerstein, and S A Teichmann. Structure and evolution of transcriptional regulatory networks. *Curr Opin Struct Biol*, 14(3):283–91, Jun 2004.

[133] S A Teichmann and M M Babu. Conservation of gene co-regulation in prokaryotes and eukaryotes. *Trends Biotechnol*, 20(10):407–10; discussion 410, Oct 2002.

[134] A N Surovaia, G I Gitel'zon, and G V Gurskiĭ. [the interaction of lamda-cro repressor and its mutant covalent s-s-dimer lamda-crov55c with symmetric and asymmetric dna]. *Biofizika*, 51(3):567–73, 2006.

[135] B P Westover, J D Buhler, J L Sonnenburg, and J I Gordon. Operon prediction without a training set. *Bioinformatics*, 21(7):880–8, Apr 2005.

[136] E V Koonin and A R Mushegian. Complete genome sequences of cellular life forms: glimpses of theoretical evolutionary genomics. *Curr Opin Genet Dev*, 6(6):757–62, Dec 1996.

[137] M Kozak. Comparison of initiation of protein synthesis in procaryotes, eucaryotes, and organelles. *Microbiol Rev*, 47(1):1–45, Mar 1983.

[138] I G Fotheringham, S A Dacey, P P Taylor, T J Smith, M G Hunter, M E Finlay, S B Primrose, D M Parker, and R M Edwards. The cloning and sequence analysis of the aspc and tyrb genes from escherichia coli k12. comparison of the primary structures of the aspartate aminotransferase and aromatic aminotransferase of e. coli with those of the pig aspartate aminotransferase isoenzymes. *Biochem J*, 234(3):593–604, Mar 1986.

[139] G Golderer, M Dlaska, P Gröbner, and W Piendl. Ttg serves as an initiation codon for the ribosomal protein mvas7 from the archaeon methanococcus vannielii. *J Bacteriol*, 177(20):5994–6, Oct 1995.

[140] J Nölling, T D Pihl, A Vriesema, and J N Reeve. Organization and growth phase-dependent transcription of methane genes in two regions of the methanobacterium thermoautotrophicum genome. *J Bacteriol*, 177(9):2460–8, May 1995.

[141] T Sazuka and O Ohara. Sequence features surrounding the translation initiation sites assigned on the genome sequence of synechocystis sp. strain pcc6803 by amino-terminal protein sequencing. *DNA Res*, 3(4):225–32, Aug 1996.

[142] Frederick R. Blattner, III Plunkett, Guy, Craig A. Bloch, Nicole T. Perna, Valerie Burland, Monica Riley, Julio Collado-Vides, Jeremy D. Glasner, Christopher K. Rode, George F. Mayhew, Jason Gregor, Nelson Wayne Davis, Heather A. Kirkpatrick, Michael A. Goeden, Debra J. Rose, Bob Mau, and Ying Shao. The Complete Genome Sequence of Escherichia coli K-12. *Science*, 277(5331):1453–1462, 1997.

[143] K F Genser, G Renner, and H Schwab. Molecular cloning, sequencing and expression in escherichia coli of the poly(3-hydroxyalkanoate) synthesis genes from alcaligenes latus dsm1124. *J Biotechnol*, 64(2-3):125–35, Oct 1998.

[144] G Wang, L Nie, and H Tan. Cloning and characterization of sano, a gene involved in nikkomycin biosynthesis in streptomyces ansochromogenes. *Lett Appl Microbiol*, 37(6):452–7, 2003.

[145] P Polard, M F Prère, M Chandler, and O Fayet. Programmed translational frameshifting and initiation at an auu codon in gene expression of bacterial insertion sequence is911. *J Mol Biol*, 222(3):465–77, Dec 1991.

[146] A J Spiers and P L Bergquist. Expression and regulation of the repa protein of the repfib replicon from plasmid p307. *J Bacteriol*, 174(23):7533–41, Dec 1992.

[147] D Liveris, J J Schwartz, R Geertman, and I Schwartz. Molecular cloning and sequencing of infc, the gene encoding translation initiation factor if3, from four enterobacterial species. *FEMS Microbiol Lett*, 112(2):211–6, Sep 1993.

[148] N Binns and M Masters. Expression of the escherichia coli pcnb gene is translationally limited using an inefficient start codon: a second chromosomal example of translation initiated at auu. *Mol Microbiol*, 44(5):1287–98, Jun 2002.

[149] R P Bonocora and D A Shub. A likely pathway for formation of mobile group i introns. *Curr Biol*, 19(3):223–8, Feb 2009.

[150] M Belfort, M E Reaban, T Coetzee, and J Z Dalgaard. Prokaryotic introns and inteins: a panoply of form and function. *J Bacteriol*, 177(14):3897–903, Jul 1995.

[151] T Ohama, Y Inagaki, Y Bessho, and S Osawa. Evolving genetic code. *Proc Jpn Acad Ser B Phys Biol Sci*, 84(2):58–74, 2008.

[152] F Yamao, A Muto, Y Kawauchi, M Iwami, S Iwagami, Y Azumi, and S Osawa. Uga is read as tryptophan in mycoplasma capricolum. *Proc Natl Acad Sci U S A*, 82(8):2306–9, Apr 1985.

[153] James D. Watson, Tania A. Baker, Stephen P. Bell, Alexander Gann, Michael Levine, and Richard Losick. *Molecular Biology of the Gene, Fifth Edition.* Benjamin Cummings, 5 edition, December 2003.

[154] J Ma, A Campbell, and S Karlin. Correlations between shine-dalgarno sequences and gene features such as predicted expression levels and operon structures. *J Bacteriol*, 184(20):5733–45, Oct 2002.

[155] J Shine and L Dalgarno. Determinant of cistron specificity in bacterial ribosomes. *Nature*, 254(5495):34–8, Mar 1975.

[156] G Q Hu, X Zheng, Y F Yang, P Ortet, Z S She, and H Zhu. Protisa: a comprehensive resource for translation initiation site annotation in prokaryotic genomes. *Nucleic Acids Res*, 36(Database issue):D114–9, Jan 2008.

[157] A Marchler-Bauer, J B Anderson, F Chitsaz, M K Derbyshire, C DeWeese-Scott, J H Fong, L Y Geer, R C Geer, N R Gonzales, M Gwadz, S He, D I Hurwitz, J D Jackson, Z Ke, C J Lanczycki, C A Liebert, C Liu, F Lu, S Lu, G H Marchler, M Mullokandov, J S Song, A Tasneem, N Thanki, R A Yamashita, D Zhang, N Zhang, and S H Bryant. Cdd: specific functional annotation with the conserved domain database. *Nucleic Acids Res*, 37(Database issue):D205–10, Jan 2009.

[158] D Frishman, A Mironov, H W Mewes, and M Gelfand. Combining diverse evidence for gene recognition in completely sequenced bacterial genomes. *Nucleic Acids Res*, 26(12):2941–7, Jun 1998.

[159] Y Makita, M J de Hoon, and A Danchin. Hon-yaku: a biology-driven bayesian methodology for identifying translation initiation sites in prokaryotes. *BMC Bioinformatics*, 8:47, 2007.

[160] B Chang, S Halgamuge, and S L Tang. Analysis of sd sequences in completed microbial genomes: non-sd-led genes are as common as sd-led genes. *Gene*, 373:90–9, May 2006.

[161] W R Gish. nrdb 2.0.1 – quasi-nonredundant database generator, 1998. `http://pubmlst.org/perl/mlstanalyse/mlstanalyse.pl?site=pubmlst`.

[162] T L Bailey and C Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proc Int Conf Intell Syst Mol Biol*, 2:28–36, 1994.

[163] Axel Mosig, Julian J.-L. Chen, and Peter F. Stadler. Homology search with fragmented nucleic acid sequence patterns. In Raffaele Giancarlo and Sridhar Hannenhalli, editors, *WABI*, volume 4645 of *Lecture Notes in Computer Science*, pages 335–345. Springer, 2007.

[164] A Mosig, K Sameith, and P Stadler. Fragrep: an efficient search tool for fragmented patterns in genomic sequences. *Genomics Proteomics Bioinformatics*, 4(1):56–60, Feb 2006.

[165] C L Kingsford, K Ayanbule, and S L Salzberg. Rapid, accurate, computational discovery of rho-independent transcription terminators illuminates their relationship to dna uptake. *Genome Biol*, 8(2):R22, 2007.

[166] A L Delcher, D Harmon, S Kasif, O White, and S L Salzberg. Improved microbial gene identification with glimmer. *Nucleic Acids Res*, 27(23):4636–41, Dec 1999.

[167] S L Salzberg, A L Delcher, S Kasif, and O White. Microbial gene identification using interpolated markov models. *Nucleic Acids Res*, 26(2):544–8, Jan 1998.

[168] UniProt Consortium. The universal protein resource (uniprot) 2009. *Nucleic Acids Res*, 37(Database issue):D169–74, Jan 2009.

[169] T C Hodgman. A historical perspective on gene/protein functional assignment. *Bioinformatics*, 16(1):10–5, Jan 2000.

[170] M D Ermolaeva, O White, and S L Salzberg. Prediction of operons in microbial genomes. *Nucleic Acids Res*, 29(5):1216–21, Mar 2001.

[171] T T Tran, F Zhou, S Marshburn, M Stead, S R Kushner, and Y Xu. De novo computational prediction of non-coding rna genes in prokaryotic genomes. *Bioinformatics*, 25(22):2897–905, Nov 2009.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet. Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann.

Leipzig	10. Februar 2010	
Ort	Datum	Unterschrift