Dissertação de Mestrado

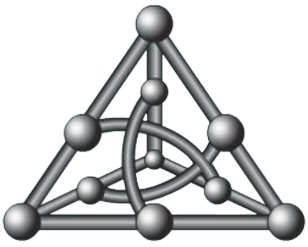
Orthologsorter: inferindo genotipagem e funcionalidade a partir de famílias de proteínas ortólogas

Nariélly Calista Farias

Orientação: Prof. Dr. Nalvo Franco de Almeida Jr.

Área de Concentração: Biologia Computacional

Palavras-chave: Bioinformática, Genotipagem, Orthologsorter, Família de Proteínas, Filogenia



Faculdade de Computação
Universidade Federal de Mato Grosso do Sul
Campo Grande, 27 de junho de 2013

Orthologsorter: inferindo genotipagem e funcionalidade
a partir de famílias de proteínas ortólogas

Campo Grande, 27 de junho de 2013

Banca Examinadora

- Prof. Dr. Nalvo Franco de Almeida Junior (FACOM/UFMS) - orientador
- Prof. Dr. Said Sadique Adi (FACOM/UFMS)
- Profa. Dra. Luciana Montera Cheung (FACOM/UFMS)
- Profa. Dra. Maria Emilia Machado Telles Walter (UnB)

Dedicatória

Dedico este trabalho a vocês que sempre me fizeram acreditar na realização dos meus sonhos e trabalharam muito para que eu pudesse realizá-los, meus pais, Doraci e Eumiro, e minha irmã, Cristielly.

Agradecimentos

A elaboração deste trabalho não teria sido possível sem a colaboração, estímulo e empenho de diversas pessoas. Gostaria, por este fato, de expressar toda a minha gratidão e apreço a todos aqueles que, direta ou indiretamente, contribuíram para que este trabalho se tornasse uma realidade. A todos quero manifestar os meus sinceros agradecimentos.

Em primeiro lugar, agradeço a Deus pela força, coragem e amor para concluir este mestrado. Ao único digno de glória e honra. Minha sincera gratidão por tudo o que tens feito em minha vida, e por tudo que ainda há de fazer.

Aos meus pais, Doraci e Eumiro, e à minha irmã Cristielly, pela dedicação, carinho, suporte e tudo mais. Por estarem sempre presentes, me apoiando em todas as minhas decisões.

Ao meu orientador, Professor Doutor Nalvo, para quem não há agradecimentos que cheguem. As notas dominantes da sua orientação foram a utilidade das suas recomendações e a cordialidade com que sempre me recebeu. Estou grata pela paciência e pela intensa dedicação durante o desenvolvimento deste trabalho, que foram decisivas para sua conclusão e em meu desenvolvimento pessoal. Muito obrigada pelo tempo dedicado e pelos ensinamentos transmitidos.

Aos meus amigos, com os quais contei nas necessárias horas de descanso e compartilhei momentos de tristeza e de alegria.

Agradeço o apoio da Faculdade de Computação da UFMS, em colaboração com o Instituto de Biologia da UnB, com o Instituto de Computação da Unicamp e com a Embrapa Gado de Corte.

Agradeço ainda aos professores da banca examinadora Maria Emília Walter Telles, Luciana Montera Cheung e Said Sadique Adi por suas sugestões e críticas, que contribuíram para a melhoria deste trabalho.

Por fim, agradeço à Capes pelo apoio financeiro durante meu mestrado.

A todos obrigada por ajudarem a concluir esse trabalho.

Resumo

Os métodos que permitem identificar e classificar espécies e subespécies de bactérias de maneira homogênea contam com uma variedade de técnicas para obter evidências biológicas. A genotipagem tem sido uma das principais metodologias utilizadas por estudos de epidemiologia, estrutura populacional e filogenética de bactérias. No entanto, tais técnicas apresentam limitações no que diz respeito ao poder de discriminação de cepas, ao tempo e esforço laborial bem como ao custo dos equipamentos e materiais necessários para sua execução. Diante disso, o presente trabalho tem por objetivo desenvolver ferramentas computacionais baseadas em comparação de sequências para a genotipagem de bactérias que contribuem para diminuir tais restrições. Mais especificamente, este trabalho criou um *pipeline* automático que fornece dois produtos finais. O primeiro consiste em uma ferramenta de busca via *web*, denominada *Orthologsorter*, que permite realizar buscas no conjunto de famílias de proteínas obtidas por algum método de clusterização de sequências a fim de obter informações funcionais e evolutivas das espécies analisadas. O segundo consiste em uma árvore filogenética construída através de uma busca específica utilizando *Orthologsorter*, contribuindo para a inferência da história evolutiva das espécies envolvidas. O *pipeline* proposto foi testado usando, em especial, os dados biológicos pertencentes a dois complexos bacterianos, *Bacillus cereus* e *Mycobaterium tuberculosis* e apresentou resultados satisfatórios, quando comparados com outras técnicas de genotipagem. Desse modo, os resultados obtidos demonstram que a ferramenta criada neste trabalho pode inferir relações filogenéticas entre espécies/cepas proximamente relacionadas baseando-se na comparação direta dos conjuntos de proteínas preditas dos organismos analisados.

Palavras-chave: bioinformática; genotipagem; orthologsorter; família de proteínas; filogenia.

Abstract

Methods to identify and classify species and subspecies of bacteria in a homogeneous way has a variety of techniques to obtain biological evidence. Genotyping has been one of the major methods used by studies of epidemiology, population structure and phylogenetics of bacteria. However, these techniques have limitations on the ability to discriminate isolates, are time and labor intensive, and demand high cost equipments and materials. This work aims to develop computational tools based on comparison of sequences for bacterial genotyping that contribute to reduce such restrictions. Specifically, this work has created an automatic pipeline that provides two final products. The first consists of a web search tool, called *Orthologsorter*, allowing search a set of protein families obtained by some method of clustering sequences in order to obtain functional and evolutionary information of the analyzed species. The second consists of a phylogenetic tree built through of a specific search utilizing *Orthologsorter*, contributing to the inference of the evolutionary history of the involved species. This tool has been tested using biological data from bacterial complexes of *Bacillus cereus* and *Mycobacterium tuberculosis*, and has shown satisfying results when compared to other genotyping techniques. Thus, the results show that the tool created in this work can successfully infer phylogenetic relationships among closely related species/isolates based is able direct comparison of the sets of predicted proteins of the analyzed organisms.

Keywords: bioinformatics; genotyping; orthologsorter; protein family; phylogeny.

Sumário

Capítulo 1

Introdução

Este trabalho está inserido na área de Bioinformática, a qual consiste no desenvolvimento e implementação de técnicas computacionais para solucionar problemas de Biologia Molecular. A importância da Bioinformática vem crescendo nas últimas décadas e tem contribuído consideravelmente para a identificação e classificação de organismos, principalmente por meio de ferramentas de comparação de genomas. Entre os principais focos da área estão o tratamento, armazenamento e análise do crescente número de sequências biológicas, com ênfase naquelas resultantes de projetos de sequenciamento de genomas.

Novas tecnologias de sequenciamento e a drástica redução de seus custos têm permitido a obtenção de um grande volume de informações biológicas dos organismos, aumentando a necessidade de ferramentas que permitam realizar análises computacionais detalhadas, além do armazenamento e gerenciamento dessas informações.

Um dos problemas cruciais para a Biologia Molecular e para a Genômica consiste em encontrar métodos que permitam identificar e classificar espécies e subespécies de bactérias de maneira homogênea. A identificação envolve reconhecer o grupo biológico ao qual o organismo pertence, previamente estabelecido pela classificação. A classificação corresponde a agrupar organismos em categorias hierárquicas, o que geralmente é realizado por meio do estudo de homologias, isto é, a partir da detecção de ancestralidade em comum [18].

Ambos os procedimentos (identificar e classificar) contam com uma variedade de

técnicas para obter evidências biológicas. Entre tais técnicas destaca-se a tipagem. Esta permite caracterizar organismos por meio da análise de caracteres expressos pela bactéria (fenotipagem) e por meio da análise do conteúdo genético (genotipagem) [32, 49].

A fenotipagem é limitada à capacidade das bactérias de imprevisivelmente mudar a expressão de determinada característica. Além disso, algumas técnicas de tipagem fenotípica tendem a ser um tanto específicas para um grupo bacteriano particular, o que impede a tipagem de outros grupos que não possuem as características investigadas pela técnica. Acrescenta-se a tais problemas o fato de que o grau de reprodutibilidade e de discriminação de tais técnicas frequentemente dificultam os processos de identificação e classificação de bactérias [32].

Apesar da sua grande utilidade na determinação do gênero e da espécie, a aplicabilidade de métodos fenotípicos de tipagem no nível da subespécie ou cepa é muito limitada. Devido ao fato de caracterizarem o fenótipo e não o genótipo, os resultados obtidos por estes métodos dependem fortemente das condições ambientais e são aplicáveis na tipagem de apenas algumas espécies.

Quanto à genotipagem, esta demonstra ser promissora para superar as dificuldades mencionadas [32]. Pode-se citar como uma das vantagens da genotipagem a estreita relação entre a diversidade genética e a variabilidade fenotípica, sendo esta última descrita pela distribuição geográfica, especificidade de patógenos, patogenicidade, resistência a drogas e virulência [49]. No entanto, a genotipagem também possui restrições no que diz respeito ao nível de distinção de isolados bacterianos.

A flexibilidade genética bacteriana constitui um fator determinante para tais restrições. Afinal, estes organismos possuem a habilidade de realizar transferência gênica horizontal (transferência de material genético entre organismos distantemente relacionados), perda ou aquisição gênica e recombinação. Assim a genotipagem, muitas vezes, não alcança a resolução requerida para distinguir isolados muito similares, pertencentes, por exemplo, à mesma linhagem. Ressalta-se ainda que uma quantidade considerável de técnicas de genotipagem exige muito tempo e esforço laborial bem como materiais caros para serem executadas [49].

Diante disso, a genotipagem necessita de abordagens que aumentem seu poder de discriminação de isolados bacterianos e que reduzam o tempo, esforço laborial e

custo para obtenção de resultados. Nesse contexto, abordagens computacionais apresentam-se como alternativas eficientes para superar tais limitações.

Uma abordagem promissora para genotipagem é a comparação de genomas, que vem sendo largamente utilizada, uma vez que facilita a análise funcional e evolutiva dos genomas. É uma poderosa ferramenta na identificação de funcionalidades comuns entre organismos, além de ser útil na determinação de propriedades filogenéticas importantes.

Muitas estratégias para comparar genomas levam em conta a preservação da ordem e do conteúdo gênico, uma vez que essas características tendem a se manter pouco alteradas entre organismos filogeneticamente próximos. Sequências conservadas entre genomas podem ser utilizadas para inferir a história evolutiva das espécies, além de facilitar a anotação genômica por meio da identificação de sequências similares as quais já tiveram identificadas suas funções bioquímicas [2, 9, 93].

Uma abordagem bastante utilizada é a determinação de famílias de proteínas comuns a vários genomas. A maioria das proteínas pode ser agrupada em famílias com base na similaridade entre suas sequências de aminoácidos por algum método de clusterização de sequências. Essas famílias dão pistas a respeito de funcionalidades comuns aos organismos e também fornecem direções para a construção da filogenia das espécies envolvidas na comparação. A construção de árvores filogenéticas auxilia na explicação dos possíveis relacionamentos entre as espécies atuais e na dedução das histórias evolutivas delas.

O objetivo deste trabalho foi o desenvolvimento de ferramentas computacionais baseadas em comparação de sequências para a genotipagem de bactérias. Mais especificamente, neste trabalho novas ferramentas foram desenvolvidas, além da utilização de outras já disponíveis, com o objetivo de criar um *pipeline* automático que fornece dois produtos finais. O primeiro consiste em uma ferramenta de busca via *web*, denominada *Orthologsorter*, que permite a realização de buscas no conjunto de famílias de proteínas obtidas por algum método de clusterização de sequências, visando obter informações funcionais e evolutivas das espécies analisadas. O segundo consiste na construção de uma árvore filogenética através de uma busca específica utilizando o *Orthologsorter*, contribuindo para a inferência da história evolutiva das espécies envolvidas. A busca específica citada aqui consiste, a grosso modo, na obtenção de famílias de proteínas contendo exatamente uma sequência de cada

organismo incluído na análise.

Tanto o *Orthologsorter* quanto essa abordagem de escolha de famílias e construção da filogenia foram utilizadas de forma não automatizada e com sucesso em vários projetos [4, 54, 63, 19, 73, 76, 96]. A construção de um *pipeline* automático envolvendo essas abordagens certamente auxiliará os pesquisadores na tarefa de análise de espécies de bactérias. A ferramenta aqui apresentada foi testada usando dois grupos de organismos: *Bacillus cereus* e *Mycobacterium tuberculosis*, e apresentou resultados satisfatórios, quando comparados com outras técnicas de tipagem.

Resultados preliminares deste trabalho foram publicados:

- em [58], no projeto de sequenciamento e análise do genoma do isolado FT9 de *Bacillus cereus*;
- em [8], como ferramentas auxiliares na determinação de um isolado de *Mycobacterium*; e
- em [55], como parte de uma abordagem alternativa de determinação de novos isolados, usada conjuntamente com a ferramenta kGC [88].

O presente texto, que pressupõe um entendimento básico de Biologia Molecular, em particular de síntese de proteínas, e de Bioinformática, encontra-se estruturado da seguinte forma. No Capítulo 2 é descrito o processo de anotação, suas etapas e principais objetivos. No Capítulo 3, são apresentados os conceitos sobre homologia e famílias de proteínas, essenciais para o entendimento do problema do qual trata o trabalho. Características e exemplos de ferramentas para detecção de famílias de proteínas também podem ser vistos nesse Capítulo. Em seguida, no Capítulo 4, conceitos relacionados a filogenia de espécies são apresentados, bem como exemplos de filogenia e a sua utilidade na determinação de especificidades dos organismos. Nosso software *Orthologsorter* é apresentado no Capítulo 5, incluindo o fluxograma do *pipeline* e as ferramentas computacionais que compõem o *pipeline*. Em seguida, no Capítulo 6, os resultados são mostrados e uma discussão acerca do que foi obtido é apresentada. Também são descritos nesse capítulo os grupos de organismos de interesse, nominalmente *Bacillus cereus* e *Mycobacterium tuberculosis*. No Capítulo 7 concluímos este trabalho. O apêndice traz o manual do usuário para o *Orthologsorter*.

Capítulo 2

Anotação e análise de genomas de bactérias

Neste capítulo descrevemos uma breve introdução sobre os processos de anotação e análise de genomas de bactérias. Na Seção 2.1 apresentamos as três categorias básicas da anotação. Na Seção 2.2 algumas ferramentas disponíveis para auxiliar o processo de anotação de uma sequência. Por fim, descrevemos os arquivos resultantes da anotação que utilizamos como entrada no *pipeline* desenvolvido.

Vale ressaltar que o foco deste trabalho não é a anotação, mas sim a produção de um ferramental que use como entrada os resultados da anotação, em particular o conjunto de proteínas preditas de um genoma. A leitura deste capítulo pressupõe conhecimentos básicos de Biologia Molecular, em especial de síntese de proteínas.

A sequência de DNA armazena informações sobre a biologia dos organismos, e as sequências transcritas em RNA que expressam proteínas devem ser anotadas de forma correta para que possamos inferir corretamente seus dados biológicos. A anotação de um genoma é o processo onde, a partir de uma sequência de DNA ou RNA produzida por sequenciamento e montagem, e empregando-se a análise e interpretação necessárias, busca-se atribuir características biológicas para o entendimento do contexto biológico em que essas se inserem, por exemplo, sua função. A análise da organização estrutural e funcional dessas sequências também compõem a anotação [46, 61, 80].

A fase de análise de um genoma anotado consiste, basicamente, na tentativa de entender quais são as principais características do organismo, a partir dos dados genotípicos obtidos pela anotação. Para alcançar este objetivo, os pesquisadores tipicamente executam programas para a análise das sequências e realizam comparações com fontes de dados externas.

Geralmente o processo de anotação de genomas é realizado em três etapas:

- identificar a localização de possíveis genes dentro do genoma;
- determinar o produto dos genes identificados na etapa anterior; e
- determinar as vias metabólicas e/ou os processos celulares aos quais os genes estão relacionados.

Os processos envolvidos em cada uma destas etapas serão detalhados nas seções seguintes.

2.1 Etapas da anotação

A anotação de um genoma é um processo de múltiplos passos, podendo ser dividido em em três categorias básicas: anotação em nível de nucleotídeos, anotação em nível de proteínas e anotação em nível de processos. A Figura 2.1 ilustra essas etapas. As informações apresentadas nesta seção foram retiradas de [80, 61].

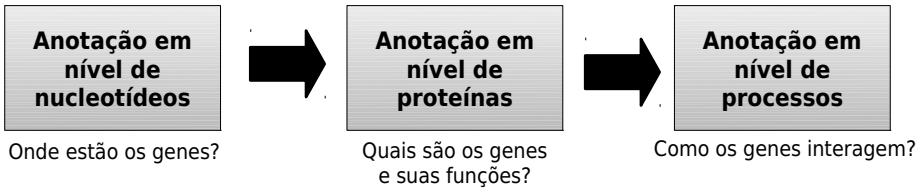


Figura 2.1: Etapas da anotação.

Anotação em nível de nucleotídeos

Uma das etapas mais importantes realizadas durante o processo de análise do genoma de um organismo é a detecção de regiões codificantes e não-codificantes. Gene

é todo o segmento de DNA que contém informação para a síntese de uma cadeia polipeptídica. A região do gene que é transcrita pode ser separada em codificante (contém informação para a estrutura primária de um polipeptídio) e não-codificante (sequências que não serão traduzidas). Quando se obtém o sequenciamento de um genoma, a primeira coisa a ser feita nessa etapa é caracterizar as sequências obtidas.

Devemos descobrir se tal sequência pertence a uma região gênica, se representa uma molécula de RNA transportador ou RNA ribossômico, se pertence a algum tipo de região repetitiva ou se apresenta algum marcador genético conhecido. O principal objetivo dessa etapa é construir um mapa do genoma do organismo, posicionando cada uma das possíveis regiões codificantes e caracterizando as regiões não-codificantes.

Em genoma de procariotos, cuja a maior parte é composta por regiões codificadoras, a localização de possíveis genes é feita, basicamente, através da identificação do conjunto de *Open Reading Frames* (ORFs). Uma ORF é uma subsequência do DNA, que não é necessariamente um gene, definida por um códon de inicialização (por exemplo, ATG) e um códon de finalização (TAA, TAG ou TGA) na mesma fase de leitura. Fase de leitura é uma das seis opções de se ler o código genético em uma molécula de DNA dupla fita. Entretanto, em bactérias, códons de inicialização alternativos podem ser empregados, tais como GTG ou TTG. O tamanho delimitado por esta janela de leitura é frequentemente utilizado para definir uma determinada região como sendo gênica ou não.

Bactérias e *Archaea* são caracterizadas por uma elevada densidade de genes, ausência de íntrons, e poucos DNA repetitivos. Portanto, o problema de encontrar genes em bactérias é relativamente simples se comparado com genomas de eucariotos.

Em eucariotos, a região do gene que é transcrita pode ser separada em codificante (contém informação para a estrutura primária de um polipeptídio) e não-codificante (sequências que não serão traduzidas). Nesses genomas, a tarefa de identificar genes é mais complicada já que apenas uma pequena porcentagem do genoma é composta por regiões gênicas. Além disso, a existência de íntrons que podem alcançar dezenas de quilobases e de formas alternativas de *splicing* dos mRNAs dificulta ainda mais a correta identificação dos genes. Alguns programas são capazes de identificar, dependendo do genoma analisado, regiões gênicas codificadoras (éxons) e não codificadoras (íntrons).

Anotação em nível de proteínas

Mapeados os genes, a etapa seguinte consiste em identificar quais proteínas são codificadas, o que consiste o processo de anotação das sequências proteicas. Nessa etapa, procura-se montar um catálogo dos genes presentes no organismo estudado, dando-lhes nomes e associando-os a prováveis funções.

Apenas um número pequeno de proteínas possui sua função bem caracterizada e, devido a isso, os softwares anotadores de sequências geralmente classificam-nas em grupos ou famílias a partir de similaridades encontradas com proteínas já anotadas de outras espécies, sendo essa uma rica fonte para a anotação funcional.

É bom lembrar, entretanto, que nem sempre proteínas da mesma família possuem funções similares. Sequências proteicas de uma mesma espécie incluídas na mesma família são consideradas parálogas. Sequências parálogas consistem em cópias de um mesmo gene, localizadas em *loci* diferentes, produzidas por duplicação de uma sequência ancestral [66]. Tipicamente têm função idêntica ou similar, mas, algumas vezes, isso não acontece devido à falta da pressão seletiva original sobre uma cópia do gene duplicado. Esta cópia é livre para sofrer mutações e adquirir novas funções. Mais detalhes acerca desses conceitos serão vistos no Capítulo 3.

O termo "família" refere-se a um grupo de sequências consideradas similares em virtude dos valores da pontuação de alinhamento, *e-value* e cobertura. O processo de detecção de famílias de proteínas a partir das relações de similaridade do conjunto de sequências proteicas é denominado "clusterização de sequência".

Uma forma comum de procurar similaridade entre proteínas é utilizando ferramentas como o Blastp [6], buscando informação em diferentes bancos de dados de proteínas. Dentre os bancos de dados de proteínas mais usados estão o Genbank [12], UniProtKB/SwissProt [45] e *Protein Data Bank* [13].

Um processo típico de anotação de proteínas irá buscar primeiramente similaridades com outras proteínas já anotadas. Uma análise complementar seria a procura de domínios funcionais. As bases de dados mais utilizadas nesse processo podem ser vistas na Tabela 2.2. Esses vários bancos de dados de padrões são altamente sobreponíveis, mas cada um possui seu próprio sistema de nomenclaturas e método de procura, o que torna difícil a interpretação dos resultados. Por isso, foi desen-

volvido um banco integrado de assinaturas de proteínas, conhecido como *InterPro*, que procura integrar as informações dos bancos anteriormente citados.

Devido aos bancos de dados de proteínas serem altamente sobrepostos, com diferentes nomenclatura e métodos de busca, é difícil termos uma informação confiável, comprovada experimentalmente da função de proteínas preditas. Uma outra abordagem para obter a função de uma proteína é construir sua estrutura tridimensional a partir da sequência. O *Protein Data Bank* (PDB) armazena estruturas $3D$ de proteínas e macromoléculas. O problema dessa abordagem é que a obtenção da estrutura $3D$ a partir da sequência é muito complexa. Dentro desse contexto, a anotação funcional por similaridade de sequência ajuda a esclarecer a função de uma proteína sem precisarmos obter sua estrutura $3D$.

Anotação em nível de processos biológicos

O maior desafio na anotação de um genoma é associar as proteínas preditas aos processos biológicos ocorridos dentro de uma célula. Esta última etapa visa identificar como os genes e as proteínas se associam ao ciclo celular, metabolismo e à manutenção da vida do organismo.

Identificados os genes, devemos agora tentar relacioná-los de modo a obtermos um mapa funcional do organismo estudado. Processos podem conter diversos genes ativos simultaneamente que possibilitam que determinada ação seja realizada pela célula. Alterações nos genes podem levar a falta de proteínas importantes impossibilitando a realização de um ou mais processos. Nesse ponto devem-se identificar quais vias bioquímicas estão completas ou incompletas no organismo e quais vias alternativas ele possui.

Nesta etapa é fundamental a participação de biólogos especialistas em diversas áreas para que se possa descobrir como o metabolismo do organismo pode influenciar seu modo de vida e seu comportamento. Esse é o momento onde é possível levantar várias hipóteses que relacionem o funcionamento dos organismos com seus dados genômicos. Tais hipóteses devem ser testadas experimentalmente, por pesquisadores que trabalhem com o organismo estudado.

Uma das principais técnicas utilizadas visa analisar mutações ocorridas em genes, e

verificar se estas mutações influenciam a ação do gene. Em caso afirmativo, é possível determinar se um gene participa ou não de um determinado processo, analisando quais ações deixaram de ocorrer em detrimento da mutação. Como resultado desta análise obtém-se uma rede de relações funcionais entre genes, denominada rede gênica.

2.2 Principais ferramentas de anotação

O processo de anotação gênica pelo qual um genoma passa é, na prática, a execução de programas de bioinformática que visam ajudar os biólogos em suas pesquisas, poupando-lhes tempo e trabalho, acelerando assim a interpretação biológica de cada uma das sequências sendo estudadas.

São diversos os softwares destinados às mais diferentes manipulações das grandes quantidades de dados gerados pelos projetos de sequenciamento. Existem softwares destinados a catalogar e comparar sequências de nucleotídeos e aminoácidos [6], a identificar e prever localização de genes [68, 71], a clusterizar [47, 88], entre outros. Os bancos de dados também são essenciais para o gerenciamento e identificação de padrões no grande volume de dados biológicos que cresce exponencialmente. O *NCBI* (*The National Center for Biotechnology Information*) nos Estados Unidos, e o *EBI* (*European Bioinformatics Institute*) na Inglaterra são os dois principais servidores de informações biológicas. Eles mantêm bancos de dados e softwares que servem como ferramentas para toda a comunidade científica, que também submete seus dados a esses bancos, diariamente, disponibilizando-os para uso público.

Como dito anteriormente, a partir do momento que obtemos a sequência de nucleotídeos a ser analisada, o primeiro passo da anotação é descobrir a localização dos genes na sequência.

Vários programas foram criados para manipular sequências de nucleotídeos a fim de realizar a predição gênica, dentre eles podemos citar o *Orfinder* [68] e o *Glimmer* [71]. Em procariotos, os programas consistem tipicamente de uma busca por ORFs (*Open Reading Frames*), que correspondem a sequências com possíveis regiões codificadoras.

Além dos genes que codificam proteínas, é preciso ainda determinar onde estão os

genes que codificam RNAs diferentes do RNAm. Estes genes são transcritos porém não são traduzidos. A própria sequência transcrita de um RNA já desempenha seu papel na célula. São várias as funções de sequências de RNA, como os tRNAs (RNAs transportadores) e os rRNAs (RNAs ribossomais), entre outros. Muitos programas e bancos estão disponíveis na literatura para tal fim. Como principal exemplo, podemos citar o programa chamado Infernal, baseado nos chamados modelos de covariância [56].

Após obtermos a localização dos genes, precisamos caracterizar biologicamente as sequências obtidas. Isto é feito por comparação de sequências, utilizando sequências previamente anotadas disponíveis em bancos públicos de sequências biológicas. A comparação entre sequências biológicas (ácidos nucleicos e proteínas) permite que relações evolutivas, estruturais e funcionais existentes entre as sequências comparadas sejam reveladas, fornecendo evidências decisivas para a caracterização das propriedades biológicas de novas sequências com base no conhecimento acumulado sobre outras já estudadas.

Blast

O Blast (*Basic Local Alignment Search Tool*) [6] é utilizado na comparação e alinhamento local de sequências de nucleotídeos ou de aminoácidos depositadas em bancos de dados. É a ferramenta computacional mais utilizada e de muita importância para a bioinformática.

O procedimento mais comum e difundido para a análise de sequências é a utilização do programa Blast para identificar rapidamente a existência de sub-sequências semelhantes entre si. Estas semelhanças podem indicar homologia e permitir a inferência de função, ou seja, através da similaridade identificada pode-se atribuir - ainda que provisoriamente - uma função à sequência sem a necessidade de realização imediata de experimento biológico.

A ferramenta Blast (*Basic local alignment search tool*) [6], como o nome indica, é utilizada na busca de similaridade local entre sequências biológicas (DNA ou aminoácidos). Para tanto, o Blast utiliza uma heurística, isto é, um algoritmo que toma atalhos (baseados em alguma restrição) para encontrar mais rapidamente uma dentre as possíveis soluções corretas de um problema. Dessa maneira, um algoritmo

heurístico não procura pelo melhor resultado possível, e sim por qualquer resultado que obedeça às restrições impostas para ser considerado como correto ou próximo do melhor resultado, em um tempo satisfatório.

O alinhamento de sequências consiste no processo de comparar duas ou mais sequências (de nucleotídeos ou aminoácidos) de forma a se observar seu nível de similaridade. No que diz respeito ao conceito de alinhamento local, este pode ser definido como a comparação de subsequências das sequências comparadas.

Uma busca utilizando Blast compara a sequência de interesse (*query* ou sequência de entrada) introduzida pelo usuário, contra um banco de dados (*subject*), tentando encontrar alinhamentos locais estatisticamente significativos. A origem da sequência de interesse, assim como a pergunta que se tenta responder, irão determinar o banco de dados a ser utilizado.

De forma bastante resumida, o algoritmo Blast pode ser dividido em três estágios básicos [5]. No primeiro estágio é gerada uma lista de todas as "palavras" encontradas na sequência que está sendo buscada. "Palavra", no caso do Blast, é um trecho de comprimento definido da sequência. Esta lista é refinada através do cálculo da pontuação do alinhamento, sem *gaps* (i.e., apenas comparando palavras inteiras, ou seja, contínuas, sem interrupção), destas palavras contra todas as palavras possíveis de uma sequência. Caso as sequências em questão sejam formadas por aminoácidos, o algoritmo calcula a pontuação bruta(*raw score*) da palavra por meio da soma da pontuação de cada par de aminoácido alinhado [5].

A referência para atribuir pontos a cada par de palavras alinhadas é através de uma matriz de substituição, a qual armazena todas as possíveis substituições para os aminoácidos existentes e a pontuação que deve ser atribuída para cada par de aminoácido alinhado. Caso as sequências sejam formadas por nucleotídeos, não são utilizadas matrizes de substituição, mas sim, um sistema de pontos para *match* (par idêntico) e *mismatch* (par não idêntico). Pontos negativos são atribuídos devido à inserção de lacunas (*gaps*) para que se possa obter um alinhamento ótimo.

Tal pontuação, obtida pela soma dos pontos de cada par, pode ser considerada como uma medida da similaridade entre as sequências comparadas [5]. Um vez que o algoritmo tenha calculado tal pontuação para todas as palavras, apenas "palavras" com pontuação acima de um determinado limiar (valor mínimo) são selecionadas,

isto é, as palavras que são idênticas ou similares a subsequências da sequência *query*. Estas palavras formam o conjunto de buscas a serem realizadas pelo algoritmo em um banco de dados.

No segundo estágio, palavras idênticas às presentes na lista são procuradas nas sequências do banco de dados. Ao encontrar alguma subsequência de uma sequência do banco de dados que alinhe a qualquer uma das palavras selecionadas, tem-se um *match*. Como o banco de dados teve sua lista de palavras pré-compilada e indexada no momento em que foi criado, não no instante em que se executa a busca, este passo é extremamente rápido.

No último estágio, as extremidades dos *matches* são estendidas. Inicialmente o algoritmo estendia ambas as extremidades de todos os *matches* encontrados como configuração padrão, porém, um refinamento posterior do algoritmo permite ao usuário escolher se as extensões serão realizadas segundo o critério de existência de dois *matches* não sobrepostos proximamente localizados [6], o que torna o tempo de execução do algoritmo ainda menor.

O algoritmo estende um *match* ou *hit* enquanto a pontuação permanecer alta para aquele alinhamento, mas é interrompida quando são detectadas variações que levem o valor abaixo de um limiar de pontuação estabelecido para as extensões [5].

Como resultado, é exibida uma lista contendo todos os *hits* acompanhados de seus alinhamentos, juntamente com o *score*, também chamado de *raw-score*, uma estimativa de significância, denominada de *e-value*, além da porcentagem de identidade.

O *score* é a pontuação atribuída pelo algoritmo e baseada no número de pareamentos perfeitos (*match*) e imperfeitos (*mismatch*) entre uma subsequência da sequência de entrada e um segmento de alguma sequência do banco de dados. É consequência do número de inserções, deleções e substituições neste pareamento. Contudo, mesmo quando o *score* é alto e, em princípio, melhor o pareamento, este valor não pode ser analisado individualmente, mas acompanhado de medida estatística (*e-value*). Enfim, o valor do *score* dá uma indicação se o alinhamento é bom ou não, sendo o seu valor positivamente correlacionado com a qualidade deste alinhamento (ou seja, quanto maior, melhor).

Uma medida do Blast a ser considerada é o *bit-score*, o qual é calculado por meio

da normalização da pontuação bruta (*raw-score*). O *bit-score* permite a comparação de *score* entre alinhamentos de sistemas de pontuação ou matrizes diferentes, bem como alinhamentos de sequências que diferem no comprimento ou no tamanho do banco de dados onde estão armazenadas [6].

Já o *e-value* representa o valor estatístico que indica se o alinhamento é real ou foi obtido meramente pelo acaso naquele banco de dados ("falso positivo"). Em outras palavras, é o número esperado de falsos positivos que obteriam *score* igual ou maior que o reportado em um determinado alinhamento entre a sequência de entrada e uma do banco. Fundamentalmente, quanto menor o *e-value*, menores as chances daquele resultado ser consequência do acaso.

Outro dado relevante é a identidade que corresponde ao número *matches* identificados no alinhamento e expresso, em porcentagem, a partir da comparação com o comprimento deste alinhamento. Nos resultados do Blast, são a soma do número de aminoácidos idênticos e aqueles que são diferentes na comparação mas que apresentam *score* positivo na tabela empregada.

De acordo com a necessidade do usuário, a natureza dos dados de entrada (nucleotídeo ou aminoácido) e do banco de referência, existe um programa Blast específico: o Blastp (compara a sequência de aminoácidos de entrada (*query*) contra um banco de dados de sequências de proteínas (*subject*)), Blastn (compara a sequência de nucleotídeos de entrada contra um banco de dados de sequências de nucleotídeos), Blastx (compara a sequência de nucleotídeos de entrada traduzida para todas as sequências de leitura possíveis contra um banco de dados de sequências de proteínas), TBlastn (compara a sequência de aminoácidos de entrada contra um banco de dados de sequências de nucleotídeos traduzidas para todas as sequências de leitura possíveis) e TBlastx (Compara as seis sequências de leitura possíveis de nucleotídeos contra um banco de dados de nucleotídeos traduzidos para todas as sequências de leitura possíveis).

Neste trabalho, o *pipeline* desenvolvido utiliza em seu primeiro passo o Blastp, cuja comparação é do tipo "todos contra todos". Para executar o programa Blastp é necessário criar o banco de dados local das proteínas preditas dos organismos estudados. Após obtido o banco de dados, as comparações do tipo "todos contra todos" envolvem a comparação de cada uma das sequências de um banco contra todas as demais sequências pertencentes ao banco. Deste modo, por meio de alinhamentos

de pares (*pairwise alignment*) de todas as sequências de cada banco, o Blastp obteve as relações de similaridade de sequências entre as proteínas de cada complexo bacteriano.

Após a identificação de cada gene, é necessário obter a sua classificação. Existe uma grande diversidade de nomes e sinônimos para entidades biológicas que resulta em uma anotação confusa. Para uniformizar o vocabulário utilizado na anotação, algumas iniciativas desenvolvem ontologias para a descrição de processos e produtos biológicos, gerando uma coleção de termos (vocabulário) controlados o que minimiza problemas de nomenclatura na anotação. O *Gene Ontology* [10] é um repositório deste tipo, sendo um mecanismo eficiente para classificação dos genes. Cada gene é classificado em três níveis: função molecular, processos celulares e localização celular.

Na Tabela 2.1 são apresentados os principais programas utilizados na anotação de genomas e a suas características principais.

Tabela 2.1: Principais programas de anotação

Programa	**Característica principal**
Orfinder http://www.ncbi.nlm.nih.gov/projects/gorf/ Glimmer http://www.cbcb.umd.edu/software/glimmer/	Busca por ORFs
Blast http://blast.ncbi.nlm.nih.gov/Blast.cgi	Comparações de sequências
Gene Ontology http://www.geneontology.org/ Interpro http://www.ebi.ac.uk/interpro	Classificação dos genes
ClustalW http://www.genome.jp/tools/clustalw/ MUSCLE http://www.drive5.com/muscle/	Alinhamento múltiplo
Mega http://www.megasoftware.net/ Phylip http://evolution.genetics.washington.edu/phylip.html	Inferências filogenéticas

Existem também, bancos de dados aplicados ao estudo de proteínas. Na Tabela 2.2 são apresentados os bancos de dados de famílias de proteínas mais conhecidos.

Tabela 2.2: Bancos de dados de famílias de proteínas

Banco de Dados	
PROSITE	http://prosite.expasy.org/
PRINTS	http://www.bioinf.manchester.ac.uk/dbbrowser/PRINTS/index.php
BLOCKS	http://blocks.fhcrc.org/blocks/blocks_search.html
PROFILES	http://hits.isb-sib.ch/cgi-bin/PFSCAN
Pfam	http://pfam.sanger.ac.uk/
ProDom	http://prodom.parabi.fr/prodom/current/html/home.php
InterPro	http://www.ebi.ac.uk/Tools/pfa/iprscan/

2.3 Resultados da anotação

Na prática, como resultado da anotação, espera-se um conjunto de arquivos que armazena todos os genes, suas respectivas posições no genoma e sequências e finalmente os respectivos produtos resultantes.

No escopo deste trabalho, estaremos interessados em dois arquivos padronizados pelo *Genbank*, com extensões '`.ptt`' e '`.faa`'.

O arquivo com extensão '`.ptt`' é um arquivo texto que contém as informações dos genes anotados de um genoma. A partir deste arquivo obtemos a localização dos genes (posição inicial e final na fita '+'), a fita em que foi obtido (fita '+' ou '−'), o número de aminoácidos, excluindo o códon de terminação, o gene e o produto gênico, análogo ao *Genbank* [12]. O gene que foi obtido a partir da fita '−' tem posição inicial e final correspondente a fita '+'. A sequência de aminoácidos deste gene é obtida pela sequência complementar reversa à fita '+'.

A utilização do arquivo '`.ptt`' é necessária em geral quando a análise comparativa envolve aspectos de sintenia compartilhada. O termo "sintenia" refere-se à conservação na ordem e no conteúdo de genes, ou grupos gênicos, entre espécies relacionadas.

No caso de genomas incompletos, onde em geral têm-se apenas os *contigs* ou *scaffolds* anotados, esse arquivo não faz sentido e portanto não é utilizado.

Já o arquivo '`.faa`' traz todas as sequências de genes já traduzidas, no formato fasta. Cada sequência representada nesse arquivo corresponde biunivocamente à uma linha no arquivo '`.ptt`'. As Figuras 2.2 e 2.3 mostram, respectivamente, trechos iniciais desses arquivos do cromossomo circular da bactéria *Bacillus cereus* ATCC 10987.

```
Bacillus cereus ATCC 10987, complete genome - 1..5224283
5603 proteins
Location Strand Length PID Gene Synonym Code COG Product
408..1748 + 446 42779082 dnaA BCE_0001 -COG0593L chromosomal replication initiator protein DnaA
1927..3066 + 379 42779083 dnaN BCE_0002 -COG0592L DNA polymerase III, beta subunit
3182..3406 + 74 42779084 -BCE_0003 -COG2501S hypothetical protein
3418..4545 + 375 42779085 recF BCE_0004 -COG1195L DNA replication and repair protein RecF
```

Figura 2.2: Trecho inicial do arquivo `NC_003909.ptt`, do cromossomo circular de *Bacillus cereus* ATCC 10987.

```
>gi|42779082|ref|NP_976329.1| chrom. rep. initiator protein DnaA [Bacillus cereus ATCC 10987]
MENISDLWNSALKELEKKVSKPSYETWLKSTTAHNLKKDVLTITAPNEFARDWLESHYSELISETLYDLT
GAKLAIRFIIPQSQAEEEIDLPPSKPNSAQDDSNHLPQSMLNPKYTFDTFVIGSGNRFAHAASLAVAEAP
AKAYNPLFIYGGVGLGKTHLMHAIGHYVIEHNPNAKVVYLSSEKFTNEFINSIRDNKAVDFRNKYRNVDV
LLIDDIQFLAGKEQTQEEFFHTFNALHEESKQIVISSDRPPKEIPTLEDRLRSRFEWGLITDITPPDLET
RIAILRKKAKAEGLDIPNEVMLYIANQIDSNIRELEGALIRVVAYSSLINKDINADLAAEALKDIIPNSK
PKIISIYDIQKAVGDVYQVKLEDFKAKKRTKSVAFPRQIAMYLSRELTDSSLPKIGEEFGGRDHTTVIHA
HEKISKLLKTDTQLQKQVEEINDILK
>gi|42779083|ref|NP_976330.1| DNA polymerase III, beta subunit [Bacillus cereus ATCC 10987]
MRFTIQKDYLVRSVQDVMKAVSSRTTIPILTGIKVVATEEGVTLTGSDADISIESFIPVEENGKEIVEVK
QSGSIVLQAKYFSEIVKKLPKETVEISVENHLMTKITSGKSEFNLNGLDSAEYPLLPQIEEHHVFKIPTD
LLKHMIRQTVFAVSTSETRPILTGVNWKVYNSELTGIATDSHRLALRKAKIEGIADEFQANVVIPGKSLN
ELSKILDESEEMVDIVITEYQVLFRTKHLLFFSRLLEGNYPDTTRLIPAESKTDIFVNTKEFLQAIDRAS
LLARDGRNNVVKLSTLEQAMLEISSNSPEIGKVVEEVQCEKVDGEELKISFSAKYMMDALKALDSTEIKI
SFTGAMRPFLIRTVNDESIIQLILPVRTY
>gi|42779084|ref|NP_976331.1| conserved hypothetical protein [Bacillus cereus ATCC 10987]
MSDFMKRIKISTEYITLGQFLKLADVIDTGGAVKWFLQEYEVYVNQELENRRGRKLYANDIIEIPGSGSF
QVQS
>gi|42779085|ref|NP_976332.1| DNA rep.repair protein RecF [Bacillus cereus ATCC 10987]
MFISEIQLKNYRNYEKLELSFEDKVNVIIGENAQGKTNLMEAIYVLAMAKSHRTSNDRELIRWDEDFGQI
KGKLQKRNSSLSLELNISKKGKKAKLNQLEQQKLSQYIGVMNVVMFAPEDLNLVKGSPQVRRRFLDMELG
QIAPVYLYELSQYQKVLTQRNHLLKKMQGNSKNEETMLDVFTLQLIEHGTKILQKRFEFLHLLQEWAAPI
HRGISRGLEELEIVYKPSVDVSESMDLSKIKEVYYESFQSVKQREIFRGTTLIGPHRDDLQFFVNSKNVQ
VFGSQGQQRTTALSLKLAEIELIYSEVKEYPILLLDDVLSELDDYRQSHLLNTIQGKVQTFVTTTSVDGI
EHETLKEAKTIHVTNGTVDCEIDRA
```

Figura 2.3: Trecho inicial do arquivo `NC_003909.faa`, do cromossomo circular de *Bacillus cereus* ATCC 10987.

O produto do gene é o material bioquímico resultante da expressão de um gene, ou seja, corresponde a função que o gene possui dentro da célula. Para o nosso projeto, a identificação de grupos ortólogos entre múltiplos genomas é essencial, revelando assim padrões filogenéticos de proteínas de linhagens diferentes, além de prover conhecimento evolucionário dos genomas estudados. A sequência proteica e principalmente o produto do gene são fundamentais para distribuir as proteínas a distintos grupos, de modo que as proteínas em um mesmo grupo são mais similares entre si do que aquelas em outros grupos.

Capítulo 3

Famílias de proteínas

Neste capítulo apresentamos uma breve introdução ao conceito de famílias de proteínas. Na Seção 3.1 descrevemos algumas definições usadas na comparação de sequências, em especial na comparação de proteínas. Por fim, na Seção 3.2 apresentamos algumas ferramentas existentes para a determinação de famílias de proteínas.

A disponibilidade de sequência completa dos genomas vem revolucionando a pesquisa de evolução molecular e ampliando o conhecimento sobre a genética e sua influência nos mecanismos celulares. Comparar os conteúdos de genes de genomas diferentes permite inferir funções de proteínas e a maneira pela qual elas evoluíram por caminhos independentes. Esse conhecimento é importante para o entendimento de como surgiram novas funções de proteínas, de como fenótipos são codificados nos genes, ou para predizer quais genes têm a mesma função em organismos diferentes. Mais ainda, a identificação de famílias de proteínas de bactérias tem possibilitado análises aprofundadas acerca de como os organismos comparados interagem com seus hospedeiros [20, 27, 30, 77]. O termo "família", adotado no presente trabalho, não assume o sentido usado em Biologia (categoria taxonômica que reúne gêneros), mas refere-se a um *cluster* ou grupo de sequências consideradas similares em virtude dos valores da pontuação de alinhamento, *e-value* e cobertura.

Durante a evolução, algumas proteínas compartilham características estruturais e funcionais, o que sugere que elas tiveram uma origem em comum, um ancestral, do qual tais proteínas tiveram origem. Nesse caso, diz-se que fazem parte da mesma família. Uma família de proteínas é um grupo de proteínas relacionadas evolutiva-

mente.

Mas a questão é como relacionar essas proteínas. Durante a evolução das proteínas, há regiões que não se alteraram. Procurar por regiões conservadas é um bom começo para se relacionar proteínas. A quantidade de diferenças entre sequências de aminoácidos pode indicar quão recentemente os dois genomas compartilharam um ancestral comum. Dois genomas que divergiram num passado recente tendem a ter menores diferenças. Uma vez separadas as proteínas em famílias, fica fácil relacionar uma nova proteína com uma família e decidir se ela é membro da família ou não. Caso seja, pode-se fazer melhores inferências sobre sua função.

Proteínas em uma mesma família descendem de um ancestral comum e normalmente têm estruturas tridimensionais, funções e sequências com similaridade significativa. Embora seja difícil avaliar a importância da semelhança funcional ou estrutural, métodos de alinhamento de sequências nos permitem avaliar a significância da similaridade entre um grupo de sequências e a identificar os possíveis membros de uma família de proteínas.

O processo de detecção de famílias de proteínas a partir das relações de similaridade do conjunto de sequências proteicas é denominado "clusterização de sequências". Essa abordagem é geralmente baseada em agrupar proteínas entre si através de uma medida de similaridade obtida a partir da comparação direta das sequências. Idealmente, os *clusters* resultantes devem corresponder a famílias de proteínas, cujos membros estão relacionados por uma história evolutiva comum.

3.1 Homologia, ortologia e paralogia

Nesta seção introduzimos algumas definições básicas usadas quando se comparam genomas através de suas proteínas preditas. Homologia, ortologia e paralogia são conceitos referentes à comparação entre genomas ou proteínas e são esses conceitos importantes para compreender como os organismos são relacionados. Homologia, o conceito mais genérico, designa uma relação de ancestralidade comum entre quaisquer entidades. Assim, genes/proteínas relacionadas por homologia são chamados de homólogos. Dentro da homologia, os termos paralogia e ortologia são comumente utilizados.

Sequências homólogas são ortólogas se elas foram separadas por um evento de especiação: quando uma espécie diverge em duas espécies separadas, as cópias de um único gene nas duas espécies resultantes são considerados ortólogos. Ortólogos, ou genes ortólogos, são genes em diferentes espécies que se originaram de um único gene do último ancestral comum. Ortólogos, frequentemente, têm a mesma função. Co-ortólogos são dois ou mais genes em uma linhagem que são, coletivamente, ortólogos a um ou mais genes de outra linhagem, devido a uma duplicação de linhagem específica. O termo linhagem refere-se ao conjunto de indivíduo que descendem de um ancestral comum.

Sequências homólogas são parálogas se elas foram separadas por um evento de duplicação de genes: se um gene de um organismo é duplicado e ocupa duas regiões diferentes no mesmo genoma, então as duas cópias são parálogas. Genes parálogos frequentemente pertencem à mesma espécie, entretanto, existem exceções: por exemplo, o gene da hemoglobina dos seres humanos e o gene da mioglobina de chimpanzés são parálogos. Parálogos podem ser divididos em in-parálogos (gerados por duplicação após especiação) e out-parálogos (gerados por duplicação antes da especiação). Parálogos tipicamente têm função idêntica ou similar, mas, algumas vezes, isso não acontece devido à falta da pressão seletiva original sobre uma cópia do gene duplicado, ficando esta cópia livre para sofrer mutações e adquirir novas funções.

Como mostrado na Figura 3.1, um gene A em uma espécie ancestral sofre uma duplicação seguida de uma especiação, originando as linhagens B e C. $C2$ e $C3$ são in-parálogos devido a suas duplicações terem ocorrido após especiação. São, portanto co-ortólogos a $B2$. $C1$ é out-parálogo a $C2$ e $C3$, já que houve duplicação de A antes da especiação, o mesmo ocorrendo entre $B1$ e $B2$. No entanto, $B1$ e $C1$ são ortólogos.

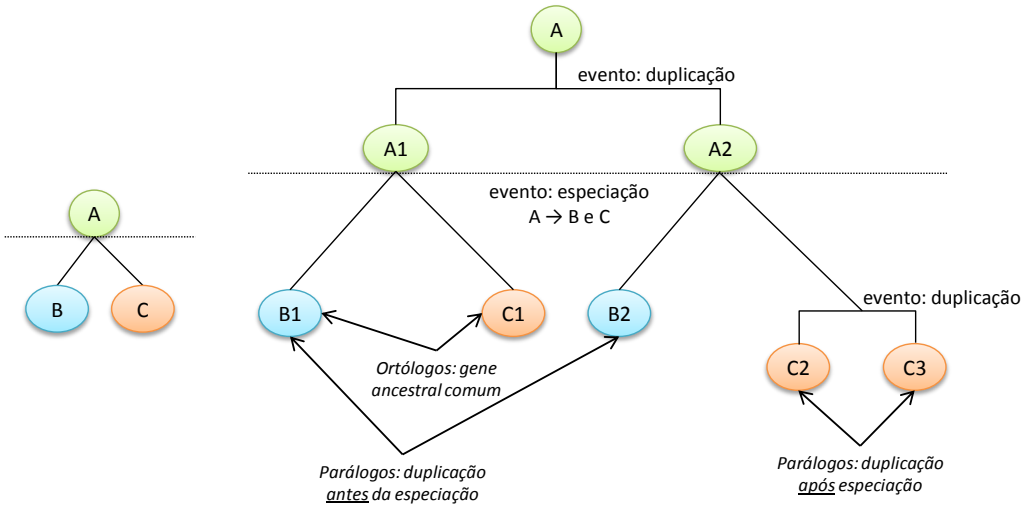


Figura 3.1: Conceitos de ortologia e paralogia. Figura transcrita de [95].

Como o número total de proteínas sequenciadas está crescendo, o interesse na análise dessas proteínas tem aumentado. Por isso, há um esforço contínuo de organizar as proteínas em famílias. A identificação correta das famílias de proteínas é fundamental para a predição confiável de funções dos membros da família, na análise filogenética, na anotação funcional, na exploração da diversidade da função da proteína em um ramo filogenético dado, entre outras aplicações. Em nosso trabalho, queremos encontrar famílias de proteínas homólogas, ortólogas e/ou parálogas, de vários genomas de espécies próximas, com o objetivo de construir, a partir dessas famílias, uma árvore filogenética. As buscas no conjunto de famílias obtidas por algum método de clusterização de sequências são feitas através do software *Orthologsorter*, descrito no Capítulo 5.

3.2 Ferramentas para detecção de famílias de proteínas

Algumas ferramentas, como o OrthoMCL e kGC descritas a seguir, foram criadas para a determinação de famílias de proteínas, a partir da comparação entre essas famílias [47, 88]. Através destas ferramentas obtemos resultados semelhantes aos almejados neste trabalho, porém utilizando procedimentos e tratamentos diferentes.

OrthoMCL

O OrthoMCL [47] é uma ferramenta capaz de construir agrupamentos de ortólogos e parálogos entre dois ou mais organismos. É uma ferramenta utilizada para encontrar regiões que preservam o conteúdo gênico em vários genomas, a partir de comparações feitas "todos contra todos" pelo Blastp. Tais regiões múltiplas, chamadas de regiões ortólogas, podem nos dar pistas sobre funcionalidades comuns de genes ou grupo de genes que se preservam em espécies próximas. OrthoMCL utiliza o Blast para comparações de sequências e o algoritmo de *Markov* (*Markov Cluster* - MCL) para a formação de grupos, o qual é baseado em probabilidade e teoria dos grafos.

O algoritmo de *Markov* recebe como entrada um conjunto de pares com pesos (pode ser visto como um grafo, onde cada elemento do par é um vértice e o peso é uma aresta ligando estes dois vértices com o peso indicado). Este algoritmo é largamente utilizado em Biologia para "clusterizar" sequências de proteínas, ou seja, atribuir as proteínas em grupos (chamados de *clusters*) de modo que as proteínas em um mesmo *cluster* são mais similares entre si do que aquelas em outros *clusters*.

O algoritmo de *Markov* (e vários outros algoritmos de *clustering*) baseia-se na seguinte ideia: considerando um grafo, haverá muitas ligações dentro de um mesmo *cluster*, e poucas ligações entre *clusters* diferentes. Isto significa que, se você tivesse que partir de um nó, e depois aleatoriamente passeasse por outro nó, você estaria mais propenso a permanecer dentro de um *cluster* do que atingir outro *cluster*. Este conceito é chamado de *random walks*. Ao realizarmos *random walks* sobre o grafo, é possível descobrir onde o fluxo se concentra, e, portanto, onde os *clusters* estão. A partir desta ideia, o algoritmo simula caminhos aleatórios em um grafo utilizando matrizes de *Markov* para determinar as probabilidades de fluxo através dos nós. Essa metodologia gera grupos de proteínas, consistindo de paralogias e ortologias entre pelo menos duas espécies. O termo "matriz de *Markov*" refere-se a uma matriz de transição, ou ainda matriz estocástica, sendo uma matriz quadrada onde todas as entradas são não-negativas e todas as colunas tem entradas cuja a soma é igual a 1. As probabilidades de transição que caracterizam um processo de *Markov* são normalmente agrupadas na matriz de *Markov*.

O OrthoMCL pode ser visto como um processo de duas etapas: a primeira envolve a aplicação de regras baseadas no conhecimento biológico do problema para determinar quais sequências podem ser incluídas, como as sequências são conectadas e como os

pesos das arestas podem quantificar o relacionamento entre duas sequências. A segunda etapa consiste do agrupamento. O algoritmo OrthoMCL é apresentado na Figura 3.2.

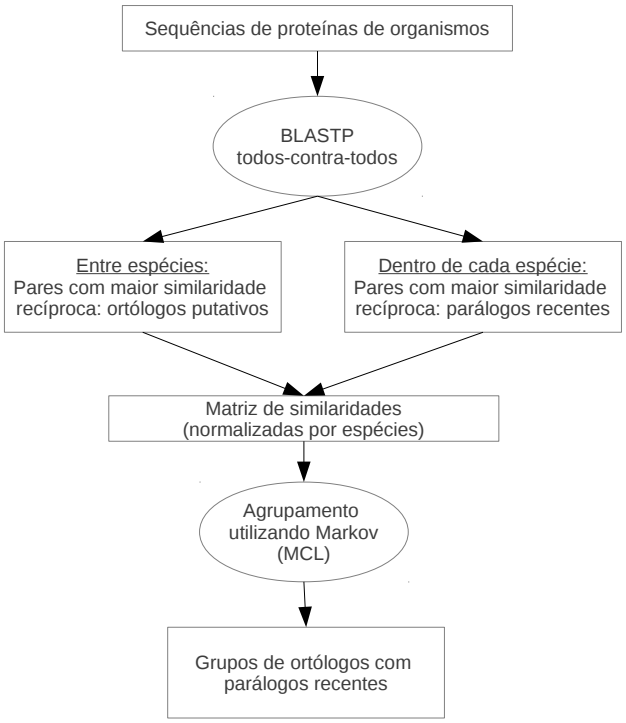


Figura 3.2: Algoritmo OrthoMCL. Figura adaptada de [47].

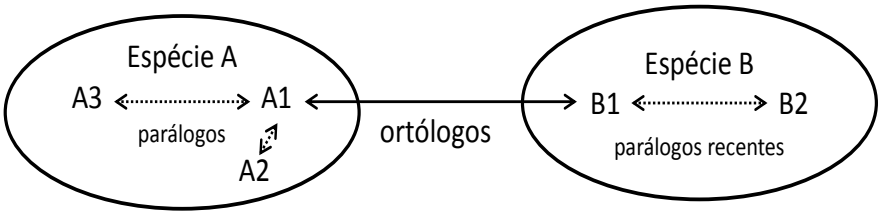
O procedimento do OrthoMCL inicia-se com comparações feitas "todos contra todos" pelo Blastp de um conjunto de sequências de proteínas com os genomas de interesse. Relacionamentos de ortologia e paralogia são identificados entre pares de genes, cujos *e-values* de alinhamento do Blastp são reciprocamente inferiores a 10^{-5}. Posteriormente, estes resultados são convertidos em um grafo no qual os nós representam os genes e as arestas seus relacionamentos.

Devido à alta similaridade obtida nas paralogias em relação à ortologia, o processo de agrupamento pode ser comprometido. Os pesos das arestas são então normalizados por espécies de forma a refletir um peso médio para todo par de ortologia entre duas espécies. A matriz de *Markov* é preenchida com esses valores normalizados através do cálculo de $-\log_{10}(\textit{e-value})$ para cada *e-value* proveniente do alinhamento de pares de genes pelo Blastp, conforme mostra a Figura 3.3.

O algoritmo de *Markov* é então aplicado sobre esta matriz de forma a obter as

ortologias e paralogias respectivamente. Como resultado, este procedimento provê grupos de genes ortólogos entre pelo menos duas espécies com seus respectivos genes parálogos.

A Figura 3.3 exemplifica um caso em que duas espécies, A e B, possuem uma ortologia entre os genes $A1$ e $B1$, além das paralogias na espécie A entre ($A1$, $A3$) e ($A1$, $A2$) e na espécie B entre ($B1$, $B2$). Note que $A2$ está muito mais próxima de $A1$ do que $A3$, e que $A1$ foi escolhido como gene ortólogo a $B1$. Estas escolhas são justificadas pelos valores encontrados na matriz de similaridade normalizada por espécies, onde os maiores valores evidenciam os pares descritos.



Matriz de Similaridade = Matriz de Markov

	A1	A2	A3	B1	B2
A1		***300***	***152***	***61***	**29**
A2	***233***		***150***	**60**	**29**
A3	***118***	***117***		**40**	**30**
B1	**69**	**68**	**45**		***88***
B2	**33**	**33**	**34**	***100***	

Pesos normalizados corrigidos por MCL

Pesos iniciais: -log$_{10}$ (e-value) similaridades BLASTP

Figura 3.3: Ilustração dos relacionamentos entre as sequências e a construção da matriz de similaridade. Setas pontilhadas representam paralogia "recente" (duplicação subsequente à especiação); setas sólidas representam ortologia. A metade superior direita da matriz contém os pesos iniciais calculados como a média de $-\log_{10}(\textit{e-value})$ a partir das comparações *pairwise* pelo WU-Blastp. A metade inferior esquerda contém os pesos corrigidos fornecidos ao algoritmo MCL. O peso da aresta de cada par de sequências w_{ij} é dividido por $\frac{W_{ij}}{W}$, onde W representa o peso médio dentre todos os ortólogos (sublinhado) e pares de parálogos "recentes" (itálico), e W_{ij} representa o peso médio da aresta dentre todos os pares de ortólogos a partir das espécies i e j. O resultado desta normalização corrige as diferenças sistemáticas na comparação entre duas espécies (por exemplo, diferenças que podem ser atribuídas ao *bias* da composição de nucleotídeos), e quando $i = j$, para minimizar o impacto de parálogos "recentes" (duplicação dentro de uma determinada espécie) no agrupamento de espécies ortólogas. Figura transcrita de [47].

A Figura 3.4 mostra o trecho inicial da saída gerada pelo OrthoMCL para o grupo *Bacillus cereus*.

```
ORTHOMCL0(41 genes,8 taxa):  gi|152974714|ref|YP_001374231.1|
(Bacillus_cytotoxicus_NVH_391_98) gi|152975323|ref|YP_001374840.1|
(Bacillus_cytotoxicus_NVH_391_98) gi|152975969|ref|YP_001375486.1|
(Bacillus_cytotoxicus_NVH_391_98) gi|152975970|ref|YP_001375487.1|
(Bacillus_cytotoxicus_NVH_391_98) gi|157691852|ref|YP_001486314.1|
(Bacillus_pumilus_SAFR_032) gi|163939082|ref|YP_001643966.1|
(Bacillus_weihenstephanensis_KBAB4) gi|163939088|ref|YP_001643972.1|
(Bacillus_weihenstephanensis_KBAB4) gi|163939867|ref|YP_001644751.1|
(Bacillus_weihenstephanensis_KBAB4) gi|163941190|ref|YP_001646074.1|
(Bacillus_weihenstephanensis_KBAB4) gi|163941191|ref|YP_001646075.1|
(Bacillus_weihenstephanensis_KBAB4) gi|163941192|ref|YP_001646076.1|
(Bacillus_weihenstephanensis_KBAB4) gi|228990265|ref|ZP_04150232.1|
(Bacillus_pseudomycoides_DSM_12442) gi|228990273|ref|ZP_04150240.1|
(Bacillus_pseudomycoides_DSM_12442) gi|228991060|ref|ZP_04151021.1|
(Bacillus_pseudomycoides_DSM_12442) gi|228992275|ref|ZP_04152207.1|
(Bacillus_pseudomycoides_DSM_12442) gi|228992276|ref|ZP_04152208.1|
(Bacillus_pseudomycoides_DSM_12442) gi|228992277|ref|ZP_04152209.1|
(Bacillus_pseudomycoides_DSM_12442) gi|229010574|ref|ZP_04167776.1|
(Bacillus_mycoides_DSM_2048) gi|229010580|ref|ZP_04167782.1|
(Bacillus_mycoides_DSM_2048) gi|229011367|ref|ZP_04168558.1|
(Bacillus_mycoides_DSM_2048) gi|229012761|ref|ZP_04169931.1|
(Bacillus_mycoides_DSM_2048) gi|229012762|ref|ZP_04169932.1|
(Bacillus_mycoides_DSM_2048) gi|229012764|ref|ZP_04169933.1|
(Bacillus_mycoides_DSM_2048) gi|30261290|ref|NP_843667.1|
(Bacillus_anthracis_AE016879) gi|30261296|ref|NP_843673.1|
(Bacillus_anthracis_AE016879) gi|30262066|ref|NP_844443.1|
(Bacillus_anthracis_AE016879) gi|30263533|ref|NP_845910.1|
(Bacillus_anthracis_AE016879) gi|30263534|ref|NP_845911.1|
(Bacillus_anthracis_AE016879) gi|30263535|ref|NP_845912.1|
(Bacillus_anthracis_AE016879) gi|42780374|ref|NP_977621.1|
(Bacillus_cereus_ATCC_10987) gi|42780380|ref|NP_977627.1|
(Bacillus_cereus_ATCC_10987) gi|42782651|ref|NP_979898.1|
(Bacillus_cereus_ATCC_10987) gi|42782652|ref|NP_979899.1|
(Bacillus_cereus_ATCC_10987) gi|42782653|ref|NP_979900.1|
(Bacillus_cereus_ATCC_10987) gi|478733102|ref|NP_978423.3|
(Bacillus_cereus_ATCC_10987) gi|49477118|ref|YP_035420.1|
(Bacillus_thuringiensis_serovar_konkukian_97_27) gi|49477123|ref|YP_035426.1|
(Bacillus_thuringiensis_serovar_konkukian_97_27) gi|49478278|ref|YP_037662.1|
(Bacillus_thuringiensis_serovar_konkukian_97_27) gi|49478282|ref|YP_037667.1|
(Bacillus_thuringiensis_serovar_konkukian_97_27) gi|49479250|ref|YP_037666.1|
(Bacillus_thuringiensis_serovar_konkukian_97_27) gi|49481243|ref|YP_036188.1|
(Bacillus_thuringiensis_serovar_konkukian_97_27)
```

Figura 3.4: Trecho inicial do arquivo de saída gerado pelo OrthoMCL para o grupo *Bacillus cereus*.

Identificação de grupos ortólogos entre múltiplos genomas pode ser bastante útil para a anotação genômica, revelando padrões filogenéticos de proteínas de linhagens diferentes, além de prover conhecimento evolucionário.

kGC

O kGC [88] consiste em um algoritmo recentemente desenvolvido para a construção de grupos de genes homólogos em múltiplos organismos simultaneamente. O algoritmo kGC propõe-se a encontrar famílias de proteínas a partir das relações de similaridade do conjunto de sequências proteicas.

Dados k genomas, onde cada genoma é um conjunto de sequências de genes, a entrada para o kGC é o resultado das comparações das sequências proteicas (contendo pontuações de similaridades de sequência) de "todos-contra-todos" no Blastp, e a partir desses, cria um grafo de similaridades de sequência.

Um grafo G é formado por dois conjuntos, um composto de elementos chamados vértices (V) e outro composto de elementos chamados arestas (E). Cada aresta corresponde a um par não orientado de vértices. Considera-se um conjunto C, pertencente ao conjunto V, uma clique do grafo G se qualquer par de vértices pertencente a C é também pertencente a E. Uma clique é dita maximal se esta deixar de ser clique devido ao acréscimo de qualquer vértice não pertencente a C [2].

Com base na identificação de cliques maximais em grafos de similaridades de sequências, a saída do kGC é uma coleção de grupos formados por sequências similares. Os experimentos, descritos em [88], mostram que o kGC é um método simples, com um pequeno número de parâmetros, tempo de execução razoável e produz resultados com qualidade, quando comparado ao OrthoMCL.

Capítulo 4

Filogenia de espécies

Todas as espécies de organismos existentes na Terra passaram por um processo de transformação ao longo dos tempos, sendo tal processo chamado de evolução. Um dos problemas centrais da Biologia, conhecido como problema da filogenia, é explicar a história evolutiva das espécies hoje existentes, bem como verificar relacionamentos entre essas espécies, a fim de determinar possíveis ancestrais comuns entre elas.

Na Seção 4.1 descrevemos o problema da construção de filogenias baseadas em distâncias. Em particular, descrevemos a construção de árvores aditivas e ultramétricas. Apresentamos também o problema da construção de filogenias baseadas em características, adotado em nosso trabalho. Na Seção 4.2 explicamos como o problema de construção de um alinhamento múltiplo e o de determinar uma árvore filogenética estão ligados. Por fim, apresentamos na Seção 4.3 publicações que também adotaram, com algumas diferenças, a abordagem utilizada nesse trabalho para construir de filogenia de espécies.

Filogenia é o termo comumente utilizado para hipóteses de relações evolutivas (ou seja, relações filogenéticas) de um grupo de objetos, isto é, a filogenia mostra relações ancestrais entre os objetos. Vamos nos referir às espécies e outras taxonomias como objetos.

Para tentar explicar fatos como os mencionados acima, frequentemente utilizamos o conceito de árvores. Uma árvore filogenética é uma representação gráfica, em forma de uma árvore, das relações evolutivas entre vários objetos que possam ter

um ancestral comum. Assim árvores filogenéticas são estruturas que expressam a ancestralidade e relacionamentos entre os objetos ou grupos de objetos [9].

Em uma árvore filogenética, cada nó com descendentes representa o mais recente antepassado comum, e os comprimentos dos ramos podem representar estimativas do tempo evolutivo. Cada nó terminal, ou folha, em uma árvore filogenética é chamado de "unidade taxonômica", representando os objetos. Os nós internos geralmente são chamados de "unidades taxonômicas hipotéticas" e representam os supostos ancestrais. As árvores filogenéticas são confeccionadas a partir de uma matriz contendo os dados disponíveis (morfológicos, químicos ou genéticos) sobre os objetos estudados. Estes dados são comparados, e os objetos agrupados de acordo com as semelhanças e diferenças entre si. Existem vários softwares disponíveis para a construção destas árvores. Dentre eles podemos citar o RAxML [78], PHYLIP [29] e MEGA [42]. Neste trabalho foi utilizado o RAxML para a construção e o NJplot [60] como software de visualização da árvore filogenética.

A construção de árvores filogenéticas auxilia na explicação dos possíveis relacionamentos entre as espécies atuais e na dedução das histórias evolutivas delas. Técnicas filogenéticas são usadas para construir as árvores as quais são usadas para compreender a origem e diferenciação de macromoléculas em grandes grupos, auxiliando na resolução de problemas práticos, tais como o controle e combate de parasitas responsáveis por doenças.

Em geral, o conhecimento biológico pode ser expresso usando o enfoque filogenético, de modo que a geração de árvores filogenéticas contribui imensamente para inferir a história evolutiva das espécies [61].

Neste trabalho, a filogenia de espécies será construída utilizando como fonte de informações o conjunto de proteínas preditas dos organismos de interesse. A filogenia é baseada em características, onde cada aminoácido representa uma característica. Uma descrição detalhada desse método de filogenia pode ser encontrada na Seção 4.1. Através do programa MUSCLE [26, 25] obtemos o alinhamento múltiplo das famílias de proteínas, detectando regiões conservadas e regiões diferentes. O programa Gblocks [22, 85] faz uma filtragem, selecionando os blocos de alinhamentos múltiplos relevantes para inferências filogenéticas. Os alinhamentos múltiplos resultantes de cada família de proteínas são concatenados e repassados como entrada para o programa RAxML [78] que realiza a construção da árvore filogenética.

Para a reconstrução de filogenias baseadas em dados moleculares, costuma-se usar a estimativa da máxima verossimilhança. Esta estimativa baseia-se na reconstrução filogenética através da busca por uma árvore que maximize a probabilidade dos dados observados na matriz. Os algoritmos computacionais empregados nesses métodos são desenvolvidos para lidar com o fato de que mutações que resultam na substituição de nucleotídeos e aminoácidos são comuns, mas que suas frequências podem ser estimadas independentemente por meio de outras informações genéticas. Uma descrição detalhada dessa estatística pode ser encontrada em [57].

4.1 Métodos para filogenia

Existem diversos métodos de construção de filogenias, cada um tratando de forma específica as hipóteses sobre o processo de evolução. Um método filogenético consiste basicamente na separação das espécies em grupos, correspondentes a subárvores, que compartilham propriedades comuns. A formação destes grupos é possível em função das transformações que ocorrem ao longo do processo evolutivo.

Na construção de filogenias, podemos utilizar duas categorias de dados: distâncias ou características. As medidas de distâncias são estimativas das distâncias evolutivas entre espécies. As características são dados relativos ao fenótipo ou mesmo à presença/ausência de certas proteínas. Cada característica pode assumir vários estados, podendo esses serem discretos ou contínuos. Ambas as categorias de dados podem ser obtidas dos conjuntos de proteínas das espécies.

Filogenia baseada em distâncias

A construção de árvores filogenéticas baseia-se, no caso da filogenia baseada em distâncias, em dados numéricos resultantes de comparações entre n objetos. A entrada é uma matriz quadrada M de ordem n, simétrica, cujo elemento M_{ij} é um número real não-negativo, chamado de distância entre os objetos i e j.

Neste trabalho, realizamos comparações entre os conjuntos de proteínas de pares de espécies para obtermos as distâncias. Ao compararmos dois conjuntos de proteínas, estamos interessados em encontrar pares de genes ortólogos, em identificar as regiões

com conservação na ordem dos genes, ou em identificar regiões específicas de um conjunto de proteínas em relação a outro, entre outros. A partir da comparação de cada par de espécies, geramos uma matriz de distâncias e a utilizamos como entrada em um algoritmo para construção de filogenias baseadas em distâncias. As matrizes de distâncias podem ser utilizadas para inferir árvores ultramétricas e árvores aditivas.

Árvores ultramétricas demonstram o tempo evolutivo decorrido de uma espécie para outra e mostram quais espécies são mais próximas ou qual espécie surgiu primeiro. Um nó interno nessa árvore representa um evento divergente, ou seja, um ponto no tempo quando as histórias evolutivas de pelo menos duas espécies divergiu.

Dada uma matriz simétrica M para n objetos, uma árvore ultramétrica para M é uma árvore enraizada, com n folhas, sendo cada folha correspondente a uma linha da matriz M. Um nó interno da árvore é rotulado com uma entrada da matriz M e tem pelo menos dois filhos. Os rótulos dos nós internos são estritamente decrescentes ao longo de qualquer caminho da raiz até uma folha. E para quaisquer duas folhas i e j na árvore, M_{ij} é o rótulo do ancestral comum mais próximo entre i e j [35]. Os conceitos apresentados anteriormente, que definem uma árvore ultramétrica, podem ser visualizados na árvore da Figura 4.1.

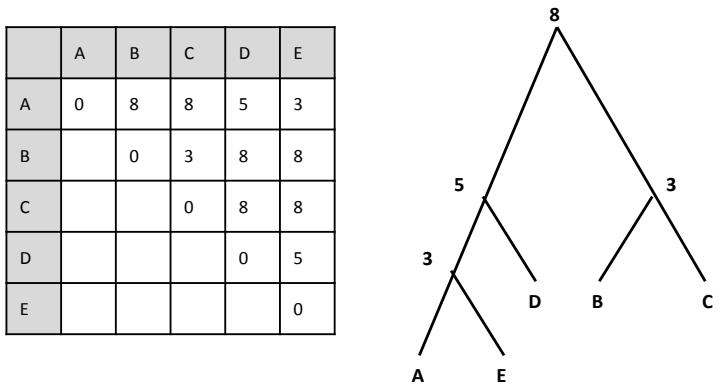


Figura 4.1: Exemplo de uma matriz simétrica M e sua respectiva árvore ultramétrica. Figura transcrita de [9].

Uma matriz simétrica M de números reais define uma distância ultramétrica se, e somente se, para quaisquer três índices i, j e k, o máximo entre M_{ij}, M_{ik} e M_{jk}

não é único. Quando M define uma distância ultramétrica, dizemos que M é uma matriz ultramétrica. O resultado abaixo caracteriza uma árvore ultramétrica.

Teorema 4.1. *Uma matriz simétrica M tem uma árvore ultramétrica se, e somente se, M é uma matriz ultramétrica.*

A prova do Teorema 4.1, descrito em [35], nos fornece um algoritmo para construir uma árvore ultramétrica. Além disso, se M é uma matriz ultramétrica, então uma árvore ultramétrica para M pode ser construída em tempo $O(n^2)$, sendo n o número de objetos.

Nem sempre é possível construir uma árvore ultramétrica. Isso acontece porque os dados reais não são ultramétricos e mesmo quando são, não necessariamente refletem verdadeiramente o tempo decorrido desde a divergência [35].

Já que não podemos construir árvores ultramétricas sempre, podemos tentar inferir árvores aditivas, que não indicam relações de ancestralidade ou direção de evolução das espécies, mas mostram a proximidade evolutiva entre elas. Com isso, essas árvores fornecem menos informações que uma árvore ultramétrica. As matrizes de distâncias utilizadas para inferir árvores aditivas são chamadas matrizes aditivas.

Seja M uma matriz simétrica com zeros na diagonal principal e números positivos nas outras posições. Seja T uma árvore com peso nas arestas e com pelo menos n nós, rotulados pelas linhas de M. T é aditiva para M se, $\forall$ par (i, j), o caminho de i até j em T tem peso $M(i, j)$.

No processo de construção de árvores filogenéticas, quando trabalhamos com distâncias, o conceito de espaço métrico é necessário.

Um espaço métrico é um conjunto C munido de uma métrica d (distância, neste caso) tal que, para qualquer $i, j, k \in C$:

1. $d(i, j) > 0$ para $i \neq j$,
2. $d(i, j) = 0$ para $i = j$,
3. $d(i, j) = d(j, i)$ para todo i, j e
4. $d(i, j) \leq d(i, k) + d(k, j)$ para todo i, j, k (desigualdade triangular)

Um espaço métrico é aditivo se,

5. para quaisquer 4 elementos do conjunto, podemos rotulá-los com i, j, k, l, de tal modo que $d(i,j) + d(k,l) = d(i,k) + d(j,l) \geq d(i,l) + d(j,k)$

Uma matriz M admite uma árvore aditiva T se, e somente se, seus elementos formam um espaço métrico aditivo. O peso do caminho entre quaisquer dois nós i e j, deve ser igual a M_{ij}. Se tal árvore T puder ser construída, dizemos que M e T são aditivas.

Os conceitos apresentados acima, que definem uma árvore aditiva, podem ser visualizados na matriz e árvore da Figura 4.2.

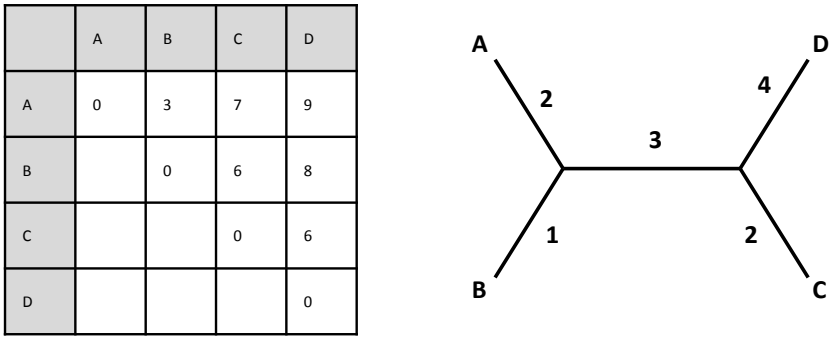


Figura 4.2: Exemplo de uma matriz simétrica M e sua respectiva árvore aditiva. Figura transcrita de [74].

Note que, através de uma árvore aditiva não há predição de ancestralidade, pois a árvore construída não possui raiz. Com isso, não conseguimos ver o nó que é o ancestral de todos ou o nó que vem antes de outro nó. Um algoritmo para construção de árvores aditivas cujo tempo de execução é $O(n^2)$, sendo n o número de objetos, pode ser visto em [74].

O objetivo é construir filogenias cuja distância observada na árvore entre duas espécies seja exatamente igual à distância armazenada numa matriz de distâncias entre os pares de genomas. Nem sempre isto é possível, pois em geral às distâncias não possuem determinadas propriedades, apresentadas anteriormente. Com isso, heurísticas são necessárias para o problema e existem várias na literatura [74, 69, 75]. Essas heurísticas não serão aqui descritas, uma vez que fogem do escopo do nosso trabalho.

Filogenia baseada em características

As filogenias são construídas com base nas comparações entre as espécies. Quando os objetos são espécies, podemos construir filogenias baseadas em características que podem ser, por exemplo: a forma do bico, número de dedos na pata, presença ou ausência de certas proteínas, hábitos alimentares, ciclo de vida. Cada característica pode ou não ter um número finito de estados. Essas características podem ser agrupadas em uma matriz, de modo que cada linha da matriz representa um objeto e cada coluna representa uma característica. Chamamos esta matriz de **matriz de estados** [74].

Na construção de filogenias baseadas em características, os seguintes aspectos são considerados:

- as características podem ser herdadas independentemente umas das outras;
- todos os estados de uma característica devem evoluir de um estado original do ancestral comum mais próximo dos objetos em estudo;
- os nós internos da árvore representam espécies ancestrais hipotéticas;
- a distância entre um nó interno e uma folha pode ser interpretada como uma estimativa do tempo que um nó (nó interno) levou para evoluir para outro nó (no caso, a folha).

Podemos então definir uma matriz de estados como sendo uma matriz M com n linhas (objetos) e m colunas (características), onde M_{ij} denota o estado que o objeto i tem para a característica j.

Tabela 4.1: Exemplo de matriz de estados.

Objeto/Característica	c_1	c_2	c_3	c_4	c_5
A	1	1	0	0	0
B	0	0	1	0	1
C	1	1	0	0	1
D	0	1	1	1	0
E	1	1	0	0	1

A construção de uma filogenia a partir de uma matriz de estados depara-se com algumas dificuldades, descritas a seguir:

- Convergência ou evolução paralela - Os métodos para reconstrução da árvore filogenética baseiam-se no fato de que objetos que compartilham o mesmo estado para uma dada característica são geneticamente mais relacionados que aqueles que não compartilham. Entretanto, existe a possibilidade que dois objetos compartilhem um estado mas não sejam geneticamente próximos. Tal fenômeno é chamado de convergência ou evolução paralela.

- Reversão de estados - Tal dificuldade diz respeito a relação entre os estados de cada característica. Considerando a matriz apresentada na Tabela 4.1, por exemplo, suponha que A e B evoluíram de um objeto ancestral X. Que estado deveríamos atribuir a X em relação à característica c_1? Podemos observar pela matriz que $c_1 = 1$ para A e $c_1 = 0$ para B. Se fizermos $c_1 = 1$ para X, e algum ancestral de X possuir o estado 0 para c_1, então o objeto B apresenta uma reversão de estados para a característica c_1.

Para evitarmos eventos de convergência e reversão de estados, o projeto de uma árvore T deve possuir a seguinte propriedade: o conjunto de todos os nós (objetos) que possuem o mesmo estado para uma determinada característica deve formar uma subárvore de T. Uma filogenia com esta propriedade é uma **filogenia perfeita**.

O problema central de reconstrução da filogenia baseado em matrizes de estados é conhecido como problema da filogenia perfeita. O problema consiste em dados um conjunto O com n objetos, um conjunto C de m características, cada característica tendo no máximo r estados, determinar se existe uma filogenia perfeita para O.

Sendo O_k o conjunto de objetos com 1 na coluna k de M, segue o Teorema 4.2.

Teorema 4.2. *M tem uma filogenia perfeita se, e somente se, $\forall$colunas i, j, ou O_i ou O_j são disjuntas ou uma contém a outra.*

Observe na Figura 4.3 uma matriz de estados e sua respectiva árvore filogenética de características.

	1	2	3	4	5	6
A	0	0	0	1	1	0
B	1	1	0	0	0	0
C	0	0	0	1	1	1
D	1	0	1	0	0	0
E	0	0	0	1	0	0

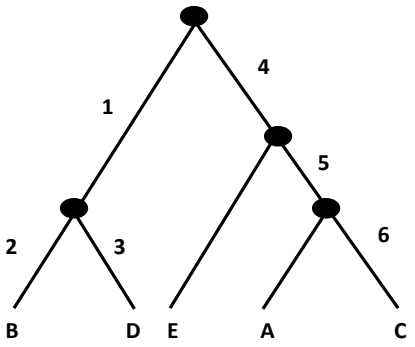


Figura 4.3: Exemplo de matriz de estados e sua respectiva árvore filogenética de características. Figura transcrita de [74].

Sempre que um conjunto de objetos definidos por uma matriz de estados admite uma filogenia perfeita dizemos que as características são **compatíveis**.

Mais uma vez, nem sempre uma matriz de estados admite uma filogenia perfeita. Então heurísticas são necessárias para o problema e existem várias na literatura [74]. Essas heurísticas não serão aqui descritas, uma vez que fogem ao escopo do nosso trabalho.

4.2 Filogenia e alinhamento múltiplo

Normalmente o alinhamento múltiplo de sequências de nucleotídeos/aminoácidos é feito com o objetivo de descobrir similaridades biológicas (estruturais/funcionais) entre as proteínas. Se a similaridade entre sequências for baixa, alinhamentos duas-a-duas podem impedir a sua identificação. A comparação simultânea de muitas sequências encontra frequentemente similaridades que não são detectadas nos alinhamentos duas-a-duas [61].

Os problemas de construção de um alinhamento múltiplo de sequências e o de determinar uma árvore filogenética para essas sequências são ligados de maneira fundamental.

O alinhamento múltiplo de sequências de nucleotídeos ou de aminoácidos permite

detectar regiões conservadas e regiões diferentes, determinar as sequências que são homólogas e descobrir a função de uma determinada sequência desconhecida. Através das informações obtidas pelo alinhamento de sequências é possível a construção de árvore filogenética.

A maior parte dos métodos de alinhamento múltiplo realiza os alinhamentos dois-a-dois entre todos os pares de sequências. A partir desses alinhamentos *pairwise*, gera-se então uma matriz que fornece uma pontuação a cada par de sequências. Após isso, ordenam-se os alinhamentos de acordo com a pontuação obtida e, em seguida, começa por alinhar as duas sequências com uma pontuação mais elevada. Prossegue-se o alinhamento com as sequências restantes. Essa abordagem para a construção de alinhamento múltiplo é conhecida como **alinhamento progressivo** [33].

Para criar uma árvore a partir de uma matriz, escolhem-se primeiro as duas espécies mais estreitamente relacionadas. Um nó para representar seu ancestral comum é inserido no conjunto. Em seguida, substituem-se as duas espécies selecionadas por esse novo elemento, e substitui-se as distâncias do par para os outros pela média das distâncias das duas espécies selecionadas para os outros. Agora temos um conjunto de diferenças entre pares, e não entre espécies individuais, mas entre os conjuntos de espécies (tendo, cada uma das espécies individuais restantes como um conjunto contendo apenas um elemento). Esse processo é repetido até que se tenham apenas dois elementos no conjunto, que são finalmente ligados por uma aresta [46].

Este método de construção da árvore é chamado UPGMA (*Unweighted Pair Group Method with Arithmetic mean*). A modificação do método de UPGMA por [69], chamado *Neighbor-Joining*, foi criado para corrigir taxas desiguais de evolução em diferentes ramos da árvore.

MUSCLE

Neste trabalho, o alinhamento múltiplo é obtido utilizando o programa *MUSCLE* (*Multiple Sequence Comparison by Log-Expectation*) [26, 25], que é um software para alinhamento múltiplo de sequências de nucleotídeos ou de aminoácidos. Neste trabalho utilizamos sequências de proteínas. O alinhamento múltiplo consiste na comparação de três ou mais sequências de proteínas ou de ácidos nucleicos, que se alinham parcial ou completamente [61].

Um vez encontradas as famílias proteicas provenientes de genes ortólogos, estas devem ser submetidas a um método de comparação que encontre as regiões que são de fato conservadas entre as sequências pertencentes a uma determinada família. Tais regiões conservadas podem ser associadas a estrutura/função similares, evidência esta que permite inferir relações evolutivas entre os organismos que possuem tais proteínas [61, 66].

Neste projeto, para encontrar tais regiões conservadas para cada uma das famílias foi utilizado o MUSCLE na geração do alinhamento múltiplo. Os alinhamentos resultantes serão a base para a construção de uma árvore filogenética para os organismos analisados.

O MUSCLE é frequentemente utilizado como um substituto para o CLUSTALW [89], uma vez que normalmente produz melhores alinhamentos de sequências e, além disso, executa alinhamentos múltiplos de um grande número de sequências de modo mais eficiente [26]. A execução do MUSCLE pode ser descrita em três passos.

No primeiro passo, o MUSCLE não realiza alinhamentos globais (comparação das sequências em todo o seu comprimento) como o CLUSTALW, pois alinhamentos globais para todos os pares não redundantes de um conjunto grande de sequências, tornaria o tempo de execução do algoritmo muito maior. A fim de diminuir o tempo de execução, o MUSCLE conta o número de subsequências comuns a cada par de sequências sem construir um alinhamento. A partir dos resultados da comparação das sequências, o MUSCLE, assim como os demais programas de alinhamento múltiplo, forma uma matriz de distância, a qual por sua vez é utilizada para construir uma árvore guia [26, 61]. Tal árvore caracteriza-se por sua topologia (ordem de ramificação) e comprimento dos ramos (proporcionais à similaridade das sequências), de modo que possua sequências cada vez mais distantes à medida que se caminha nas folhas (nós externos que correspondem às sequências) em direção à raiz [61]. A topologia da árvore guia determina a ordem com que as sequências serão adicionadas ao alinhamento múltiplo no próximo passo. Ressalta-se que esta árvore guia não possui nenhum significado filogenético.

No segundo passo, o MUSCLE realiza alinhamento múltiplo progressivo. No método progressivo, para cada nó da árvore guia, um alinhamento *pairwise* (entre duas sequências) global é realizado, progredindo das folhas em direção à raiz [26]. O primeiro alinhamento é feito entre duas sequências mais similares, e os próximos

alinhamentos podem ser de 3 tipos: sequência-sequência, perfil-sequência e perfil-perfil [26]. Neste contexto, perfil corresponde a um grupo alinhado de sequências conectadas a um nó interno (nó entre a raiz e as folhas) da árvore [26].

Finalmente, o terceiro passo constitui-se de refinamentos do alinhamento múltiplo realizado. Para tanto, são calculadas as identidades *pairwise* de cada par de sequências, e uma nova matriz de distância é formada bem como uma nova árvore guia a partir desta matriz [26]. A árvore do alinhamento anterior é comparada com a nova, e as subárvores que estiverem diferentes são submetidas novamente ao alinhamento múltiplo progressivo [26]. O processo é repetido até que todas as subárvores sejam estabilizadas, isto é, permaneçam com a mesma topologia, ou até que um número máximo de iterações seja atingido [26]. O conjunto de sequências que formam a árvore anterior modificada é então dividido em dois (eliminando os nós a partir das folhas até chegar na metade da árvore) e cada um destes subconjuntos dá origem a um perfil, baseado no alinhamento múltiplo atual. Estes dois perfis são então realinhados um com o outro usando a comparação *pairwise*. Um *score* é calculado, que é uma medida da qualidade do alinhamento [26]. Caso o *score* objetivo do atual alinhamento seja mais elevado, este é mantido, senão este alinhamento múltiplo é descartado e mantém-se o alinhamento anterior.

Gblocks

A análise filogenética exige, como um primeiro passo, o alinhamento de sequências de nucleotídeos ou de aminoácidos de tal maneira que elas sejam homólogas em cada posição. No entanto, nem todas as posições do alinhamento de sequências homólogas são úteis para a inferência filogenética porque, para fazer uma árvore filogenética confiável, as sequências não devem ser nem tão similares, já que estas são desprovidas de informação filogenética, e nem tão divergentes, já que muitas posições são saturadas por várias substituições. Uma vez que nem todas as regiões de um gene evoluem na mesma proporção, uma situação muito comum ocorre quando algumas partes de um alinhamento são bem conservadas e, por conseguinte, adequado para a análise filogenética, ao passo que outras são muito divergentes e repletas de *gaps*, de tal modo que a homologia posicional não pode ser precisamente determinada (afinal substituições múltiplas encobrem a informação filogenética destas regiões) [22, 85].

Em tais casos, é recomendado que as regiões divergentes sejam removidas antes da construção da árvore filogenética. O programa Gblocks [22, 85] recebe como entrada o alinhamento múltiplo e faz uma filtragem, selecionando os blocos de alinhamentos múltiplos relevantes para inferências filogenéticas. A Figura 4.4 mostra a saída do programa para uma família de 13 proteínas de genomas do gênero *Mycobacterium*. O método define um conjunto de blocos conservados a partir de um alinhamento múltiplo através da eliminação de todas as regiões divergentes e mal alinhadas seguindo 6 passos.

```
Gblocks 0.91b Results
Number of sequences: 13
Alignment assumed to be: Protein
New number of positions: 134 (selected positions are underlined in blue)

                    10        20        30        40        50        60
               =========+=========+=========+=========+=========+=========+
118464605      ------MKRLWIPLVIVAVVVAGGLTVSRLHGIFGSEKRPSYADTRQQDTKPFNPKHVKY
41407339       ------MKRLWIPLVIVAVVVAGGLTVSRLHGIFGSEKRPSYADTRQQDTKPFNPKHVKY
99915328        ---------MWIPLLVLVVLGAGAFTVLRLHRVFGSETRPAYADTELEERKPYDPKQLVY
118473924      ------MRRLWIPLLVIAVVAVGGFTVSRLHNVFGAEKRPSYADTKAADSKPFDPKRLTY
183980130      MTALKILGKLWIPLVIVTVVGVTAFSVSRVHGFFGSQKPELYTDGSSDDATPFNPKQLTY
121636365      -----MLMRTWIPLVILVVVIVGGFTVHRIRGFFGSENRPSYSDTNLENSKPFNPKHLTY
148660216      -----MLMRTWIPLVILVVVIVGGFTVHRIRGFFGSENRPSYSDTNLENSKPFNPKHLTY
15607592       -----MLMRTWIPLVILVVVIVGGFTVHRIRGFFGSENRPSYSDTNLENSKPFNPKHLTY
224988837      -----MLMRTWIPLVILVVVIVGGFTVHRIRGFFGSENRPSYSDTNLENSKPFNPKHLTY
31791629       -----MLMRTWIPLVILVVVIVGGFTVHRIRGFFGSENRPSYSDTNLENSKPFNPKHLTY
339630519      -----MLMRTWIPLVILVVVIVGGFTVHRIRGFFGSENRPSYSDTNLENSKPFNPKHLTY
378770199      -----MLMRTWIPLVILVVVIVGGFTVHRIRGFFGSENRPSYSDTNLENSKPFNPKHLTY
183980796      -----MLSRTWIVLVIVAVIIVGGFSVHRIRGFFASEKRESYSDSNLDN-KPFNPKEITY
                     ###################################### ##########

                    70        80        90       100       110       120
               =========+=========+=========+=========+=========+=========+
118464605      EVFGPAGAMADISYFDANGEPQHINGVELPWTFDISTTLPSIVGNVVAQGNSDSLGCRIL
41407339       EVFGPAGAMADISYFDANGEPQHINGVELPWTFDISTTLPSIVGNVVAQGNSDSLGCRIL
99915328       EVFGPPGTVANISYFDVDAEPQFVEGASLPWSLKFPMSEATSMVNVIAQGDSNRIGCRII
118473924      EVFGPPGTLADISYFDEDSNPQFVEKVSLPWSLHFDIGKTTAVGSIMAQGDSDTIGCRII
183980130      EIFGPAGTVADIDYFDVNAQPRRVDGAHLPWSLEIKSTSASLTGNIVAQGDSNRLGCRII
121636365      EIFGPPGTVADISYFDVNSEPQRVDGAVLPWSLHITTNDAAVMGNIVAQGNSDSIGCRIT
148660216      EIFGPPGTVADISYFDVNSEPQRVDGAVLPWSLHITTNDAAVMGNIVAQGNSDSIGCRIT
15607592       EIFGPPGTVADISYFDVNSEPQRVDGAVLPWSLHITTNDAAVMGNIVAQGNSDSIGCRIT
224988837      EIFGPPGTVADISYFDVNSEPQRVDGAVLPWSLHITTNDAAVMGNIVAQGNSDSIGCRIT
31791629       EIFGPPGTVADISYFDVNSEPQRVDGAVLPWSLHITTNDAAVMGNIVAQGNSDSIGCRIT
339630519      EIFGPPGTVADISYFDVNSEPQRVDGAVLPWSLHITTNDAAVMGNIVAQGNSDSIGCRIT
378770199      EIFGPPGTVADISYFDVNSEPQRVDGAVLPWSLHITTNDAAVMGNIVAQGNSDSIGCRIT
183980796      EVFGPPGTVADISYFDVNSDPQQVEGAVLPWTLHFTTNLAAVMGNLVAQGNTNSIGCRIT
               ################################################################

                      130       140
               =========+=========+=====
118464605      VDGVVKTERISHELNAFTYCVLTAT
41407339       VDGVVKTERISHELNAFTYCVLTAT
99915328       IDDKVKSERVENGESAFTYCLLKAA
118473924      VDDEVKAEKVSNQTNAFTSCLLKAA
183980130      VDGVIKAERISQEVNAYTFCIVKSA
121636365      VDGKVRAERVSNEVNAYTYCLVKSA
148660216      VDGKVRAERVSNEVNAYTYCLVKSA
15607592       VDGKVRAERVSNEVNAYTYCLVKSA
224988837      VDGKVRAERVSNEVNAYTYCLVKSA
31791629       VDGKVRAERVSNEVNAYTYCLVKSA
339630519      VDGKVRAERVSNEVNAYTYCLVKSA
378770199      VDGKVRAERVSNEVNAYTYCLVKSA
183980796      VDGKVKAERVSNEVNAYTYCLVKSA
               ########################

Parameters used
Minimum Number Of Sequences For A Conserved Position: 7
Minimum Number Of Sequences For A Flanking Position: 11
Maximum Number Of Contiguous Nonconserved Positions: 8
Minimum Length Of A Block: 10
Allowed Gap Positions: None
Use Similarity Matrices: Yes
Flank positions of the 2 selected block(s)
Flanks: [11  49]  [51  145]
New number of positions:  134  (92% of the original 145 positions)
```

Figura 4.4: Saída do Gblocks para a família de 13 proteínas, de *Mycobacterium*. As colunas marcadas com '#' são as mantidas para a construção da filogenia.

No primeiro passo, o grau de conservação de todas as posições de um alinhamento múltiplo é avaliado e classificado em não conservado (menos de 50% + 1 das sequências possuem aminoácidos idênticos ou possuem um *gap*), conservado (mais de 50% + 1 e menos de 85% das sequências possuem aminoácidos idênticos) e altamente conservado (85% ou mais das sequências possuem aminoácidos idênticos). No segundo passo, todos os trechos de posições não conservadas contíguas maiores que oito aminoácidos são descartados. Nesses trechos, os alinhamentos são normalmente ambíguos e várias substituições escondidas os tornam inadequados para uma análise filogenética. No terceiro passo, as extremidades dos blocos restantes do alinhamento são examinadas, e posições destas extremidades são removidas até que nas extremidades restem apenas posições altamente conservadas, o que permite que as extremidades de tais blocos possam ser alinhadas com alta confiança [22]. No quarto passo, apenas blocos de alinhamento com tamanho maior ou igual a 15 posições são mantidos, a fim de evitar pequenas regiões nas quais a qualidade do alinhamento é difícil de avaliar. No quinto passo, posições com *gaps* e posições não conservadas adjacentes aos *gaps* são removidas até que se alcance posições conservadas, pois regiões adjacentes a *gaps* dificultam o alinhamento. No último passo, apenas blocos com tamanho maior ou igual a 10 posições são mantidos [22].

RAxML

A partir do alinhamento resultante da eliminação das colunas não desejadas, pode-se então usar um programa qualquer de construção de filogenia. Neste trabalho, usamos o programa RAxML (*Randomized Axelerated Maximum Likelihood*) [78].

As sequências dos alinhamentos múltiplos resultantes do Gblocks são concatenadas em um único alinhamento a fim de executar a posterior análise filogenética de um conjunto maior de dados. Esse alinhamento único é obtido por meio da "ligação" de uma sequência ao final de outra. Tais sequências concatenadas constituem a entrada para o programa RAxML. O RAxML constrói árvores filogenéticas baseando-se no princípio de máxima verossimilhança.

Em uma árvore filogenética, as folhas correspondem às unidades taxonômicas operacionais (OTU), que neste caso são as espécies e os nós internos representam ancestrais hipotéticos das OTUs. A topologia da árvore mostra as relações entre as espécies

em termos de ancestralidade [46, 61].

As árvores construídas para cada complexo bacteriano não possuem raiz, porém, a presença de um grupo externo proporciona certa compreensão a respeito do estado de características do ancestral comum mais recente, afinal qualquer estado de características observado no grupo externo e no grupo de estudo é considerado um estado ancestral.

A árvore filogenética a ser construída pelo RAxML é obtida a partir da concatenação das sequências dos alinhamentos múltiplos das famílias de proteína contendo exatamente um gene de cada genoma *ingroup* e no máximo um gene dos genomas *outgroup*. Como dito antes, a ideia contida nesta abordagem é não permitir que genes parálogos atrapalhem a construção da árvore.

Com relação ao método de máxima verossimilhança, este atribui probabilidades de eventos mutacionais para cada aminoácido. A partir dessas matrizes de probabilidades, são construídas várias árvores e para cada topologia de árvore possível, as taxas de substituição de aminoácidos assumidas sofrem variações para encontrar os parâmetros que forneçam a maior probabilidade de gerar as sequências observadas. A árvore ótima é aquela que apresenta a maior probabilidade de gerar os dados observados [46].

Considerando o número de sequências a serem analisadas e a necessidade de diminuir o tempo de execução, o RAxML utiliza-se de heurísticas para reduzir o espaço de topologias potenciais de árvores a serem avaliadas, o que reduz o número de topologias avaliadas e, portanto, diminui o número de invocações da função de máxima verossimilhança, a qual normalmente exige um longo tempo de execução. Além disso, o RAxML restringe o número de posições consideradas no cálculo de probabilidade de mudança de posições (aminoácidos) e de outros parâmetros, e dessa maneira, minimiza o número de cálculos realizados para cada nó interno durante a avaliação das árvores [79].

No que diz respeito à análise estatística da árvore ótima gerada pelo RAxML, este permite utilizar o teste *bootstrapping*. A análise *bootstrapping* não avalia a acurácia da árvore, mas descreve a robustez da topologia da árvore. O que o *bootstrapping* testa é se rearranjos aleatórios do conjunto original de dados permitem chegar na mesma topologia. Os rearranjos são na verdade amostras aleatórias de colunas do

alinhamento múltiplo com muitas posições repetidas a fim de produzir uma amostra com mesmo tamanho que o conjunto original de dados. O resultado do *bootstrapping* é a frequência com que cada clado (ancestral comum mais recente e todos os seus descendentes) é observado na árvore original. Logo, quanto maior a porcentagem de topologias iguais, mais robusta é a hipótese que confirma a árvore [61, 46].

Ao final, o RAxML devolve como resultado a árvore, em um formato conhecido como Newick, o qual é lido por um programa chamado NJplot descrito a seguir.

NJplot

O NJplot [60] é um programa capaz de desenhar qualquer árvore filogenética expressa no formato Newick. A árvore no formato Newick é lida pelo programa NJplot a fim de desenhar a árvore de correlação entre as espécies em estudo. O NJplot gera o gráfico da árvore conforme vistos em figuras mostradas no próximo capítulo.

4.3 Filogenia de espécies e concatenação de alinhamentos

Como será visto em detalhes na Seção 5.1, a árvore produzida como subproduto de nosso *pipeline* é construída a partir de um alinhamento múltiplo resultante da concatenação de alinhamentos. Cada alinhamento representa uma família de proteínas contendo genes ortólogos e de cópia única dos genomas comparados. A construção de árvores filogenéticas desse tipo, também conhecidas na literatura como *genome tree* ou *whole-genome tree*, já são amplamente conhecidas na literatura, contrastando com árvores de espécies baseadas em genes específicos, na maioria das vezes em genes de RNA ribossomal, como 16s rRNA.

Árvores baseadas em genes 16s rRNA são ainda muito utilizadas porque esses genes possuem uma boa combinação de propriedades: estão presentes em praticamente todas as bactérias, possuem variabilidade compatível com o que se espera na diferenciação de espécies, são facilmente amplificados em laboratório e raramente são transferidos lateralmente. Entretanto, essas árvores apresentam falhas quando os

organismos separados são muito próximos evolutivamente, como é o caso de cepas de mesma espécie [43]. Assim, com a facilidade na obtenção de genomas completos, ou incompletos mas com anotação suficientemente determinada, o uso de árvores baseadas em coleções de genes ficou mais frequente.

Martin Wu e colegas [100, 101] desenvolveram uma ferramenta denominada AMPHORA (*AutoMated PHylogenOmic inferRence*). Essa ferramenta, desenvolvida primordialmente para dados de metagenômica, escolhe genes de cada genoma que exercem funcionalidades importantes e pré-definidas, como genes *housekeeping*. Além disso, fazem filtragens adicionais aos alinhamentos antes da construção da árvore. O mesmo é feito em [43]. Genes *housekeeping* são descritos com mais detalhes na Seção 6.1.

No trabalho apresentado em [99], os autores propõem uma técnica muito parecida com a apresentada neste trabalho, incluindo o fato de escolherem famílias com genes ortólogos, mas construindo uma árvore inicial para cada família e eliminando aquelas que não obedecem a alguns critérios. Além disso, uma análise de frequência de aminoácidos é feita em cada alinhamento, antes do alinhamento ser usado na construção da árvore final.

Rodriguez [67] e colegas propuseram uma metodologia específica para o gênero *Xanthomonas* e produziram uma árvore para espécies desse gênero, usando o programa AMPHORA. Eles obtiveram a mesma topologia obtida pela aplicação da técnica aqui proposta e apresentada para genomas do mesmo gênero em [54].

Vale ressaltar novamente que a árvore aqui proposta é apenas um subproduto do nosso *pipeline*, uma vez que resulta de uma busca específica feita no *Orthologsorter*. Dessa forma, ela apresenta apenas critérios básicos de filtragem pelo uso do Gblocks, bem menos sofisticados dos que aqueles apresentados nas publicações descritas nos parágrafos acima. No entanto, mesmo fazendo uso apenas desses critérios, essa técnica possibilita a determinação de uma árvore com alto grau de confiança, como pode ser visto nos testes de *bootstrap* apresentados nos diversos projetos onde a técnica foi usada [4, 54, 63, 19, 73, 76, 96].

Capítulo 5

Orthologsorter

Neste capítulo descrevemos o *pipeline* proposto neste trabalho, que gera uma ferramenta de busca via *web* de famílias de interesse, denominado *Orthologsorter*. Na Seção 5.1 são descritas as ferramentas computacionais utilizadas no *pipeline*. Na Seção 5.2, descrevemos o *Orthologsorter* de forma geral. Na Seção 5.3, tendo como guia os três passos a seguir, o *pipeline* é descrito detalhadamente. Finalmente, abordamos alguns aspectos acerca da implementação do *Orthologsorter* na Seção 5.4.

5.1 Ferramentas computacionais utilizadas

A Figura 5.1 mostra a sequência em que as ferramentas computacionais são utilizadas. Ao final, o objetivo é criar uma ferramenta de busca via *web* de famílias de proteínas de interesse e, através de uma busca específica, construir uma árvore filogenética que mostra as espécies estudadas e seus relacionamentos. Essa árvore é obtida a partir de um alinhamento múltiplo de sequências de proteínas dos genomas comparados. Mais detalhes sobre a árvore serão vistos ao final deste capítulo.

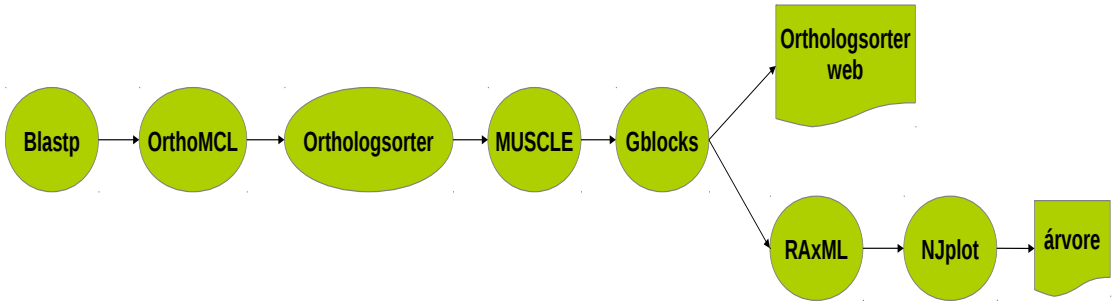


Figura 5.1: *Pipeline* para a construção da árvore filogenética.

O *pipeline* tem como entrada os arquivos no formato fasta contendo os genes anotados das espécies analisadas e um arquivo texto, com formato previamente estabelecido, contendo uma lista com os nomes das espécies comparadas e a identificação dos organismos externos ao grupo. Organismos externos ao grupo são necessários em qualquer análise filogenética e serão aqui denominados *outgroup*.

Uma das saídas do *pipeline* é uma ferramenta de busca via *web*, denominada *Orthologsorter*, composta por vários *scripts* cuja execução permite realizar buscas no conjunto de famílias obtidas a partir dos resultados do método de clusterização de sequências do OrthoMCL a fim de explicar os relacionamentos entre as espécies analisadas.

Também obtemos como saída do *pipeline* uma árvore filogenética que contribui para a inferência da história evolutiva. Dentro dessa abordagem, o *pipeline* possui três grandes fases:

- **Passo 1**: agrupamento de proteínas de acordo com os resultados da comparação das sequências proteicas (contendo pontuações de similaridades de sequência) produzidos pelo Blastp (comparação do tipo "todos-contra-todos");
- **Passo 2**: criação e inserção de dados no SGBD MySQL; e
- **Passo 3**: construção de uma ferramenta de busca via *web* de famílias de interesse, além de execução de buscas específicas para a construção e disponibilização da árvore filogenética dos genomas comparados.

5.2 Visão geral do Orthologsorter

O *Orthologsorter* é uma ferramenta *web* composta por vários *scripts* que permite realizar buscas no conjunto de famílias obtidas pelo OrthoMCL a fim de obter informações funcionais e evolutivas dos organismos analisados.

Para implementação das buscas utilizou-se a linguagem de programação *Perl* e a interface foi construída através da linguagem de formatação *HTML*. As buscas realizadas podem obter:

- famílias contendo pelo menos uma sequência de cada organismo incluído na análise;
- famílias contendo nenhuma sequência de determinados organismos;
- famílias contendo exatamente uma sequência de cada organismo incluído na análise;
- famílias com números de proteínas de cada genoma especificados pelo usuário; ou ainda
- todas as famílias.

O foco do *Orthologsorter* é fornecer mecanismos para buscas em um conjunto de famílias de proteínas de tal maneira que seja possível inferir as relações filogenéticas entre espécies próximas, auxiliando na tipagem de bactérias, mas também inferir funcionalidades a partir de famílias de proteínas ortólogas ou parálogas.

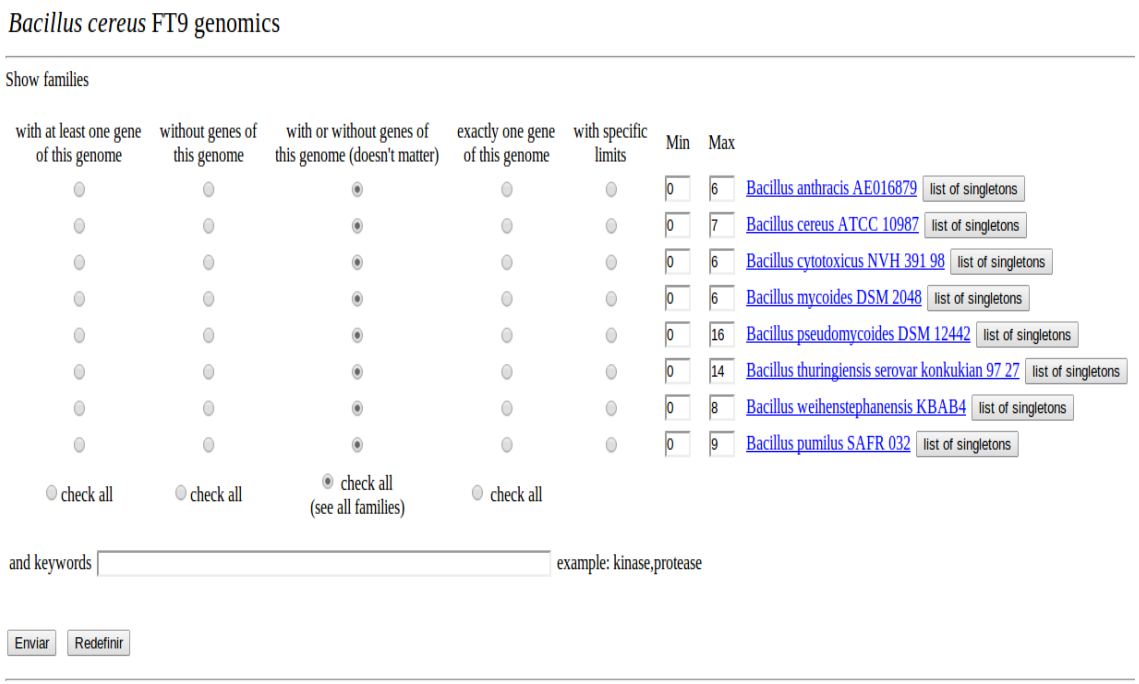


Figura 5.2: Tela principal do *Orthologsorter*.

Na Figura 5.2 podemos observar que o aplicativo *web* está organizado para facilitar a navegação e a realização de buscas. Uma descrição detalhada das possíveis buscas pode ser observada abaixo.

- 'with at least one gene of this genome' - permite mostrar famílias que tenham pelo menos um gene do organismo incluído na análise. O campo 'Min' é setado automaticamente com 1 e o campo 'Max' é setado automaticamente com o número máximo de genes daquele organismo encontrado nas famílias de proteínas.

- 'without genes of this genome' - permite mostrar famílias que não contém nenhum gene do organismo incluído na análise. O campo 'Min' e 'Max' são setados automaticamente com 0.

- 'with or without genes of this genome (doesn't matter)' - permite mostrar famílias contendo ou não genes do organismo incluído na análise. O campo 'Min' é setado automaticamente com 0 e o campo 'Max' é setado automaticamente com o número máximo de genes daquele organismo encontrado nas famílias de proteínas.

- ‘exactly one gene of this genome’ - permite mostrar famílias contendo exatamente 1 gene do organismo incluído na análise. O campo ‘Min’ e ‘Max’ são setados automaticamente com 1.
- ‘with specific limits’ - permite mostrar famílias com a quantidade de genes especificados pelo usuário nos campos ‘Min’ e ‘Max’ do organismo de interesse.

Após definir a busca, basta clicar no botão ‘Enviar’ e então o resultado da busca é mostrado na tela. Os *links*, com os nomes dos organismo, ao serem acessados, mostram o arquivo no formato fasta contendo os genes anotados do respectivo organismo.

Ao clicar no botão ‘list of singletons’, como mostrado na Figura 5.2, pode-se visualizar uma lista das sequências proteicas isoladas de determinado organismo que não se agrupam em nenhuma família de proteína.

Como exemplo de busca, observe a Figura 5.3. As opções de busca escolhidas foram exatamente uma sequência proteica de cada organismo, exceto para o organismo externo ao grupo (*Bacillus pumilus* SAFR 032). Para o organismo externo escolhemos as famílias contendo no máximo uma sequência proteica deste organismo.

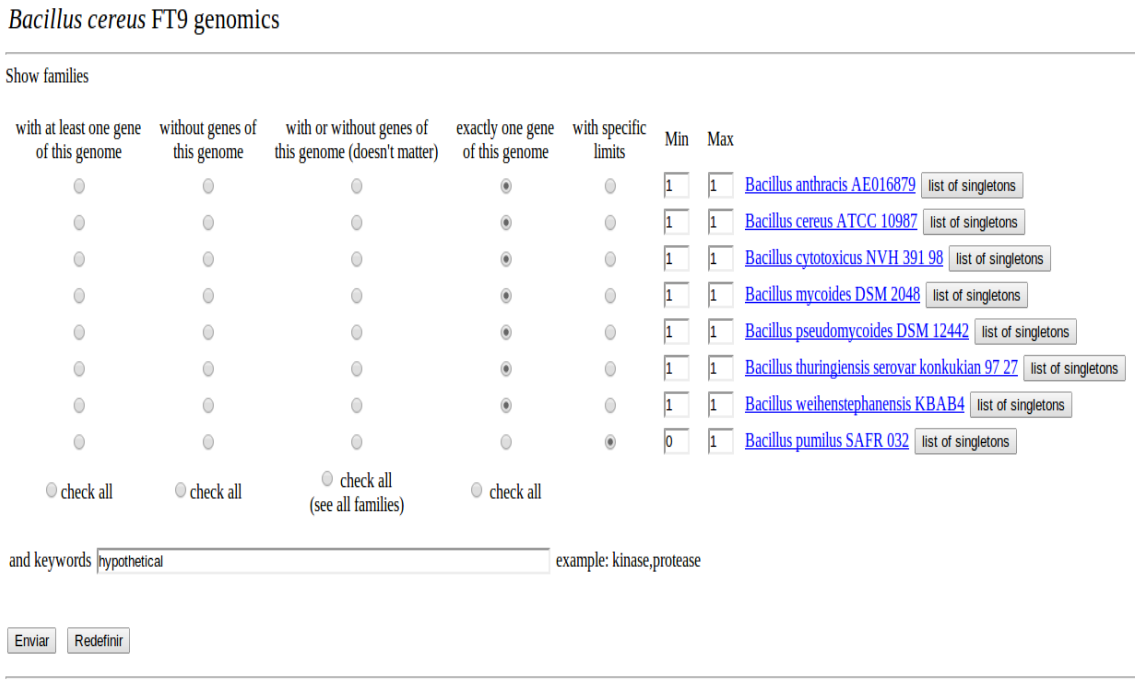


Figura 5.3: *Orthologsorter* - Exemplo de busca

Além de determinar a quantidade de genes de um organismo que se deseja na busca por famílias de proteínas, é possível também especificar palavras-chave que deverão pertencer ao produto de um gene das famílias de proteínas encontradas. Observe a Figura 5.3, onde na busca foi especificada a palavra-chave "*hypothetical*".

```
Result: 978 families

=======================================
>Family 196  -- Gblocks 205/207 (99%)
30265503            nr   jag protein [Bacillus anthracis str. Ames]
42784681            nr   jag protein [Bacillus cereus ATCC 10987]
152977684           nr   single-stranded nucleic acid binding R3H domain-containing protein [Bacillus cytotoxicus NVH 391-98]
229014651           nr   hypothetical protein bmyco0001_50510 [Bacillus mycoides DSM 2048]
228994204           nr   hypothetical protein bpmyx0001_49270 [Bacillus pseudomycoides DSM 12442]
49480476            nr   jag protein (SpoIIIJ-associated protein) [Bacillus thuringiensis serovar konkukian str. 97-27]
163943165           nr   single-stranded nucleic acid binding R3H domain-containing protein [Bacillus weihenstephanensis KBAB4]
157694476           nr   SpoIIIJ-associated protein [Bacillus pumilus SAFR-032]
-------------------------
number of proteins = 8

=======================================
>Family 203  -- Gblocks 192/224 (85%)
30265496            nr   hypothetical protein BA_5728 [Bacillus anthracis str. Ames]
42784674            nr   conserved hypothetical protein [Bacillus cereus ATCC 10987]
152977677           nr   hypothetical protein Bcer98_4018 [Bacillus cytotoxicus NVH 391-98]
229014644           nr   hypothetical protein bmyco0001_50440 [Bacillus mycoides DSM 2048]
228994197           nr   hypothetical protein bpmyx0001_49200 [Bacillus pseudomycoides DSM 12442]
49480473            nr   hypothetical protein BT9727_5159 [Bacillus thuringiensis serovar konkukian str. 97-27]
163943158           nr   hypothetical protein BcerKBAB4_5271 [Bacillus weihenstephanensis KBAB4]
157694468           nr   hypothetical protein BPUM_3726 [Bacillus pumilus SAFR-032]
-------------------------
number of proteins = 8

=======================================
>Family 205  -- Gblocks 65/68 (95%)
30265494            nr   hypothetical protein BA_5725 [Bacillus anthracis str. Ames]
42784672            nr   conserved hypothetical protein [Bacillus cereus ATCC 10987]
152977675           nr   hypothetical protein Bcer98_4016 [Bacillus cytotoxicus NVH 391-98]
229014642           nr   hypothetical protein bmyco0001_50420 [Bacillus mycoides DSM 2048]
228994195           nr   hypothetical protein bpmyx0001_49180 [Bacillus pseudomycoides DSM 12442]
49480472            nr   hypothetical protein BT9727_5157 [Bacillus thuringiensis serovar konkukian str. 97-27]
163943156           nr   hypothetical protein BcerKBAB4_5269 [Bacillus weihenstephanensis KBAB4]
157694464           nr   hypothetical protein BPUM_3722 [Bacillus pumilus SAFR-032]
-------------------------
number of proteins = 8

=======================================
>Family 210  -- Gblocks 309/320 (96%)
30265489            nr   hypothetical protein BA_5720 [Bacillus anthracis str. Ames]
42784667            nr   conserved hypothetical protein [Bacillus cereus ATCC 10987]
152977670           nr   hypothetical protein Bcer98_4011 [Bacillus cytotoxicus NVH 391-98]
229014637           nr   hypothetical protein bmyco0001_50370 [Bacillus mycoides DSM 2048]
228994190           nr   hypothetical protein bpmyx0001_49130 [Bacillus pseudomycoides DSM 12442]
49480817            nr   hypothetical protein BT9727_5152 [Bacillus thuringiensis serovar konkukian str. 97-27]
163943151           nr   hypothetical protein BcerKBAB4_5264 [Bacillus weihenstephanensis KBAB4]
157694449           nr   hypothetical protein BPUM_3707 [Bacillus pumilus SAFR-032]
-------------------------
number of proteins = 8
                                                                    ...
```

Figura 5.4: *Orthologsorter* - Resultado da busca

O resultado da execução do *Orthologsorter* utilizando os parâmetros descritos na Figura 5.3 pode ser observado na Figura 5.4. Primeiramente é mostrado o número de famílias encontradas na busca. No exemplo, foram encontradas 978 famílias. Em

seguida, observamos uma lista das famílias de proteínas com:

- um *link* 'Family X' para o arquivo no formato fasta contendo as sequências proteicas que formam esta família;
- um *link* 'Gblocks' para o alinhamento das sequências proteicas da respectiva família gerado pelo MUSCLE e filtrada pelo Gblocks;
- um *link* com o identificador do gene (*gi*) onde obtemos mais informações da sequências proteica disponível no NCBI;
- um *link* 'nr' para o resultado de um Blastp *on line* do gene contra a base *nr* do *GenBank*; e
- o produto do gene.

No Capítulo 6, foi incluída uma série de figuras que propiciarão um melhor entendimento dos recursos disponíveis pelo *Orthologsorter*.

No presente estudo, o *Orthologsorter* foi utilizado para buscar, a partir dos resultados do OrthoMCL, famílias contendo exatamente uma sequência proteica de cada organismo incluído na análise e que pode abranger no máximo uma sequência proteica para os organismos externos ao grupo. Partindo-se da premissa de que tais famílias sejam constituídas por sequências proteicas homólogas (similares devido à ancestralidade em comum). No entanto, o termo homologia consiste em um conceito um tanto geral em filogenias, de modo que o termo ortologia torna-se mais apropriado para nomear a relação filogenética inferida entre tais sequências. Sequências ortólogas referem-se a cópias de uma mesma sequência ancestral no mesmo locus, presente em espécies diferentes [66].

Famílias contendo mais de uma sequência proteica de certa espécie não poderiam ser consideradas famílias proteicas proveniente de genes ortólogos, pois as sequências de uma mesma espécie incluídas na mesma família são consideradas parálogas. Sequências parálogas consistem em cópias de um mesmo gene, localizadas em loci diferentes, produzidas por duplicação de uma sequência ancestral [66]. Caso o *Orthologsorter* não fosse usado para "filtrar" as famílias de sequências proteicas, seria possível que famílias provenientes de genes parálogos fossem incluídas para a construção das árvores filogenéticas, o que não é válido para construção de filogenias, afinal a inferência filogenética molecular baseia-se em genes ortólogos [66].

Através desta busca específica, citada acima, é possível a construção da árvore filogenética a fim de que o usuário obtenha informações funcionais e evolutivas dos organismos analisados.

5.3 Fluxograma de utilização

Nesta seção descrevemos como o *pipeline* mostrado na Figura 5.1 de forma detalhada, em termos de entradas e saídas e todos os passos necessários. O fluxograma é ilustrado na Figura 5.5.

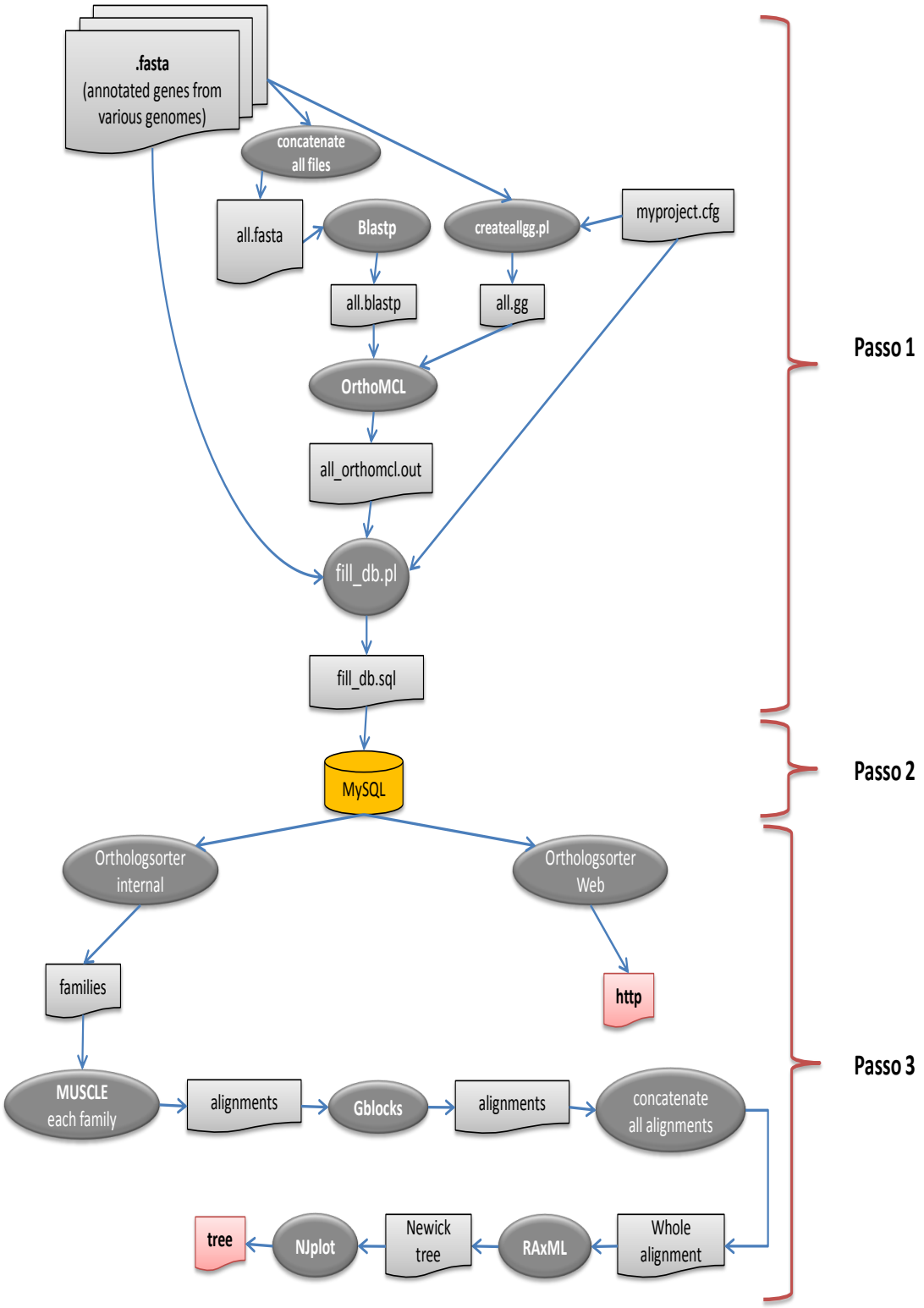


Figura 5.5: Fluxograma representando a execução do *pipeline*.

Agrupamento de proteínas

A entrada deste *pipeline* é um conjunto de arquivos no formato fasta contendo as proteínas preditas das espécies de interesse e um arquivo de configuração, denominado `myproject.cfg`. O arquivo `myproject.cfg`, fornecido pelo usuário, é um arquivo de configuração do *pipeline* durante sua execução. Primeiramente, é descrito o título do projeto. Em seguida, é descrito um genoma por linha. Na primeira coluna usa-se '1' se o genoma for *ingroup* ou '0' se for *outgroup*. Na segunda coluna usa-se '1 'se o genoma pertence ao projeto ou '0' caso contrário (encontra-se disponível no NCBI). Na terceira coluna é descrito o nome do genoma separado pelo símbolo _ e, por fim, informações do respectivo genoma, separado por espaço, estão na quarta coluna. Além disso, o usuário deverá fornecer o domínio onde estão disponíveis os diretórios `cgi-bin` e `/var/www`.

Os arquivos de entrada do *pipeline* possuem a extensão `*.faa` já que correspondem aos arquivos no formato fasta de aminoácidos. A concatenação dos arquivos fasta gera um único arquivo chamado `all.fasta`.

Dados o arquivo `all.fasta` e o arquivo de configuração, nossa primeira tarefa é a de realizar a comparação das proteínas "todos contra todos", utilizando Blastp, a fim de obter as primeiras informações sobre genes anotados.

Para executar o programa Blastp é necessário construir o banco de dados utilizando o seguinte comando `makeblastdb -in all.fasta -dbtype prot`. Depois de criado o banco de proteínas podemos executar o programa Blastp e como saída obtemos o arquivo `all.blastp`. O próprio pacote, quando instalado, incluirá os executáveis do Blastp localmente, juntamente com os executáveis do pacote.

O arquivo `myproject.cfg` é aberto e, a partir deste arquivo, obtemos o nome do genoma. Busca-se no arquivo `.faa` correspondente o nome do genoma e todos os identificadores dos genes daquele determinado genoma que são armazenados no arquivo de saída `all.gg`.

Usamos a saída do Blastp, denominada `all.blastp`, e o arquivo `all.gg` como entrada para o programa OrthoMCL. O OrthoMCL é responsável por gerar agrupamentos de maneira que cada proteína seja agrupada de acordo com as pontuações de similaridade obtidas pelo Blastp. Como saída do OrthoMCL obtemos o arquivo

`all_orthomcl.out`. Este arquivo contém todas as famílias de proteínas encontradas.

Após obtermos estas informações, podemos agora executar o *script* chamado `fill_db.pl`. Este *script* recebe como entrada os arquivos no formato fasta contendo as proteínas preditas das espécies de interesse, o arquivo `all_orthomcl.out` e o arquivo `myproject.cfg`. O *script* `fill_db.pl` gera o arquivo `fill_db.sql`, descrito na próxima seção.

O Passo 1, responsável por todos estas etapas descritas anteriormente, é descrito da seguinte forma:

Passo 1. Dado o conjunto de arquivos com os genes anotados de vários genomas e o arquivo de configuração, execute o script `createDatabase.pl`, disponível no pacote, de acordo com o seguinte comando:
$ perl `createDatabase.pl` *<myproject>*

Gerar o banco de dados MySQL

O arquivo `fill_db.sql`, gerado no passo anterior, é responsável por:

- criar o banco de dados do respectivo projeto;
- ler o arquivo `myproject.cfg` e carregar a tabela `genome` do banco de dados informando o código de identificação, nome e informações adicionais do genoma. Além disso, informar se o genoma é *ingroup* e se está disponível no NCBI.
- ler os arquivos fasta contendo as proteínas preditas e o `myproject.cfg` a fim de carregar a tabela `gene` do banco de dados informando a identificação do gene, a identificação do genoma ao qual aquele gene pertence, o produto do gene, a sequência proteica do gene e a identificação da família ao qual aquele gene pertence.
- ler os arquivos `myproject.cfg` e `all_orthomcl.out` a fim de carregar a tabela `family` do banco de dados informando a identificação da família e a quantidade de genes de cada genoma que pertence àquela respectiva família.

Este arquivo deverá ser executado pelo usuário *root* do MySQL a fim de criar e carregar o banco de dados MySQL utilizado no projeto. O Passo 2 pode ser descrito como segue.

Passo 2. Faça *login* no MySQL, como *root*, e execute o arquivo `fill_db.sql` de acordo com os comandos abaixo:
$ mysql -u <user> -p
$ source ./projects/<myproject>/temp/MySQL/fill_db.sql
$ exit

Construção do *Orthologsorter* e da árvore filogenética

Uma das saídas do *pipeline* é uma ferramenta *web*, denominada *Orthologsorter*, onde será possível realizar diversas buscas que possam explicar os relacionamentos entre as espécies de interesse.

Outra saída obtida a partir do *pipeline* é uma árvore filogenética que pode contribuir para inferência da história evolutiva das espécies estudadas. Para a construção da árvore filogenética executamos o programa MUSCLE em cada família, Gblocks, RAxML e por fim o NJplot.

A árvore a ser construída por RAxML será feita a partir da concatenação das famílias contendo exatamente um gene de cada genoma *ingroup* e no máximo um gene dos genomas *outgroup*. Tais famílias encontradas são alinhadas através do MUSCLE e, em seguida, cada alinhamento é filtrado através do GBlocks. Todos os alinhamentos são concatenados e repassados ao RAxML, responsável por construir a árvore no formato Newick. O NJplot é um programa capaz de desenhar qualquer árvore filogenética expressa no formato Newick para o formato padrão de árvore filogenéticas, como mostrado nas figuras do Capítulo 6.

Finalmente, o Passo 3 é responsável pela construção da ferramenta *web Orthologsorter* e da árvore filogenética das espécies de interesse, e pode ser descrito como segue.

Passo 3. Execute o script `createOrthologsorter.pl`, disponível no pacote, de acordo com o seguinte comando:
$ perl `createOrthologsorter.pl` *<myproject>*

5.4 Implementação

Nesta seção abordamos alguns aspectos acerca da implementação do *Orthologsorter*, em especial as tabelas e consultas utilizadas na busca das famílias de proteínas.

Como pode ser visto na Figura 5.5, o elemento central de todo o *pipeline* é o *Orthologsorter*, sendo esse o mecanismo de busca por famílias em um banco de dados MySQL, famílias essas que foram encontradas pelo OrthoMCL. Essas buscas podem ser feitas pelo usuário por meio da página HTML também gerada automaticamente, enquanto que uma busca em particular é feita durante a execução do *pipeline* visando a construção da árvore de espécies. Assim, é importante termos um banco de dados cuja arquitetura favoreça essas buscas.

5.4.1 Arquitetura do banco

O Sistema Gerenciador de Banco de Dados escolhido para a criação e manutenção da base de dados foi o MySQL. O MySQL é um servidor robusto de bancos de dados SQL (*Structured Query Language*), além de ser um software de utilização livre.

O banco de dados desenvolvido é composto por 3 tabelas: a tabela `genome`, a tabela `gene` e a tabela `family`. Descreveremos a seguir essas três tabelas.

Na tabela `genome`, são mantidas as seguintes informações:

- `genome_id`: inteiro que armazena uma identificação para o organismo;
- `genome_name`: *string* que identifica o arquivo que contém o conjunto de proteínas preditas do organismo;

- `genome_info`: *string* que armazena as informações adicionais a respeito do organismo;
- `genome_ingroup`: inteiro que armazena se o organismo é *ingroup* (1) ou *outgroup* (0);
- `genome_mygenome`: inteiro que armazena se as sequências do organismo foram já depositadas no NCBI (0) ou não (1);

Na tabela `gene`, as informações armazenadas são:

- `gene_gi`: *string* que armazena o identificador da sequência (*gi*);
- `gene_genome`: inteiro que armazena o número de identificação do genoma ao qual o gene pertence;
- `gene_product`: *string* que armazena o produto do gene;
- `gene_sequence`: *string* que armazena a sequência proteica;
- `gene_family`: inteiro que identifica o número da família ao qual o gene pertence;

Na tabela `family`, são mantidas as seguintes informações:

- `family_id`: inteiro que armazena uma identificação para a família de proteína;
- `family_set`: *string* que armazena a quantidade de genes de cada genoma que pertence à respectiva família;

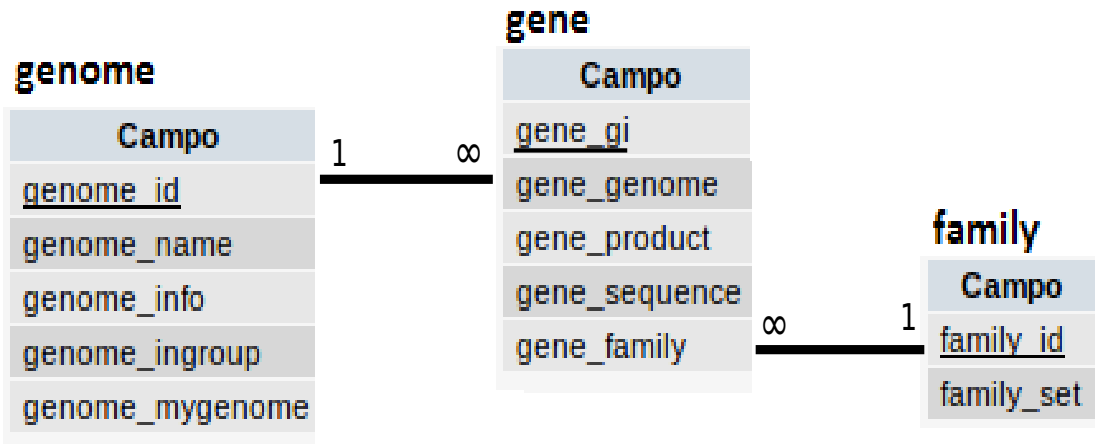


Figura 5.6: Modelagem do banco de dados para o complexo *B. cereus* com as ligações entre as tabelas.

Na Figura 5.6 são mostradas as ligações entre as três tabelas. O campo `genome_id` na tabela `genome` corresponde ao campo `gene_genome` na tabela `gene`. O campo `gene_family` na tabela `gene` corresponde ao campo `family_id` na tabela `family`. As Figuras 5.7, 5.8 e 5.9 mostram, respectivamente, trechos iniciais das tabelas `genome`, `gene` e `family` para o complexo *B. cereus*.

5.4.2 Como funciona uma busca

Os atributos da tabela `family` permitem uma busca eficiente por famílias, uma vez que o campo `family_set` é uma *string* simulando um vetor com o número de posições correspondente ao total de genomas. As posições são separadas por vírgulas e, em cada posição, é armazenada a quantidade de genes de cada genoma que à respectiva família possui. Por exemplo, na primeira posição consta o número de genes que o genoma com `genome_id = 1` possui naquela família.

A Figura 5.9 mostra o trecho inicial da tabela `family` contendo 6383 famílias para o complexo *B. cereus*. Considerando a família de proteína com `family_id = 2`, observe que a primeira posição mostra que a família contém 2 genes do genoma com `genome_id = 1`, denominado *Bacillus anthracis* AE016879. Em seguida, a segunda posição mostra que a família contém 4 genes do genoma com `genome_id = 2`,

denominado *Bacillus cereus* ATCC 10987 e assim por diante.

Observe também, na Figura 5.9, as famílias que possuem o campo `family_id` com valor negativo representam a quantidade de *singletons* que o respectivo genoma possui. Considerando `family_id = -1` observe que a primeira posição mostra que o genoma *Bacillus anthracis* AE016879 possui 210 *singletons*. Já o genoma *Bacillus cereus* ATCC 10987 possui 852 *singletons*, como mostra a `family_id = -2`.

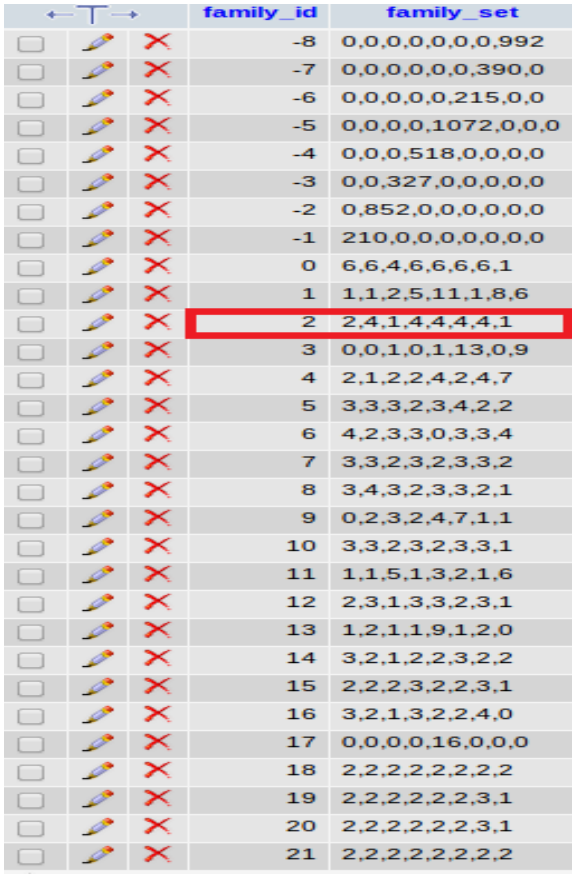
genome_id	genome_name	genome_info	genome_ingroup	genome_mygenome
1	Bacillus anthracis AE016879	Bacillus_anthracis_AE016879	1	0
2	Bacillus cereus ATCC 10987	Bacillus_cereus_ATCC_10987	1	0
3	Bacillus cytotoxicus NVH 391 98	Bacillus_cytotoxicus_NVH_391_98	1	0
4	Bacillus mycoides DSM 2048	Bacillus_mycoides_DSM_2048	1	0
5	Bacillus pseudomycoides DSM 12442	Bacillus_pseudomycoides_DSM_12442	1	0
6	Bacillus thuringiensis serovar konkukian 97 27	Bacillus_thuringiensis_serovar_konkukian_97_27	1	0
7	Bacillus weihenstephanensis KBAB4	Bacillus_weihenstephanensis_KBAB4	1	0
8	Bacillus pumilus SAFR 032	Bacillus_pumilus_SAFR_032	0	0

Figura 5.7: Trecho inicial da tabela `genome` contendo 8 genomas para o complexo *B. cereus*.

gene_gi	gene_genome	gene_product	gene_sequence	gene_family
30260196|ref|NP_842573.1|	1	chromosomal replication initiation protein [Bacill...	MENISDLWNSALKELEKKVSKPSYETWLKSTTAHNLKKDVLTITAPNEFA...	1829
30260197|ref|NP_842574.1|	1	DNA polymerase III subunit beta [Bacillus anthraci...	MRFSIQKDYLVRSVQDVMKAVSFRTTIPILTGIKVVATEEGVTLTGSDAD...	1828
30260198|ref|NP_842575.1|	1	hypothetical protein BA_0003 [Bacillus anthracis s...	MKRIKISTEYITLGQFLKLADVIDTGGAVKWFLQEYEVYVNQELENRRGR...	1827
30260199|ref|NP_842576.1|	1	recombination protein F [Bacillus anthracis str. A...	MFISEIQLKNYRNYEKLELSFEDKVNVIIGENAQGKTNLMEAIYVLAMAK...	1826
30260200|ref|NP_842577.1|	1	DNA gyrase subunit B [Bacillus anthracis str. Ames...	MEQKQMQENSYDESQIQVLEGLEAVRKRPGMYIGSTSGKGLHHLVWEIVD...	23
30260201|ref|NP_842578.1|	1	DNA gyrase subunit A [Bacillus anthracis str. Ames...	MSDNQQQARIREINISHEMRTSFLDYAMSVIVSRALPDVRDGLKPVHRRV...	22
30260202|ref|NP_842579.1|	1	inosine 5-monophosphate dehydrogenase [Bacillus an...	MWESKFVKEGLTFDDVLLVPAKSDVLPREVSVKTVLSESLQLNIPLISAG...	1825
30260203|ref|NP_842580.1|	1	D-alanyl-D-alanine carboxypeptidase [Bacillus anth...	MFCKRFIALVTMLTLVCSMLLPYSNASAETGSALNIEAGAAILVEANSGK...	75

Figura 5.8: Trecho inicial da tabela `gene` contendo 40850 genes para o complexo *B. cereus*.

Por meio da página HTML o usuário pode realizar buscas por famílias em um banco de dados MySQL. Por exemplo, a busca mostrada na Figura 6.1 procura todas as famílias contendo exatamente um gene de todos os genomas (*ingroup*), exceto para o *Bacillus pumilus* SAFR 032 (*outgroup*), que deve conter no máximo um gene.



family_id	family_set
-8	0,0,0,0,0,0,0,992
-7	0,0,0,0,0,0,390,0
-6	0,0,0,0,0,215,0,0
-5	0,0,0,0,1072,0,0,0
-4	0,0,0,518,0,0,0,0
-3	0,0,327,0,0,0,0,0
-2	0,852,0,0,0,0,0,0
-1	210,0,0,0,0,0,0,0
0	6,6,4,6,6,6,6,1
1	1,1,2,5,11,1,8,6
2	2,4,1,4,4,4,4,1
3	0,0,1,0,1,13,0,9
4	2,1,2,2,4,2,4,7
5	3,3,3,2,3,4,2,2
6	4,2,3,3,0,3,3,4
7	3,3,2,3,2,3,3,2
8	3,4,3,2,3,3,2,1
9	0,2,3,2,4,7,1,1
10	3,3,2,3,2,3,3,1
11	1,1,5,1,3,2,1,6
12	2,3,1,3,3,2,3,1
13	1,2,1,1,9,1,2,0
14	3,2,1,2,2,3,2,2
15	2,2,2,3,2,2,3,1
16	3,2,1,3,2,2,4,0
17	0,0,0,0,16,0,0,0
18	2,2,2,2,2,2,2,2
19	2,2,2,2,2,2,3,1
20	2,2,2,2,2,2,3,1
21	2,2,2,2,2,2,2,2

Figura 5.9: Trecho inicial da tabela `family` contendo 6383 famílias para o complexo *B. cereus*. As famílias que possuem o campo `family_id` com valor negativo representam a quantidade de *singletons* que o respectivo genoma possui.

A busca primeiramente cria dois vetores: um com a quantidade mínima de genes que cada genoma deve conter, denominado 'MIN', e outro com a quantidade máxima de genes que cada genoma deve conter, denominado 'MAX'. Para cada família retornada do banco, é criado um vetor intermediário com a quantidade de genes de cada genoma que a respectiva família possui. Na Figura 5.10 são mostrados os vetores 'MIN', 'MAX' e o vetor intermediário construídos para as famílias com `family_id = 2` e `family_id = 2004`. A família com `family_id = 2` será desconsiderada pois as quantidades de genes não estão dentro dos limites determinados na busca. Já a família com `family_id = 2004` é considerada e mostrada nos resultados. O processo de construir o vetor intermediário, realizar as devidas comparações com o vetor 'MIN' e 'MAX' e verificar se a família está dentro dos limites determinados pelo usuário na busca é feito para cada família de proteína do banco de dados. O resultado é mostrado como na Figura 6.2.

MIN		FAMÍLIA 2		MAX	MIN		FAMÍLIA 2004		MAX
1	≤	2	≤	**1**	**1**	≤	1	≤	**1**
1	≤	4	≤	**1**	**1**	≤	1	≤	**1**
1	≤	1	≤	**1**	**1**	≤	1	≤	**1**
1	≤	4	≤	**1**	**1**	≤	1	≤	**1**
1	≤	4	≤	**1**	**1**	≤	1	≤	**1**
1	≤	4	≤	**1**	**1**	≤	1	≤	**1**
1	≤	4	≤	**1**	**1**	≤	1	≤	**1**
0	≤	1	≤	**1**	**0**	≤	0	≤	**1**

Figura 5.10: Exemplo de busca com `family_id = 2` e `family_id = 2004` para o complexo *B. cereus*.

Capítulo 6

Resultados e discussão

Neste capítulo os resultados da utilização do *Orthologsorter* são mostrados para dois grupos de genomas. A Seção 6.1 contém vários genomas do grupo *Bacillus cereus* e está descrito uma comparação da técnica de MLST com o *pipeline* desenvolvido. A Seção 6.2 contém genomas do grupo *Mycobacterium* e está descrito uma comparação das técnica de espoligotipagem e MIRU-VNTR com o *pipeline* desenvolvido. A Seção 6.3 traz algumas considerações acerca dos tempos de execução.

6.1 Grupo *Bacillus cereus*

O grupo *Bacillus cereus* lato sensu reúne espécies de bactérias Gram-positivas proximamente relacionadas, entre as quais se incluem *B.cereus, B. thuringiensis, B. anthracis, B. mycoides, B. pseudomycoides, B. cytotoxicus* e *B. weihenstephanensis* [41, 40].

Algumas das principais características comuns ao grupo são metabolismo anaeróbico facultativo, formação de endosporos e distribuição ubíqua [86]. O grupo recebeu uma atenção especial, uma vez que possui patógenos de insetos como *B. thuringiensis* e patógenos humanos como *B. cereus* e *B. anthracis* [41].

Existe um número considerável de pesquisas que exploram diferentes aspectos do grupo. Entre estes pode-se citar a seleção de cepas que possam ser usadas como con-

trole biológico de insetos [11], manipulação de genes codificadores de enterotoxinas envolvidas em doenças causadas por ingestão de alimentos contaminados [28, 50, 34], busca por relações filogenéticas entre as espécies [102] e investigação da estrutura populacional do complexo *B. cereus* [41].

Diferentes métodos confirmam a alta diversidade genética do grupo *B. cereus* [21, 36, 90, 94]. Entretanto, tais métodos apresentam limitações para a padronização de parâmetros entre laboratórios e para a comparação de resultados de coleções extensas de cepas [37]. Desse modo, a técnica MLST (*Multi Locus Sequence Typing*) mostrou-se como uma eficiente alternativa padronizada de análise genética de cepas do grupo *B. cereus* [37, 91].

O MLST pode ser definido como um sistema de tipagem baseado no sequenciamento e comparação de fragmentos de genes *housekeeping*. O método tem sido padronizado para várias espécies bacterianas, e bancos de dados fornecem dados de referência de tipagem [51, 3, 91].

Genes *housekeeping* (tal como GAPDH, HSP90 e β-actina) referem-se a genes envolvidos em funções celulares essenciais à vida da célula e continuamente expressos em todas as células de um organismo [16, 84].

Para realizar uma análise de MLST, esquemas com diferentes composições de genes *housekeeping* podem ser utilizados para estudar a estrutura populacional ou a dinâmica epidemiológica de um mesmo grupo de espécies bacterianas.

Existem cinco esquemas MLST desenvolvidos para o grupo *B. cereus*, totalizando 25 genes *housekeeping*, disponíveis no banco de dados *SuperCAT* [91]. O atual banco de dados contém um total de 1484 isolados e 26 fragmentos dos 25 genes *housekeeping*. Embora cada esquema seja capaz de discriminar espécies, foi utilizado um esquema que corresponde à combinação de todos os cinco esquemas, para construir, utilizando as ferramentas online do banco de dados *SuperCAT*, a árvore baseada em análise de MLST de genes *housekeeping*. A Tabela 6.1 apresenta os cinco esquemas e seus respectivos genes *housekeeping*.

Os experimentos relativos ao grupo *B. cereus* foram realizados por meio da análise de MLST e da abordagem computacional, a partir de sequências nucleotídicas e proteicas, respectivamente. Os resultados destes experimentos podem ser vistos com

Tabela 6.1: Genes *housekeeping* de cada esquema do grupo *B. cereus*.

Esquema	Genes
Tourasse-Helgason	*adk, ccpA, glpF, glpT, panC, pta, pycA*
Helgason	*adk, ccpA, ftsA, glpT, pyrE, recF, sucC*
Ko	*gyrB, mbl, mdh, mutS, pycA, rpoB*
Priest	*glpF, gmk, ilvD, pta, purH, pycA, tpi*
Candelon-Sorokin	*clpC, dinB, gdpD, panC, purF, yhfL*

mais detalhes em [55]. Para ambos os métodos, os experimentos foram realizados utilizando o conjunto de cepas mostrado na Tabela 6.2.

Tabela 6.2: Conjunto de cepas do grupo *Bacillus cereus* utilizado nos experimentos.

B. cereus ATCC 10987
B. thuringiensis serovar konkukian 97-27
B. anthracis AE016879
B. mycoides DSM 2048
B. pseudomycoides DSM 12442
B. cytotoxicus NVH 391/98
B. weihenstephanensis KBAB4

Foi incluído o genoma *B. pumilus* SAFR 032 como *outgroup*, de modo que este proporcionasse uma noção de onde, na árvore, se encontra o principal grupo de organismos. Todos os genomas foram obtidos do repositório NCBI.

Para a análise de MLST, foi obtida uma árvore filogenética contendo cepas do grupo *B. cereus*, a qual consiste em uma subárvore de uma superárvore construída por meio das ferramentas do banco *SuperCAT*. A superárvore consiste de uma árvore filogenética que contém todos os organismos armazenados no banco *SuperCAT*. Para obter a subárvore, os cinco esquemas MLST foram usados simultaneamente, e os resultados de tal análise permitiram que para cada gene fosse construída uma árvore pelo método de máxima verossimilhança (cujo conceito foi exposto na seção 4.2). A subárvore contendo as cepas selecionadas foi extraída da superárvore e é mostrada na Figura 6.8.

Os genomas do grupo *B. cereus* foram comparados um contra o outro por meio da ferramenta Blastp [6] e agrupados pelo OrthoMCL [47], resultando em 6375 famílias de produtos gênicos. O arquivo no formato fasta de cada família é gerado.

No próximo passo do *pipeline*, utilizando o *Orthologsorter* são selecionadas as

famílias contendo exatamente uma proteína de cada genoma e no máximo uma proteína do *outgroup* (*Bacillus pumilus* SAFR 032) resultando em 2429 famílias, como mostrado nas Figuras 6.1 e 6.2.

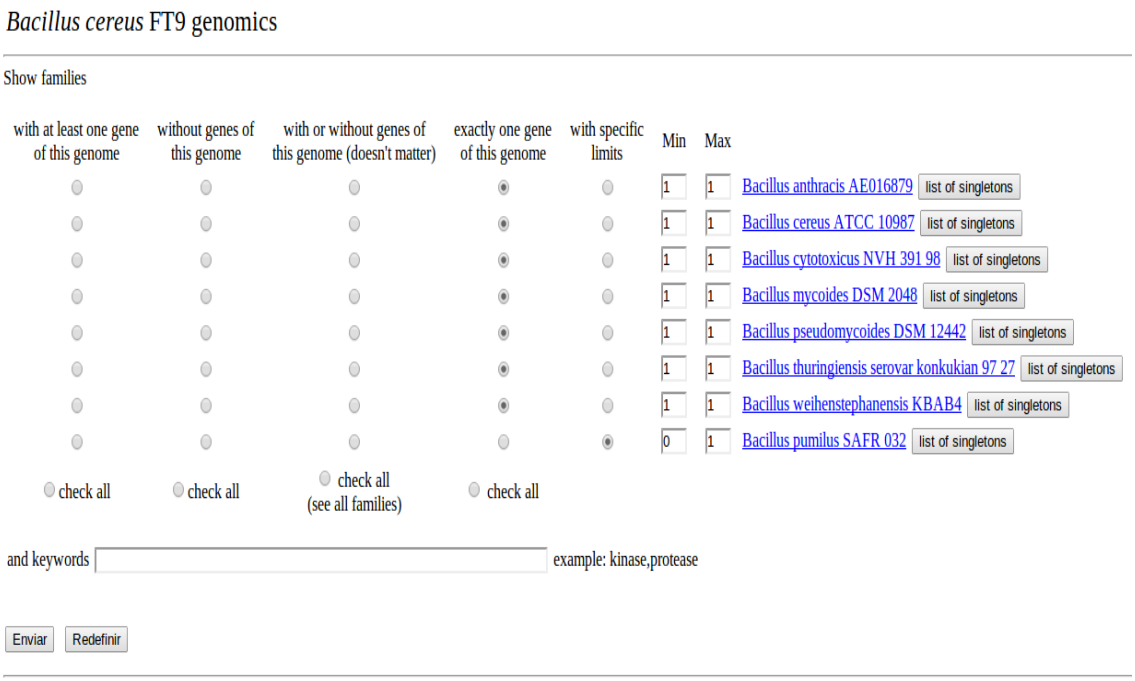
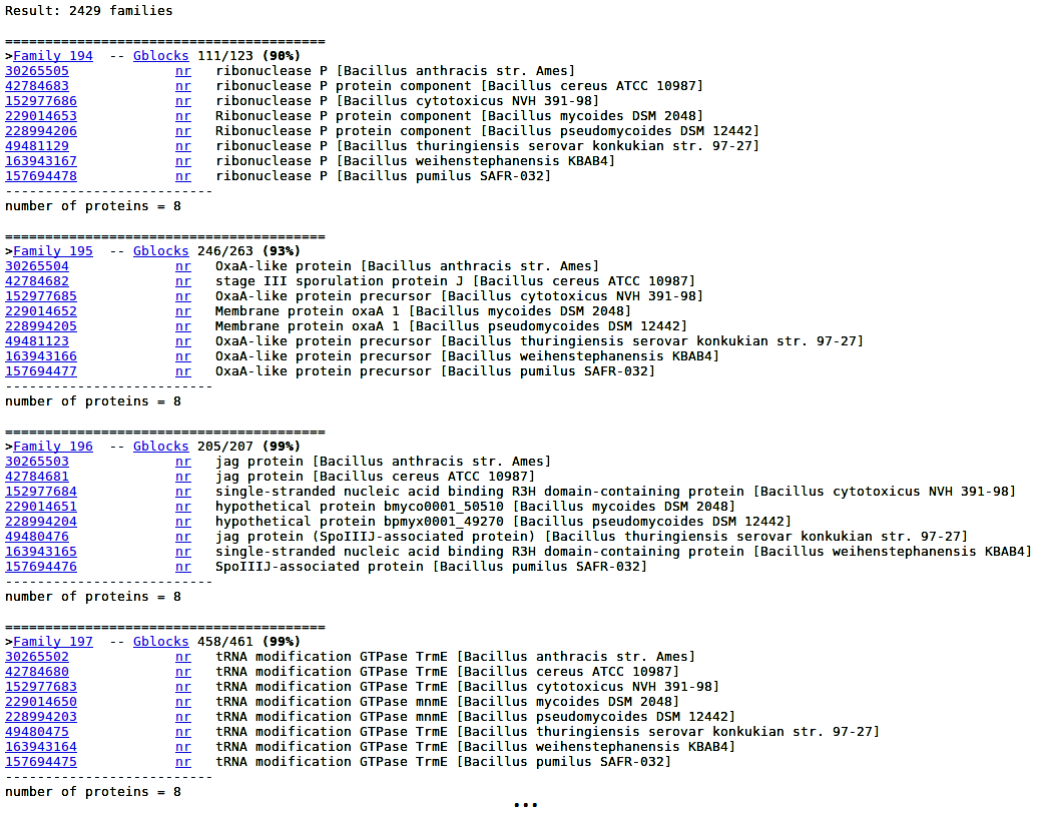


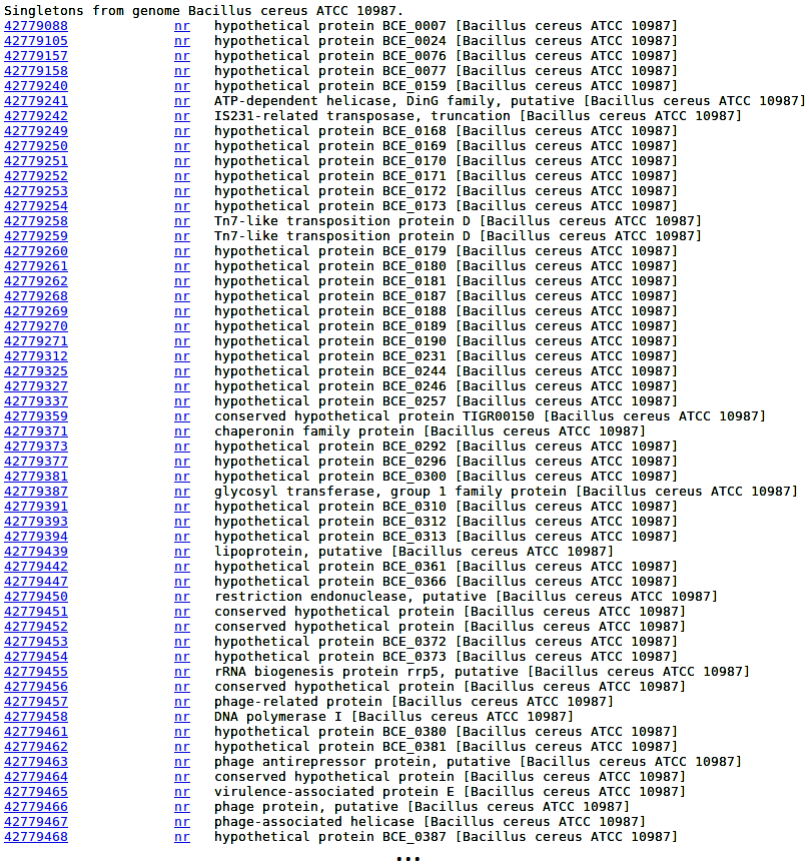
Figura 6.1: Tela principal do *Orthologsorter* para o grupo *B. cereus*.



```
Result: 2429 families

================================================
>Family 194  -- Gblocks 111/123 (90%)
30265505              nr   ribonuclease P [Bacillus anthracis str. Ames]
42784683              nr   ribonuclease P protein component [Bacillus cereus ATCC 10987]
152977686             nr   ribonuclease P [Bacillus cytotoxicus NVH 391-98]
229014653             nr   Ribonuclease P protein component [Bacillus mycoides DSM 2048]
228994206             nr   Ribonuclease P protein component [Bacillus pseudomycoides DSM 12442]
49481129              nr   ribonuclease P [Bacillus thuringiensis serovar konkukian str. 97-27]
163943167             nr   ribonuclease P [Bacillus weihenstephanensis KBAB4]
157694478             nr   ribonuclease P [Bacillus pumilus SAFR-032]
---------------------------
number of proteins = 8

================================================
>Family 195  -- Gblocks 246/263 (93%)
30265504              nr   OxaA-like protein [Bacillus anthracis str. Ames]
42784682              nr   stage III sporulation protein J [Bacillus cereus ATCC 10987]
152977685             nr   OxaA-like protein precursor [Bacillus cytotoxicus NVH 391-98]
229014652             nr   Membrane protein oxaA 1 [Bacillus mycoides DSM 2048]
228994205             nr   Membrane protein oxaA 1 [Bacillus pseudomycoides DSM 12442]
49481123              nr   OxaA-like protein precursor [Bacillus thuringiensis serovar konkukian str. 97-27]
163943166             nr   OxaA-like protein precursor [Bacillus weihenstephanensis KBAB4]
157694477             nr   OxaA-like protein precursor [Bacillus pumilus SAFR-032]
---------------------------
number of proteins = 8

================================================
>Family 196  -- Gblocks 205/207 (99%)
30265503              nr   jag protein [Bacillus anthracis str. Ames]
42784681              nr   jag protein [Bacillus cereus ATCC 10987]
152977684             nr   single-stranded nucleic acid binding R3H domain-containing protein [Bacillus cytotoxicus NVH 391-98]
229014651             nr   hypothetical protein bmyco0001_50510 [Bacillus mycoides DSM 2048]
228994204             nr   hypothetical protein bpmyx0001_49270 [Bacillus pseudomycoides DSM 12442]
49480476              nr   jag protein (SpoIIIJ-associated protein) [Bacillus thuringiensis serovar konkukian str. 97-27]
163943165             nr   single-stranded nucleic acid binding R3H domain-containing protein [Bacillus weihenstephanensis KBAB4]
157694476             nr   SpoIIIJ-associated protein [Bacillus pumilus SAFR-032]
---------------------------
number of proteins = 8

================================================
>Family 197  -- Gblocks 458/461 (99%)
30265502              nr   tRNA modification GTPase TrmE [Bacillus anthracis str. Ames]
42784680              nr   tRNA modification GTPase TrmE [Bacillus cereus ATCC 10987]
152977683             nr   tRNA modification GTPase TrmE [Bacillus cytotoxicus NVH 391-98]
229014650             nr   tRNA modification GTPase mnmE [Bacillus mycoides DSM 2048]
228994203             nr   tRNA modification GTPase mnmE [Bacillus pseudomycoides DSM 12442]
49480475              nr   tRNA modification GTPase TrmE [Bacillus thuringiensis serovar konkukian str. 97-27]
163943164             nr   tRNA modification GTPase TrmE [Bacillus weihenstephanensis KBAB4]
157694475             nr   tRNA modification GTPase TrmE [Bacillus pumilus SAFR-032]
---------------------------
number of proteins = 8
                                                                                ...
```

Figura 6.2: Parte do resultado da busca por famílias contendo exatamente uma proteína de cada genoma e no máximo uma proteína do *outgroup* (*Bacillus pumilus* SAFR 032) para o grupo *B. cereus*.

Ao clicar no botão 'list of singletons', como mostrado na Figura 6.1, pode-se visualizar as sequências do respectivo organismo que não pertencem a nenhuma família de proteína. Um exemplo para o genoma '*Bacillus cereus* ATCC 10987' pode ser visto na Figura 6.3.



```
Singletons from genome Bacillus cereus ATCC 10987.
42779088            nr   hypothetical protein BCE_0007 [Bacillus cereus ATCC 10987]
42779105            nr   hypothetical protein BCE_0024 [Bacillus cereus ATCC 10987]
42779157            nr   hypothetical protein BCE_0076 [Bacillus cereus ATCC 10987]
42779158            nr   hypothetical protein BCE_0077 [Bacillus cereus ATCC 10987]
42779240            nr   hypothetical protein BCE_0159 [Bacillus cereus ATCC 10987]
42779241            nr   ATP-dependent helicase, DinG family, putative [Bacillus cereus ATCC 10987]
42779242            nr   IS231-related transposase, truncation [Bacillus cereus ATCC 10987]
42779249            nr   hypothetical protein BCE_0168 [Bacillus cereus ATCC 10987]
42779250            nr   hypothetical protein BCE_0169 [Bacillus cereus ATCC 10987]
42779251            nr   hypothetical protein BCE_0170 [Bacillus cereus ATCC 10987]
42779252            nr   hypothetical protein BCE_0171 [Bacillus cereus ATCC 10987]
42779253            nr   hypothetical protein BCE_0172 [Bacillus cereus ATCC 10987]
42779254            nr   hypothetical protein BCE_0173 [Bacillus cereus ATCC 10987]
42779258            nr   Tn7-like transposition protein D [Bacillus cereus ATCC 10987]
42779259            nr   Tn7-like transposition protein D [Bacillus cereus ATCC 10987]
42779260            nr   hypothetical protein BCE_0179 [Bacillus cereus ATCC 10987]
42779261            nr   hypothetical protein BCE_0180 [Bacillus cereus ATCC 10987]
42779262            nr   hypothetical protein BCE_0181 [Bacillus cereus ATCC 10987]
42779268            nr   hypothetical protein BCE_0187 [Bacillus cereus ATCC 10987]
42779269            nr   hypothetical protein BCE_0188 [Bacillus cereus ATCC 10987]
42779270            nr   hypothetical protein BCE_0189 [Bacillus cereus ATCC 10987]
42779271            nr   hypothetical protein BCE_0190 [Bacillus cereus ATCC 10987]
42779312            nr   hypothetical protein BCE_0231 [Bacillus cereus ATCC 10987]
42779325            nr   hypothetical protein BCE_0244 [Bacillus cereus ATCC 10987]
42779327            nr   hypothetical protein BCE_0246 [Bacillus cereus ATCC 10987]
42779337            nr   hypothetical protein BCE_0257 [Bacillus cereus ATCC 10987]
42779359            nr   conserved hypothetical protein TIGR00150 [Bacillus cereus ATCC 10987]
42779371            nr   chaperonin family protein [Bacillus cereus ATCC 10987]
42779373            nr   hypothetical protein BCE_0292 [Bacillus cereus ATCC 10987]
42779377            nr   hypothetical protein BCE_0296 [Bacillus cereus ATCC 10987]
42779381            nr   hypothetical protein BCE_0300 [Bacillus cereus ATCC 10987]
42779387            nr   glycosyl transferase, group 1 family protein [Bacillus cereus ATCC 10987]
42779391            nr   hypothetical protein BCE_0310 [Bacillus cereus ATCC 10987]
42779393            nr   hypothetical protein BCE_0312 [Bacillus cereus ATCC 10987]
42779394            nr   hypothetical protein BCE_0313 [Bacillus cereus ATCC 10987]
42779439            nr   lipoprotein, putative [Bacillus cereus ATCC 10987]
42779442            nr   hypothetical protein BCE_0361 [Bacillus cereus ATCC 10987]
42779447            nr   hypothetical protein BCE_0366 [Bacillus cereus ATCC 10987]
42779450            nr   restriction endonuclease, putative [Bacillus cereus ATCC 10987]
42779451            nr   conserved hypothetical protein [Bacillus cereus ATCC 10987]
42779452            nr   conserved hypothetical protein [Bacillus cereus ATCC 10987]
42779453            nr   hypothetical protein BCE_0372 [Bacillus cereus ATCC 10987]
42779454            nr   hypothetical protein BCE_0373 [Bacillus cereus ATCC 10987]
42779455            nr   rRNA biogenesis protein rrp5, putative [Bacillus cereus ATCC 10987]
42779456            nr   conserved hypothetical protein [Bacillus cereus ATCC 10987]
42779457            nr   phage-related protein [Bacillus cereus ATCC 10987]
42779458            nr   DNA polymerase I [Bacillus cereus ATCC 10987]
42779461            nr   hypothetical protein BCE_0380 [Bacillus cereus ATCC 10987]
42779462            nr   hypothetical protein BCE_0381 [Bacillus cereus ATCC 10987]
42779463            nr   phage antirepressor protein, putative [Bacillus cereus ATCC 10987]
42779464            nr   conserved hypothetical protein [Bacillus cereus ATCC 10987]
42779465            nr   virulence-associated protein E [Bacillus cereus ATCC 10987]
42779466            nr   phage protein, putative [Bacillus cereus ATCC 10987]
42779467            nr   phage-associated helicase [Bacillus cereus ATCC 10987]
42779468            nr   hypothetical protein BCE_0387 [Bacillus cereus ATCC 10987]
                         ...
```

Figura 6.3: Parte da lista das sequências proteicas que não pertencem a nenhuma família de proteínas para o genoma '*Bacillus cereus* ATCC 10987'.

Os alinhamentos, usando MUSCLE [26] e, filtrados com o Gblocks [22], foram concatenados resultando em 687,990 colunas. A partir do alinhamento produzido, o RAxML [78] permitiu obter uma árvore filogenética de máxima verossimilhança.

Observando a Figura 6.2:

- Ao acessar o *link* 'Family X' o arquivo no formato fasta da respectiva família é mostrado. Um exemplo deste tipo de arquivo pode ser visto na Figura 6.4.
- Ao acessar o *link* 'Gblocks' o alinhamento gerado pelo Gblocks da respectiva família é mostrado. A Figura 6.5 mostra um exemplo deste tipo de arquivo.
- Ao acessar o *link* com o identificador do gene (*gi*) é mostrado o resultado da busca no NCBI sobre o gene. A Figura 6.6 mostra um exemplo deste resultado.

- Ao acessar o *link* 'nr' é mostrado o resultado de um Blastp *online* do gene contra a base *nr* do GenBank. A Figura 6.7 mostra um exemplo deste resultado.

```
>152977686|ref|YP_001377203.1| ribonuclease P [Bacillus cytotoxicus NVH 391-98]
MKKKNRIKKNDEFQAVFQKGRSSANRQFVIYQLDKETQPYFRIGLSVSKKIGNAVVRNRI
KRMIRQSITELKDEIDSGKDFVIIARKPCAEMTYEEFKKSLIHAFKRSGIKITNK
>157694478|ref|YP_001488940.1| ribonuclease P [Bacillus pumilus SAFR-032]
MKKRNRLKKNEEFQKVFKKGQSMANRQFVIYQLDQPNQDELRLGLSVSKKIGNAVMRNRI
KRLIRQVFLEEGHKLKQEKDYIVIARKPASELTFEETKKSLQHLFRKSAVYQQQPKKSS
>163943167|ref|YP_001648051.1| ribonuclease P [Bacillus weihenstephanensis KBAB4]
MKKKNRIKKNDEFQAVFQKGQSNANRQFVVYKLDKEKQPHFRIGLSVSKKIGNAVVRNRI
KRMIRQSIIELKDEIDSGKDFVIIARKPCAEMTYEELKKSLIHVFKRSGMKRIKK
>228994206|ref|ZP_04154106.1| Ribonuclease P protein component [Bacillus pseudomycoides DSM 12442]
MSIDMKKKNRIKKNDEFQAVFQNGKSNANRQFVVYQLEKAEQSYFRIGLSVSKKLGNAVV
RNRIKRMIRQSIAELKDEIDSGKDFVIIARKPCADMTYEQLKKSLIHAFKRSGIKITNK
>229014653|ref|ZP_04171767.1| Ribonuclease P protein component [Bacillus mycoides DSM 2048]
MSNDMKKKNRIKKNDEFQAVFQKGQSNANRQFVVYKLDKEKQPHFRIGLSVSKKIGNAVV
RNRIKRMIRQSIIELKDEIDSGKDFVIIARKPCAEMTYEELKKSLIHVFKRSGMKRIKK
>30265505|ref|NP_847882.1| ribonuclease P [Bacillus anthracis str. Ames]
MKKKHRIKKNDEFQTVFQKGKSNANRQFVVYQLDKEEQPNFRIGLSVSKKIGNAVVRNRI
KRMIRQSITELKDEIDSGKDFVIIARKPCAEMTYEELKKSLIHVFKRSGMKRIKSSVRK
>42784683|ref|NP_981930.1| ribonuclease P protein component [Bacillus cereus ATCC 10987]
MKKKHRIKKNDEFQTVFQKGKSNANRQFVVYQLDKEEQPNFRIGLSVSKKIGNAVVRNRI
KRMIRQSITELKDEIDSGKDFVIIARKPCAEMTYEELKKSLIHVFKRSGMKRIKK
>49481129|ref|YP_039477.1| ribonuclease P [Bacillus thuringiensis serovar konkukian str. 97-27]
MKKKHRIKKNDEFQTVFQKGKSNANRQFVVYQLDKEEQPNFRIGLSVSKKIGNAVVRNRI
KRMIRQSITELKDEIDSGKDFVIIARKPCAEMTYEELKKSLIHVFKRSGMKRIKSSVRK
                                    ...
```

Figura 6.4: Parte do arquivo no formato fasta contendo as proteínas agrupadas na família 194 para o grupo *B. cereus*.

```
Gblocks 0.91b Results
Number of sequences: 8
Alignment assumed to be: Protein
New number of positions: 111 (selected positions are underlined in blue)

                    10        20        30        40        50        60
               =========+=========+=========+=========+=========+=========+
157694478      ----MKKRNRLKKNEEFQKVFKKGQSMANRQFVIYQLDQPNQDELRLGLSVSKKIGNAVM
228994206      MSIDMKKKNRIKKNDEFQAVFQNGKSNANRQFVVYQLEKAEQSYFRIGLSVSKKLGNAVV
152977686      ----MKKKNRIKKNDEFQAVFQKGRSSANRQFVIYQLDKETQPYFRIGLSVSKKIGNAVV
163943167      ----MKKKNRIKKNDEFQAVFQKGQSNANRQFVVYKLDKEKQPHFRIGLSVSKKIGNAVV
229014653      MSNDMKKKNRIKKNDEFQAVFQKGQSNANRQFVVYKLDKEKQPHFRIGLSVSKKIGNAVV
30265505       ----MKKKHRIKKNDEFQTVFQKGKSNANRQFVVYQLDKEEQPNFRIGLSVSKKIGNAVV
49481129       ----MKKKHRIKKNDEFQTVFQKGKSNANRQFVVYQLDKEEQPNFRIGLSVSKKIGNAVV
42784683       ----MKKKHRIKKNDEFQTVFQKGKSNANRQFVVYQLDKEEQPNFRIGLSVSKKIGNAVV
                   ############################################################

                    70        80        90       100       110       120
               =========+=========+=========+=========+=========+=========+
157694478      RNRIKRLIRQVFLEEGHKLKQEKDYIVIARKPASELTFEETKKSLQHLFRKSAVYQQQPK
228994206      RNRIKRMIRQSIAELKDEIDSGKDFVIIARKPCADMTYEQLKKSLIHAFKRSGIKITNK-
152977686      RNRIKRMIRQSITELKDEIDSGKDFVIIARKPCAEMTYEEFKKSLIHAFKRSGIKITNK-
163943167      RNRIKRMIRQSIIELKDEIDSGKDFVIIARKPCAEMTYEELKKSLIHVFKRSGMKRIKK-
229014653      RNRIKRMIRQSIIELKDEIDSGKDFVIIARKPCAEMTYEELKKSLIHVFKRSGMKRIKK-
30265505       RNRIKRMIRQSITELKDEIDSGKDFVIIARKPCAEMTYEELKKSLIHVFKRSGMKRIKSS
49481129       RNRIKRMIRQSITELKDEIDSGKDFVIIARKPCAEMTYEELKKSLIHVFKRSGMKRIKSS
42784683       RNRIKRMIRQSITELKDEIDSGKDFVIIARKPCAEMTYEELKKSLIHVFKRSGMKRIKK-
               ############################################################

               ===
157694478      KSS
228994206      ---
152977686      ---
163943167      ---
229014653      ---
30265505       VRK
49481129       VRK
42784683       ---

Parameters used
Minimum Number Of Sequences For A Conserved Position: 5
Minimum Number Of Sequences For A Flanking Position: 6
Maximum Number Of Contiguous Nonconserved Positions: 8
Minimum Length Of A Block: 10
Allowed Gap Positions: With Half
Use Similarity Matrices: Yes
Flank positions of the 1 selected block(s)
Flanks: [5  115]
New number of positions:  111  (90% of the original 123 positions)
```

Figura 6.5: Saída do Gblocks para a família de 8 proteínas, de *Bacillus*. As colunas marcadas com '#' são as mantidas para a construção da filogenia.

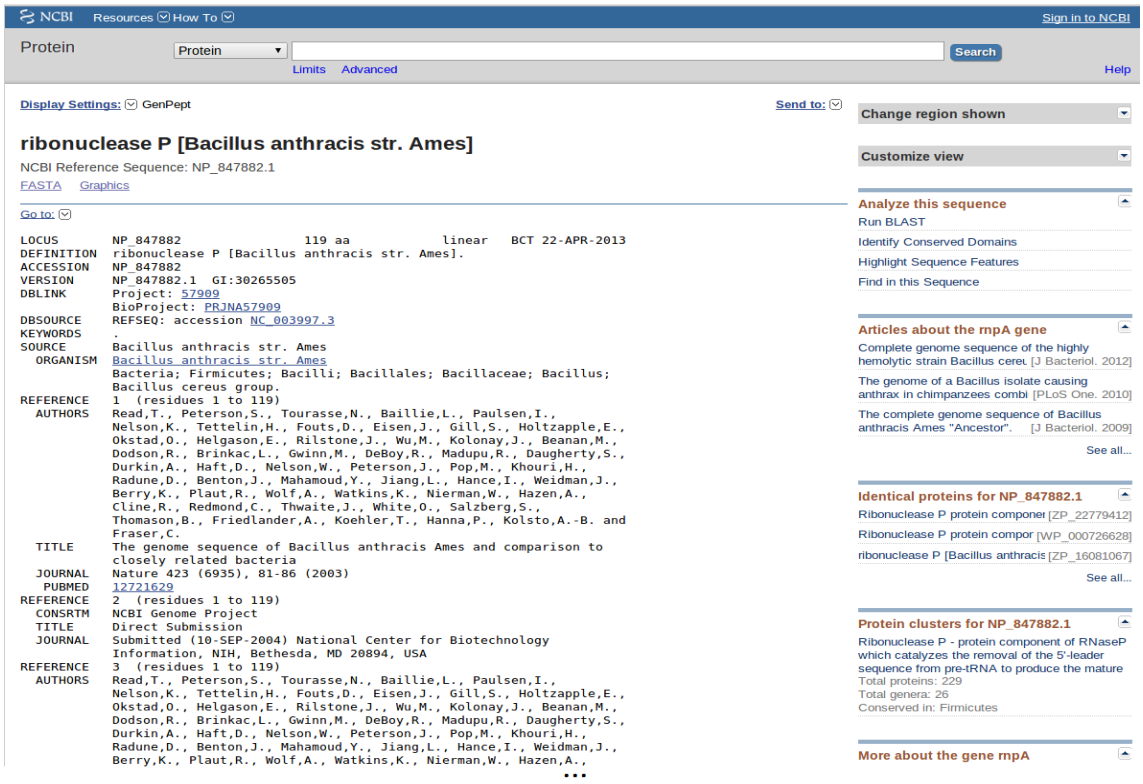
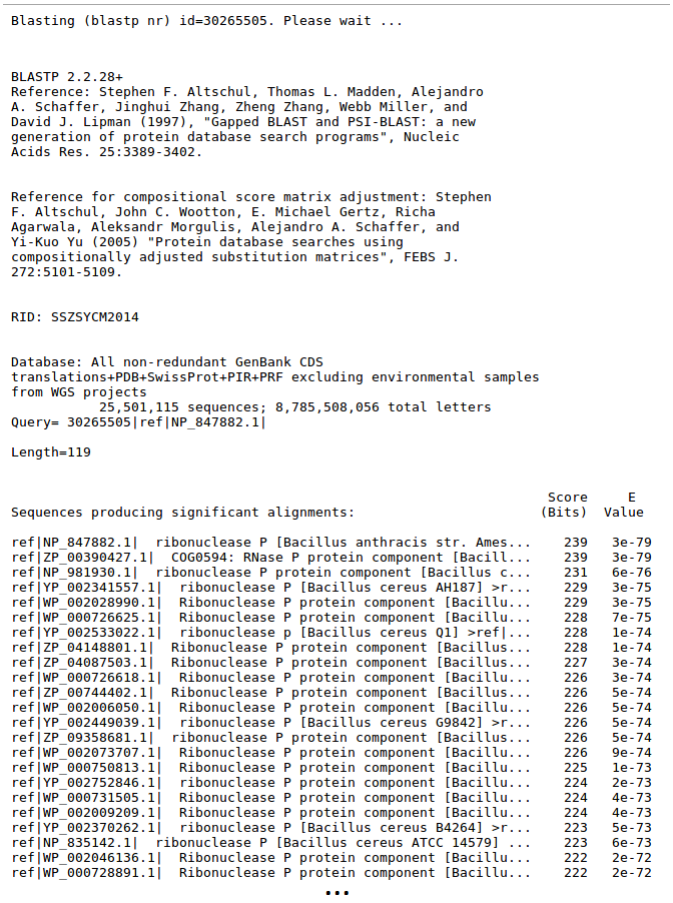


Figura 6.6: Parte do resultado da busca no NCBI da proteína com identificador do gene (*gi*) igual a '30265505'.



```
Blasting (blastp nr) id=30265505. Please wait ...


BLASTP 2.2.28+
Reference: Stephen F. Altschul, Thomas L. Madden, Alejandro
A. Schaffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and
David J. Lipman (1997), "Gapped BLAST and PSI-BLAST: a new
generation of protein database search programs", Nucleic
Acids Res. 25:3389-3402.


Reference for compositional score matrix adjustment: Stephen
F. Altschul, John C. Wootton, E. Michael Gertz, Richa
Agarwala, Aleksandr Morgulis, Alejandro A. Schaffer, and
Yi-Kuo Yu (2005) "Protein database searches using
compositionally adjusted substitution matrices", FEBS J.
272:5101-5109.


RID: SSZSYCM2014


Database: All non-redundant GenBank CDS
translations+PDB+SwissProt+PIR+PRF excluding environmental samples
from WGS projects
           25,501,115 sequences; 8,785,508,056 total letters
Query= 30265505|ref|NP_847882.1|

Length=119


                                                                  Score     E
Sequences producing significant alignments:                      (Bits)  Value

ref|NP_847882.1|  ribonuclease P [Bacillus anthracis str. Ames...   239   3e-79
ref|ZP_00390427.1|  COG0594: RNase P protein component [Bacill...   239   3e-79
ref|NP_981930.1|  ribonuclease P protein component [Bacillus c...   231   6e-76
ref|YP_002341557.1|  ribonuclease P [Bacillus cereus AH187] >r...   229   3e-75
ref|WP_002028990.1|  Ribonuclease P protein component [Bacillu...   229   3e-75
ref|WP_000726625.1|  Ribonuclease P protein component [Bacillu...   228   7e-75
ref|YP_002533022.1|  ribonuclease p [Bacillus cereus Q1] >ref|...   228   1e-74
ref|ZP_04148801.1|  Ribonuclease P protein component [Bacillus...   228   1e-74
ref|ZP_04087503.1|  Ribonuclease P protein component [Bacillus...   227   3e-74
ref|WP_000726618.1|  Ribonuclease P protein component [Bacillu...   226   3e-74
ref|ZP_00744402.1|  Ribonuclease P protein component [Bacillus...   226   5e-74
ref|WP_002006050.1|  Ribonuclease P protein component [Bacillu...   226   5e-74
ref|YP_002449039.1|  ribonuclease P [Bacillus cereus G9842] >r...   226   5e-74
ref|ZP_09358681.1|  ribonuclease P protein component [Bacillus...   226   5e-74
ref|WP_002073707.1|  Ribonuclease P protein component [Bacillu...   226   9e-74
ref|WP_000750813.1|  Ribonuclease P protein component [Bacillu...   225   1e-73
ref|YP_002752846.1|  ribonuclease P protein component [Bacillu...   224   2e-73
ref|WP_000731505.1|  Ribonuclease P protein component [Bacillu...   224   4e-73
ref|WP_002009209.1|  Ribonuclease P protein component [Bacillu...   224   4e-73
ref|YP_002370262.1|  ribonuclease P [Bacillus cereus B4264] >r...   223   5e-73
ref|NP_835142.1|  ribonuclease P [Bacillus cereus ATCC 14579] ...   223   6e-73
ref|WP_002046136.1|  Ribonuclease P protein component [Bacillu...   222   2e-72
ref|WP_000728891.1|  Ribonuclease P protein component [Bacillu...   222   2e-72
                                     ...
```

Figura 6.7: Parte do resultado de um Blastp *online* da proteína com identificador do gene (*gi*) igual a '30265505' contra a base *nr* do GenBank.

Foi realizada a comparação da topologia entre a árvore obtida pelas ferramentas do banco *SuperCAT* e a árvore construída a partir de comparações de genomas. A árvore fornecida pelo banco *SuperCAT* para o grupo *B. cereus* é mostrada na Figura 6.8. A árvore construída a partir de comparações de genomas é apresentada na Figura 6.9. Tais árvores possuem a mesma topologia e ambas têm 100% de concordância *bootstrapping*.

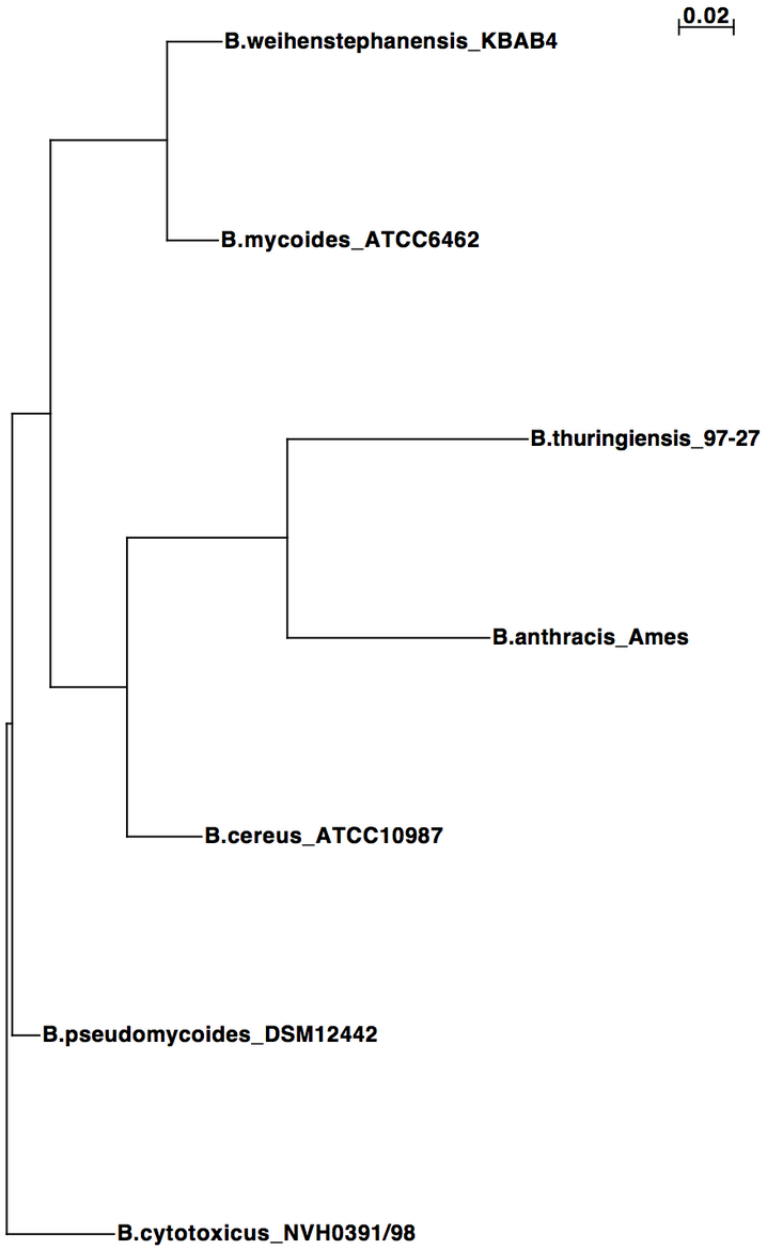


Figura 6.8: Árvore baseada nos genes *housekeeping* de *B. cereus*, obtida pelas ferramentas disponíveis no *SuperCAT Database*.

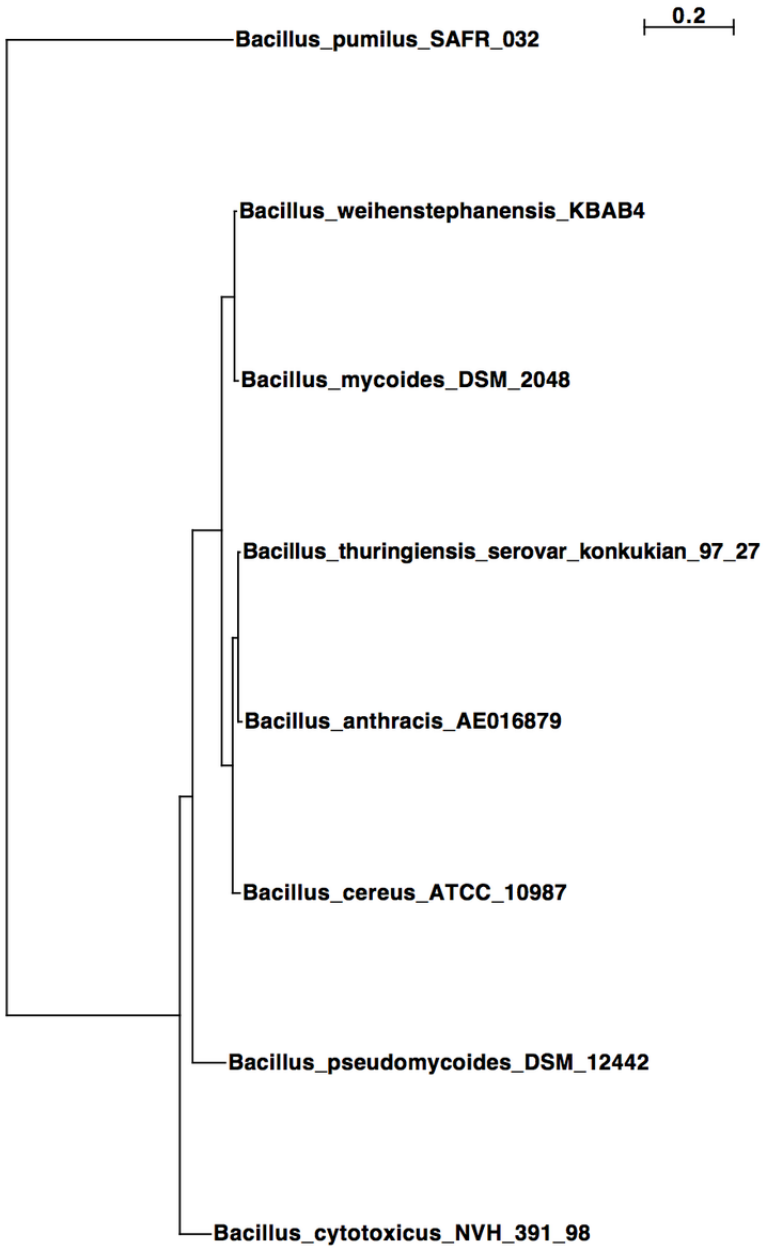


Figura 6.9: Árvore obtida segundo o *pipeline* descrito na Seção 5.3. O genoma de *B. pumilus* SAFR 032 foi utilizado como externo ao grupo *B. cereus*.

6.2 Grupo *Mycobacterium*

O complexo *Mycobacterium tuberculosis* (MTBC) reúne espécies de bactérias gram-positivas com forma de bastonete, proporcionando ao grupo uma ampla diversidade

de hospedeiros e de graus de patogenicidade [15]. O complexo é composto pelas espécies *M. tuberculosis, M. bovis, M. caprae, M. africanum, M. pinnipedii, M. microti* e *M. canetti* [31].

A tuberculose é uma doença crônica causada por bactérias do gênero *Mycobacterium* que ataca ruminantes, suínos, aves, animais silvestres e humanos. A tuberculose bovina é responsável por perdas econômicas significativas além de constituir uma das mais importantes zoonoses de relevância para a saúde pública. Com a globalização econômica, as barreiras tarifárias foram substituídas por barreiras sanitárias. Há uma pressão crescente de mercados importadores para o diagnóstico definitivo da tuberculose em bovinos [8].

Mycobacterium bovis é o principal causador da tuberculose bovina, mas há outras espécies de *Mycobacterium* que ocasionalmente infectam o gado causando lesões compatíveis com tuberculose (LCT), como *M. africanum* [97, 64], *M. caprae* [24], o *M. tuberculosis* [7], e outras micobactérias que podem estar associadas com LCT, como *M. fortuitum*, *M. smegmatis* e *M. phlei* [59].

Considerando a relevância do complexo e sua restrita variação genética, técnicas moleculares tem sido desenvolvidas a fim de encontrar uma maneira adequada de identificar cepas de MTBC. Os métodos tradicionais de identificação de cepas para este complexo exploram a variação de marcadores neutros, porções do genoma menos sujeitas a pressão seletiva [1], fato este que possibilita mutações relativamente recentes permanecerem no DNA e, assim, serem observadas.

Sequências repetidas tem sido consideradas uma fonte útil de variação para MTBC [23]. Os métodos de espoligotipagem e MIRU-VNTR analisam *loci* contendo sequências repetidas. O primeiro, tipagem de oligonucleotídeos espaçadores, baseia-se nos padrões de presença/ausência de 43 espaçadores do *locus* de repetição direta [39]. Tal *locus* possui cópias de uma sequência de 36pb, separadas por sequências conservadas e não repetitivas de 35–41pb, denominadas espaçadores [38, 92]. Apesar do poder de distinção de linhagens de MTBC, a espoligotipagem não pode diferenciar cepas pertencentes à mesma linhagem (com o mesmo espoligotipo ou espoligotipo similar). Outro aspecto a ser ressaltado é a ocorrência de homoplasias em árvores baseadas em espoligotipagem, como resultado de remoção independente de determinado espaçador em linhagens diferentes [52]. O termo homoplasia refere-se a uma característica similar compartilhada por duas ou mais espécies que não derivadas de

um ancestral comum. A homoplasia ocorre quando diferentes ambientes ou pressões seletivas desencadeiam o aparecimento ou o reaparecimento de características semelhantes em organismos que não compartilham um ancestral comum mais recente.

O segundo método, unidade repetitiva intercalada de *Mycobacterium*, tem como objeto de análise sequências formadas por repetição em *tandem* de número variável de 40–100pb exclusivas dos membros de MTBC [82]. Os padrões de MIRU-VNTR são formados por uma sequência de dígitos, em que cada dígito corresponde ao número de cópias de uma sequência de repetição em *tandem* de um certo locus [83]. Tal técnica, assim como a espoligotipagem, pode gerar algumas relações filogenéticas falso-positivas em virtude da existência de padrões similares em linhagens diferentes [52].

A fim de explorar ao máximo as possibilidades de genotipagem na epidemiologia molecular da tuberculose e também para estudar a filogenia das bactérias causadoras de maneira confiável, a aplicação da análise do sequenciamento de todo o genoma (WGS) para as espécies de micobactérias e/ou isolados tornou-se uma boa solução. O uso de WGS para tipagem molecular de patógenos tem sido explorado e rendeu informações adicionais importantes sobre a diversidade das cepas em comparação com os métodos clássicos de tipagem de DNA. Com a redução do custo do sequenciamento de DNA, é possível que o WGS torne-se a única ferramenta de diagnóstico da tuberculose para a identificação e caracterização genética [72].

Os experimentos relativos ao complexo *Mycobacterium* foram realizados por meio da análise do conjunto de cepas mostrado na Tabela 6.3. Mais detalhes destes experimentos podem ser vistos em [8].

Durante inspeções de rotina em um matadouro oficial, um touro de 36 meses, sem histórico de tuberculose, do estado do Rondônia, Brasil, apresentou lesões descritas como semi-sólidas e sem calcificação. As micobactérias cultivadas tiveram resposta negativa aos testes de rotina para *Mycobacterium bovis*. Esse isolado é denominado *Mycobacterium* 1407. Os demais genomas foram obtidos do repositório NCBI. Foi incluído o genoma *M. marinum* M como *outgroup*, de modo que este proporcionasse uma noção de onde, na árvore, encontra-se o principal grupo de organismos.

O conjunto de cepas do complexo *Mycobacterium* foram comparadas usando o Blastp [6] e agrupados pelo OrthoMCL [47], resultando em 7126 famílias de pro-

Tabela 6.3: Conjunto de cepas do complexo *Mycobacterium* utilizado nos experimentos.

M. 1407
M. africanum GM041182
M. avium 104
M. avium subsp *paratuberculosis* K 10
M. bovis AF2122 97
M. bovis BCG str Mexico
M. bovis BCG str Pasteur 1173P2
M. bovis BCG str Tokyo 172
M. smegmatis str MC2 155
M. tuberculosis H37Ra
M. tuberculosis H37Rv
M. marinum M

dutos gênicos. O arquivo no formato fasta de cada família é gerado.

No próximo passo do *pipeline*, utilizando o *Orthologsorter* são selecionadas as famílias contendo exatamente uma proteína de cada genoma e no máximo uma proteína do *outgroup* (*M. marinum* M) resultando em 1653 famílias, como mostrado nas Figuras 6.10 e 6.11. No caso da sequência proteica ser destacada na cor vermelha, como ocorre com as sequências '99907366', '99906450' e '99905939' mostradas na Figura 6.11, o respectivo genoma ainda não está disponível no NCBI.

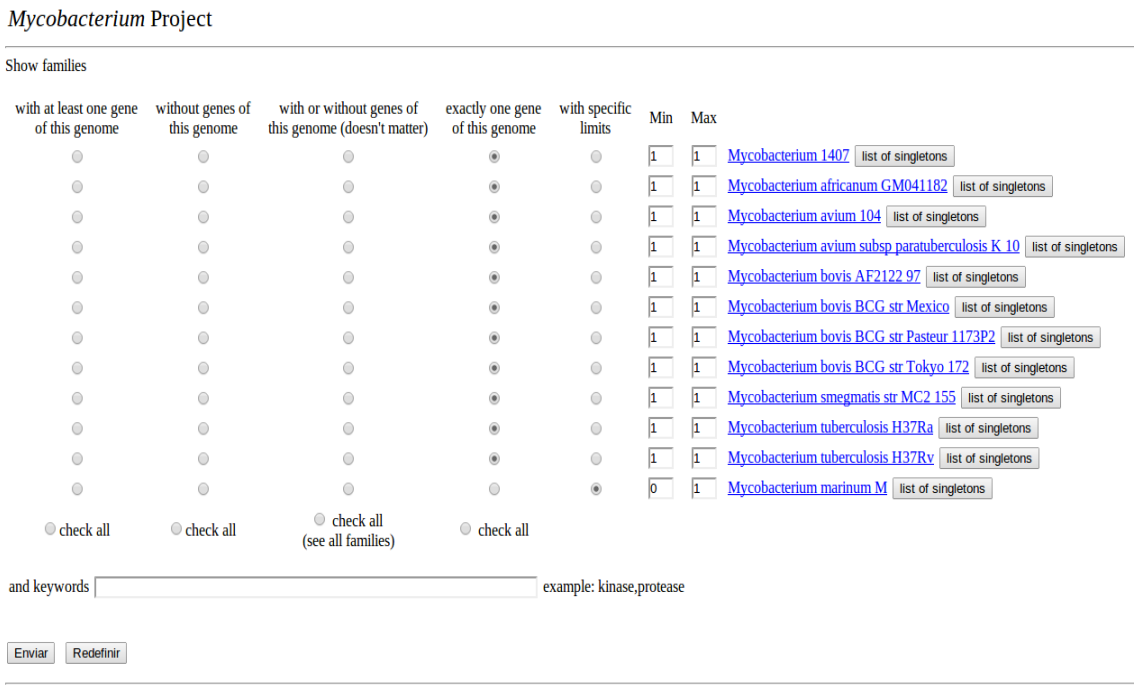
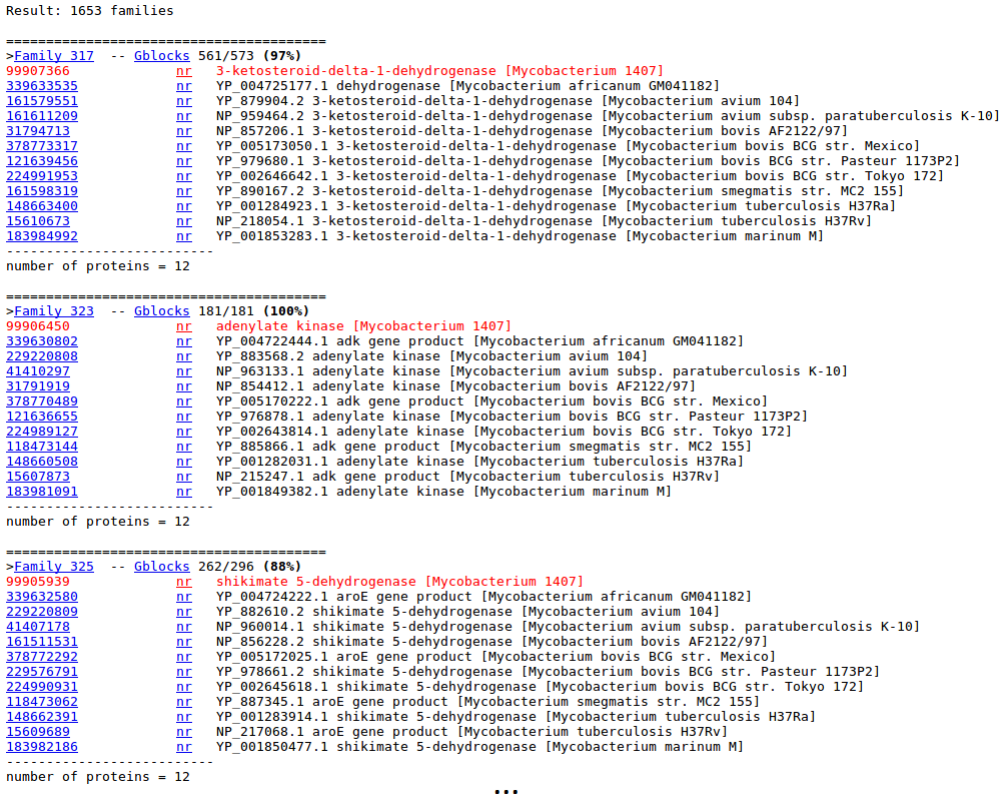


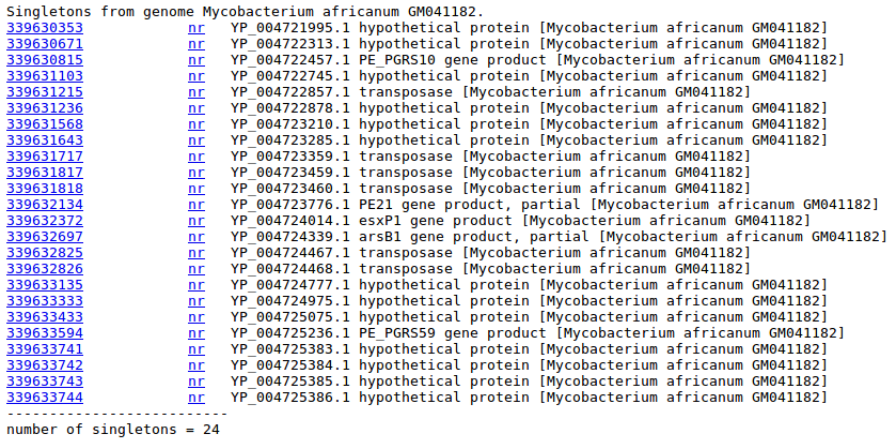
Figura 6.10: Tela principal do *Orthologsorter* para o complexo *Mycobacterium*.

Ao clicar no botão 'list of singletons', como mostrado na Figura 6.10, pode-se visualizar as sequências proteicas do respectivo organismo que não pertencem a nenhuma família de proteína. Um exemplo para o genoma '*M. africanum* GM041182' pode ser visto na Figura 6.12.



```
Result: 1653 families

========================================
>Family_317  -- Gblocks 561/573 (97%)
99907366            nr   3-ketosteroid-delta-1-dehydrogenase [Mycobacterium 1407]
339633535           nr   YP_004725177.1 dehydrogenase [Mycobacterium africanum GM041182]
161579551           nr   YP_879904.2 3-ketosteroid-delta-1-dehydrogenase [Mycobacterium avium 104]
161611209           nr   NP_959464.2 3-ketosteroid-delta-1-dehydrogenase [Mycobacterium avium subsp. paratuberculosis K-10]
31794713            nr   NP_857206.1 3-ketosteroid-delta-1-dehydrogenase [Mycobacterium bovis AF2122/97]
378773317           nr   YP_005173050.1 3-ketosteroid-delta-1-dehydrogenase [Mycobacterium bovis BCG str. Mexico]
121639456           nr   YP_979680.1 3-ketosteroid-delta-1-dehydrogenase [Mycobacterium bovis BCG str. Pasteur 1173P2]
224991953           nr   YP_002646642.1 3-ketosteroid-delta-1-dehydrogenase [Mycobacterium bovis BCG str. Tokyo 172]
161598319           nr   YP_890167.2 3-ketosteroid-delta-1-dehydrogenase [Mycobacterium smegmatis str. MC2 155]
148663400           nr   YP_001284923.1 3-ketosteroid-delta-1-dehydrogenase [Mycobacterium tuberculosis H37Ra]
15610673            nr   NP_218054.1 3-ketosteroid-delta-1-dehydrogenase [Mycobacterium tuberculosis H37Rv]
183984992           nr   YP_001853283.1 3-ketosteroid-delta-1-dehydrogenase [Mycobacterium marinum M]
-------------------------
number of proteins = 12

========================================
>Family_323  -- Gblocks 181/181 (100%)
99906450            nr   adenylate kinase [Mycobacterium 1407]
339630802           nr   YP_004722444.1 adk gene product [Mycobacterium africanum GM041182]
229220808           nr   YP_883568.2 adenylate kinase [Mycobacterium avium 104]
41410297            nr   NP_963133.1 adenylate kinase [Mycobacterium avium subsp. paratuberculosis K-10]
31791919            nr   NP_854412.1 adenylate kinase [Mycobacterium bovis AF2122/97]
378770489           nr   YP_005170222.1 adk gene product [Mycobacterium bovis BCG str. Mexico]
121636655           nr   YP_976878.1 adenylate kinase [Mycobacterium bovis BCG str. Pasteur 1173P2]
224989127           nr   YP_002643814.1 adenylate kinase [Mycobacterium bovis BCG str. Tokyo 172]
118473144           nr   YP_885866.1 adk gene product [Mycobacterium smegmatis str. MC2 155]
148660508           nr   YP_001282031.1 adenylate kinase [Mycobacterium tuberculosis H37Ra]
15607873            nr   NP_215247.1 adk gene product [Mycobacterium tuberculosis H37Rv]
183981091           nr   YP_001849382.1 adenylate kinase [Mycobacterium marinum M]
-------------------------
number of proteins = 12

========================================
>Family_325  -- Gblocks 262/296 (88%)
99905939            nr   shikimate 5-dehydrogenase [Mycobacterium 1407]
339632580           nr   YP_004724222.1 aroE gene product [Mycobacterium africanum GM041182]
229220809           nr   YP_882610.2 shikimate 5-dehydrogenase [Mycobacterium avium 104]
41407178            nr   NP_960014.1 shikimate 5-dehydrogenase [Mycobacterium avium subsp. paratuberculosis K-10]
161511531           nr   NP_856228.2 shikimate 5-dehydrogenase [Mycobacterium bovis AF2122/97]
378772292           nr   YP_005172025.1 aroE gene product [Mycobacterium bovis BCG str. Mexico]
229576791           nr   YP_978661.2 shikimate 5-dehydrogenase [Mycobacterium bovis BCG str. Pasteur 1173P2]
224990931           nr   YP_002645618.1 shikimate 5-dehydrogenase [Mycobacterium bovis BCG str. Tokyo 172]
118473062           nr   YP_887345.1 aroE gene product [Mycobacterium smegmatis str. MC2 155]
148662391           nr   YP_001283914.1 shikimate 5-dehydrogenase [Mycobacterium tuberculosis H37Ra]
15609689            nr   NP_217068.1 aroE gene product [Mycobacterium tuberculosis H37Rv]
183982186           nr   YP_001850477.1 shikimate 5-dehydrogenase [Mycobacterium marinum M]
-------------------------
number of proteins = 12
...
```

Figura 6.11: Parte do resultado da busca por famílias contendo exatamente uma proteína de cada genoma e no máximo uma proteína do *outgroup* (*M. marinum* M) para o complexo *Mycobacterium*.



```
Singletons from genome Mycobacterium africanum GM041182.
339630353           nr   YP_004721995.1 hypothetical protein [Mycobacterium africanum GM041182]
339630671           nr   YP_004722313.1 hypothetical protein [Mycobacterium africanum GM041182]
339630815           nr   YP_004722457.1 PE_PGRS10 gene product [Mycobacterium africanum GM041182]
339631103           nr   YP_004722745.1 hypothetical protein [Mycobacterium africanum GM041182]
339631215           nr   YP_004722857.1 transposase [Mycobacterium africanum GM041182]
339631236           nr   YP_004722878.1 hypothetical protein [Mycobacterium africanum GM041182]
339631568           nr   YP_004723210.1 hypothetical protein [Mycobacterium africanum GM041182]
339631643           nr   YP_004723285.1 hypothetical protein [Mycobacterium africanum GM041182]
339631717           nr   YP_004723359.1 transposase [Mycobacterium africanum GM041182]
339631817           nr   YP_004723459.1 transposase [Mycobacterium africanum GM041182]
339631818           nr   YP_004723460.1 transposase [Mycobacterium africanum GM041182]
339632134           nr   YP_004723776.1 PE21 gene product, partial [Mycobacterium africanum GM041182]
339632372           nr   YP_004724014.1 esxP1 gene product [Mycobacterium africanum GM041182]
339632697           nr   YP_004724339.1 arsB1 gene product, partial [Mycobacterium africanum GM041182]
339632825           nr   YP_004724467.1 transposase [Mycobacterium africanum GM041182]
339632826           nr   YP_004724468.1 transposase [Mycobacterium africanum GM041182]
339633135           nr   YP_004724777.1 hypothetical protein [Mycobacterium africanum GM041182]
339633333           nr   YP_004724975.1 hypothetical protein [Mycobacterium africanum GM041182]
339633433           nr   YP_004725075.1 hypothetical protein [Mycobacterium africanum GM041182]
339633594           nr   YP_004725236.1 PE_PGRS59 gene product [Mycobacterium africanum GM041182]
339633741           nr   YP_004725383.1 hypothetical protein [Mycobacterium africanum GM041182]
339633742           nr   YP_004725384.1 hypothetical protein [Mycobacterium africanum GM041182]
339633743           nr   YP_004725385.1 hypothetical protein [Mycobacterium africanum GM041182]
339633744           nr   YP_004725386.1 hypothetical protein [Mycobacterium africanum GM041182]
-------------------------
number of singletons = 24
```

Figura 6.12: Lista das sequências proteicas que não pertencem a nenhuma família de proteínas para o genoma '*M. africanum* GM041182'.

Cada família foi alinhada usando MUSCLE [26] e, por meio do Gblocks [22], foram eliminadas as posições mal alinhadas e regiões divergentes dos alinhamentos. Os alinhamentos foram concatenados resultando em 543,689 colunas. A partir do

alinhamento produzido, o RAxML [78] permitiu obter uma árvore filogenética de máxima verossimilhança.

De forma análoga ao B. *cereus*, observando a Figura 6.11:

- Ao acessar o link 'Family X' o arquivo no formato fasta da respectiva família é mostrado. Um exemplo deste tipo de arquivo pode ser visto na Figura 6.13.
- Ao acessar o link 'Gblocks' o alinhamento gerado pelo Gblocks da respectiva família é mostrado. A Figura 6.14 mostra um exemplo deste tipo de arquivo.
- Ao acessar o link com o identificador do gene (*gi*) é mostrado o resultado da busca no NCBI sobre o gene. A Figura 6.15 mostra um exemplo deste resultado.
- Ao acessar o link 'nr' é mostrado o resultado de um Blastp *online* do gene contra a base *nr* do GenBank. A Figura 6.16 mostra um exemplo deste resultado.

```
>121639456 YP_979680.1 3-ketosteroid-delta-1-dehydrogenase [Mycobacterium bovis BCG str. Pasteur 1173P2]
MTVQEFDVVVVGSGAAGMVAALVAAHRGLSTVVVEKAPHYGGSTARSGGGVWIPNNEVLK
RRGVRDTPEAARTYLHGIVGEIVEPERIDAYLDRGPEMLSFVLKHTPLKMCWVPGYSDYY
PEAPGGRPGGRSIEPKPFNARKLGADMAGLEPAYGKVPLNVVVMQQDYVRLNQLKRHPRG
VLRSMKVGARTMWAKATGKNLVGMGRALIGPLRIGLQRAGVPVELNTAFTDLFVENGVVS
GVYVRDSHEAESAEPQLIRARRGVILACGGFEHNEQMRIKYQRAPITTEWTVGASANTGD
GILAAEKLGAALDLMDDAWWGPTVPLVGKPWFALSERNSPGSIIVNMSGKRFMNESMPYV
EACHHMYGGEHGQGPGPGENIPAWLVFDQRYRDRYIFAGLQPGQRIPSRWLDSGVIVQAD
TLAELAGKAGLPADELTATVQRFNAFARSGVDEDYHRGESAYDRYYGDPSNKPNPNLGEV
GHPPYYGAKMVPGDLGTKGGIRTDVNGRALRDDGSIIDGLYAAGNVSAPVMGHTYPGPGG
TIGPAMTFGYLAALHIADQAGKR
>148663400 YP_001284923.1 3-ketosteroid-delta-1-dehydrogenase [Mycobacterium tuberculosis H37Ra]
MTVQEFDVVVVGSGAAGMVAALVAAHRGLSTVVVEKAPHYGGSTARSGGGVWIPNNEVLK
RRGVRDTPEAARTYLHGIVGEIVEPERIDAYLDRGPEMLSFVLKHTPLKMCWVPGYSDYY
PEAPGGRPGGRSIEPKPFNARKLGADMAGLEPAYGKVPLNVVVMQQDYVRLNQLKRHPRG
VLRSMKVGARTMWAKATGKNLVGMGRALIGPLRIGLQRAGVPVELNTAFTDLFVENGVVS
GVYVRDSHEAESAEPQLIRARRGVILACGGFEHNEQMRIKYQRAPITTEWTVGASANTGD
GILAAEKLGAALDLMDDAWWGPTVPLVGKPWFALSERNSPGSIIVNMSGKRFMNESMPYV
EACHHMYGGEHGQGPGPGENIPAWLVFDQRYRDRYIFAGLQPGQRIPSRWLDSGVIVQAD
TLAELAGKAGLPADELTATVQRFNAFARSGVDEDYHRGESAYDRYYGDPSNKPNPNLGEV
GHPPYYGAKMVPGDLGTKGGIRTDVNGRALRDDGSIIDGLYAAGNVSAPVMGHTYPGPGG
TIGPAMTFGYLAALHIADQAGKR
>15610673 NP_218054.1 3-ketosteroid-delta-1-dehydrogenase [Mycobacterium tuberculosis H37Rv]
MTVQEFDVVVVGSGAAGMVAALVAAHRGLSTVVVEKAPHYGGSTARSGGGVWIPNNEVLK
RRGVRDTPEAARTYLHGIVGEIVEPERIDAYLDRGPEMLSFVLKHTPLKMCWVPGYSDYY
PEAPGGRPGGRSIEPKPFNARKLGADMAGLEPAYGKVPLNVVVMQQDYVRLNQLKRHPRG
VLRSMKVGARTMWAKATGKNLVGMGRALIGPLRIGLQRAGVPVELNTAFTDLFVENGVVS
GVYVRDSHEAESAEPQLIRARRGVILACGGFEHNEQMRIKYQRAPITTEWTVGASANTGD
GILAAEKLGAALDLMDDAWWGPTVPLVGKPWFALSERNSPGSIIVNMSGKRFMNESMPYV
EACHHMYGGEHGQGPGPGENIPAWLVFDQRYRDRYIFAGLQPGQRIPSRWLDSGVIVQAD
TLAELAGKAGLPADELTATVQRFNAFARSGVDEDYHRGESAYDRYYGDPSNKPNPNLGEV
GHPPYYGAKMVPGDLGTKGGIRTDVNGRALRDDGSIIDGLYAAGNVSAPVMGHTYPGPGG
TIGPAMTFGYLAALHIADQAGKR
>161579551 YP_879904.2 3-ketosteroid-delta-1-dehydrogenase [Mycobacterium avium 104]
MSAQEYDVVVVGSGGAGMVAALTAAHRGLSTIVIEKAPHFGGSTARSGGGVWIPNNEVLK
RDGVKDTPEAARTYLHGIIGDVVEPERIDTYLERGPEMLSFVLKHTPLKMCWVPRYSDYY
PESPGGRAEGRSIEPKPFNARKLGPDEAGLEPAYGKVPLNVVVMQQDYVRLNQLKRHPRG
VLRSLKVGARTMWAKATGKNLVGMGRALIGPLRIGLQRAGVPVVLNTALTDLYLEDGVVR
GVYVRDSQAAESAEPRLIRARRGVILASGGFEHNEQMRVKYQRAPITTEWTVGAKANTGD
GILAAEKLGAALELMEDAWWGPTVPLVGAPWFALSERNSPGSIIVNMSGKRFMNESMPYV
EACHHMYGGEYGQGPGPGENIPAWLVFDQQYRDRYIFAGLQPGQRIPRKWLESGVIIQAE
TLEELASKAGLPVDEFLATVQRFNGFARTGIDEDYHRGESAYDRYYGDPTNKPNPNLGEI
SHPPYYAAKMVPGDLGTKGGIRTDIHGRALRDDGSIIEGLYAAGNVSAPVMGHTYPGPGG
TIGPAMTFGYLAALHIAGEN
>161598319 YP_890167.2 3-ketosteroid-delta-1-dehydrogenase [Mycobacterium smegmatis str. MC2 155]
MTGQEYDVVVVGSGAAGMVAALTAAHQGLSTVVVEKAPHYGGSTARSGGGVWIPNNEILK
RDGVKDTPDEARKYLHAIIGDVVPAEKIDTYLDRGPEMLSFVLKHSPLKLCWVPGYSDYY
PETPGGKPTGRSVEPKPFDANKLGPDLKGLEPPYGKVPMNMVVMQQDYVRLNQLKRHPRG
VLRSLKVGIRATWGKVSGKNLVGMGRALIAPLRIGLREAGVPVLLNTALTDLYVEDGRVL
GIYVRDTTAGDDAEPRLIRARHGVILGSGGFEHNEQMRVKYQRAPITTEWTVGAVANTGD
GIVAAEKLGAALELMEDAWWGPTVPLVGAPWFALSERNSPGSIIVNMSGKRFMNESMPYV
EACHHMYGGQYGQGPGPGENIPAWLIFDQQYRDRYIFAGLQPGQRIPSKWLESGVIVKAD
TLVELAEKTGLPADQFTSTIERFNGFARSGVDEDFHRGESAYDRYYGDPTNKPNPNLGEI
KHGPFYAAKMVPGDLGTKGGIRTDVRGRALRDDDSVIEGLYAAGNVSSPVMGHTYPGPGG
TIGPAMTFGYLAALDIAATAAASRKG
```

...

Figura 6.13: Parte do arquivo no formato fasta contendo as proteínas agrupadas na família 317 para o complexo *Mycobacterium*.

```
Gblocks 0.91b Results

Processed file: family2347.afa
Number of sequences: 11
Alignment assumed to be: Protein
New number of positions: 129 (selected positions are underlined in blue)

                    10        20        30        40        50        60
               =========+=========+=========+=========+=========+=========+
118463706      MSPGACELDGGMQLPQGLARFNRHVTNPIQRMWAGWLPPFGIVEHVGRRSGKPYRTPVNL
41410116       MSPGACELDGGMQLPQGLARFNRHVTNPIQRMWAGWLPPFGIVEHVGRRSGKPYRTPVNL
121636441      -----------MQLPQWLARFNRYVTNPIQRLWAGWLPAFAILEHVGRRSGKPYRTPLNV
148660292      -----------MQLPQWLARFNRYVTNPIQRLWAGWLPAFAILEHVGRRSGKPYRTPLNV
15607663       -----------MQLPQWLARFNRYVTNPIQRLWAGWLPAFAILEHVGRRSGKPYRTPLNV
224988913      -----------MQLPQWLARFNRYVTNPIQRLWAGWLPAFAILEHVGRRSGKPYRTPLNV
31791705       -----------MQLPQWLARFNRYVTNPIQRLWAGWLPAFAILEHVGRRSGKPYRTPLNV
339630595      -----------MQLPQWLARFNRYVTNPIQRLWAGWLPAFAILEHVGRRSGKPYRTPLNV
378770275      -----------MQLPQWLARFNRYVTNPIQRLWAGWLPAFAILEHVGRRSGKPYRTPLNV
118468773      -----------MQLPQWLARFNRHVTNPIQRIWAGWAPTFGILEHVGRKSGKHYRTPLSV
99903902       -----------MQLPQSLARFNRHVTNPIQRLWAGWAPTFGILEHVGRKSGTTYRTPLSV
                          ################################################

                    70        80        90       100       110       120
               =========+=========+=========+=========+=========+=========+
118463706      FSADVDGKPGVAILLTYGPDRDWLKNLTAAGGGRIRSRGKTFGVADPRVVSKAEAASHLR
41410116       FSADVDGKPGVAILLTYGPDRDWLKNLTAAGGGRIRSRGKTFGVADPRVVSKAEAASHLR
121636441      FSADVDGRAGVAILLTYGPNRDWLKNITAAGGGRMRRYGKTFGVANPRRLTKAEAAPYVS
148660292      FSADVDGRAGVAILLTYGPNRDWLKNITAAGGGRMRRYGKTFGVANPRRLTKAEAAPYVS
15607663       FSADVDGRAGVAILLTYGPNRDWLKNITAAGGGRMRRYGKTFGVANPRRLTKAEAAPYVS
224988913      FSADVDGRAGVAILLTYGPNRDWLKNITAAGGGRMRRYGKTFGVANPRRLTKAEAAPYVS
31791705       FSADVDGRAGVAILLTYGPNRDWLKNITAAGGGRMRRYGKTFGVANPRRLTKAEAAPYVS
339630595      FSADVDGRAGVAILLTYGPNRDWLKNITAAGGGRMRRYGKTFGVANPRRLTKAEAAPYVS
378770275      FSADVDGRAGVAILLTYGPNRDWLKNITAAGGGRMRRYGKTFGVANPRRLTKAEAAPYVS
118468773      FGTD----EGVAILLTYGPDRDWLKNLTAAGTADMRRHGKTIRVTDPRVVPRDEAVARVR
99903902       FSTD----DGVAIMLTYGPDRDWLKNITSAGNAKIRRHGKTIEVRDPRVVSKAEAAEHVT
               ############################################################

                   130       140
               =========+=========+==
118463706      PGVRGVFARLPFEQAVLLTRTP
41410116       PGVRGVFARLPFEQAVLLTRTP
121636441      SRWRPVFARLPFDEAVLLTKAD
148660292      SRWRPVFARLPFDEAVLLTKAD
15607663       SRWRPVFARLPFDEAVLLTKAD
224988913      SRWRPVFARLPFDEAVLLTKAD
31791705       SRWRPVFARLPFDEAVLLTKAD
339630595      SRWRPVFARLPFDEAVLLTKAD
378770275      SRWRPVFARLPFDEAVLLTKAD
118468773      GPIRKVLARLPFEQAVLLRRV-
99903902       GAMRRPFARLPFEQAVLLKKV-
               ####################

Parameters used
Minimum Number Of Sequences For A Conserved Position: 6
Minimum Number Of Sequences For A Flanking Position: 9
Maximum Number Of Contiguous Nonconserved Positions: 8
Minimum Length Of A Block: 10
Allowed Gap Positions: With Half
Use Similarity Matrices: Yes
Flank positions of the 1 selected block(s)
Flanks: [12  140]
New number of positions:  129  (90% of the original 142 positions)
```

Figura 6.14: Saída do Gblocks para a família de 11 proteínas, de *Mycobacterium*. As colunas marcadas com '#' são as mantidas para a construção da filogenia.

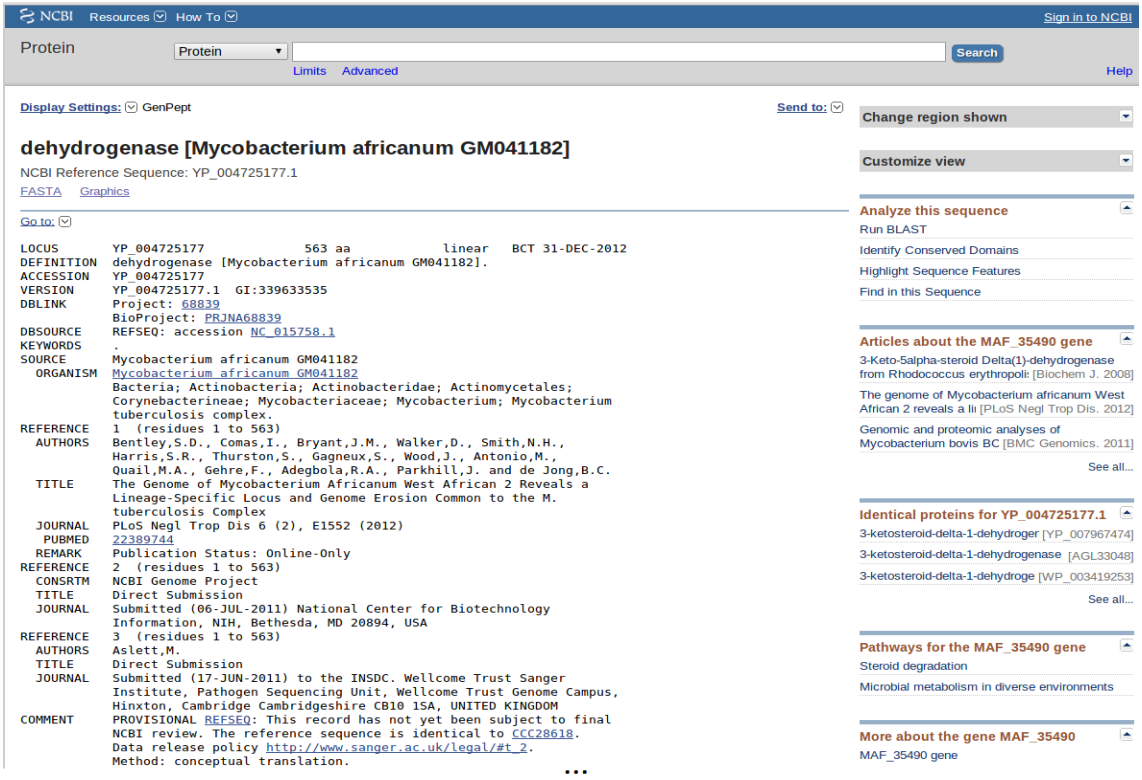
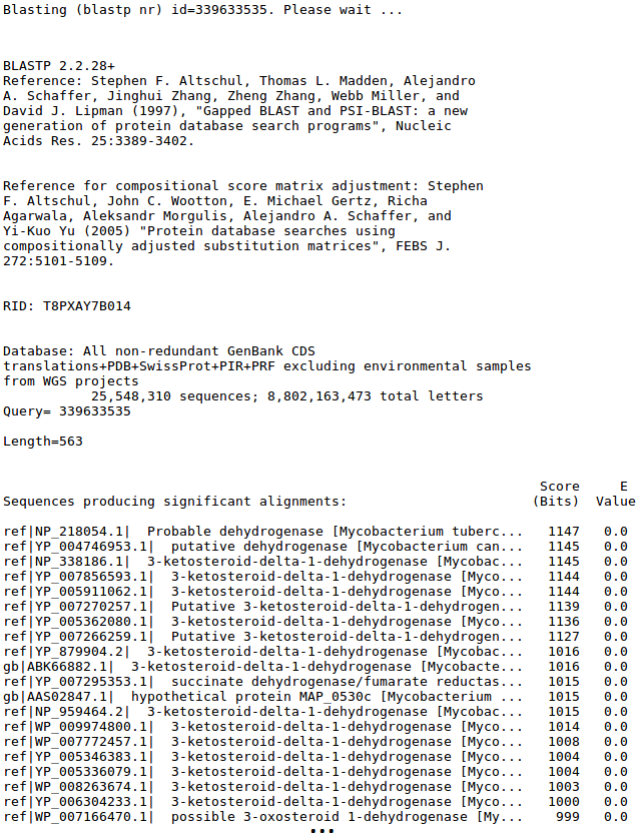


Figura 6.15: Parte do resultado da busca no NCBI da proteína com identificador do gene (*gi*) igual a '339633535'.



```
Blasting (blastp nr) id=339633535. Please wait ...


BLASTP 2.2.28+
Reference: Stephen F. Altschul, Thomas L. Madden, Alejandro
A. Schaffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and
David J. Lipman (1997), "Gapped BLAST and PSI-BLAST: a new
generation of protein database search programs", Nucleic
Acids Res. 25:3389-3402.


Reference for compositional score matrix adjustment: Stephen
F. Altschul, John C. Wootton, E. Michael Gertz, Richa
Agarwala, Aleksandr Morgulis, Alejandro A. Schaffer, and
Yi-Kuo Yu (2005) "Protein database searches using
compositionally adjusted substitution matrices", FEBS J.
272:5101-5109.


RID: T8PXAY7B014


Database: All non-redundant GenBank CDS
translations+PDB+SwissProt+PIR+PRF excluding environmental samples
from WGS projects
           25,548,310 sequences; 8,802,163,473 total letters
Query= 339633535

Length=563


                                                                    Score     E
Sequences producing significant alignments:                        (Bits)  Value

ref|NP_218054.1|  Probable dehydrogenase [Mycobacterium tuberc...   1147   0.0
ref|YP_004746953.1|  putative dehydrogenase [Mycobacterium can...   1145   0.0
ref|NP_338186.1|  3-ketosteroid-delta-1-dehydrogenase [Mycobac...   1145   0.0
ref|YP_007856593.1|  3-ketosteroid-delta-1-dehydrogenase [Myco...   1144   0.0
ref|YP_005911062.1|  3-ketosteroid-delta-1-dehydrogenase [Myco...   1144   0.0
ref|YP_007270257.1|  Putative 3-ketosteroid-delta-1-dehydrogen...   1139   0.0
ref|YP_005362080.1|  3-ketosteroid-delta-1-dehydrogenase [Myco...   1136   0.0
ref|YP_007266259.1|  Putative 3-ketosteroid-delta-1-dehydrogen...   1127   0.0
ref|YP_879904.2|  3-ketosteroid-delta-1-dehydrogenase [Mycobac...   1016   0.0
gb|ABK66882.1|  3-ketosteroid-delta-1-dehydrogenase [Mycobacte...   1016   0.0
ref|YP_007295353.1|  succinate dehydrogenase/fumarate reductas...   1015   0.0
gb|AAS02847.1|  hypothetical protein MAP_0530c [Mycobacterium ...   1015   0.0
ref|NP_959464.2|  3-ketosteroid-delta-1-dehydrogenase [Mycobac...   1015   0.0
ref|WP_009974800.1|  3-ketosteroid-delta-1-dehydrogenase [Myco...   1014   0.0
ref|WP_007772457.1|  3-ketosteroid-delta-1-dehydrogenase [Myco...   1008   0.0
ref|YP_005346383.1|  3-ketosteroid-delta-1-dehydrogenase [Myco...   1004   0.0
ref|YP_005336079.1|  3-ketosteroid-delta-1-dehydrogenase [Myco...   1004   0.0
ref|WP_008263674.1|  3-ketosteroid-delta-1-dehydrogenase [Myco...   1003   0.0
ref|YP_006304233.1|  3-ketosteroid-delta-1-dehydrogenase [Myco...   1000   0.0
ref|WP_007166470.1|  possible 3-oxosteroid 1-dehydrogenase [My...    999   0.0
                                    ...
```

Figura 6.16: Parte do resultado de um Blastp *online* da proteína com identificador do gene (*gi*) igual a '339633535' contra a base *nr* do GenBank.

A árvore fornecida pela abordagem computacional para o complexo *Mycobacterium* a partir de comparações de genomas é apresentada na Figura 6.17.

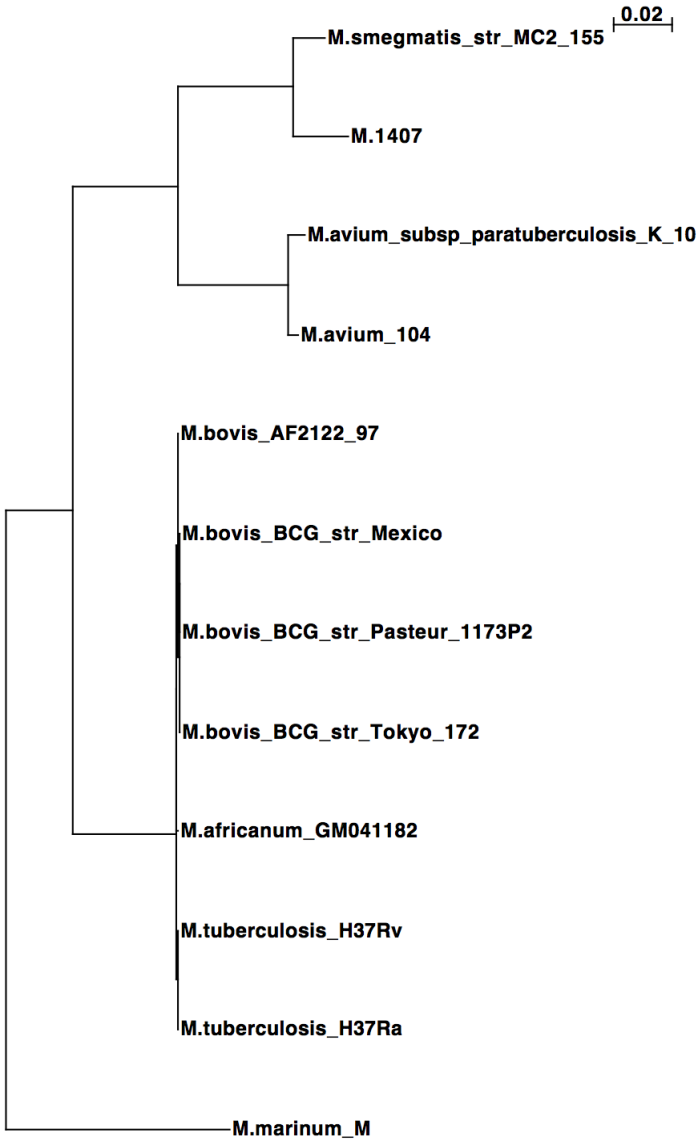


Figura 6.17: Árvore obtida segundo o *pipeline* descrito na Seção 5.3. O genoma de *M. marinum* M foi utilizado como externo ao complexo *Mycobacterium*.

Para a tipagem de cepas do complexo *Mycobacterium tuberculosis*, dois métodos foram avaliados e comparados à nossa abordagem computacional: de espoligotipagem e MIRU-VNTR.

Para construir árvores filogenéticas baseadas em dados de espoligotipagem e MIRU-VNTR, tais padrões foram obtidos de registros disponíveis na literatura [14, 70, 48, 17, 53, 62, 81, 44], para dois conjuntos de cepas S_1 e S_2, como mostrado na Tabela 6.4, tais que S_1 é formado pelas 10 cepas e S_2, por sua vez, é formado pelas 6 primeiras cepas listadas.

Tabela 6.4: Conjunto de cepas do complexo *Mycobacterium* utilizado para construir árvores filogenéticas baseadas em dados de espoligotipagem e MIRU-VNTR.

Cepas	Conjunto(s) a que pertence
M. tuberculosis H37Ra	S_1, S_2
M. tuberculosis H37Rv	S_1, S_2
M. tuberculosis KZN 1435	S_1, S_2
M. tuberculosis KZN 605	S_1, S_2
M. tuberculosis F11	S_1, S_2
M. bovis BCG Pasteur 1173 P2	S_1, S_2
M. tuberculosis KZN 4207	S_1
M. tuberculosis CDC 1551	S_1
M. bovis AF2122 97	S_1
M. bovis BCG Tokyo 172	S_1

A divisão das cepas desse complexo em dois conjuntos foi feita em função da disponibilidade na literatura de dados de espoligotipagem e MIRU-VNTR. Para S_1 apenas dados de espoligotipagem estão disponíveis, enquanto que para S_2 tanto dados de espoligotipagem quanto dados de MIRU-VNTR estão disponíveis.

Os bancos de dados usados para confirmar padrões, linhagens, nomes de cepas e país de isolamento foram o SITVIT2 (Intitute of Guadalupe), TB Database [65], SpolTools [87] e MIRU-VNTR*plus* [98]. Foram criados arquivos MS Excel com as seguintes combinações de dados e cepas: espoligotipagem e MIRU-VNTR (S_2), apenas espoligotipagem (S_1 e S_2) e apenas MIRU-VNTR (S_2). Cada um destes quatro arquivos foi carregado no banco de dados MIRU-VNTR*plus*. As ferramentas do banco MIRU-VNTR*plus* foram utilizadas para calcular uma árvore filogenética pelo método de *Neighbor Joining* (NJ) [61] a partir dos dados do arquivo MS Excel. Além disso, tal método calcula estimativas dos comprimentos dos ramos, os quais são proporcionais ao número de mudanças ocorridas entre duas unidades taxonômicas, e considera ótima a árvore que tiver a menor soma dos comprimentos dos ramos [61].

Assim, as comparações a serem feitas aqui são as seguintes:

- para o conjunto S_1:
  - árvore espoligotipagem *vs.* árvore *Orthologsorter*
- para o conjunto S_2:
  - árvore espoligotipagem *vs.* árvore *Orthologsorter*

- árvore MIRU-VNTR *vs.* árvore *Orthologsorter*
- árvore espoligotipagem combinado com MIRU-VNTR *vs.* árvore *Orthologsorter*

As árvores mostradas nas Figura 6.18 e 6.19 apresentam os resultados de espoligotipagem e *Orthologsorter*, respectivamente, considerando o conjunto S_1, formado pelas 10 cepas do complexo *M. tuberculosis*. Ressalta-se que no caso de *Orthologsorter*, é necessária a inclusão de um genoma externo ao grupo, que neste caso foi o *M. canettii*.

Nota-se uma pequena diferença na topologia da árvore de espoligotipagem e a subárvore de *Orthologsorter*, excluindo o genoma externo, diferença essa relacionada à posição relativa de *M. tuberculosis* F11. Entretanto, como observamos na Figura 6.20, todas as outras cepas foram agrupadas perfeitamente.

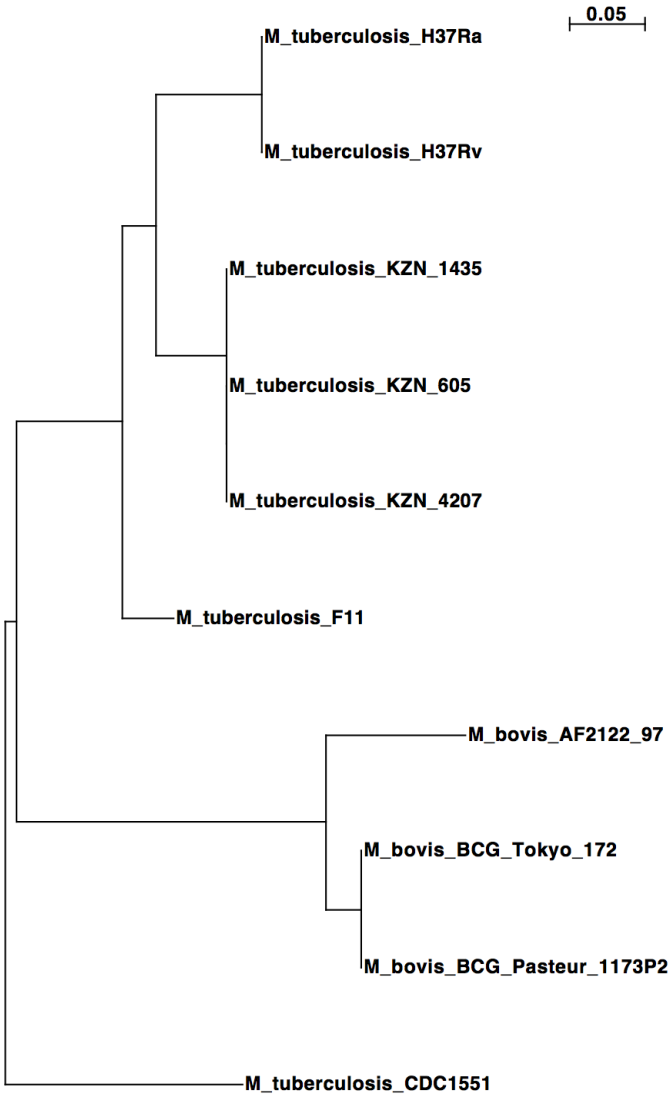


Figura 6.18: Árvore resultante de espoligotipagem para o conjunto S_1.

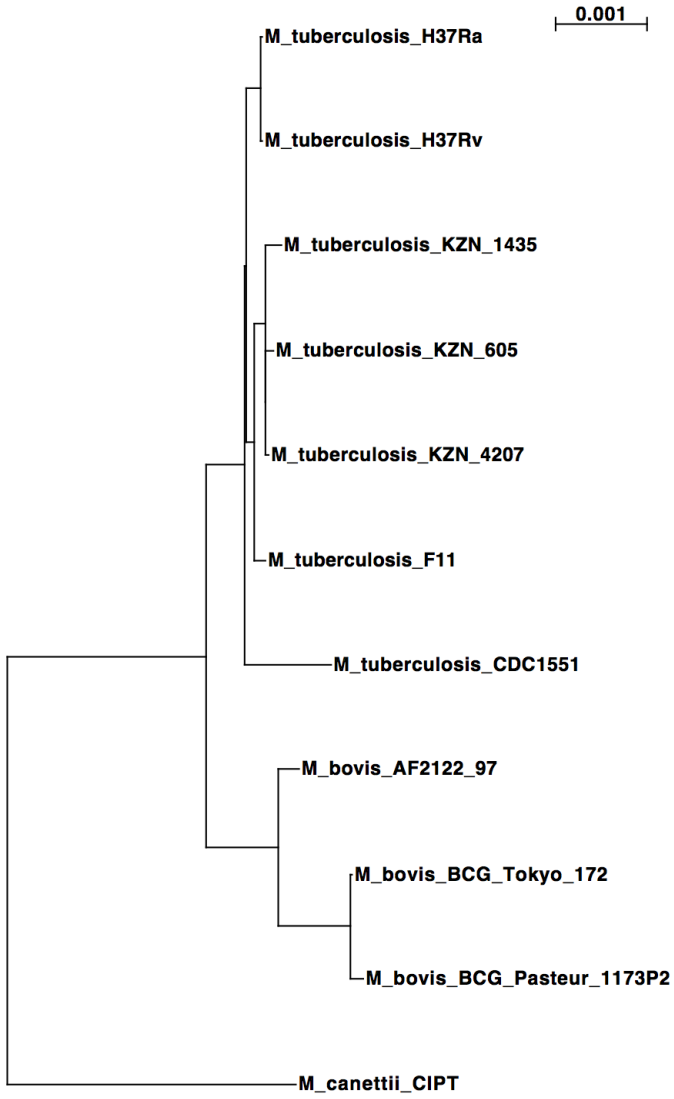


Figura 6.19: Árvore resultante de *Orthologsorter* para o conjunto S_1.

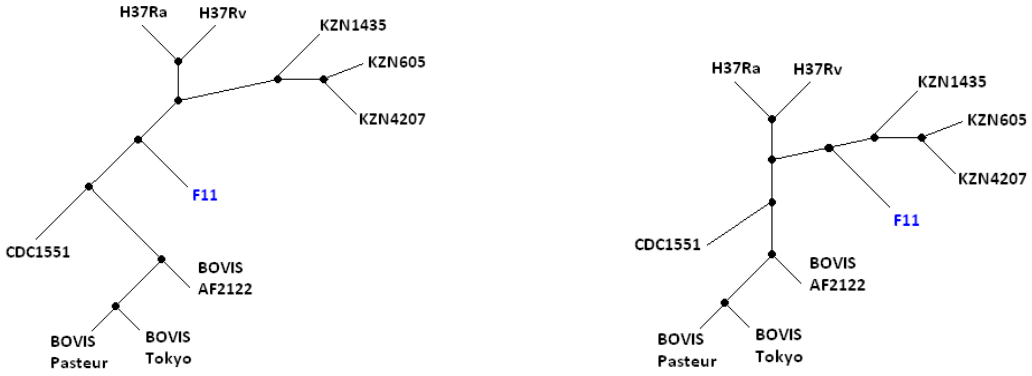


Figura 6.20: Árvore baseada em dados de espoligotipagem para o conjunto S_1(esquerda). Árvore construída pelo *Orthologsorter* para o conjunto S_1(direita).

Para o conjunto S_2, as árvores mostradas nas Figuras 6.21, 6.22 e 6.23 apresentam os resultados de espoligotipagem, MIRU-VNTR e uma combinação das duas técnicas, respectivamente. Todas essas árvores devem ser comparadas à sub-árvore da árvore *Orthologsorter*, mostrada na Figura 6.24, removendo o *outgroup*, *M. canettii*.

Nota-se total congruência entre as topologias em todas as comparações, mostrando que a metodologia computacional aqui proposta pode ser aplicada perfeitamente à determinação da tipagem das cepas consideradas.

Outra observação importante é que por vezes espoligotipagem e/ou MIRU-VNTR não são capazes de diferenciar duas cepas. Como exemplo, vê-se que as cepas *M. tuberculosis* H37Rv e *M. tuberculosis* H37Ra possuem exatamente o mesmo padrão de espoligotipagem e MIRU, e portanto, não são discriminadas por tais técnicas, como demonstram todas árvores construídas a partir de tais dados (Figuras 6.18 e 6.22). As cepas *M. bovis* BCG Tokyo 172 e *M. bovis* BCG Pasteur 1173P2 também possuem o mesmo padrão de espoligotipagem, logo, não são diferenciadas nas árvores construídas com dados de espoligotipagem (Figura 6.18). Situação similar ocorre para as cepas *M. tuberculosis* KZN 1435, 605 e 4207 (Figura 6.18).

Espera-se, a partir de uma análise mais criteriosa (como a análise feita no [99]) das famílias encontradas por *Orthologsorter*, que seja possível obter tal discriminação, uma vez que a entrada para o *Orthologsorter* contém todas as proteínas preditas em cada genoma. Mais ainda, as famílias usadas pelo *Orthologsorter* na construção da árvore são tais que apenas genes representativos estão presentes. Dessa forma, a abordagem computacional torna-se promissora na discriminação dos pares de cepas

indistinguíveis para espoligotipagem e/ou MIRU, reforçando assim a efetividade da abordagem computacional.

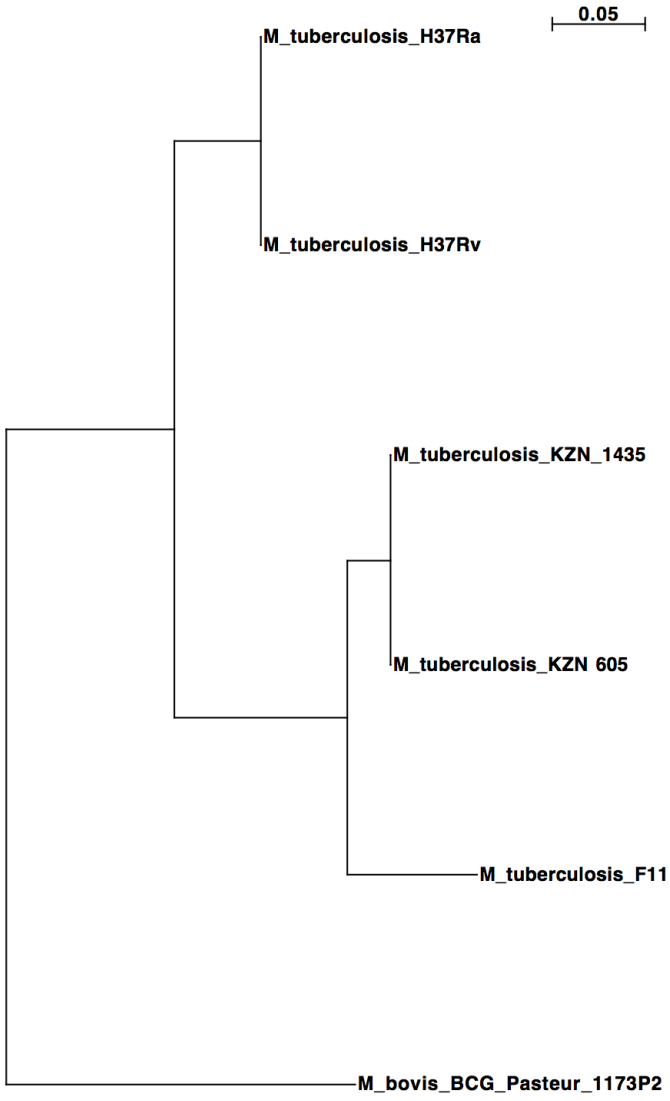


Figura 6.21: Árvore resultante de espoligotipagem para o conjunto S_2.

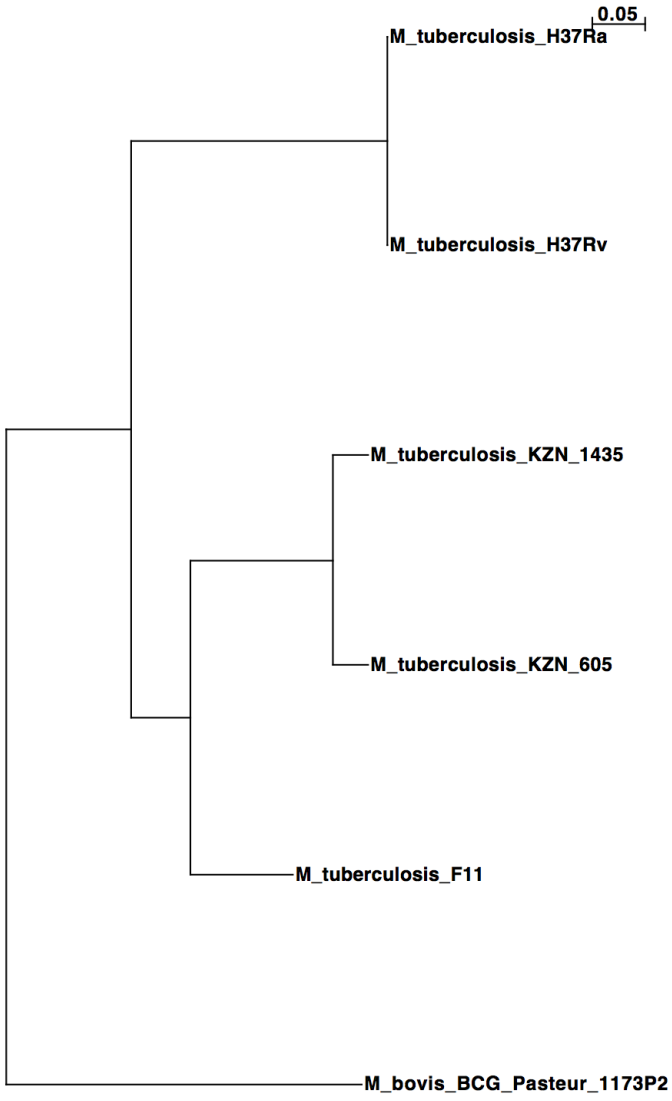


Figura 6.22: Árvore resultante de MIRU-VNTR para o conjunto S_2.

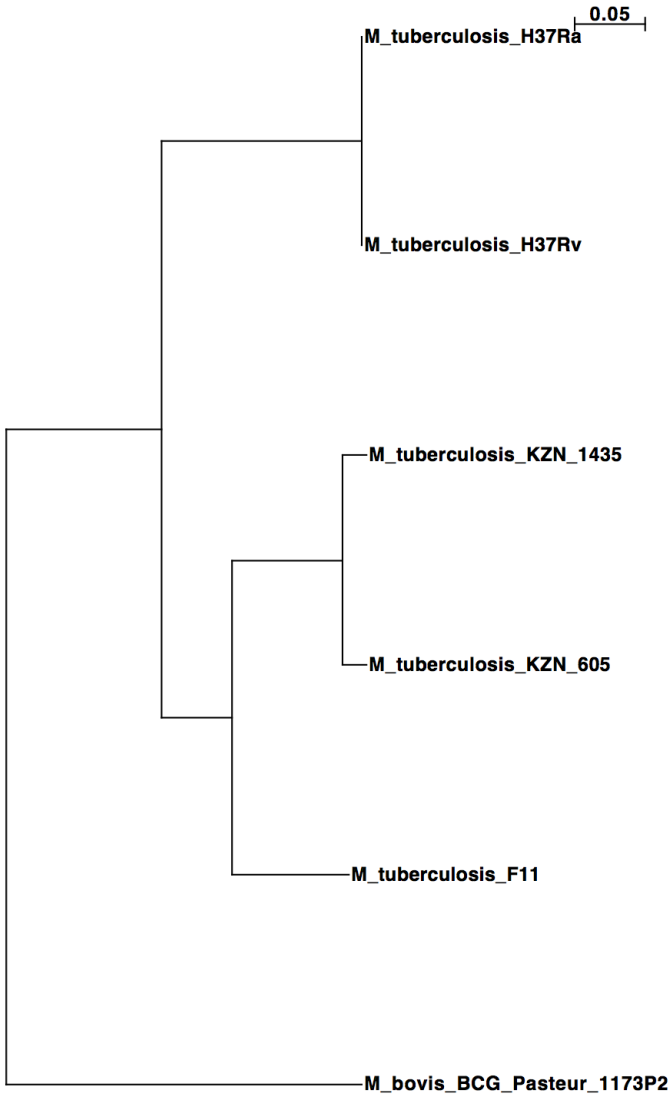


Figura 6.23: Árvore resultante da combinação de Espoligotipagem e VNTR para o conjunto S_2.

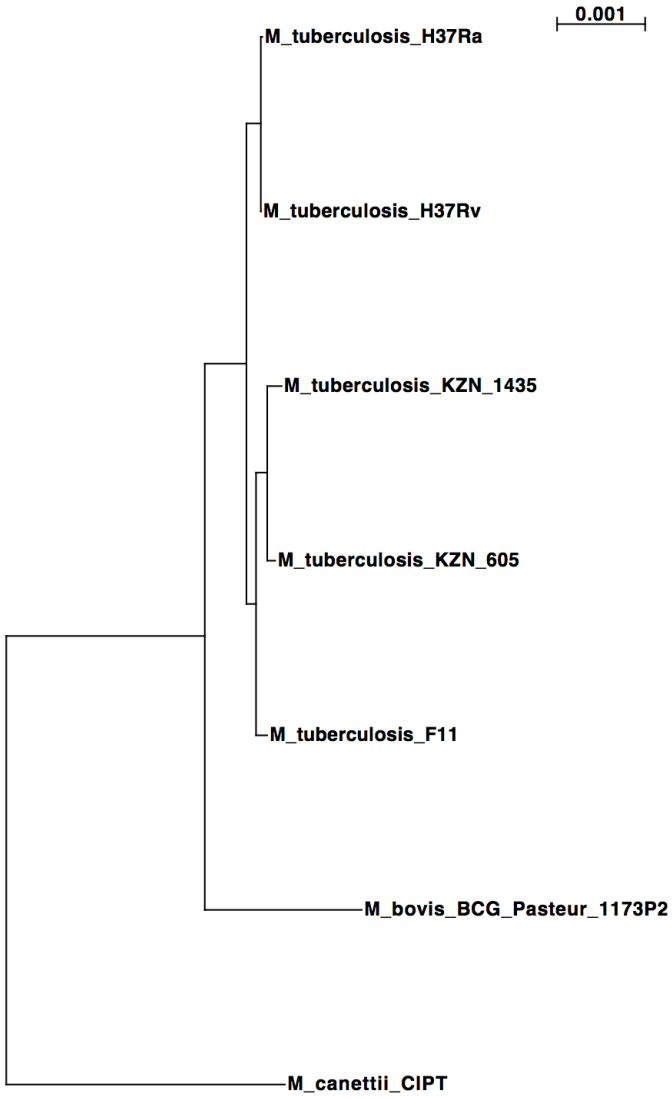


Figura 6.24: Árvore resultante de *Orthologsorter* para o conjunto S_2.

6.3 Tempos de execução

A configuração da máquina utilizada foi: Intel Xeon Processador E5-2620 2.00GHz 64-bit com 6 cores, 24 threads, 64 GB de memória RAM e 15 MB de memória cache. Os programas desenvolvidos foram escritos na linguagem *Perl*.

Na Tabela 6.5 a seguir é apresentado o tempo de execução de cada ferramenta com-

putacional utilizada no *pipeline*, descritas na Seção 5.1, para o complexo bacteriano *Mycobacterium* usado nos resultados.

Tabela 6.5: Tempo de execução de cada ferramenta computacional utilizada no *pipeline* para o complexo bacteriano *Mycobacterium* usado nos resultados.

	Mycobacterium Conjunto S_2	*Mycobacterium* Conjunto S_1
	7 genomas	11 genomas
	28011 genes	44058 genes
Blastp	3h07m06s	7h24m06s
OrthoMCL	20m27s	40m31s
Orthologsorter	0.154654s	0.194923s
MUSCLE	19m11s	21m30s
Gblocks	9s	10s
RAxML	5m44s	9m50s
NJplot	0.00202s	0.00211s
total de tempo de execução	3h52m37s	8h36m07s

Capítulo 7

Conclusão

Novas tecnologias de sequenciamento a baixo custo têm permitido o sequenciamento completo do DNA de um número cada vez maior de organismos. Este trabalho propôs a utilização de uma abordagem estritamente computacional para a tipagem de espécies de bactérias, para as quais se tem as sequências de proteínas preditas nos genomas correspondentes.

Dados os conjuntos de proteínas preditas das espécies de interesse, a abordagem consiste, de maneira simplificada, na execução das seguintes etapas:

1. comparação de todas essas proteínas preditas ("todas-contra-todas");
2. agrupamento das proteínas em famílias;
3. escolha de famílias predominantemente formadas por genes ortólogos;
4. alinhamento de cada uma das famílias escolhidas;
5. concatenação de todos os alinhamentos; e
6. construção de árvore filogenética das espécies de interesse.

A principal característica da abordagem computacional é o fato de que, durante sua execução, é criada uma ferramenta *web*, denominada *Orthologsorter*, onde é possível realizar diversas buscas nas famílias de proteínas a fim de explicar os relacionamentos entre as espécies de interesse.

Nossa abordagem apresenta como resultado não apenas uma ferramenta *web*, mas também uma árvore filogenética gerada a partir de uma busca específica através do *Orthologsorter*. O objetivo é mostrar, por meio da árvore filogenética obtida, as relações de proximidade entre as espécies. Dessa forma é possível, para bactérias que já foram sequenciadas e anotadas, auxiliar em tipagem ainda não confirmada.

Os experimentos foram feitos com o uso de dois grupos de genomas. O primeiro contém vários genomas do grupo *Bacillus cereus*, enquanto que o segundo contém genomas do grupo *Mycobacterium*. Para testar a efetividade da abordagem, comparou-se os resultados obtidos, nominalmente as árvores obtidas para cada conjunto de genomas de entrada, com as respectivas árvores obtidas por técnicas utilizadas nesses mesmo conjuntos. Como foi visto no Capítulo 6, o método apresentou resultados equivalentes às técnicas conhecidas, o que mostra a sua efetividade.

A abordagem mostrou-se útil não somente para inferir relações filogenéticas entre espécies proximamente relacionadas por meio da comparação direta de proteínas, solucionando restrições das técnicas de genotipagem comentadas, mas também para inferir funcionalidades a partir de famílias de proteínas ortólogas.

Com relação aos custos, a abordagem é útil e vantajosa mesmo quando não se dispõe das sequências proteicas preditas em algum banco de dados, pois atualmente sequenciamentos de genomas completos custam bem menos que os equipamentos e materiais necessários às técnicas moleculares tradicionais. Além disso, as tecnologias de sequenciamentos seguem a tendência de automatização de processos, o que implica menor esforço laborial para a obtenção de sequências biológicas.

Resultados preliminares deste trabalho foram publicados em [55, 58, 8], em 2012.

Trabalhos futuros

O *pipeline* proposto neste trabalho foi implementado de maneira que a busca específica feita para a construção da árvore filogenética selecione as famílias contendo exatamente uma proteína de cada genoma e no máximo uma proteína do organismo externo ao grupo. Algumas vezes, encontrar famílias de proteínas para o conjunto de organismos de interesse é uma tarefa difícil. Com relação a trabalhos futuros, uma

possibilidade consiste no caso dessa abordagem não conseguir recuperar famílias com essas características, o *pipeline* então relaxa gradativamente essas restrições, permitindo, por exemplo, a presença de mais de um gene dos genomas, até que se consiga um número expressivo de famílias. Esse número poderá ser objeto de estudo em trabalhos futuros.

Uma outra possibilidade consiste na verificação da abordagem para grandes conjuntos de genomas filogeneticamente muito distantes. Considerando que a abordagem computacional recebe como entrada sequências proteicas, tal proposta é viável, já que proteínas tem sequências mais conservadas que o DNA ao longo do tempo evolutivo em virtude de mudanças em suas sequências serem mais seriamente julgadas pela pressão seletiva [61]. Logo, as sequências proteicas possuem muito provavelmente evidências filogenéticas suficientes para a comparação entre organismos distantes filogeneticamente.

O *pipeline* recebe como entrada um conjunto de arquivos no formato fasta contendo as proteínas preditas das espécies de interesse resultantes da anotação. Desse modo, uma possibilidade seria a inclusão da etapa de anotação ao *pipeline*.

A árvore filogenética construída é apenas um subproduto do nosso *pipeline*, uma vez que resulta de uma busca específica feita no *Orthologsorter* e apresenta apenas critérios básicos de filtragem pelo uso do Gblocks, bem menos sofisticados dos que aqueles apresentados nas publicações descritas na Seção 4.3. Como trabalho futuro complementar a este, pode-se sugerir a melhoria da árvore através de análises estatísticas.

Por fim, o presente projeto tem como objetivo disponibilizar o pacote desenvolvido no *SourceForge*.

Referências Bibliográficas

[1] D. Alland, T.S. Whittam, M.B. Murray, M.D. Cave, M.H. Hazbon, K. Dix, M. Kokoris, A. Duesterhoeft, J.A. Eisen, C.M. Fraser, and R.D. Fleishmann. Modeling bacterial evolution with comparative-genome-based marker systems: application to *Mycobacterium tuberculosis* evolution and pathogenesis. *Journal of Bacteriology*, 185(11):3392–3399, 2003.

[2] N.F. Almeida. *Ferramentas para comparação genômica*. PhD thesis, Instituto de Computação, IC-Unicamp, 2002.

[3] N.F. Almeida, S. Yan, R. Cai, C.R. Clarke, C.E. Morris, N.W. Schaad, E.L. Schuenzel, G.H. Lacy, X. Sun, J.B. Jones, J.A. Castillo, C.T. Bull, S. Leman, D. S. Guttman, J.C. Setubal, and B.A. Vinatzer. PAMDB, a multilocus sequence typing and analysis database and website for plant-associated microbes. *Phytopathology*, 100(3):208–215, 2010.

[4] N.F. Almeida, S. Yan, M. Lindeberg, D.J. Studholme, D.J. Schneider, B. Condon, H. Liu, C.J. Viana, A. Warren, C. Evans, E. Kemen, D. MacLean, A. Angot, G.B. Martin, J.D. Jones, A. Collmer, J.C. , and B.A. Vinatzer. A Draft Genome Sequence of *Pseudomonas syringae* pv. tomato T1 Reveals a Type III Effector Repertoire Significantly Divergent from that of *P. syringae* pv. tomato DC3000. *Molecular Plant-Microbe Interactions*, 22(1):52–62, Jan 2009.

[5] S.F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[6] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped Blast and Psi-Blast: a new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.

[7] G. Ameni, M. Vordermeier, R. Firdessa, A. Aseffa, G. Hewinson, S.V. Gordon, and S. Berg. *Mycobacterium tuberculosis* infection in grazing cattle in central Ethiopia. *Vet J.*, 188(3):359–361, 2011.

[8] F.R. Araújo, A.B. Castelão, A.A. Fonseca-Jr., M.A. Hodon, M.A. Issa, A.K. Ramalho, G.M.T. Mendes, C.A.N. Ramos, E.B. Sales, P.M. Soares-Filho, N.C. Farias, C. Nishibe, and N.F. Almeida. Typing of a Brazilian *Mycobacterium* isolate by whole-genome sequencing. In M.P. de Souto and M. Kann, editors, *BSB2012 Digital Proceedings*, pages 15–20. VII Brazilian Symposium on Bioinformatics, 2012. ISSN 2316-1248.

[9] G.S. Araújo. Filogenia de Proteomas. Master's thesis, Universidade Federal de Mato Grosso do Sul, FACOM, 2003.

[10] 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. Harris, D. 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. Genetics*, 25(1):25–29, 2000.

[11] A.O. Azambuja, G.C. Alles, L.L. Fritz, M.H.R. Reche, and L.M. Fiuza. Ecologia de Bacillus entomopatogênicos. *Biotecnologia Ciência e Desenvolvimento*, 38:14–23, 2009.

[12] D.A. Benson, I. Karsch-Mizrachi, D.J. Lipman, J. Ostell, and E.W. Sayers. GenBank. *Nucleic Acids Research*, 37:D26–31, 2009.

[13] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28:235–242, 2000.

[14] P. Bifani, S. Moghazeh, B. Shopsin, J. Driscoll, A. Ravicovitch, and B.N. Kreiswirth. Molecular characterization of *Mycobacterium tuberculosis* H37Rv/Ra variants: distinguishing the mycobacterial laboratory strain. *Journal of Clinical Microbiology*, 38(9):3200–3204, 2000.

[15] R. Brosch, S. Gordon, M. Marmiesse, P. Brodin, C. Buchrieser, K. Eiglmeier, T. Garnier, C. Gutierrez, G. Hewinson, K. Kremer, L.M. Parsons, A. Pym, S. Samper, D. van Soolingen, and S. Cole. A new evolutionary scenario for the *Mycobacterium tuberculosis* complex. *PNAS*, 99(6):3684–3689, 2012.

[16] T.A. Brown. *Genomes*. John Wiley and Sons, 2nd edition, 2002.

[17] K. Brudey, J.R. Driscoll, L. Rigouts, W.M. Prodinger, and et al. *Mycobacterium tuberculosis* complex genetic diversity: mining the fourth international spoligotyping database (SpolDB4) for classification, population genetics and epidemiology. *BMC Microbiology*, 6(23), 2006.

[18] H. Busse, E.B.M. Denner, and W. Lubitz. Classification and identification of bacteria: current approaches to an old problem. Overview of methods used in bacterial systematics. *Journal of Biotechnology*, 47:3–38, 1996.

[19] R. Cai, J. Lewis, S. Yan, H. Liu, C. Clarke, F. Campanile, N.F. Almeida, D.J. Studholme, M. Lindeberg, D. Schneider, M. Zaccardelli, J.C. Setubal, N.P. Morales-Lizcano, A. Bernal, G. Coaker, C. Baker, C.L. Bender, S. Leman, and B.A. Vinatzer. The Plant Pathogen *Pseudomonas syringae* pv. tomato Is Genetically Monomorphic and under Strong Selection to Evade Tomato Immunity. *PLoS Pathogens*, 7(8), 2011.

[20] S.B. Cannon and N.B. Young. OrthoParaMap: Distinguishing orthologs from paralogs by integrating comparative genome data and gene phylogenies. *BioMed Central Bioinform.*, 4:35, 2003.

[21] C.R. Carlson, D.A. Caugant, and A. Kolstϕ. Genotypic diversity among *Bacillus cereus* and *Bacillus thuringiensis* strains. *Applied and Environmental Microbiology*, 60(6):1719–1725, 1994.

[22] J Castresana. Selection of Conserved Blocks from Multiple Alignments for Their Use in Phylogenetic Analysis. *Molecular Biology and Evolution*, 17(4):540–552, 2000.

[23] A. Cubillos-Ruiz, A. Sandoval, V. Ritacco, B. López, J. Robledo, N. Correa, I. Hernandez-Neuta, M.M. Zambrano, and P. Dell Portillo. Genomic signatures of the Haarlem lineage of *Mycobacterium tuberculosis*: implications of strain genetic variation in drug and vaccine development. *Journal of Clinical Microbiology*, 48(10):3614–3623, 2010.

[24] Z. Cvetnic, V. Katalinic-Jankovic, and *et al*. *Mycobacterium caprae* in cattle and humans in Croatia. *Int J Tuberc Lung Dis*, 11(6):652–658, 2007.

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

[26] R.C. Edgar. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32:1792–1797, 2004.

[27] A.J. Enright, S. Van Dongen, and C.A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30:1575-1584(7), 2002.

[28] A. Fagerlund, T. Lindbäck, A.K. Storset, P.E. Granum, and S.P. Hardy. *Bacillus cereus* Nhe is a pore-forming toxin with structural and functional properties similar to the ClyA (HlyE, SheA) family of haemolysins, able to induce osmotic lysis in epithelia. *Microbiology*, 154:693–704, 2008.

[29] J. Felsenstein. PHYLIP - Phylogeny Inference Package (Version 3.2). *Cladistics*, 5:164–166, 1989.

[30] W. M. Fitch. Homology a personal view on some of the problems. *Trends in genetics : TIG*, 16(5):227–231, May 2000.

[31] M.A. Forrellad, L.I. Klepp, A. Gioffré, J.S. García, H.R. Morbidoni, M.P. Santangelo, A.A. Cataldi, and F. Bigi. Virulence factors of the *Mycobacterium tuberculosis* complex. *Virulence*, 4(1):1–64, 2012.

[32] C. Gemmel. Where is typing going? *J. Hospital Infection*, 43:S89–S92, 1999.

[33] A.M. Griffin and H.G. Griffin. *Computer analysis of sequence data, Part II*, volume 25 of *Methods in Molecular Biology*. Blackwell Science Ltd, 1994.

[34] M.H. Guinebretiere, V. Broussolle, and C. Nguyen-The. Enterotoxigenic Profiles of Food-Poisoning and Food-Borne *Bacillus cereus* Strains. *Journal of clinical microbiology*, 40(8):3053–3056, 2002.

[35] D. Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, NY, USA, 1997.

[36] E. Helgason, D.A. Caugant, M. Lecadet, Y. Chen, J. Mahillon, A. Lövgren, I. Hegna, K. Kvalϕy, and A. Kolstϕ. Genetic diversity of *Bacillus cereus/ Bacillus thuringiensis* isolates from natural sources. *Current Microbiology*, 37:80–87, 1998.

[37] E. Helgason, N.J. Tourasse, R. Meisal, D.A. Caugant, and A. Kolstϕ. Multilocus Sequence Typing Scheme for bacteria of the *Bacillus cereus* group. *Applied and Environmental Microbiology*, 70(1):191–201, 2004.

[38] P. Hermans, D. van Soolingen, E. Bick, P. de Haas, J. Dale, and J. van Embden. Insertion element IS987 from *Mycobacterium bovis* BCG is located in a hot-spot integration region for insertion elements in *Mycobacterium tuberculosis* complex strains. *Infection and Immunity*, 59(8):2695–2705, 1991.

[39] J. Kamerbeek, L. Schouls, A. Kolk, M. van Agterveld, D. van Soolingen, S. Kuijper, A. Bunschoten, H. Molhuizen, R. Shaw, M. Goyal, and J. van Embden. Simultaneous detection and strain differentiation of *Mycobacterium tuberculosis* for diagnosis and epidemiology. *Journal of Clinical Microbiology*, 35(4):907–914, 1997.

[40] A. Klevan, N. Tourasse, F. Stabell, A. Kolstϕ, and O. Φkstad. Exploring the evolution of the *Bacillus cereus* group repeat element bcr1 by comparative genome analysis of closely related strains. *Microbiology*, 153(11):3894–3908, 2007.

[41] K.S. Ko, J.W. Kim, J.M. Kim, W. Kim, S. Chung, I.J. Kim, and Y. Kook. Population Structure of the *Bacillus cereus* Group as Determined by Sequence Analysis of Six Housekeeping Genes and the plcR Gene. *Infection and Immunity*, 72(9):5253–5261, 2004.

[42] S. Kumar, K. Tamura, and M. Nei. MEGA: Molecular Evolutionary Genetics Analysis software for microcomputers. *Computer Applications in the Biosciences*, 10:189–191, 1994.

[43] J. Lang, A. Darling, and J. Eisen. Phylogeny of bacterial and archeal genomes using conserved genes: supertrees and supermatrices. *PLOS ONE*, 8(4):e62510, 2013.

[44] W. Lee, F. Liang, J. Huang, T. Jaing, C. Wang, T. Lin, Y. Huang, W. Huang, R. Jou, M. Hsieh, J. Shia, and T. Wu. Immunologic analysis of HIV-uninfected taiwanese children with BCG-induced disease. *Journal of Clinical Immunology*, 29:319–329, 2009.

[45] R. Leinonen, F. Nardone, W. Zhu, and R. Apweiler. UniSave: the UniProtKB Annotation Version database. *Bioinformatics*, 22(10):1284–1285, 2006.

[46] A.M. Lesk. *Introdução à Bioinformática*. Porto Alegre: Artmed, 2008.

[47] L. Li, C.J. Stoeckert, and D.S. Roos. OrthoMCL: identification of ortholog groups for eukaryotic genomes. *Genome Research*, 13(9):2178–89, 2003.

[48] Q. Li, C.C. Whalen, J.M. Albert, R. Larkin, L. Zukowski, M.D. Cave, and R.F. Silver. Differences in rate and variability of intracellular growth of a panel of *Mycobacterium tuberculosis* clinical isolates within a human monocyte model. *Infection and Immunity*, 70(11):6489–6493, 2002.

[49] W. Li, D. Raoult, and P. Fournier. Bacterial strain typing in the genomic era. *FEMS Microbiology Reviews*, 33:892–916, 2009.

[50] A.C. López and A.M. Alippi. Enterotoxigenic gene profiles of *Bacillus cereus* and *Bacillus megaterium* isolates recovered from honey. *Revista Argentina de Microbiología*, 42(3):216–225, 2010.

[51] M.C.J. Maiden. Multilocus Sequence Typing of Bacteria. *Annu. Rev. Microbiol.*, 60:561–588, 2006.

[52] B. Mathema, N.E. Kurepina, P.J. Bifani, and B.N. Kreiswirth. Molecular epidemiology of tuberculosis: current insights. *Clinical Microbiology Reviews*, 19(4):658–685, 2006.

[53] S. Mignard, C. Pichat, and G. Carret. *Mycobacterium bovis* infection, Lion, France. *Emerging Infectious Diseases*, 12(9):1431–1433, 2006.

[54] L.M. Moreira, N.F. Almeida Jr, N. Potnis, L.A. Digiampietri, S.S. Adi, J.C. Bortolossi, A.C. da Silva, A.M. da Silva, F.E. de Moraes, J.C. de Oliveira, R.F. de Souza, A.P. Facincani, A.L. Ferraz, M.I. Ferro, L.R. Furlan, D.F. Gimenez, J.B. Jones, E.W. Kitajima, M.L. Laia, R.P. Leite Jr, M.Y. Nishiyama, J. Rodrigues Neto, L.A. Nociti, D.J. Norman, E.H. Ostroski, H.A. Pereira Jr, B.J. Staskawicz, R.I. Tezza, J.A. Ferro, B.A. Vinatzer, and J.C. Setubal. Novel insights into the genomic basis of citrus canker based on the genome sequences of two strains of *Xanthomonas fuscans* subsp. *aurantifolii*. *BMC Genomics*, 11:238+, 2010.

[55] H. Muniz, N.C. Farias, T. Raiol, A.L. Cunha-Laura, G.P. Telles, and N.F. Almeida. A comparison of two approaches for species identification. In M.P. de Souto and M. Kann, editors, *BSB2012 Digital Proceedings*, pages 50–55. VII Brazilian Symposium on Bioinformatics, August 2012. ISSN 2316-1248.

[56] E.P. Nawrocki, D.L. Kolbe, and S.R. Eddy. Infernal 1.0: Inference of RNA alignments. *Bioinformatics*, 25:1335–1337, 2009.

[57] M. Nei and S. Kumar. *Molecular evolution and phylogenetics*. Oxford University Press, 2000.

[58] J.V.A. Oliveira, T. Raiol, N.C. Farias, N.F. Almeida, G.P. Telles, J.C. Setubal, M.L. Araujo, F.P. Lima, C. Benoit-Pilven, M.M. Brigido, M.T. Souza, L.M.P. Moraes, and M.E.M.T. Walter. Genome sequence of a Brazilian *Bacillus cereus* isolate. In M.P. de Souto and M. Kann, editors, *BSB2012 Digital Proceedings*, pages 9–14. VII Brazilian Symposium on Bioinformatics, 2012. ISSN 2316-1248.

[59] D. Parashar, V.S. Chauhan, V.D. Sharma, and V.M. Katoch. Applications of real-time PCR technology to mycobacterial research. *Indian J. Med. Res.*, 124(4):385–398, 2006.

[60] G. Perrière and M. Gouy. WWW-query: An on-line retrieval system for biological sequence banks. *Biochimie*, 78(5):364–369, 1996.

[61] J. Pevsner. *Bioinformatics and Functional Genomics*. John Willey and Sons Inc., 2009.

[62] M. Pillay and A.W. Sturm. Evolution of the extensively drug-resistant F15/LAM4/KZN strain of *Mycobacterium tuberculosis* in KwaZulu-Natal, South Africa. *Clinical Infectious Diseases*, 45:1409–1414, 2007.

[63] N. Potnis, K. Krasileva, V. Chow, N.F. Almeida, P. Patil, R. Ryan, M. Sharlach, F. Behlau, J.M. Dow, WhiteF., J. Preston, B. Vinatzer, R. Koebnik, J.C. Setubal, D.J. Norman, B. Staskawicz, and J.B. Jones. Comparative genomics reveals diversity among *Xanthomonas* infecting tomato and pepper. *BMC Genomics*, 12:146, 2011.

[64] Z. Rahim, M. Möllers, and *et al.* Characterization of *Mycobacterium africanum* subtype I among cows in a dairy farm in Bangladesh using spoligotyping. *Southeast Asian J Trop Med Public Health*, 38(4):706–713, 2007.

[65] T.B.K. Reddy, R. Riley, F. Wymore, P. Montgomeri, D. Decaprio, R. Engels, M. Gellesh, J. Hubble, D. Jen, H. Jin, M. Koersen, and et al. TB database: an integrated platform for tuberculosis research. *Nucleic Acids Research*, 37:D499–D508, 2009.

[66] M. Ridley. *Evolução*. Porto Alegre: Artmed, 2008.

[67] L. Rodriguez-R, A. Grajales, M. Arrieta-Ortiz, C. Salazar, S. Restrepo, and A. Bernal. Genomes-based phylogeny of the genus *Xanthomonas*. *BMC Microbiology*, 12:43, 2012.

[68] I.T. Rombel, K.F. Sykes, S. Rayner, and S.A. Johnston. ORF-FINDER: A vector for high-throughput gene identification. *Gene*, 282:33–41, 2002.

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

[70] M.P.U. Sales, G.M. Taylor, S. Hughes, M. Yates, G. Hewinson, D.B. Young, and R.J. Shaw. Genetic diversity among *Mycobacterium bovis* isolates: a preliminary study of strains from animal and human sources. *Journal of Clinical Microbiology*, 39(12):4558–4562, 2001.

[71] S.L. Salzberg, A.L. Delcher, S. Kasif, and O. White. Microbial gene identification using interpolated Markov models. *Nucleic Acids Research*, 26(2):544–548, jan 1998.

[72] A.C. Schürch and D. van Soolingen. DNA fingerprinting of *Mycobacterium tuberculosis*: from phage typing to whole-genome sequencing. *Infect Genet Evo.*, 12(4):602–609, 2012.

[73] J.C. Setubal, P. dos Santos, B.S. Goldman, H. Ertesvag, G. Espin, L. Rubio, S. Valla, N.F. Almeida, D. Balasubramanian, L. Cromes, L. Curatti, Z. Du, E. Godsy, B. Goodner, K. Hellner-Burris, J.A. Hernandez, K. Houmiel, J. Imperial, C. Kennedy, T.J. Larson, P. Latreille, L. Ligon, J. Lu, M. Mark, N. Miller, S. Norton, I.P. OCarroll, I. Paulsen, E.C. Raulfs, R. Roemer, J. Rosser, D. Segura, S. Slater, S. Stricklin, D. Studholme, J. Sun, C.J. Viana, E. Wallin, B. Wang, and C. Wheeler. The genome sequence of *Azotobacter vinelandii*, an obligate aerobe specialized to support diverse anaerobic metabolic processes. *J. Bacteriology*, 191(14):4534–4545, July 2009.

[74] J.C. Setubal and J. Meidanis. *Introduction to computational molecular biology*. PWS Publishing Company, 1997.

[75] R. Shamir. Algorithms for Molecular Biology - Lecture 10, 1999.

[76] S. Slater, B. Goldman, B. Goodner, J.C. Setubal, S. Farrand, E. Nester, T. Burr, L. Banta, A. Dickerman, I. Paulsen, L. Otten, G. Suen, R. Welch,

N.F. Almeida, F. Arnold, O. Burton, Z. Du, A. Ewing, E. Godsy, S. Heisel, K. Houmiel, J. Jhaveri, J. Lu, N. Miller, S. Norton, Q. Chen, W. Phoolcharoen, V. Ohlin, D. Ondrusek, N. Pride, S. Stricklin, J. Sun, C. Wheeler, L. Wilson, H. Zhu, and D. Wood. Genome Sequences of Three *Agrobacterium Biovars* Help Elucidate the Evolution of Multichromosome Genomes in Bacteria. *Journal of Bacteriology*, 191(8):2501–2511, April 2009.

[77] E.L. Sonnhammer and E.V. Koonin. Orthology, paralogy and proposed classification for paralog subtypes. *Trends Genet*, 18(12):619–20, 2002.

[78] A. Stamatakis. RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics*, 22(21):2688–2690, 2006.

[79] A.P. Stamatakis, T. Ludwig, H. Meier, and M.J. Wolf. AxML: a fast program for sequential and parallel phylogenetic tree calculations based on the maximum likelihood method. In *Computer Society Bioinformatics Conference*, pages 21–28. Computer Society Bioinformatics, 2002.

[80] L. Stein. Genome annotation: from sequence to biology. *Nature reviews. Genetics*, 2(7):493–503, jul 2001.

[81] E.M. Streicher, T.C. Victor, G. van der Spuy, C. Sola, N. Rastogi, P.D. van Helden, and R.M. Warren. Spoligotype signatures in the *Mycobacterium tuberculosis* complex. *Journal of Clinical Microbiology*, 45(1):237–240, 2007.

[82] P. Supply, J. Magdalena, S. Himpens, and C. Locht. Identification of novel intergenic repetitive units in a mycobacterial two-component system operon. *Molecular Microbiology*, 26(5):991–1003, 1997.

[83] P. Supply, E. Mazars, S. Lesjean, V. Vincent, B. Gicquel, and C. Locht. Variable human minisatellite-like regions in the *Mycobacterium tuberculosis* genome. *Molecular Microbiology*, 36(3):762–771, 2000.

[84] A. Szabo, C.M. Perou, M. Karaca, L. Perreard, J.F. Quackenbush, and P.S. Bernard. Statistical modeling for selecting housekeeper genes. *Genome Biology*, 5(8):R59, 2004.

[85] G. Talavera and J. Castresana. Improvement of phylogenies after removing divergent and ambiguously aligned blocks from protein sequence alignments. *Systematic Biology*, 56(4):564–577, Aug 2007.

[86] A.M. Tallent, K.M. Kotewicz, E.A. Strain, and R.W. Bennett. Efficient Isolation and Identification of *Bacillus cereus* Group. *Journal of AOAC International*, 95(2):446–451, 2012.

[87] C. Tang, J.F. Reyes, F. Luciani, A.R. Francis, and M.M. Tanaka. SpolTools: online utilities for analyzing spoligotypes of the *Mycobacterium tuberculosis* complex. *Bioinformatics Applications Note*, 24(20):2414–2415, 2008.

[88] G.P. Telles, N.F. Almeida, M.M. Brigido, P.A. Alvarez, and M.E. Walter. kGC: Finding groups of homologous genes across multiple genomes. In O.N. de Souza, G.P. Telles, and M.J. Palakal, editors, *Advances in Bioinformatics and Computational Biology*, volume 6832 of *LNBI*, pages 78–82. Springer, 2011.

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

[90] L.O. Ticknor, A. Kolstϕ, K.K. Hill, P. Keim, M.T. Laker, M. Tonks, and P.J. Jackson. Fluorescent amplified fragment length polymorphism analysis of norwegian *Bacillus cereus* and *Bacillus thuringiensis* soil isolates. *Applied and Environmental Microbiology*, 67(10):4863–4873, 2001.

[91] N.J. Tourasse and A.B. Kolsto. SuperCAT: a supertree database for combined and integrative multilocus sequence typing analysis of the *Bacillus cereus* group of bacteria. *Nucleic Acids Research*, 36:D461–D468, 2008.

[92] J. van Embden, T. van Gorkom, K. Kremer, R. Jansen, B.A.M. van der Zeijst, and L. Schouls. Genetic variation and evolutionary origin of the direct repeat locus of *Mycobacterium tuberculosis* complex bacteria. *Journal of Bacteriology*, 182(9):2393–2401, 2000.

[93] C.J.M. Viana. Aspectos de genômica comparativa. Master's thesis, Universidade Federal de Mato Grosso do Sul, FACOM, 2006.

[94] V. Vilas-Boas, V. Sanchis, D. Lereclus, M.V.F. Lemos, and D. Bourguet. Genetic Differentiation between Sympatric Populations of *Bacillus cereus* and *Bacillus thuringiensis*. *Applied and Environmental Microbiology*, 68(3):1414–1424, 2002.

[95] M.E.M.T. Walter and M.M. Brígido. Genômica Comparativa: aspectos biológicos e técnicas computacionais. In *EBB 2012*. Escola Brasileira de Bioinformática EBB2012, 2012.

[96] A.R. Wattam, K. Williams, E. Snyder, N.F. Almeida, M. Shukla, A. Dickerman, O. Crasta, R. Kenyon, J. Lu, J. Shallom, H. Yoo, T. Ficht, R. Tsolis, C. Munk, R. Tapia, C. Han, J. Detter, D. Bruce, T. Brettin, B. Sobral, S. Boyle, and J.C. Setubal. Analysis of ten *Brucella* genomes reveals evidence for horizontal gene transfer despite a preferred intracellular lifestyle. *Journal of Bacteriology*, 191(11):3569–3579, June 2009.

[97] A. Weber, U. Reischl, and L. Naumann. Demonstration of *Mycobacterium africanum* in a bull from North Bavaria. *Berl Munch Tierarztl Wochenschr*, 111(1):6–8, 1998.

[98] T. Weniger, J. Krawczyk, P. Supply, S. Niemann, and D. Harmsen. MIRU-VNTRplus: a web tool for polyphasic genotyping of *Mycobacterium tuberculosis* complex bacteria. *Nucleic Acids Research*, 38:W326–W331, 2010.

[99] K. Williams, J. Gillespie, B. Sobral, E. Nordberg, E. Snyder, J. Shallom, and A. Dickerman. Phylogeny of Gammaproteobacteria. *Journal of Bacteriology*, 192(9):2305–2314, 2010.

[100] M. Wu and J. Eisen. A simple, fast, and accurate method of phylogenomic inference. *Genome Biology*, 9(10):R151, 2008.

[101] M. Wu and A. Scott. Phylogenomic analysis of bacterial and archeal sequences with AMPHORA2. *Bioinformatics*, 28(7):1033–1034, 2012.

[102] M.E. Zwick, S.J. Joseph, X. Didelot, P.E. Chen, K.A. Bishop-Lilly, A.C. Stewart, K. Willner, N. Nolan, S. Lentz, M.K. Thomason, S. Sozhamannam, A.J. Mateczum, L. Du, and T.D. Read. Genomic characterization of the *Bacillus cereus* sensu lato species: backdrop to the evolution of *Bacillus anthracis*. *Genome Research*, 22(8):1512–1524, 2012.

Apêndice

Nas páginas seguintes o manual do usuário para o *pipeline* é transcrito.

Orthologsorter User's Guide
Version 2.0

==========
========== Introduction ==========
==========

This document describes the Orthologsorter software.

With the development of genomic research and frequent discoveries of new organisms to be studied, increases the necessity of tools that allow their detailed analyzes, beyond the storage and management of large amount of information generated.

An essential step is the analysis of the biological sequences in order to obtain functional characterizations and evolutionary details.

Orthologsorter is automatic pipeline which aims to compare and analyze bacterial genomes. The program is based on the comparison of protein sequences in order to determine the typing of bacteria.

The input to Orthologsorter is the set of predicted proteins of the species of interest and a configuration file created by the user.

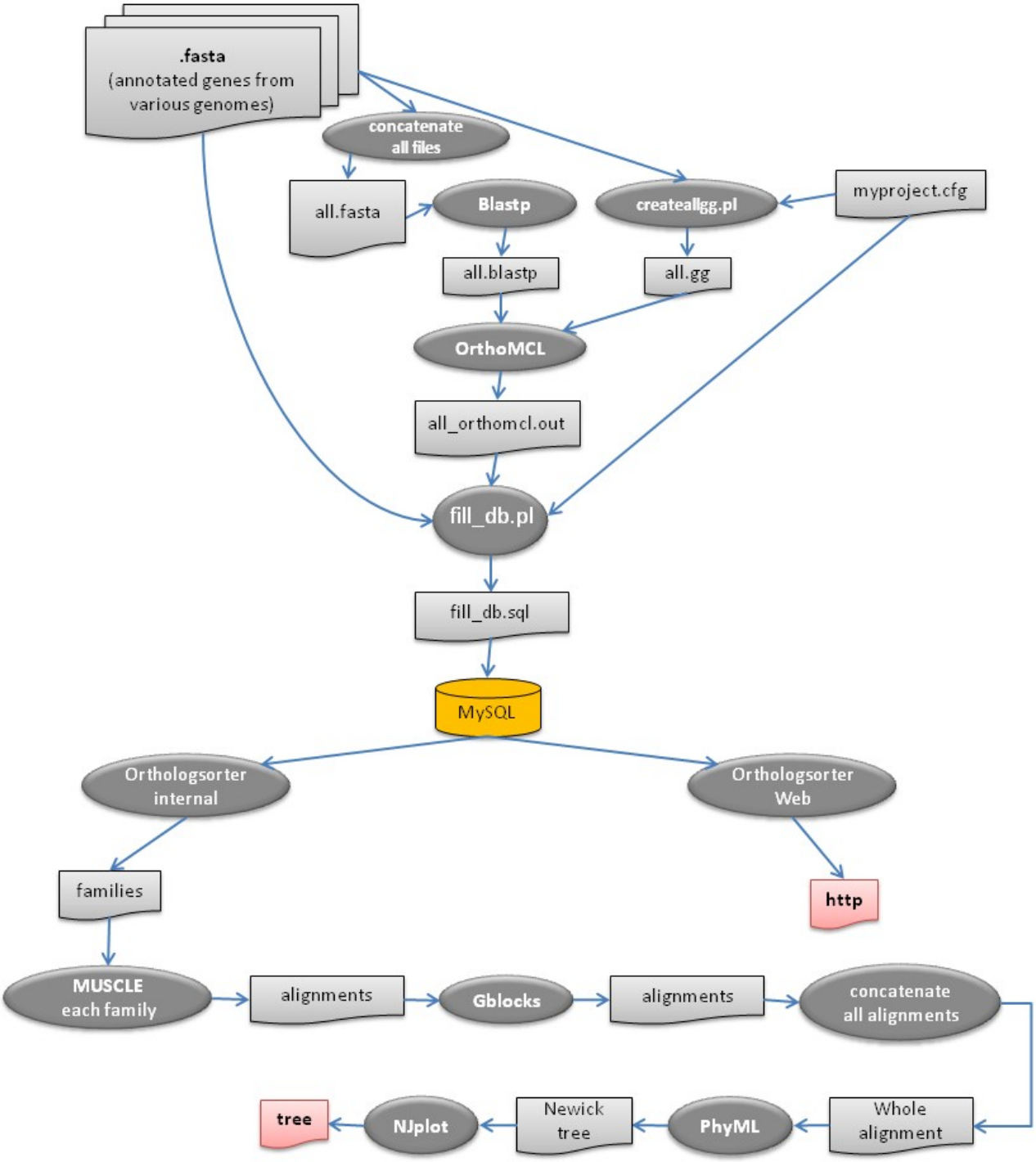
The output can be divided in two parts:

- The first consists of a web search tool, called Orthologsorter, allowing search a set of protein families obtained by some method of clustering sequences in order to obtain functional and evolutionary information of the analyzed species.
- The second is to construct a phylogenetic tree through of a specific search utilizing Orthologsorter, contributing to the inference of the evolutionary history of the species involved.

There are five overall stages:

- all-v-all Blastp -- alignment of the sequences so as to observe their level of similarity;
- the OrthoMCL program -- creates cluster of proteins;
- MySQL -- creates a database;
- MUSCLE and Gblocks program -- The families of proteins are aligned by MUSCLE. Then, each alignment is filtered by GBlocks; and
- PhyML and NJplot -- phylogenetic tree construction.

The flowchart is shown in the figure that follows.



These stages are executed in a series of eight steps detailed in the Section 'Steps in detail'. Most simply involve running a provided program. They are broken into steps for ease of backtracking and recoverability.

==========
========= Requirements ==========
==========

(1) LINUX

- The Orthologsorter program has only been tested on LINUX.

(2) Relational Database

- The Orthologsorter program runs in a relational database. Supported vendor is:
  - MySQL;
- If you don"t already have installed, install MySQL, which can be done for free and without significant systems administration support.

(3) Perl

- standard perl;
- DBI libraries;
- BioPerl libraries;

(4) Hardware

- The hardware requirements vary dramatically with the size of your dataset.

(5) Apache

- The Apache HTTP Server, commonly referred to as Apache, is a web server software.

==========
========= Installation ==========
==========

Copy the orthologsorter tgz file to a directory that is accessible from your computer. To extract .tgz files. Use this command to unpack the tgz files:

```
$ tar -zxvf orthologsorter_2.0.tgz
```

==========
========= File Formats ==========
==========

The Orthologsorter takes as input the fasta files (.faa) and a configuration file. These files must follow the format described below.

- fasta files (.faa)

Input files must be in FASTA format. Each sequence starts with an annotation line, which is recognized by having a greater-than symbol ">" as its first character. The sequence itself follows on one or more subsequent lines, and is terminated either by the next annotation line or by the end of the file. The input file consists of amino acids sequences. The standard single-letter amino acid alphabet is used.

Example:

>gi|229015392|ref|ZP_04172400.1| hypothetical protein bmyco0001_57200 [Bacillus mycoides DSM 2048]
MSRFIAKQIIINDRFLALIFGETDMKNKNIEVPDIRKNGIT
>gi|229015391|ref|ZP_04172399.1| hypothetical protein bmyco0001_57210 [Bacillus mycoides DSM 2048]
MVGSFVSYTNEASTEFTYNNPLYFWKGFFFIQIIPLFVLFVALIWWFIRKGKEKIDT

- configuration file

The configuration file, called "myproject.cfg", provided by the user, is used in the pipeline during execution. First, is described project title. Then, a genome is described by line. In the first column is used "1" if the genome is ingroup or "0" if outgroup. In the second column is used to "0" if the genome is available at NCBI or "1" otherwise. In the third column is described name of the genome separated by the symbol "_" and, in the fourth column, the genome information, separated by a space. Furthermore, the user must provide the url where the directories cgi-bin and /var/www are available.

Example:

#Project title
Bacillus cereus FT9 genomics

#Genomes information

1	*0*	*Bacillus_anthracis_AE016879*	*Bacillus anthracis AE016879*
1	*0*	*Bacillus_cereus_ATCC_10987*	*Bacillus cereus ATCC 10987*
1	*0*	*Bacillus_cytotoxicus_NVH_391_98*	*Bacillus cytotoxicus NVH 391 98*
1	*0*	*Bacillus_mycoides_DSM_2048*	*Bacillus mycoides DSM 2048*
1	*0*	*Bacillus_pseudomycoides_DSM_12442*	*Bacillus pseudomycoides DSM 12442*
1	*0*	*Bacillus_weihenstephanensis_KBAB4*	*Bacillus weihenstephanensis KBAB4*
0	*0*	*Bacillus_pumilus_SAFR_032*	*Bacillus pumilus SAFR 032*

#url
http://pacu.facom.ufms.br

==========
========== Overview of steps ==========
==========

This is an overview of the eight steps to run orthologsorter. Details are in the next sections.

All programs except MySQL Server, Apache and Perl are provided as part of the Orthologsorter download.

(1) install and configure the Orthologsorter suite of programs.

(2) create the project directory and upload the fastas files (.faa).

(3) configure Orthologsorter by creating the <myproject>.cfg file.

(4) run createDatabase.pl to create a sql script that loads the data into the database.

(5) logged in to MySQL, execute the sql script created in the previous step to create and load the database used in the project.

(6) run createOrthologsorter.pl to create a web page and a phylogenetic tree.

(7) create a project folder in the cgi-bin directory of the local machine.

(8) create a project folder in the /var/www/ directory of the local machine.

We recommend you save the output of each step so that you can easily redo it if things go wrong.

==========
========== Steps in detail ==========
==========

========== Step 1: install Orthologsorter ========
Input:
- orthologsorter_2.0.tgz

Output:
- bin directory of executable programs
- doc directory of documentation files
- projects directory to store project data
- scripts directory to run the orthologsorter
- createDatabase.pl file
- createOrthologsorter.pl file
- README

Use this command to unpack the tgz files:

```
$ tar -zxvf orthologsorter_2.0.tgz
```

The result will be this:

```
orthologsorter/
    bin/
      blast/
      command_line_njplot/
      gblocks/
      muscle/
      orthomcl/
      phyml/
    doc/
      example.cfg
      Orthologsorter_UserGuide.txt
    projects/
    scripts/
      create_tables_from_faa.pl
      edit_fastas.pl
      fill_db.pl
      generate_call_web_blast.pl
      generate_html.pl
      generate_list_singletons.pl
      generate_orthologsorter.pl
      orthomcl2fastas.pl
      web_blast.pl
    createDatabase.pl
    createOrthologsorter.pl
    README
```

Enter in the orthologsorter directory.

```
$ cd orthologsorter_2.0
```

IMPORTANT: The next steps must be run from the orthologsorter directory. When you find <myproject>, change by the project name replacing blanks by underline.

========== Step 2: Upload the fastas files (.faa) ========
Create a directory to store the project data, replacing <myproject> by the name of their project. Run the following command:

$ mkdir projects/<myproject>

Create a directory to store the fastas files. Run the following command:

$ mkdir projects/<myproject>/fastas

In this directory "projects/<myproject>/fastas", uploads the fastas files (.faa) of the genomes to be analyzed.

========== Step 3: configure Orthologsorter ========
Create the <myproject>.cfg file in the following directory "projects/<myproject>".

The <myproject>.cfg file is used to get settings of the Orthologsorter program during its execution.

The example of the <myproject>.cfg file can be viewed in the doc directory.

========== Step 4: createDatabase ========
Input:

- projects/<myproject>/<myproject>.cfg
- fasta files as acquired from the genome resource (projects/<myproject>/fastas)

Output:

- temporary directory (projects/<myproject>/temp)
- log directory (projects/<myproject>/temp/log)
- projects/<myproject>/temp/gitoid.txt
- projects/<myproject>/fastas/all.fasta
- projects/<myproject>/temp/blast/all.blastp
- projects/<myproject>/temp/orthomcl/all.gg
- projects/<myproject>/temp/orthomcl/all_orthomcl.out
- projects/<myproject>/temp/MySQL/fill_db.sql

First, this step checks the fastas files and edits to a standard. If the fasta file isn't standardized, the gitoid.txt file (projects/<myproject>/temp) is created. In this file can be obtained a relationship between the old ID number and the new ID number of each gene in the genome.

Second, create the "all.fasta" file. This file contains the annotated genes of all organisms analyzed. Concatenates the ".faa" files in a single file, called "all.fasta".

It created the log directory where are stored the logs of each program.

After, generate the "all.gg" file. This file contains a list of genomes with the identification numbers of the genes of this organism. This file is required to run the OrthoMCL program.

Then run the Blastp and OrthoMCL programs. The Blastp and OrthoMCL output, called "all.blastp" and "all_orthomcl.out" respectively, are parsed and the input files mentioned above are analyzed in order to create a script to build a database, called "fill_db.sql".

IMPORTANT: If you already have the output of Blastp, rename your file to “all.blastp” and store it in the directory “orthologsorter/projects/<myproject>/temp/blast/”. Similarly, if you already have the output of OrthoMCL, rename your file to “all_orthomcl.out” and store it in the directory “orthologsorter/projects/<myproject>/temp/orthomcl/”.

Use the createDatabase script as described below.

$ perl createDatabase.pl <myproject>

========== Step 5: MySQL ========
Input:
- projects/<myproject>/temp/MySQL/fill_db.sql
Output:
- database

Log in MySQL as root by running the commands below in order to create and load the database used in the project.

$ MySQL -u <user> -p

$ source ./projects/<myproject>/temp/MySQL/fill_db.sql

$ exit

========== Step 6: createOrthologsorter ========
Input:
- projects/<myproject>/<myproject>.cfg
- projects/<myproject>/temp/orthomcl/all_orthomcl.out
- projects/<myproject>/fastas/all.fasta
Output:
- projects/<myproject>/results/www/orthologsorter/index.html
- fastas of families directory (projects/<myproject>/results/www/orthologsorter/fastas_of_families)
- projects/<myproject>/results/cgi-bin/call_web_blast.pl
- projects/<myproject>/results/cgi-bin/list_singletons.pl
- projects/<myproject>/results/cgi-bin/orthologsorter.pl
- muscle output directory (projects/<myproject>/temp/muscle_output)
- paths file (bin/gblocks/paths)
- gblocks output directory (projects/<myproject>/results/www/orthologsorter/gblocks_results)
- phyml output directory (projects/<myproject>/temp/phyml)
- <myproject>.pdf (projects/<myproject>/results/tree)

First, through the script "generate_html.pl", creates a web page where the biologist can make searches in the set of protein families obtained by OrthoMCL in order to obtain functional and evolutionary information of the analyzed species.

Through the script "orthomcl2fastas.pl" is generated the file in the format fasta of each proteins family.

Through the script "generate_call_web_blast" is generated the "call_web_blast.pl" file. The "web_blast.pl" and "call_web_blast.pl" files are used to run the online Blastp of a gene against the base nr of the GenBank.

The "list_singletons.pl" file, generated by script "generate_list_singletons.pl", is used for listing of the protein sequences that do not belong to any of the proteins family of a genome.

The "orthologsorter.pl" file, generated by script "generate_orthologsorter.pl", is used to show the search result accomplished in the set of protein families.

After, run the MUSCLE and Gblocks programs. The "paths" file contains a list with the names of the output files of the MUSCLE, used in the execution of the Gblocks.

Then parses MUSCLE and Gblocks output and concatenates all alignments in a single alignment by creating the Fams.phyrel file. Fams.phyrel file is used as input for the PhyML program that provides a phylogenetic tree, in the Newick format, of the species analyzed. This file, in the Newick format, is read by the NJplot program to draw any binary tree expressed in the standard phylogenetic tree format, called "<myproject>.pdf".

Use the createOrthologsorter script as described below.

```
$ perl createOrthologsorter.pl <myproject>
```

========== Step 7: cgi-bin ========
IMPORTANT: The commands below must be run as root.

Creates a project folder in the cgi-bin directory of the local machine.

```
$ mkdir <path/cgi-bin>/<myproject>
```

Run the command below to copy the files to the cgi-bin directory on the local machine.

```
$ cp ./projects/<myproject>/results/cgi-bin/* <path/cgi-bin>/<myproject>
```

========== Step 8: www ========
IMPORTANT: The commands below must be run as root.

Creates a project folder in the /var/www directory of the local machine.

```
$ mkdir <path/var/ww/>/<myproject>
```

Run the commands below to copy the files to the /var/www/ directory on the local machine.

```
$ cp -r ./projects/<myproject>/results/www/orthologsorter <path/var/ww/>/<myproject>
```

==
============ Using Orthologsorter ======================
==

Now you can access Orthologsorter by your browser through the link below.

```
http://<domain>/<myproject>/orthologsorter/
```

For example, given the domain 'http://pacu.facom.ufms.br/' and the project name 'cereus' the link would be:

```
http://pacu.facom.ufms.br/cereus/orthologsorter/
```

The phylogenetic tree generated, called '<myproject>.pdf', is available in the following directory:

```
projects/<myproject>/results/tree/<myproject>.pdf
```

Narielly Calista Farias, naryfarias@gmail.com

School of Computing - Facom
Federal University of Mato Grosso do Sul