UNIVERSIDADE ESTADUAL PAULISTA
Instituto de Biociências
Campus de Rio Claro

MILENE FERRO

DESENVOLVIMENTO DE *PIPELINES* PARA ANÁLISE DE EST, BIBLIOTECAS ITS E 16S PARA ESTUDOS GENÔMICOS EM FORMIGAS ATTINI

Tese apresentada ao Instituto de Biociências da Universidade Estadual Paulista Júlio de Mesquita Filho, para obtenção do titulo de Doutora.

Rio Claro
2013

MILENE FERRO

DESENVOLVIMENTO DE *PIPELINES* PARA ANÁLISE DE EST, BIBLIOTECAS ITS E 16S PARA ESTUDOS GENÔMICOS EM FORMIGAS ATTINI

Tese apresentada ao Instituto de Biociências do Campus de Rio Claro, Universidade Estadual Paulista Júlio de Mesquita Filho, como parte dos requisitos para obtenção do titulo de Doutor em Ciências Biológicas (Área: Biologia Celular e Molecular).

Orientador: Prof. Dr. Maurício Bacci Júnior.

Rio Claro
2013

574.0285 Ferro, Milene
F395d Desenvolvimento de pipelines para análise de EST, bibliotecas ITS e 16S para estudos genômicos em formigas Attini / Milene Ferro. - Rio Claro : [s.n.], 2013
132 f. : il., figs., tabs., fots.

Tese (doutorado) - Universidade Estadual Paulista, Instituto de Biociências de Rio Claro
Orientador: Maurício Bacci Júnior

1. Biologia - Programas de computador. 2. Bioinformática. 3. Processamento encadeado. 4. Transcriptoma. 5. Anotação de sequências. 6. Formiga Attini. I. Título.

Ficha Catalográfica elaborada pela STATI - Biblioteca da UNESP
Campus de Rio Claro/SP

unesp **UNIVERSIDADE ESTADUAL PAULISTA**

CAMPUS DE RIO CLARO

INSTITUTO DE BIOCIÊNCIAS DE RIO CLARO

CERTIFICADO DE APROVAÇÃO

TÍTULO: DESENVOLVIMENTO DE PIPELINES PARA ANÁLISE DE EST, BIBLIOTECAS 16S E ITS PARA ESTUDOS GENÔMICOS EM FORMIGAS ATTINI

AUTORA: MILENE FERRO

ORIENTADOR: Prof. Dr. MAURÍCIO BACCI JUNIOR

Aprovada como parte das exigências para obtenção do Título de DOUTOR EM CIENCIAS BIOLÓGICAS (BIOLOGIA CELULAR E MOLECULAR) , pela Comissão Examinadora:

Prof. Dr. MAURÍCIO BACCI JUNIOR
Departamento de Bioquimica e Microbiologia / Instituto de Biociências / UNESP - Rio Claro

Prof. Dr. ANDRE RODRIGUES
Departamento de Bioquimica e Microbiologia / Instituto de Biociências / UNESP - Rio Claro

Prof. Dr. HENRIQUE FERREIRA
Departamento de Ciências Biológicas / Faculdade de Ciencias Farmaceuticas / UNESP - Araraquara

Prof. Dr. FLAVIO HENRIQUE DA SILVA
Departamento de Genética e Evolução / Universidade Federal de Sao Carlos

Prof. Dr. CLAUDIO JOSÉ VON ZUBEN
Departamento de Zoologia / Instituto de Biociências / UNESP - Rio Claro

Data da realização: 27 de março de 2013.

À minha mãe Vera, vó Thereza e
ao meu esposo Erik,
grandes incentivadores e
que merecem toda essa conquista,
dedico esta Tese.

Agradecimentos

Agradecimentos

Antes de tudo quero agradecer a Deus pelo presente da minha vida, por todos os aprendizados diários e pelas pessoas que coloca em meu caminho. Agradeço pelo milagre de acordar todos os dias e por tudo que habita este mundo.

Agradeço ao meu marido Erik por ter compartilhado comigo as tarefas desse trabalho, me auxiliando com ideias e com conceitos de Computação. Também é um companheiro maravilhoso, dedicado e fiel.

Agradeço a minha mãe Vera e a minha avó Thereza que, para mim, são as mulheres mais fortes, guerreiras e exemplos de tudo que uma pessoa possa ter de melhor, nunca evitaram esforços para me dar sempre o melhor. Aos meus irmãos Marcos, Marcel e Magda, por serem companheiros e estarem sempre presentes em todas as etapas da minha vida.

Quero agradecer também cada um dos meus familiares, tios (Cido, Sérgio C. e Sérgio G.), tias (Zezé, Sandra e Gracilene), primos (Rafa e Thiago), primas (Isa, Taís, Roberta e Carol), cunhados (Didi), cunhadas (Fer e Josi) e sobrinhos (Ju e Evelyn), com os quais sempre compartilho momentos importantes, sejam estes felizes ou tristes, sérios ou descontraídos, enfim, agradeço pela convivência e pelo carinho.

Agradeço ao Dr. Maurício Bacci Júnior que já em 2002, antes dos meus estudos da graduação, me concedeu o privilégio de fazer estágio de Bioinformática em seu laboratório e depois de quase oito anos me acolheu novamente para o doutorado e me confiou a realização desse e de outros projetos. Também agradeço por toda a infraestrutura oferecida e por todo aprendizado.

Tive o privilegio de conhecer a Prof. Dora que me acolheu nas aulas de Biologia Celular em 2002 quando ainda não estava na faculdade. Agradeço por tudo.

Durante esses quatro anos no LEM aprendi o que é ter amigos e o que é fazer pesquisa. Conheci pessoas inesquecíveis (Cy, Carol, Cintia, Joana, Mari, Tássio, Shao, Miagui, Paula, Franco, Alexandre, Diego, Wélliton, Suzana) com as quais dividi momentos incríveis, cada um com seu jeito e sua pesquisa me ensinou muito, trocamos experiências, ideias, sonhos, medos... cada dia no laboratório foi mágico. Agradeço a todos pelos momentos maravilhosos.

Cy e Carol, passamos momentos fantásticos juntas, jamais vou esquecer... congressos, viagens, trabalhos, festinha, jogos... sempre dividimos tudo e participamos das conquistas, só posso agradecer e desejar muito sucesso, realizações, paz, amor e saúde sempre. Sei que temos muito ainda para compartilhar.

Quero agradecer de forma especial ao Wélliton de Souza meu estagiário de Bioinformática desde 2010 que me ajudou nesse trabalho e tem me auxiliado em muitas outras tarefas do laboratório com maturidade. Por seu esforço e dedicação a este e outros trabalhos merece receber grandes conquistas.

Também quero lembrar de dois grandes mestres de bioinfo, Arthur Gruber e Alan Mitchell Durham da USP de São Paulo, que me ensinaram grande parte do que sei hoje.

Agradeço a todos por existirem e por fazerem parte da minha vida.

Ao Instituto de Biociências da UNESP de Rio Claro, pela infraestrutura e apoio durante o doutorado.

À Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), pela bolsa de doutorado e auxílios concedidos durante esses anos.

"Aprender significa entender que a vida
não começa quando nascemos.
Outros estiveram aqui antes
e caminhamos sobre suas pegadas.
O que lemos foi escrito por gerações de pais e filhos,
mães e filhas, professores e alunos.
Somos a soma de suas experiências e de suas buscas."

"A vida é a soma de suas escolhas."

Eliécer Wiesel e Albert Camus

Resumo

Resumo

Ferro M. Desenvolvimento de *Pipelines* para Análise de EST, Bibliotecas 16S e ITS para Estudos Genômicos em Formigas Attini [tese]. Rio Claro: Instituto de Biociências da Universidade Estadual Paulista “Júlio de Mesquita Filho”, 2013.

As formigas Attini são destacadas pragas agrícolas e invertebrados modelos em estudos em genômica, metagenômica, sistemática e evolução molecular. Tais estudos têm gerado milhares de sequências nucleotídicas, cuja análise demanda o uso de diferentes programas de Bioinformática em processamentos automatizados. Pensando nisso, foram desenvolvidos protocolos para análises de ESTs e de bibliotecas de sequências ITS e 16S obtidas de formigas Attini. Foi desenvolvida uma arquitetura baseada no padrão MVC e J2EE responsável por todos os processos cliente-servidor. Os protocolos foram implementados na forma de *pipelines* em que cada componente é um serviço *web* em REST acoplado a arquitetura desenvolvida. *bESTscan* realiza a anotação de sequências EST transcriptômicas, *16Scan* e *ITScan* realizam análises de ecologia microbiana, para bactérias (sequências 16S) e para fungos (sequências ITS) respectivamente, bem como identifica as sequências através de pesquisas em bancos de dados públicos. O sistema computacional desenvolvido automatiza a anotação de sequências para um pequeno volume de sequências como as obtidas por Sanger ou para grande volume de dados como as geradas por Sequenciadores de Nova Geração, reduzindo o tempo de processamento e facilitando a análise dos resultados. Todos os *pipelines* desenvolvidos foram validados com sequências de Attini, obtidas por Sanger e Illumina, geradas a partir de diferentes projetos do nosso laboratório e estão disponíveis na *web* nos endereços http://evol.rc.unesp.br/lem/?q=bioinformatics.

Palavras chaves: Bioinformática. Processamento encadeado. Transcriptoma. Anotação de sequências. Formigas Attini.

Abstract

Abstract

Ferro M. Pipeline Developing for Analysis of ESTs, ITS and 16S libraries for genomic studies in Attini ants [thesis]. Rio Claro: Instituto de Biociências da Universidade Estadual Paulista “Júlio de Mesquita Filho”; 2013.

The Attini ants are agricultural pests and invertebrate models for studies on genomics, metagenomics, molecular systematics and evolution. These studies have generated thousands of nucleotide sequences whose analysis requires the use of different Bioinformatics tools and automated processing. We developed automated sequence annotation protocols for ESTs and ITS and 16S libraries analysis for Attini ants. We developed an architecture model based on J2EE and MVC patterns which is responsible for all client-server processes and each tool in the pipeline is a REST web service coupled in the architecture model developed. The *bESTscan* pipeline to perform the EST transcriptome annotation, *16Scan* e *ITScan* realize microbial ecology studies for both, bacteria (16S sequences) and for fungi (ITS sequences), as well as identify sequences through public databases. The computer system developed automates the annotation for both a small volume of sequences obtained by Sanger and for a large number of data generated by the Next Generation Sequencers (NGS), reducing processing time and with high performance. Pipelines were developed and validated using sequences Sanger and Illumina generated with different projects from our laboratory and are available at http://evol.rc.unesp.br/lem/?q=bioinformatics.

Keywords: Bioinformatics. Pipeline. Transcriptome. Sequence Annotation. Attini Ants.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API - *Application Programming Interface* (Interface de Programação de Aplicativos)

BLAST - *Basic Local Alignment Search Tool* (Ferramenta de busca de alinhamento local básica)

CAP3 - *Contig Assembly Program* (Programa de Montagem de Contigs)

CDs - *Coding Sequence* (Sequência Codificadora)

CPU - *Central Processing Unit* (Unidade Central de Processamento)

DNA - Deoxyribonucleic Acid (Ácido desoxirribonucleico)

cDNA - *Complementary DNA* (DNA complementar)

CUDA - *Compute Unified Device Architecture* (Arquitetura Unificada de Dispositivo de Computação)

EC - *Enzyme Comission* (Comissão de enzimas)

EST - E*xpressed Sequence Tags* (Etiquetas de Sequências Expressas)

GO - *Gene Ontology* (Ontologia Gênica)

GPU - *Graphic Processing Units* (Unidades de Processamento Gráfico)

HTML - *HyperText Markup Language* (Linguagem de Marcação de Hipertexto)

ITS - *Internal Transcribed Spacer* (Espaçador Transcrito Interno)

JSON - *JavaScript Object Notation* (Notação de Objeto JavaScript)

LSU - *Large Subunit* (Subunidade Maior do Ribossomo)

MPI - *Message Passing Interface* (Interface de Passagem de Mensagem)

NTS - *Non-Transcribed Spacer* (Espaçador Não Transcrito)

NCBI - *National Center for Biotechnology Information* (Centro Nacional de Informação Biotecnológica)

NGS - Next Generation System (Sequenciadores de Nova Geração)

OpenMP - *Open Multiprocessing* (Multiprocessamento Aberto)

OMG - *Object Management Group* (Grupo de Gerenciamento de Objetos)

ORF - O*pen Reading Frame* (Fase aberta de leitura)

OTU - *Operational Taxonomic Unit* (Unidade Taxonômica Operacional)

PC - *Personal Computer* (Computador Pessoal)

RNA - *Ribonucleic acid* (Ácido Ribonucleico)

REST- *Representation State Transfer* (Transferência de Estado Representativo)

SSU - *Small Subunit (Subunidade Menor do Ribossomo)*

mRNA - messenger RNA (RNA mensageiro)

URL - *Uniform Resource Locator* (Localizador Uniforme de Recursos)

UML - *Unified Modeling Language* (Linguagem de Modelagem Unificada)

VM - *Virtual Machine* (Máquina Virtual)

XML - *eXtensible Markup Language* (Linguagem de Marcação Estendida)

LISTA DE SÍMBOLOS

GB – Gigabytes

pb - pares de bases

TB - Terabytes

SUMÁRIO

Introdução

1 INTRODUÇÃO

1.1 Formiga Cortadeira *Atta laevigata*

As formigas da tribo Attini compreendem um grupo com mais de 230 espécies descritas, todas obrigatoriamente dependentes do cultivo de fungos mutualistas para sua alimentação, em câmaras presentes nos ninhos (WEBER, 1972; HÖLLDOBLER; WILSON, 1990; SCHULTZ & BRADY, 2008). Dentre estas formigas, *Atta* e *Acromyrmex* destacam-se por sua grande herbivoria, cortando e utilizando folhas como substrato para seu fungo mutualista. Devido a tal habito, esses insetos são conhecidos como formigas cortadeiras (WETTERER *et al*., 1998).

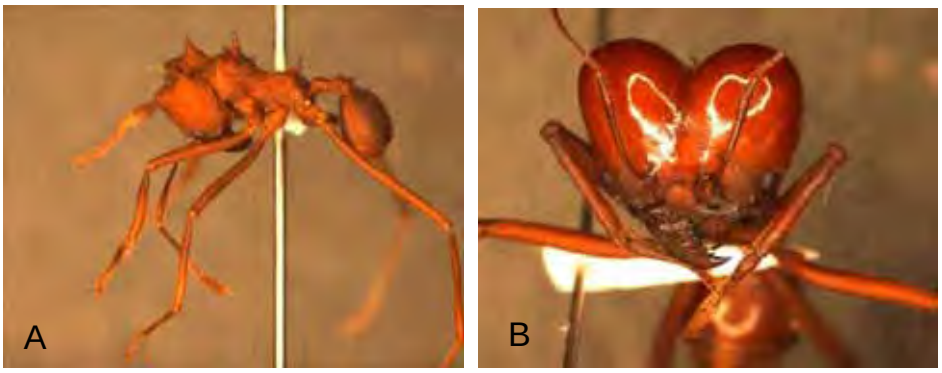
As formigas do gênero *Atta* compreendem aproximadamente 19 espécies (BOLTON, 2006) e são conhecidas como saúvas. As operárias são caracterizadas por três pares de espinhos no dorso do tronco e a superfície dorsal do gáster lisa, sem tubérculos (ANJOS *et al*., 1998). Elas se distribuem geograficamente desde o Sul dos Estados Unidos até o Norte da Argentina (HÖLLDOBLER; WILSON, 1990).

As formigas cortadeiras apresentam uma grande importância ecológica devido à associação com diferentes organismos como, por exemplo, fungo mutualista (WEBER, 1972; MUELLER *et al*., 2001) e a bactéria *Pseudonocardia* (CURRIE *et al*., 1999). Também fazem a aeração do solo e criam passagens para drenagem e penetração de água (WEBER, 1972), realizam a poda da vegetação e ciclagem de nutrientes ao levarem matéria orgânica para os ninhos subterrâneos (HÖLLDOBLER; WILSON, 1990) e trazerem minerais presentes nas partes mais profundas do solo para a superfície (WEBER, 1972).

São responsáveis pelo aumento no número de plantas que crescem ao redor e sobre os ninhos, o que contribui para a manutenção da biodiversidade em algumas situações (GARRETSON *et al*., 1998; WIRTH *et al*., 2003).

Há um grande polimorfismo na classe operária com divisão de tarefas. As operárias maiores (soldados) são especializadas na defesa da colônia, as operárias médias cortam e carregam as folhas, removem dejetos e indivíduos mortos e escavam os ninhos, enquanto as operárias menores processam as folhas, cuidam dos jardins de fungo e da prole (WEBER, 1972; WILSON, 1980). As larvas se alimentam do micélio do fungo mutualista, o que é importante para seu crescimento (BASS; CHERRET, 1995). A Figura 1 apresenta uma operária e um soldado de *Atta laevigata*.

Figura 1 – *A. laevigata.* A – Operária. B – Soldado. (Fotos: Joaquim Martins Júnior, disponíveis em http://omega.rc.unesp.br/formiga).



Os ninhos maduros são compostos por até oito milhões de operárias e uma única rainha. Como a colônia apresenta muitos indivíduos, a quantidade de matéria vegetal fresca (folhas, flores, frutos) consumida é enorme e supera o que é consumido por outros herbívoros (HÖLLDOBLER; WILSON, 1990). Muitas das espécies de *Atta* são consideradas pragas agrícolas por causarem prejuízos à agricultura (WEBER, 1972; FOWLER *et al*., 1986; CAMERON, 1985). Além disso, estas podem explorar uma grande variedade de espécies de plantas

(VASCONCELOS, 1990), alcançar altas densidades populacionais (FOWLER *et al*., 1986) e produzir rainhas longevas capazes de botar ovos constantemente por mais de 15 anos (KELLER, 1998).

1.2 Sequenciamento e Genoma

Embora exista um crescimento exponencial de sequenciamentos e vários insetos já tenham seu genoma determinado como a mosca de fruta (*Drosophila melanogaster* – ADAMS *et al*., 2000*),* abelha melífera (*Apis mellifera* – THE HONEYBEE GENOME SEQUENCING CONSORTIUM, 2006*)*, bicho da seda (*Bombyx mori* – MITA *et al*., 2004*),* vespas parasitóides (*Nasonia* – THE *NASONIA* GENOME WORKING GROUP, 2010) e o besouro castanho (*Tribolium castaneum)*, não se conhece até o presente momento o genoma completo de qualquer formiga, sendo que a única espécie de praga agrícola sequenciada e disponível é a do besouro *Tribolium castaneum*.

No caso das formigas, esses estudos se limitam devido à falta de bancos de sequências e de ferramentas moleculares (WANG *et al*., 2007). Algumas das sequências existentes para esse grupo são derivadas de um estudo com algumas EST para *Solenopsis invicta* (TIAN *et al*., 2004) e *Camponotus festinatus* (GOODISMAN *et al*., 2005).

1.3 Sequências EST e Transcriptoma

Uma biblioteca de cDNA é uma coleção de clones contendo regiões transcritas do genoma, sendo, por isso, menor que uma biblioteca genômica. Este cDNA é sintetizado no laboratório a partir de um mRNA molde e, portanto, não contém íntrons. A ação da transcriptase reversa leva a síntese de uma molécula de

DNA molde (GRIFFITHS *et al*., 1998). As moléculas de cDNA são ligadas a vetores e cada clone determina uma sequência única. A abundância de transcritos de cada gene é usada para se estimar a expressão gênica em uma determinada célula ou tecido (BOUCK; VISION, 2007).

As sequências EST são obtidas a partir de bibliotecas de cDNA de um tecido ou estágio evolutivo de um organismo e permitem a caracterização de um subconjunto de genes expressos (ADAMS *et al*., 1991). A análise de ESTs possui diversas aplicações biológicas, biotecnológicas, ecológicas e evolutivas, constituindo um dos tipos mais diversos e abundantes de informações disponíveis em bancos de dados moleculares (BOUCK; VISION, 2007).

O transcriptoma refere-se ao conjunto de RNAs mensageiros (mRNAs) ou transcritos, que são expressos em um organismo em um determinado momento. Reflete a população de genes ativos naquele momento, que pode variar conforme as diferentes condições como o ambiente, o estágio de desenvolvimento, tecido de origem, entre outros.

Como as formigas *Atta laevigata* são pragas agrícolas, a obtenção e análise do transcriptoma pode fornecer importantes informações sobre metabolismo, genes regulatórios e proteínas e enzimas expressas, abrindo imensas possibilidades para a identificação de alvos moleculares para o controle de cortadeiras pragas agrícolas.

1.4 Fungos e sua Biodiversidade

Os fungos são organismos eucariotos pertencentes ao Reino Fungi, que se distribuem mundialmente e se desenvolvem em diferentes habitats. Destacam-se por (1) serem decompositores na maioria dos ecossistemas terrestres, (2) possuírem relações de simbiose com outros organismos, (3) serem patógenos de muitos tipos

de plantas e animais, (4) oferecerem sistemas genéticos bem desenvolvidos servindo como modelos para estudos e (5) serem essenciais em processos de fermentação e em indústrias de biotecnologia (BRUNS *et al*., 1991).

Segundo Petti e colaboradores (2005), os métodos tradicionais para a identificação dos micro-organismos exigem reconhecer as diferenças através da morfologia, atividade enzimática, metabolismo e crescimento. No entanto, nem sempre é possível verificar todas essas características.

Assim como ocorre para a maioria dos micro-organismos, também é difícil para os fungos serem cultivados por técnicas padrões de cultivo em laboratório. Assim, tem-se utilizado técnicas de sequenciamento de DNA para estudos de biodiversidade.

O sequenciamento de DNA extraído diretamente do ambiente tem se mostrado uma ferramenta poderosa para a caracterização da diversidade de fungos, já que grande parte das espécies de fungos não cresce nos meios de cultura. Usando essas sequências é possível se identificar espécies raras e determinar associações em uma comunidade microbiana (SCHOCH *et al*., 2012).

Atualmente, muitos estudos de biodiversidade de fungos são feitos usando-se a região ITS (*Internal Transcribed Spacer*) do DNA ribossômico (rDNA) (FONTANA *et al*., 2012;. ORT *et al*., 2012).

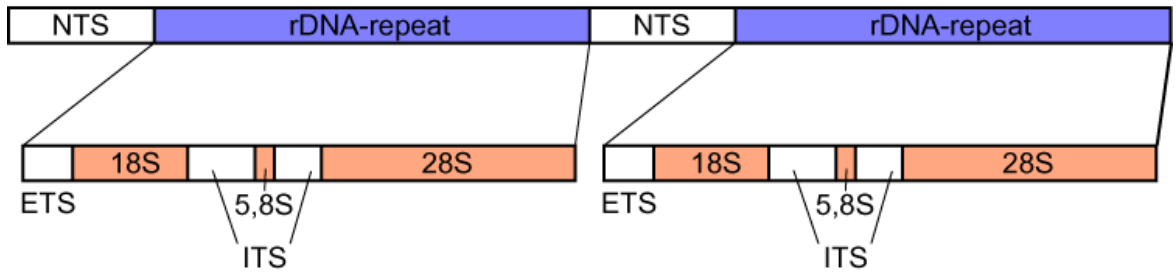
1.5 Sequências ITS

O ribossomo é formado por duas unidades, sendo uma subunidade menor (SSU) e uma subunidade maior (LSU). No caso de fungos, o DNA ribossômico (rDNA) que codifica o RNA ribossômico (rRNA), constitui-se de um cluster de genes (18S, 5,8S e 28S) . Como esse cluster de genes aparece repetidas vezes no

genoma de fungos, essas regiões podem ser amplificadas e sequenciadas com facilidade.

Os genes no rDNA são interpostos por espaçadores denominados ITS1 e ITS2 que são segmentos curtos (500 a 800 pb) que são transcritos e processados para originar o rRNA maduro. A Figura 2 apresenta um esquema da estrutura do rDNA que aparece em duas cópias entre os espaçadores não-transcritos (NTS), com seus respectivos genes e regiões ITS.

Figura 2 – Estrutura do rDNA eucariótico (fungos) com repetições do cluster gênico. Fonte: http://www.absoluteastronomy.com/topics/Ribosomal_DNA.



Os genes 18S e 28S, bem como a região ITS, apresentam diferentes taxas de evolução de suas bases. Assim, o gene 18S é mais conservado e pode ser utilizado na comparação de organismos menos relacionados. Já o 28S é mais variável e, portanto apropriado para comparar diferentes espécies ou gêneros. As regiões ITS são usadas para espécies relacionadas ou ainda variedades em uma mesma espécie, porque evoluem mais rapidamente (BELLEMAIN *et al*., 2010).

A região ITS foi recentemente proposta como um marcador padrão para a identificação molecular de fungos e avaliação da diversidade de fungos (SCHOCH *et al*., 2012). Além disso, está sendo usada como um marcador universal para *barcoding* molecular de fungos (SCHOCH *et al*., 2012). Um banco de dados de sequências *barcode* para fungos está disponível em http://www.fungalbarcoding.org/.

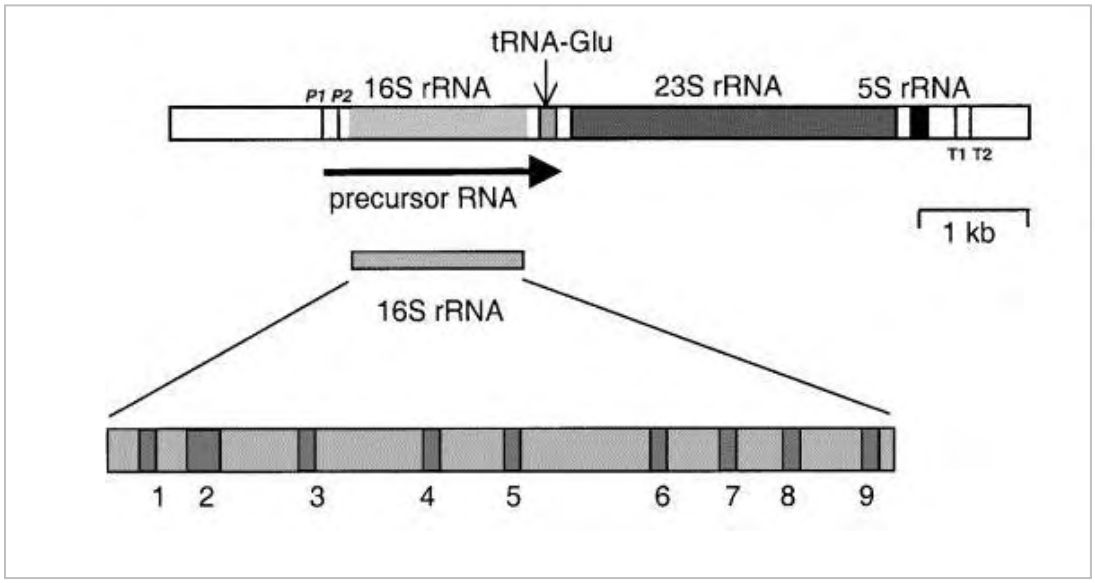
1.6 Bactérias e sua Diversidade

As bactérias compreendem um domínio da vida, constituída por micro-organismos procariontes e unicelulares, anaeróbicos ou aeróbicos. Possuem diferentes formas e podem se agrupar formando colônias. Apresentam importância (1) ecológica como decompositoras e fixadoras de nitrogênio na indústria alimentícia, (2) econômica em indústrias de bebidas, alimentícia e farmacêutica, (3) como patógenos e (4) por serem simbiontes de algumas espécies de plantas e animais.

1.7 Sequências 16S

O RNA ribossômico 16S, com aproximadamente 1500 bases, é um componente da subunidade menor do ribossomo 30S de bactérias. Esta subunidade é codificada pelo *operon* rRNA, que está localizado no DNA genômico bacteriano A Figura 3 apresenta a região do DNA genômico contendo o *operon* rRNA.

Figura 3 – Estrutura do rRNA bacteriano com suas regiões: promotores (P1-P2), terminadores (T1 e T2), gene 16S e regiões hipervariáveis (1 - 9). Adaptado de: http://www.sciencedirect.com/science/article/pii/S0924224400000303.



Conforme pode ser observado na Figura 3, o gene 16S contém regiões hipervariáveis denominadas de V1 até V9. Essas demonstram uma considerável diversidade de sequências entre espécies distintas e, portanto, podem ser usadas na identificação dessas espécies (VAN DE PEER *et al*., 1996).

Muitos estudos recentes (FONTANA *et al*. 2012, WESTERMAN *et al*., 2012), inclusive para micro-organismos não cultiváveis, têm utilizado a região 16S para identificar gêneros de bactérias presentes em amostras, bem como fazer estudos filogenéticos.

1.8 Análises de Sequências: Pré-processamento e Anotação

Todas as sequências geradas sejam pelo método de Sanger, seja por NGS, precisam passar por um pré-processamento antes de serem analisadas. Isso pode ser demonstrado usando-se as sequências EST. Sabe-se que essas sequências são obtidas a partir de bibliotecas de cDNA e que permitem caracterizar um subconjunto de genes expressos (ADAMS *et al*., 1991). No entanto, essa informação contida nos ESTs geralmente apresenta-se de forma fragmentada e redundante, já que vários ESTs podem representar um mesmo gene e podem estar em inúmeras cópias. Para organizar essa informação, é necessário agrupar as ESTs em *clusters*, cada um deles representado por uma sequência consenso. Esse processo de agrupamento e montagem de ESTs recebe o nome de reconstrução de transcritos (GRUBER, 2007).

As etapas citadas acima fazem parte do que denominamos pré-processamento. Nessa fase, as sequências geradas passam por filtragem por qualidade, mascaramento contra possíveis contaminantes como, por exemplo, vetor e *primer*, e finalmente a montagem dos clusters. Após o término dessa etapa, são usados programas computacionais específicos para se fazer a anotação.

A anotação de sequências biológicas consiste na atribuição de características que descrevem tais sequências (STEIN, 2001). É importante ressaltar que tem sido cada vez mais frequente em anotações, o uso de vocabulários controlados como o da ontologia gênica (GO) para a padronização de termos disponíveis em três ontologias distintas: processo biológico, componente celular e função molecular (GENE ONTOLOGY CONSORTIUM, 2006). No entanto, a análise de um grande conjunto de sequências exige sistemas automatizados que processem cada uma das sequências de forma contínua e as submetam às mesmas ferramentas para análise.

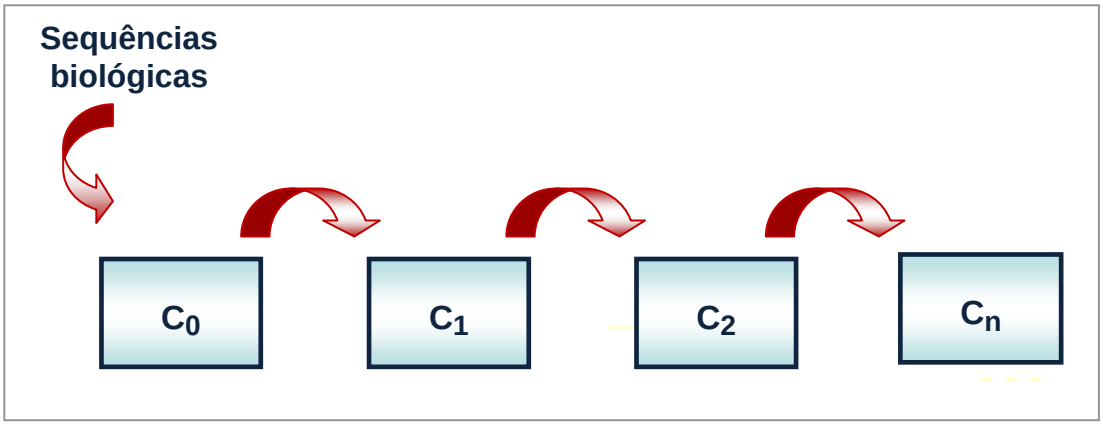
É importante ressaltar que uma etapa fundamental na análise de qualquer tipo de sequência refere-se a elaboração de um protocolo de anotação específico, onde são escolhidas as ferramentas conforme as especificidades das sequências e a ordem das etapas envolvidas no processamento. Nessa fase, devem ser consideradas as características biológicas das sequências em estudo. Para preparar o protocolo de anotação de ESTs, foi usado um modelo de um protocolo de anotação para ESTs desenvolvido no Laboratório da USP de São Paulo (FERRO, 2008). No caso de ITS e 16S, foram desenvolvidos protocolos baseados nas necessidades de anotação dos projetos do nosso laboratório.

1.9 Sistemas de *Pipelines*

Com o advento dos Sequenciadores de Nova Geração (METZKER, 2010), houve um aumento da quantidade de sequências biológicas geradas. Assim, há a necessidade de se implementar sistemas computacionais capazes de lidar com grandes lotes de sequências e em um curto intervalo de tempo. Para se analisar tais sequências devem ser usadas diferentes ferramentas ou componentes de

Bioinformática em um fluxo contínuo e ordenado, de forma que os dados de saída de cada componente sejam a entrada para o processamento da próxima etapa. Essa estrutura computacional recebe o nome de *pipeline* e a Figura 4 apresenta um modelo ilustrativo de como seria essa estrutura.

Figura 4 - Esquema ilustrativo de um *pipeline*, onde cada ícone representa um programa em um fluxo de processamento pré-determinado pelo usuário.



Através de sistemas de *pipelines* é possível processar de forma eficiente uma grande quantidade sequências, tanto nas etapas de pré-processamento quanto na anotação. Geralmente em um *pipeline*, os resultados são integrados de forma não visível ao usuário, facilitando ao final do processo a visualização e a análise de dados obtidos em cada etapa do *pipeline*.

Após a execução de um componente individualmente ou do *pipeline* completo, é comum realizar um processo denominado validação. Nessa etapa, são feitos vários testes de execução, sendo que os resultados de cada componente e do *pipeline* completo são analisados de forma manual, verificando se estão corretos e compatíveis com o esperado.

1.10 Ferramentas para Pré-processamento e Anotação de Sequências

Atualmente existe uma série de sistemas automatizados para pré-processamento e anotação de sequências biológicas, como GARSA (DÁVILA *et al.*, 2005), annot8r (SCHMID; BLAXTER, 2008), EST Express (SMITH *et al.*, 2008), EGene (DURHAM *et al.*, 2005) e dCAS (GUO *et al*., 2009).

Para análise de sequências metagenômicas baseada em bibliotecas 16S e ITS, algumas das ferramentas disponíveis são: DOTUR (SCHLOSS; HANDELSMAN, 2004), UniFrac (LOZUPONE; KNIGHT, 2005), TreeClimber (SCHLOSS; HANDELSMAN, 2006a) e SONS (SCHLOSS; HANDELSMAN, 2006b), além dos programas FastGroup (SEGURITAN; ROHWER, 2001) e uma versão mais recente FastGroupII (YU *et al*., 2006).

Essas ferramentas citadas acima funcionam muito bem para um pequeno volume de sequências como no caso de dados gerados por Sequenciamento Sanger. No entanto, quando se trabalha com milhões de sequências, como as geradas pelos NGS (Illumina, 454 e SOLID), as ferramentas utilizadas são outras. No caso da montagem de sequências, para cada tipo de sequenciador é necessário se utilizar uma ferramenta diferente. A Tabela 1 apresenta algumas das ferramentas utilizadas para montagem de sequências na etapa de pré-processamento para um grande volume de dados.

No caso dos sistemas para anotação de sequências não há mais a preocupação em saber qual o tipo de sequenciador de nova geração usado na geração das sequências, pois os dados de entrada são sempre sequências (contigs, singlets ou reads) que já passaram pelo pré-processamento.

Tabela 1 – Ferramentas para pré-processamento de grande quantidade de sequências advindas de NGS.

Ferramenta de Bioinformática	Illumina	454	SOLID	Referência
Velvet/Oases	X		X	ZERBINO; BIRNEY, 2008
Newbler		X		Roche
MIRA		X		CHEVREUX *et al*., 2004
SOAPdenovo	X		X	LUO *et al*., 2012
AbySS	X		X	SIMPSON *et al*., 2009

No caso da anotação de ESTs há ferramentas como BLAST2GO (CONESA *et al*., 2005) e FastAnnotator (CHEN *et al*., 2012) que podem lidar com grande volume de dados. No caso de análise de sequências ITS, recentemente foi lançada a ferramenta CloVR-ITS (WHITE *et al*., 2013) que realiza o pré-processamento, montagem, detecção de quimeras, gera OTUs e índice de diversidade alfa e realiza a classificação usando BlastN. Porém, esse pipeline precisa rodar em uma Máquina Virtual (VM), localmente ou em um Servidor nas Nuvens (*Cloud Computing*).

1.11 Serviços web ou *Web Services*

De acordo com o *world wide web consortium*[1] (W3C), um *web service* deve ter uma interface bem definida, descrevendo os recursos disponíveis em um formato padrão e também prover um meio de comunicação usando o protocolo padrão da *web* (HTTP), ou seja, um *web service* pode ser compreendido como um sistema de informação distribuído, projetado para dar suporte a interoperabilidade entre máquinas em um ambiente de rede (KALIN, 2009; RICHARDSON; RUBY, 2007).

[1] Disponível em http://www.w3.org/TR/ws-arch/

Katayama e colaboradores (2010) sustentam que há duas categorias principais de *web services* em Bioinformática: acesso a dados e análise. O primeiro trata de bases de dados de acesso público, tais como NCBI, DDBJ e KEGG. A segunda categoria refere-se a serviços que exigem grandes processamentos, algumas vezes com complexas estruturas de dados de entrada e saída, como ocorre com o BLAST (ALTSCHUL *et al*., 1997). Além do tipo de categoria empregada para o desenvolvimento de um *web service* em Bioinformática, deve-se considerar os estilos de desenvolvimento do *web service*, que pode ser SOAP *Web Services* ou REST *(Representation State Transfer).*

REST tem se tornado uma alternativa a abordagem de desenvolvimento de *web services* SOAP. Na abordagem REST, um serviço publicado na *web* é decomposto em recursos identificáveis únicos e sem estados, de tal forma que estes possam ser chamados como uma URL. A vantagem do REST sobre SOAP *Web Services* diz respeito a sua simplicidade de construção via HTTP, não exigindo o uso de bibliotecas extensas e complexas, bem como de técnicas sofisticadas de decodificação (*parsers*) (KATAYAMA *et al*., 2010).

1.12 Desempenho Computacional

Multithreads

Cada vez é maior a preocupação na área de Bioinformática em se desenvolver ferramentas que consigam ter um melhor desempenho o que inclui menor tempo de processamento e capaz de lidar com grande volume de dados. Trabalhos da área computacional que comparam o desempenho de aplicações usando múltiplas *threads* mostraram um acentuado ganho de *performance*. Isso

pode ser observado no trabalho utilizando *JavaParty* como mecanismo de controle nativo de *multithreads* (GUO *et al*., 2008). Nesse trabalho os autores mostram que a utilização de um ambiente de *threads* da linguagem Java (KALIN, 2009) aumenta a eficiência de métodos heurísticos para busca e alinhamento de sequências com os programas BLAST, FASTA (PEARSON, 2004) e Smith-Watermam (SMITH; WATERMAN, 1981) combinado com a programação dinâmica em um ambiente distribuído.

Paralelismo

A Lei de Moore mostrou que aumentar o número de transistores também aumentaria o desempenho computacional. Assim por anos os profissionais da área trabalharam em programas sequenciais e construíram processadores com velocidades mais altas. Entretanto, cada vez mais arquiteturas complexas de diversas áreas têm atingido o limite computacional. A solução para esse limite computacional tem sido aumentar o número de processadores (ou núcleos) ao invés de aumentar a velocidade em um único núcleo. A utilização de processadores com vários núcleos (*multicore*) altera o modelo de programação, expondo o paralelismo ao programador. Os modelos de programação paralela comumente utilizada são propensos a reproduzir erros sutis (*bugs*). Esses modelos fazem uso de modelos de memórias complexas dificultando a implementação de novas aplicações. As dificuldades encontradas no modelo de programação paralela fazem com que o número de profissionais efetivos seja pequeno e o investimento baixo.

Segundo Blaise Barney, autor do guia “Introdução a Computação Paralela” disponível em https://computing.llnl.gov/tutorials/parallel_comp/, a computação paralela é um paradigma que refere-se ao uso simultâneo de múltiplos recursos

computacionais para resolver um problema. Esses recursos podem ser várias CPUs (*Central Processing Unit*) em uma ou mais máquinas. Assim, um problema pode ser quebrado em partes menores que podem ser resolvidas de forma paralela, na qual cada parte realiza uma série de instruções executadas simultaneamente em diferentes CPUs.

A computação paralela tem sido utilizada com sucesso em diferentes domínios do conhecimento, como a computação de alto desempenho, servidores, placas gráficas e sistemas embarcados. A Bioinformática é uma área que tem se beneficiado com os esforços no desenvolvimento de aplicações concorrentes.

1.13 Métodos de Avaliação de Desempenho

Speedup

Segundo Grama e colaboradores (2003), *speedup* é definido pela razão entre o tempo decorrido durante a execução de um programa em um único processador (T_1), e o tempo de execução quando *n* processadores estão disponíveis (T_n). Para resultados iguais ou abaixo de 1, significa que o programa concorrente, em termos de desempenho, é equivalente ou pior em relação ao programa serial, enquanto resultados acima de 1 mostram que há aumento de desempenho. A Equação de *Speedup* (1) está representada pela fórmula:

$$S(n) = \frac{T_1}{T_n} \qquad (1)$$

onde S(n) é o aumento de desempenho, T1 é o tempo da execução serial e Tn é o tempo da execução concorrente do programa.

Eficiência

Eficiência é definida como a razão entre o resultado do cálculo de *speedup* e a quantidade de elementos de processamento utilizado. Elementos de processamento podem ser CPUs, núcleos, processos, *threads*, etc. Quando o resultado do cálculo mantém-se igual a 1 ao adicionar mais elementos de processamento, significa que o aumento de desempenho é linear, que é o caso ideal. Geralmente, o *speedup* linear não é obtido porque, juntamente com um aumento de desempenho vem uma diminuição na eficiência: quanto mais elementos de processamento são dedicados à execução de um *software*, espera-se um aumento na quantidade de tempo ocioso do processador, devido a fatores como a contenção, comunicação e estrutura do *software*. A Equação de Eficiência (2) está apresentada pela fórmula abaixo.

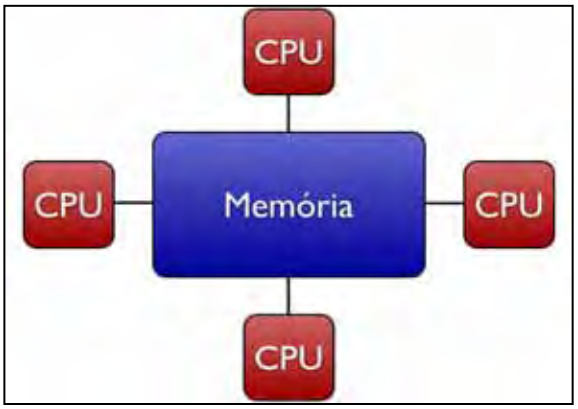
$$E(n) = \frac{S(n)}{n} \qquad (2)$$

onde E(n) é a eficiência do programa, S(n) é resultado do cálculo de *speedup* e n é a quantidade de elementos de processamento.

1.14 Arquiteturas na Computação Paralela

Ainda segundo Blaise Barney, na computação paralela há duas arquiteturas de memória, sendo que cada uma apresenta um modelo de programação. Também existem arquiteturas híbridas. Na arquitetura de Memória Compartilhada, todos os processadores têm a habilidade de acessar toda a memória como espaço de endereço global. Como a memória é compartilhada, as alterações na memória efetuada por um processador são visíveis a todos os processadores (Figura 5).

Figura 5 – Arquitetura de Memória Compartilhada.

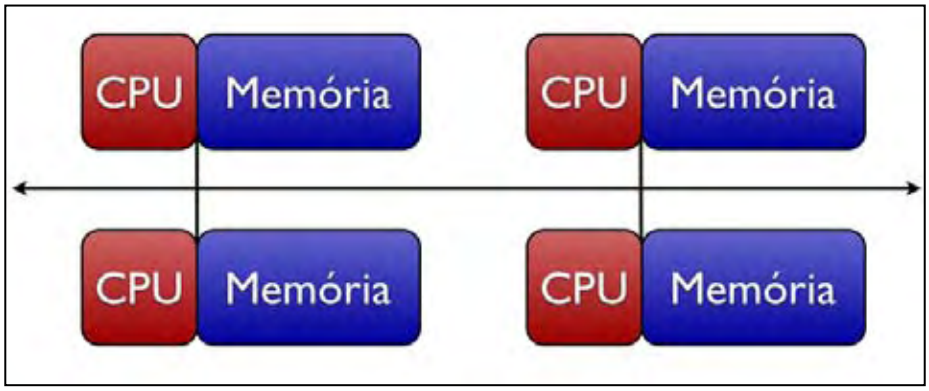


O espaço de endereçamento global oferece uma perspectiva de programação amigável ao usuário do acesso à memória. Entretanto, a principal desvantagem desse modelo de arquitetura é a falta de escalabilidade entre memória e CPUs. A adição de mais CPUs pode aumentar geometricamente o tráfego no caminho entre memória compartilhada e CPU. Para sistemas com Coerência de *Cache* (um processador atualiza uma região da memória compartilhada e todos os outros processadores irão saber sobre a atualização), aumenta geometricamente o tráfego associado com o gerenciamento de cache/memória. Essa desvantagem torna cada vez mais difícil e caro projetar e produzir máquinas com memória compartilhada com um número cada vez maior de processadores. Um exemplo de modelo de programação disponível que utiliza memória compartilhada é a API (*Application Programming Interface*) OpenMP (*Open Multiprocessing*).

A outra arquitetura é a Memória Distribuída, em que os processadores possuem memória local própria, ou seja, o endereço de memória não é mapeado para outro processador; então nesse caso não há o conceito de espaço de endereçamento global para todos os processadores. Por possuírem memória local própria, os processadores operam de forma independente. Alterações feitas na

memória local não afetam a memória de outro processador (Figura 6). Por isso, o conceito de Coerência de *Cache* não é aplicável a essa arquitetura.

Figura 6 – Arquitetura de Memória Distribuída.



Quando um processador precisa acessar os dados de outro processador, o programador define como e quando comunicar-se. A sincronização entre tarefas é também responsabilidade do programador. Nesse modelo, a memória é escalável com o número de processadores e, dessa forma, pode-se aumentar o número de processadores e tamanho de memória proporcionalmente. Cada processador pode rapidamente acessar sua própria memória sem interferência e sem despesas incorridas com a tentativa de manter a coerência de *cache*. Um exemplo de especificação para esse modelo de memória é o MPI (*Message Passing Interface*).

Existe também a arquitetura híbrida, que é uma rede com múltiplas máquinas de memória compartilhada, que conhecem apenas a própria memória, mas não a memória de outras máquinas.

Uma característica em comum entre as arquiteturas de memória é a dificuldade em projetar e desenvolver sistemas massivamente paralelos e eficazes. Alguns livros e tutoriais apresentam os problemas encontrados na computação paralela e os meios de evitar essas deficiências.

Problemas comuns na computação paralela

Segundo Subramaniam (2011) existem três problemas que devem ser evitados em programas concorrentes:

- **Inanição (*Starvation*):** ocorre quando uma linha de execução (*thread*) espera por um evento que pode demorar muito para acontecer (ou nunca acontecer). Para solucionar esse problema utiliza-se tempo máximo de espera, que então pode buscar alternativas para continuar.
- **Impasses (*Deadlocks*):** ocorrem quando duas ou mais linhas de execução aguardam pela mesma ação ou recurso. Utilizar tempo de espera não ajuda a evitar impasses, pois é possível que cada thread vai pedir novamente o mesmo recurso, repetindo os passos, e ocorrendo impasses. A melhor alternativa seria evitar bloqueios explícitos e trabalhar com estados imutáveis.
- **Condições de corrida:** ocorre quando duas *threads* competem pelo uso de um mesmo recurso ou dados. Uma condição de corrida não acontece quando duas *threads* modificam os dados e sim quando uma linha de execução está alterando os dados enquanto a outra linha de execução está tentando ler. Condições de corrida podem tornar o comportamento de um programa imprevisível, produzir execução incorreta e produzir resultados incoerentes.

Para evitar esses problemas é necessário conhecer boas práticas de programação concorrente.

Boas práticas de Programação Concorrente

Ainda segundo Subramaniam (2011), para aumentar o desempenho de uma aplicação com técnicas de paralelismo, assegurando a consistência e precisão, é

necessário verificar se é possível dividir o problema em pequenas tarefas que podem ser executadas ao mesmo tempo. Depois de dividido o problema, deve-se identificar os trechos de execução que podem evitar o compartilhamento de objetos mutáveis, sendo dessa forma possível impedir que ocorram problemas como Impasses e Condições de Corrida. Posteriormente, deve-se implementar as funções de forma que não alterem os dados de entrada e assegurar que os objetos mutáveis sejam locais a linha de execução. Por último é importante identificar o número de *threads* necessárias para obter o melhor desempenho computacional e esta etapa deve levar em consideração o tipo do problema e os recursos computacionais disponíveis.

1.15 Técnicas de Programação Concorrente na Bioinformática

A ferramenta BLAST - *Basic Local Alignment Search Tool* compara sequências de nucleotídeos ou proteicas contra um banco de dados de sequências e calcula a significância estatística do alinhamento (KORF, 2003). Pode ser usado para inferir relações funcionais e evolutivas entre as sequências, assim como ajudar a identificar os membros de famílias de genes. Devido a sua importância, pesquisadores da área têm buscado técnicas de programação concorrente e estudado novas estratégias computacionais.

Camacho e colaboradores (2009) realizaram um trabalho que apresenta a descrição de etapas do programa BLAST com o objetivo de aumentar o desempenho da aplicação. Uma das estratégias utilizadas foi manter a tabela de busca e a matriz do processador em *cache*. A justificativa para essa estratégia é que a tabela e a matriz são utilizadas na maior parte do tempo na fase de rasterização

da sequência de busca e, dessa forma, é possível evitar a barreira de memória ao ler ou escrever dados que estão na memória principal e aumentar a velocidade de acesso.

ClustalW (THOMPSON *et al*., 1994) é uma ferramenta para alinhamento múltiplo de sequências proteicas ou de nucleotídeos. Em um trabalho (LI, 2003) foi desenvolvido uma versão distribuída e paralelizada do programa ClustalW utilizando MPI que roda em computadores paralelos e em ambientes distribuídos (*cluster*).

Diferente de programas como ClustalW e BLAST, MUMmer (DELCHER *et al*, 1999) é uma ferramenta para alinhamento de genomas inteiros. Como trata uma enorme quantidade de dados, tem-se buscado otimizar o programa a cada versão lançada. Schatz e colaboradores (2007) desenvolveram uma versão do MUMmer para executar em placas gráficas (*GPU*) que suportam a arquitetura CUDA (*Compute Unified Device Architecture*). Segundo Sanders e Kandrot (2010), CUDA é uma arquitetura pelo qual a NVIDIA constrói GPUs que podem executar tanto tarefas tradicionais de renderização de gráficos quanto tarefas de propósito geral.

Objetivos

2 OBJETIVOS

2.1 Objetivo Geral

Desenvolver protocolos para análise *in silico* de sequências EST advindas de projetos transcriptomas e sequências de bibliotecas ITS e 16S e implementá-los na forma de *pipelines*, onde cada ferramenta corresponde a um serviço *web* de alto desempenho que estará acoplada em uma arquitetura cliente-servidor.

2.2 Objetivos Específicos

- Pesquisar, instalar e testar programas para anotação de sequências, para ESTs, bibliotecas ITS e 16S;

- Desenvolver um modelo de arquitetura computacional capaz de receber diferentes *pipelines* de forma flexível;

- Elaborar protocolos para análise de sequências EST e bibliotecas 16S e ITS;

- Desenvolver cada componente de forma a ser um serviço *web* usando a tecnologia REST;

- Estudar técnicas de programação concorrente para melhorar o desempenho computacional dos componentes;

- Aplicar as técnicas de programação concorrente nos componentes;

- Desenvolver *pipelines* apropriados para a execução de cada protocolo, usando os componentes elaborados;

- Validar os *pipelines* desenvolvidos usando sequências da formiga *Atta laevigata* e outras espécies representativas de diferentes linhagens de Attini.

- Realizar uma curadoria manual verificando a qualidade e veracidade da descrição automatizada das sequências.

Material e Métodos

3 MATERIAL E MÉTODOS

3.1 Ambiente de Desenvolvimento

Tanto a arquitetura como os componentes e *pipelines* foram desenvolvidos em um PC com sistema operacional GNU Linux/Debian. As linguagens de programação utilizadas foram Perl (SCHWARTZ; PHOENIX, 2011), Python e Java.

Foram usadas as tecnologias REST (RICHARDSON; RUBY, 2007) e SOAP (SNELL *et al*., 2001) para a comunicação de serviço na *web*.

Para aplicação da técnica de programação concorrente foi usado o ambiente de desenvolvimento Eclipse (http://www.eclipse.org/).

Os testes de desempenho e estresse foram realizados em um servidor com 4 processadores *quad-core* de 2.40 GHz (16 núcleos disponíveis) com 32 GB de memória. Este utiliza o sistema operacional Linux Debian e o servidor de aplicação que está sendo utilizado é o Glassfish versão 3. Por padrão, o Glassfish limita o número de requisições simultâneas em 5, sendo que para o teste de estresse realizado esse valor foi alterado para 10. Testes também foram realizados em um notebook HP com 6GB de memória.

Para executar os testes de estresse foi utilizado o JMeter 2.6 (http://jakarta.apache.org/jmeter/). JMeter é uma ferramenta que realiza testes de carga e medições de desempenho em aplicações *web* para diferentes tipos de parâmetros e protocolos de rede. Para realizar os testes foram criados planos de teste, que contêm *Samplers* (amostras que realizam tarefas específicas como fazer uma requisição a um determinado recurso *web*) e

Listeners (ouvintes que capturam o comportamento dos *Samplers* e registram os dados em formatos específicos como tabelas e gráficos).

3.2 Ferramentas para Análise de Sequências

Para realizar a primeira etapa do desenvolvimento do *pipeline* para a análise de ESTs, foram instalados alguns sistemas automáticos para pré-processamento e anotação de sequências biológicas: (DÁVILA *et al.*, 2005), annot8r (SCHMID; BLAXTER, 2008), EST Express (SMITH *et al.*, 2008), EGene (DURHAM *et al.*, 2005), dCAS (GUO *et al*., 2009), BLAST2Go (CONESA *et al*., 2005) e FastAnnotator (CHEN et *al*., 2012).

Foram também instalados os aplicativos de terceiros requeridos pelos sistemas, como Phred/Phrap/Consed (EWING E GREEN, 1998; GORDON *et al*, 1998), CAP3 (HUANG; MADAN, 1999) e BLAST, entre outros.

Cada sistema instalado foi analisado quanto a existência de componentes prontos para o pré-processamento e anotação automática de sequências e a forma de relacionamento desses componentes, bem como os formatos de anotação disponíveis e a forma de armazenamento dos dados gerados, informações estas apresentadas na sessão Resultados.

3.3 Elaboração de Protocolos para Análise de Sequências

Inicialmente foi elaborado um protocolo de anotação com as tarefas necessárias e programas de terceiros requeridos para cada tipo de sequência a ser processada (EST, ITS e 16S).

Para anotação de ESTs, foi utilizado como modelo para desenvolvimento o protocolo desenvolvido no Laboratório de Coccídias da USP de São Paulo (FERRO, 2008).

3.4 Desenvolvimento da Arquitetura

Foi utilizado *multithreads* visando aumentar a *performance* e diminuir o tempo de processamento associados aos programas utilizados no Sistema de Anotação. Este procedimento deve resultar em melhorias em pontos críticos, como pesquisas em bancos de dados usando o BLAST e o programa InterProScan (QUEVILLON *et al*., 2005) que realiza pesquisa em bancos de proteínas para a busca de domínios de proteínas. Essas tarefas representam um gargalo de eficiência de processos de anotação de genomas (FERRO, 2008).

3.5 Desenvolvimento dos *Pipelines*

O núcleo do Sistema de Anotação foi elaborado baseado na arquitetura apresentada na Figura 2, usando a linguagem de programação Java. Os formatos de intercâmbio da informação foram em XML ou JSON.

Ao longo de todo o trabalho, foram desenvolvidos novos programas para a descrição de sequências conforme a necessidade de cada etapa do projeto. Na implementação do *pipeline* foi utilizada a linguagem de programação Perl e a interface do usuário foi feita utilizando a linguagem Java, ambos no sistema operacional Linux.

3.6 Anotação Automática e Curadoria Manual

Os protocolos desenvolvidos foram implementados na forma de *pipelines* e executados para as sequências em estudo. No caso de ESTs, foram usadas cerca de 30.000 EST geradas no Projeto FAPESP 2008/54386-9 intitulado “Filogenia Molecular, Genômica e Metagenômica em algumas Formigas Attini”, pela estudante Cynara M. Rodovalho. Esses dados foram utilizados para a realização de análises manuais (curadoria) para a detecção de possíveis erros e validação da anotação automática.

3.7 Desenvolvimento de *scripts*

Durante todo o projeto foi necessário desenvolver alguns pequenos programas computacionais. Esses *scripts* foram desenvolvidos conforme a necessidade em cada etapa do projeto e assim serão apresentados na seção de Resultados e Discussão.

Resultados e Discussão

4 RESULTADOS E DISCUSSÃO

4.1 Instalação de Ferramentas e Testes Iniciais

Como esse trabalho propôs o desenvolvimento de um sistema de pré-processamento e anotação de sequências, antes de iniciar o desenvolvimento foi preciso conhecer e testar alguns dos sistemas já existentes e verificar suas limitações, tanto em relação à parte computacional, já que novas tecnologias são lançadas constantemente, como também quanto aos resultados gerados.

Foi desenvolvido inicialmente um protocolo de pré-processamento para realizar a análises preliminares de limpeza e montagem das sequências EST em formigas Attini.

As etapas selecionadas foram:

- Avaliação de qualidade,
- Mascaramento de vetores,
- Mascaramento de *primers*,
- Aparamento de pontas de baixa qualidade,
- Filtragem por tamanho,
- Montagem com o programa CAP3.

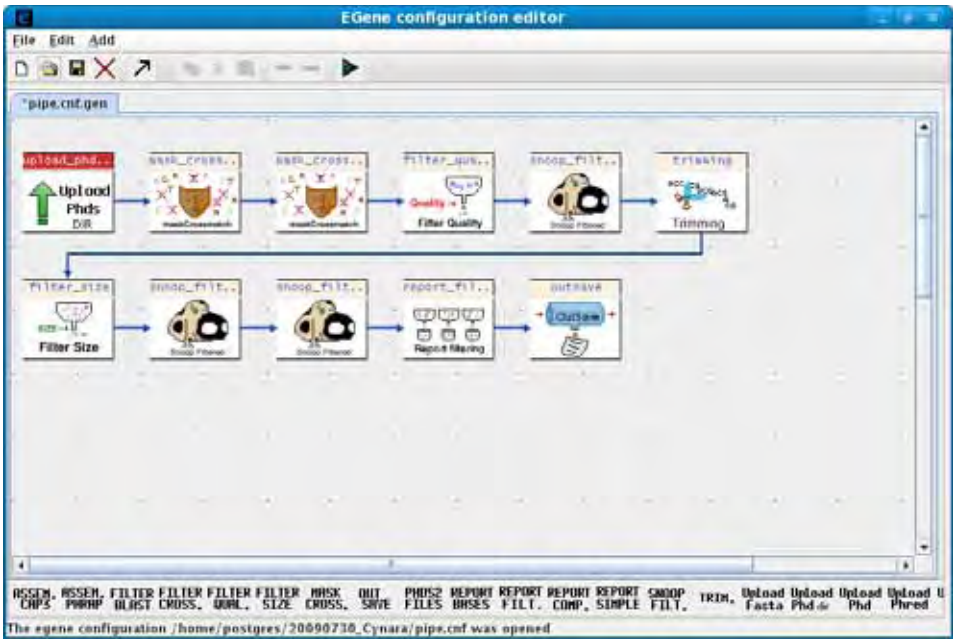
O primeiro sistema testado para essa tarefa foi o EGene.

EGene

Foi instalado inicialmente o sistema EGene, responsável por realizar etapas de pré-processamento de sequências. Pode-se notar que esse sistema é modular e permite que uma série de programas (módulos) sejam utilizados no *pipeline* conforme a necessidade de cada projeto. Além disso, há uma série

de componentes disponíveis, como módulos para mascaramento de vetor e *primer*, aparamento de pontas de baixa qualidade e agrupamento e montagem com o programa CAP3. Possui um editor gráfico denominado CoEd (Figura 3), que facilita a construção e a execução do *pipeline*.

Figura 7 – Exemplo de interface gráfica do EGene (CoEd) para a configuração dos componentes de pré-processamento das sequências.



É importante ressaltar que o EGene não possui módulos disponíveis para a anotação de sequências e não é um *web service*, como propomos que os componentes sejam em nosso sistema. Dessa forma, para que o *pipeline* funcione no EGene, todos os programas precisam estar fisicamente instalados.

Utilizando-se inicialmente cerca de 300 sequências EST de formigas Attini, foram obtidos como resultado do pré-processamento dois arquivos texto no formato FASTA, um com os *contigs* e outro com os *singlets*. Essas sequências foram então utilizadas como entrada nos sistemas de anotação testados.

No caso de sequências geradas por Sanger, o sistema EGene se mostrou eficiente, não havendo a necessidade de se desenvolver um novo sistema para pré-processamento e assim toda a atenção foi focada na descrição das sequências.

Annot8r

O primeiro sistema para a anotação instalado foi o Annot8r, que diz realizar a anotação associando resultados de BLAST e GO (Gene Ontology).

Annot8r realiza busca de similaridade usando o programa BLAST e posteriormente adiciona esses dados em uma base de dados que ele mesmo gera. Assim, ao final do processamento, temos um arquivo texto com a saída do BLAST e uma base de dados PostGree.

Notou-se que essas saídas são bastante complicadas de serem manipuladas por um usuário não-familiarizado. Além disso, o sistema não possui interface gráfica e só pode ser executado via linha de comando, o que torna ainda mais difícil sua utilização. Mas, como queríamos testar suas funcionalidades, foi elaborado um *pipeline* inicialmente sem interface gráfica.

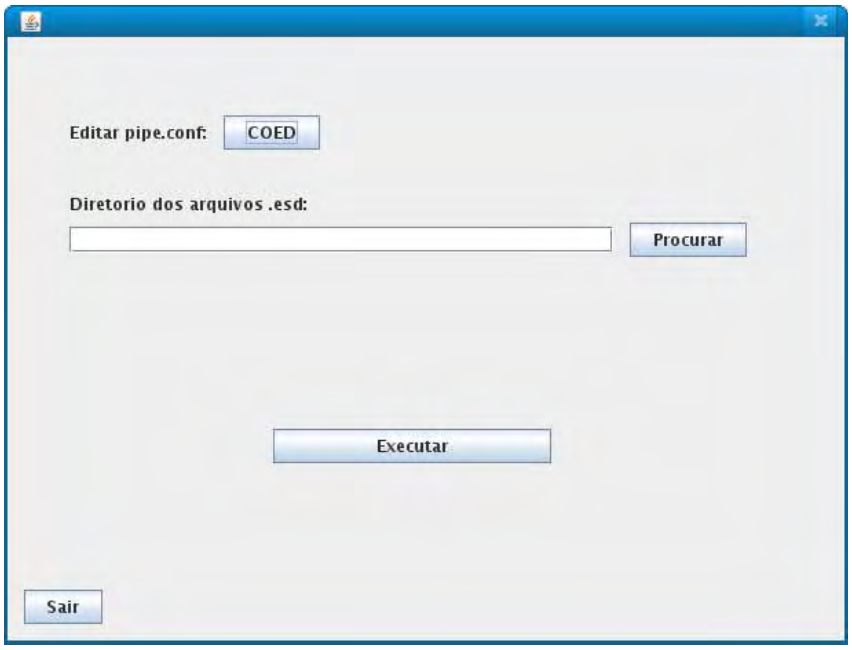
Utilizando o EGene para pré-processamento e o Annot8r para a anotação, um fluxograma inicial foi elaborado (Figura 8), mostrando como seriam ordenados os processos no *pipeline* teste.

Figura 8 – Fluxograma dos programas que compõem o *pipeline* teste elaborado.

O *script map2slim.pl* (CHRIS MUNGALL, não publicado; http://search.cpan.org/~cmungall/go-perl) foi adicionado ao *pipeline* por ter a função de mapear e contar os termos GO obtidos para cada sequência em cada uma das 3 categorias gênicas (processo biológico, componente celular e função molecular).

Com o protocolo inicial montado, o *pipeline* foi construído no sistema operacional Linux. Para facilitar a execução e a configuração dos componentes requeridos no processamento automático, foi desenvolvida uma interface gráfica simples (Figura 9) em Java.

Figura 9 - Tela da interface gráfica de execução do *pipeline* teste.



Se observarmos a Figura 9, podemos notar que há um botão que abre diretamente o EGene através de sua interface, o CoEd (Figura 7), permitindo ao usuário construir e editar os componentes de pré-processamento. Isso porque o sistema EGene já é um *pipeline,* então não haveria a necessidade de

refazê-lo. Essa é uma solução apropriada, mas que não traz grandes novidades ou funcionalidades ao *pipeline* desenvolvido.

Há também um botão EXECUTAR onde todas as etapas são executadas de forma oculta ao usuário, assim apenas os resultados são apresentados. Esta solução foi viável para finalidade de teste do anotador, mas é importante que o processo de montagem do *pipeline* seja visível ao usuário.

Depois de alguns testes com o *pipeline*, pode-se notar que o programa Annot8r é um coletor de evidências e não um anotador como se esperava. O Annot8r apenas gera dados de BLAST e GO, o que podemos chamar de evidências. Esses resultados não aparecem de forma integrada para cada sequência, mas sim em arquivos separados e isolados. Dessa forma, realizar essas tarefas de forma manual ou pelo sistema demanda praticamente o mesmo trabalho e gera os mesmos resultados finais. Além disso, não é trivial para o usuário nem o uso e nem a instalação do sistema, que exigem conhecimentos básicos de programação e de banco de dados.

Como acabamos construindo um *pipeline* teste e ao final notamos que não seria possível utilizá-lo para fazer a anotação, decidimos apenas instalar e testar os próximos sistemas, porém sem a construção de *pipeline*, já que isso demandaria mais esforço e tempo.

Foi necessário desenvolver alguns programas para realizar os testes apresentados acima. Se observarmos a Figura 8, podemos notar que o *pipeline* teste inicia-se com um *script*. Foi verificado que as sequências de DNA de entrada haviam sido geradas pelo sequenciador MegaBace e que portanto estavam no formato ESD. Este tipo de formato não é aceito como entrada para a execução do EGene. Desta forma foi necessário criar um programa que

utiliza o programa de terceiro Phred (Ewing *et al*., 1998 e Ewing; Green, 1998) para converter os arquivos do formato ESD para PHD.

Além disso, verificou-se que o EGene aceita como entrada um único diretório no formato PHD, sendo que havia um diretório para cada placa sequenciada. Para resolver esse problema foi necessária a elaboração de um *script* que realiza a concatenação dos arquivos em um único diretório, renomeando tais arquivos de forma a diferenciá-los (acrescentando o nome da placa sequenciada), já que os arquivos apresentavam os mesmos nomes em cada diretório.

Após realizar a montagem de sequências com o programa CAP3 no sistema EGene, eram gerados dois arquivos, um contendo os *contigs* e outro os *singlets*. Para que o Annot8r fosse executado de uma só vez, houve a necessidade de agrupar essas sequências em um único arquivo e para isso foi criado outro *script*.

O que se pode notar, independentemente do sistema de anotação que está sendo utilizado, é que há sempre a necessidade do desenvolvimento de algum *script* computacional para algumas adaptações.

GARSA

O próximo sistema testado foi o GARSA. O artigo da ferramenta GARSA mostrava que essa ferramenta aceitava como entrada cromatogramas e integrava e interpretava os dados de sequências geradas por análises no banco de dados CDD (*Conserved Domain Database*) e BLAST. Para esse programa não foi possível encontrar o *site* de disponibilização e *download* do sistema.

dCAS

Instalamos outro sistema, o dCAS, que exige a instalação do programa CAP3 para montagem, o programa BLAST e o banco de dados, sendo opcional a instalação do programa Phred. O dCAS possui uma interface gráfica amigável, porém, após os testes, notamos que o sistema não é flexível e modular, pois não é possível acrescentar novos módulos, apenas é possível desativar um componente que ele possui ou o programa para a busca de peptídeo sinal (SignalP) ou de domínio transmembranar (TMHMM). Além disso, não é possível alterar a ordem de execução, assim o sistema possui um fluxo de processamento (*workflow*) fixo.

Como resultado do dCAS é gerada uma planilha Excel. Inicialmente começamos os testes utilizando 384 reads, e o programa foi executado corretamente; no entanto, a planilha Excel com os resultados não mostrou dados de Gene Ontology (GO), KOG e dados das bases SMART e PFAM, como proposto pelo programa. Tentamos executar novamente com mais sequências, utilizamos 1.920 *reads* e descobrimos mais falhas. Dessa vez, o programa concluiu a execução, mas nenhum *link* da planilha pode ser aberto e os resultados de GO, KOG e dados das bases SMART e PFAM continuaram ausentes na planilha. Tentamos contactar o autor, mas não foi obtida nenhuma resposta que nos auxiliasse a resolver esses problemas.

Dessa forma, notamos que não seria possível realizar a anotação usando o dCAS, pois havia limitações como a quantidade de sequências utilizadas e os erros gerados na planilha. Além disso, o sistema mostrou-se bastante instável e, embora possua código fonte aberto, foi desenvolvido em Java e exige experiência e pessoal especializado para realizar alterações.

ESTExpress

O último sistema para anotação testado foi ESTExpress. Ele pode ser instalado no Linux ou Windows, exige a instalação do Cross_match e BLAST, além de conhecimento em PHP, MySQL e programas de terceiros. A instalação e execução, mesmo com interface de configuração, não se mostrou trivial.

Os resultados são apresentados em HTML e tabelas relacionais, mas a compreensão desses dados não é fácil. Pode-se notar também que existem módulos para programas de pré-processamento e anotação, porém a coleta de evidências se restringe a componentes para BLAST e GO. Além disso, a arquitetura do sistema é bastante restrita e pouco modular.

Conforme apresentado acima, os sistemas existentes apresentam uma série de limitações, tanto de execução quanto em relação aos resultados de anotação. Dessa forma, ficou clara a decisão da construção de um novo sistema que pudesse sanar essas limitações, apresentando a anotação de forma correta, integrando os resultados de todos os programas rodados (evidências) para cada sequência e gerando uma forma de realizar facilmente uma consulta pelo usuário. Além disso, todos esses sistemas funcionam para pequeno volume de sequências (Sanger) e como nosso grupo de pesquisa já contava com dados de NGS era necessário se pensar em formas de executar grande volume de dados de maneira eficiente.

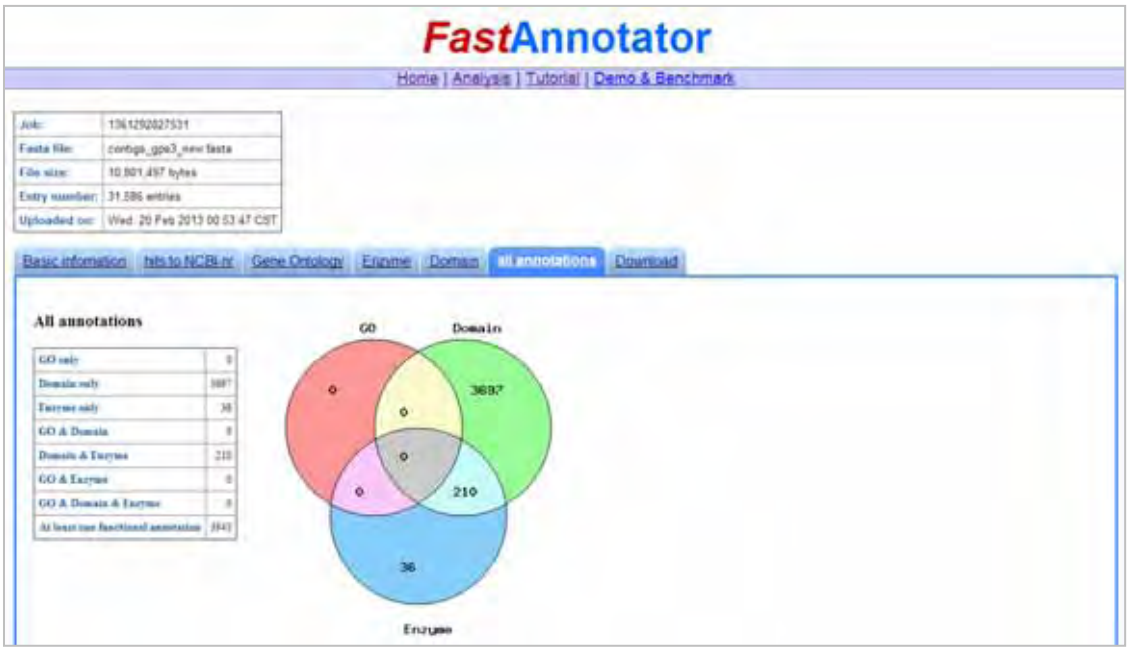
FastAnnotator

Recentemente foi publicado um sistema *web* denominado FastAnnotator (Chen *et al*., 2012) para anotação de transcriptomas ou metatranscriptomas

para sequências advindas de sequenciadores de nova geração. A entrada dessa ferramenta são sequências no formato FASTA pré-processadas.

Para testar a ferramenta foram usados 31.586 contigs obtidos por tecnologia Illumina a partir de dados de nosso laboratório de um transcriptoma de uma formiga cortadeira. Foi feito *upload* do arquivo multifasta e este foi processado usando um servidor remoto. Após um dia, os resultados foram verificados e notou-se que não foram gerados resultados para o programa Blast contra o banco de dados NR (não-redundante) e GO, mas voltaram resultados para *Enzime Comission* (EC) e domínios de proteínas. Dessa forma, apenas 3.943 contigs foram anotados funcionalmente (Figura 10).

Figura 10 - Tela da interface gráfica da ferramenta FastAnnotator mostrando um sumário com os resultados da anotação de ESTs de formigas Attini.



Essa análise mostrou que embora sejam lançadas ferramentas que se propõem a fazer a análise de transcriptomas ou de bibliotecas ITS ou 16S,

dada a quantidade de sequências a serem processadas, há limitações para processar e até apresentar essas informações.

Pensando em desenvolver sistemas que pudessem resolver essas limitações, foi desenvolvido um novo modelo de arquitetura.

4.2 Arquitetura Desenvolvida

Foi definida uma arquitetura baseada no padrão *Model-View-Controller* (MVC), sendo que esse padrão foi usado como um motivador e também um roteiro (*guideline*) para a identificação de interesses de cada uma das partes funcionais do sistema. Esse tipo de padrão auxilia na elaboração de modelos mais complexos de sistemas, mas, ainda assim, mantém a simplicidade na identificação dos elementos que compõem o sistema de forma objetiva. Além disso, pode-se verificar que uma vantagem no uso de um padrão como o MVC é a facilidade na comunicação entre os desenvolvedores durante a sua concepção (Fowler, 2002).

O padrão MVC prevê três elementos principais - três camadas: (1) à visão do usuário ou a interface gráfica de usuário (*view*); (2) um modelo de dados (*model*) e estrutura inter-relacional de objetos que descrevem de forma semântica como o sistema pode representar os dados e informações e (3) camada de controle que recebe e reenvia de modo planejado e controlado os eventos provenientes das camadas anteriores. No entanto, conforme se desenvolviam os *pipelines*, esse modelo foi melhorado e alguns nomes que associam as três principais camadas do padrão MVC foram adaptados para melhor refletir o sistema em desenvolvimento. Além disso, outras duas camadas foram adicionadas com o intuito de se detectar falhas no sistema de

pipeline no lado cliente e também do serviço *web*. A Tabela 2 apresenta a relação entre cada uma das camadas, comparando-se o padrão MVC e a arquitetura projetada neste trabalho.

Tabela 2 – Relacionamento e comparação das camadas que compõem o padrão MVC e o sistema em desenvolvimento.

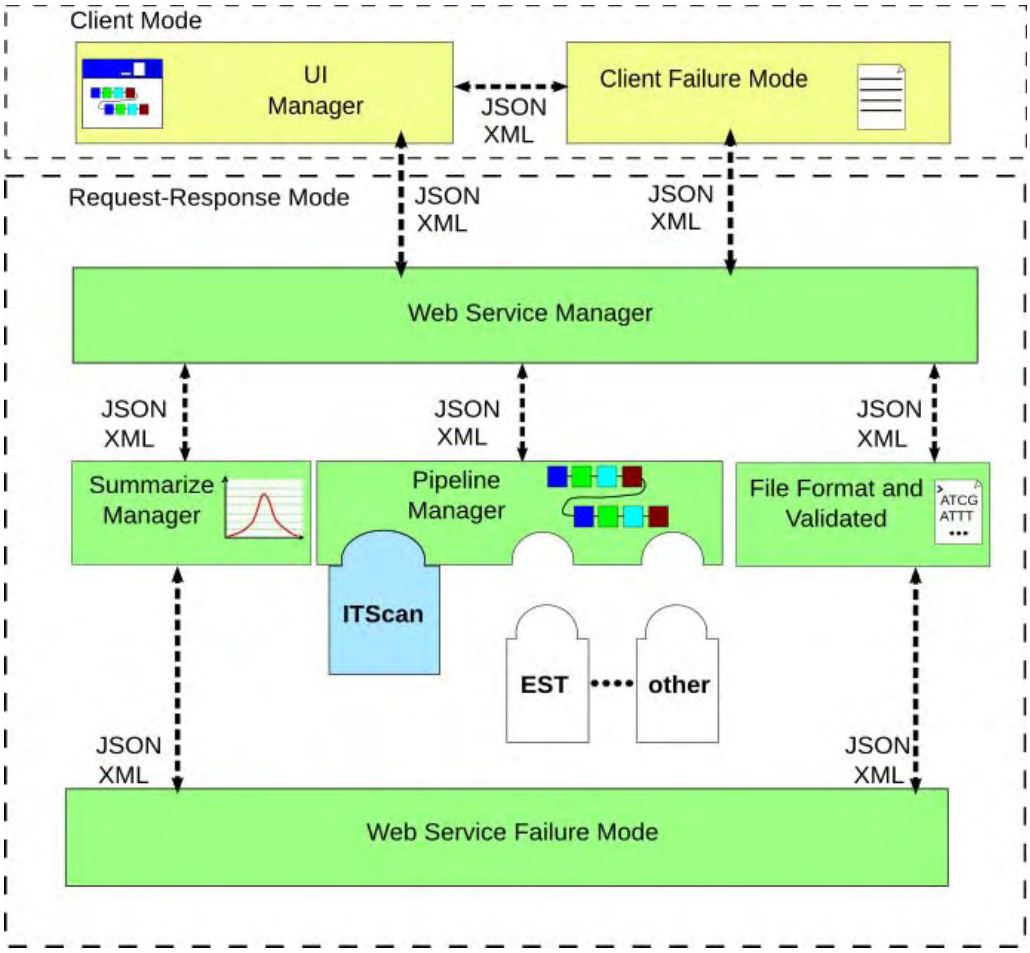
	Padrão MVC	**Arquitetura Proposta**
1ª Camada	*View*	*View*
2ª Camada	*Controller*	*Manager*
3ª Camada	*Model*	*Model Computation*
	—	*Failure Mode*

O padrão MVC foi mantido, sendo que a Figura 11 mostra a arquitetura do sistema de *pipeline* definida, onde se amplia e modifica a arquitetura inicialmente proposta.

Nessa arquitetura é possível identificar — em linhas tracejadas — duas regiões, uma que delimita o lado cliente (*Client Mode*) e a outra com as operações que são executadas do lado do servidor (*Request-Response Mode*). Essa separação é importante, pois permite identificar mais claramente os processos que são executados de cada lado, retratando em uma mesma figura a arquitetura física (das máquinas utilizadas).

Pode-se notar que existem quatro elementos denominados *managers,* sendo que cada um desses *managers* é responsável por realizar um gerenciamento, seja do lado do cliente ou do servidor, para a montagem e envio de mensagens entre cada uma dos gerenciadores de interesse.

Figura 11 – Modelo de arquitetura desenvolvida seguindo padrão MVC.



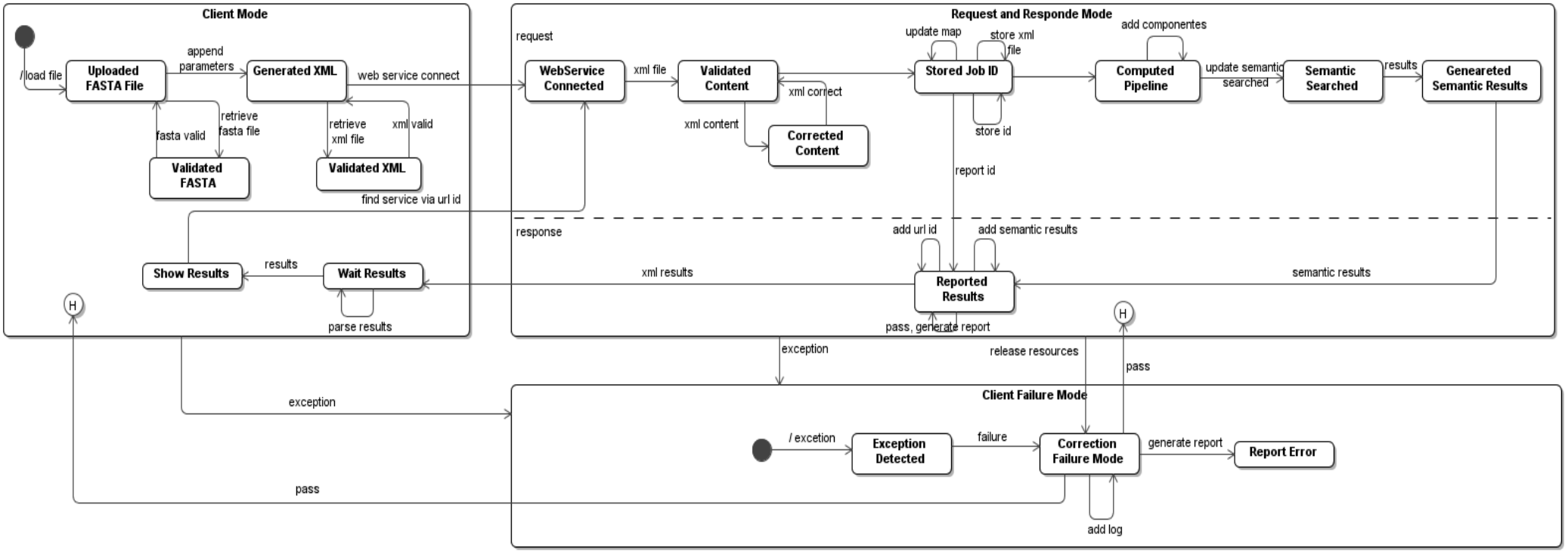
Na parte superior (em amarelo) é possível observar a visão *Client-Mode*, onde há um gerenciador *UI-Manager* e a camada *Client-Failure Mode*, sendo esta responsável por tratar os erros do lado do cliente, bem como validar os dados no formato JSON que é o formato de intercâmbio de informações em todo o sistema.

Na parte inferior da Figura 11 (em verde) temos a visão *Request-Response Mode* que ilustra as operações do lado do servidor. O *Pipeline Manager* é responsável por alocar os diferentes tipos de modelos de computação (*Model Computation*), de forma que o *Pipeline Manager* tem a função de um *framework.*

Para ilustrar, um *framework* refere-se a uma classe de aplicações computacionais que provê um ponto de acesso extensível e adaptável para a execução das suas funcionalidades (PRESSMAN, 2002). Se observarmos ainda a Figura 11, podemos identificar esses pontos adaptáveis na forma de peças de um quebra-cabeça, para os diferentes tipos de *pipelines* – para análise de sequências ITS, 16S e EST.

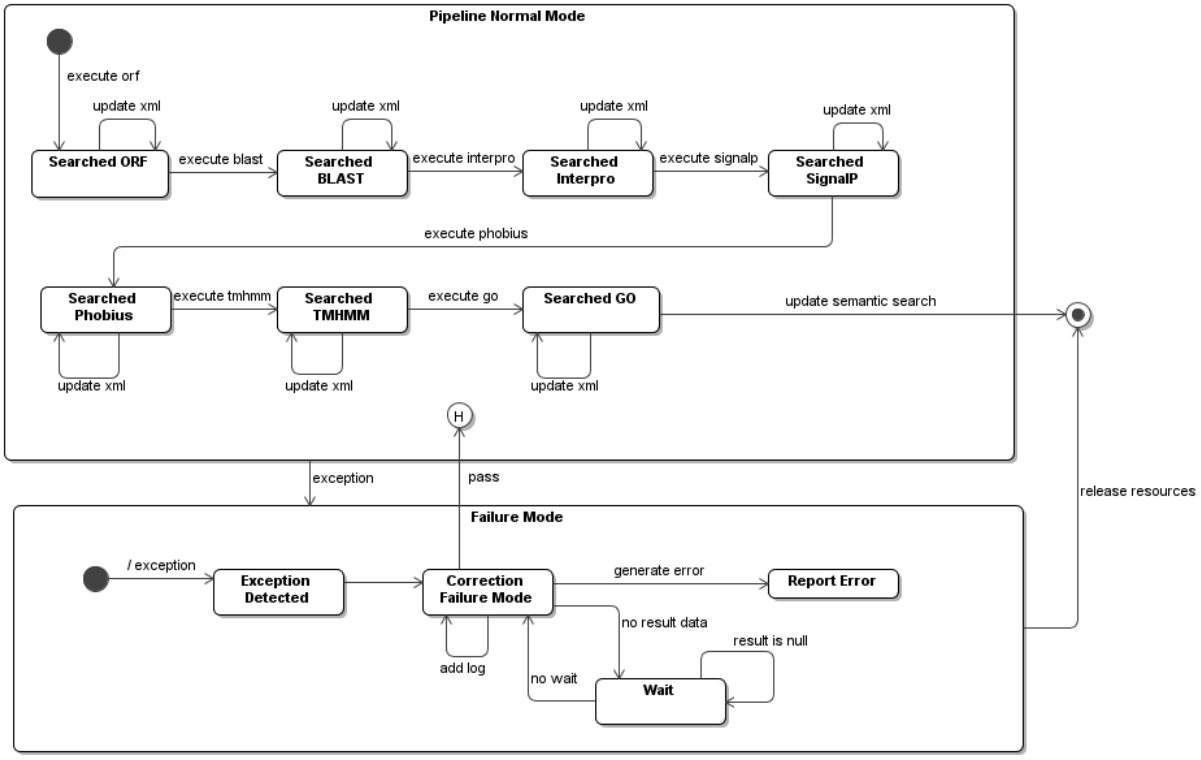
Para melhor compreender o funcionamento do sistema de *pipeline*, a Figura 12 apresenta os principais estados e eventos disparados entre a visão cliente-servidor. Essa figura representa um diagrama UML/Statecharts (UML, 2011). Nesse diagrama é possível visualizar os eventos paralelos (concorrentes) que são disparados por meio de *threads* da linguagem Java. A primeira linha de *thread — request mode* — lida com o pedido de requisição do lado do cliente para o *web service*. Essa requisição entra em um ciclo de vida de seis estados que vai de *web service connected* (conexão do serviço web) até *generated semantic results* (geração de resultados semânticos). No estado *Computed Pipeline* é realizada a computação efetiva do *pipeline*. Essa computação foi feita com base em sete componentes de terceiros que compõem o protocolo de anotação de ESTs.

Figura 12 - Diagrama UML/Statecharts para a interação entre o lado cliente e servidor do sistema de *pipeline* para anotação de ESTs.



Se ampliarmos o estado *Computed Pipeline* (Figura 13), onde se localiza o pipeline propriamente dito, no caso para análise de ESTs, podemos visualizar melhor as interações entre os componentes deste.

Figura 13 - Diagrama UML/Statecharts para os componentes do *pipeline* de ESTs.

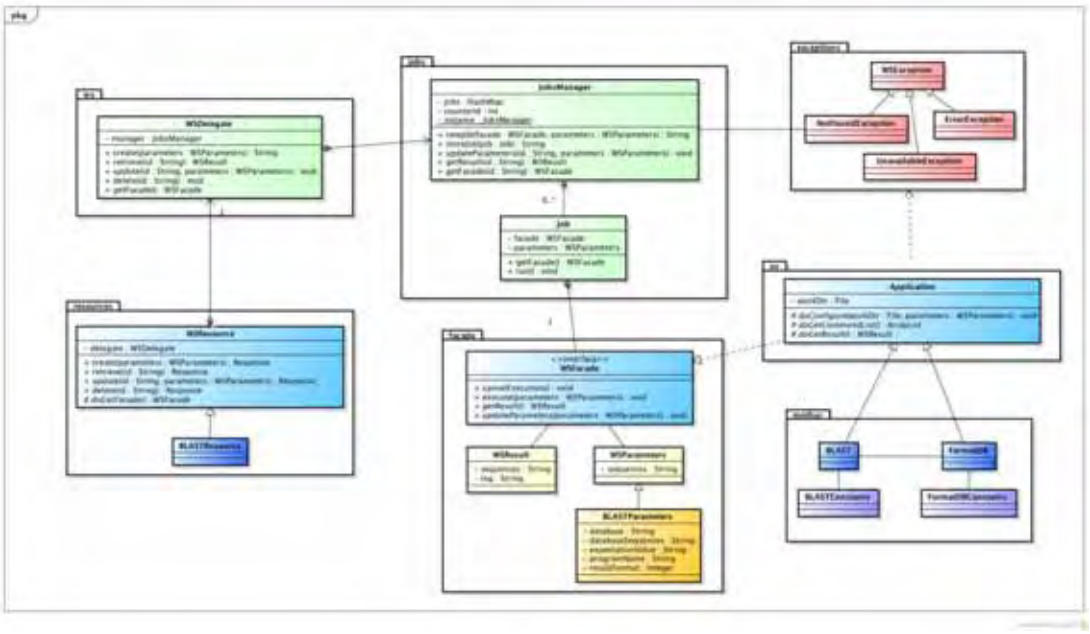


É importante notar que com essa arquitetura definida, fica estabelecido um modelo onde basta apenas plugar ou encaixar novas peças (*pipelines*) ao sistema de forma simples e eficiente. Todos os componentes desenvolvidos, tanto para anotação de ESTs quanto sequências de bibliotecas ITS e 16S, seguiram esse modelo de arquitetura.

Apenas para ilustrar a estrutura de um componente do *pipeline* usando como modelo o *pipeline* para análise de EST desenvolvido, a Figura 14 mostra o diagrama de classes Java do componente para busca de similaridade usando o BLAST. Pode-

se notar os relacionamentos que ocorrem desde a conexão com o *web service*, passando pela escolha de parâmetros pelo usuário até a apresentação dos resultados obtidos.

Figura 14 - Diagrama de classes Java do componente que executa o programa de terceiro BLAST.



Todas essas etapas de relacionamento entre as classes foram feitas para todos os componentes, para os três *pipelines* elaborados.

4.3 Desenvolvimento do Protocolo de Pré-processamento

Sequências obtidas por Sanger

Para preparar o protocolo de pré-processamento de sequências Sanger, foi usado o programa EGene. Esse protocolo apresenta os seguintes passos:

- Utilizam-se sequências no formato PHD, utilizando-se o componente upload_phd_dir.pl.

- Mascaramento da sequência do *primer*. As sequências são submetidas a um mascaramento contra o *primer* utilizado na construção da biblioteca de ESTs. Para isso, utilizou-se o componente mask_cross_match.pl do EGene e o programa Cross_match.

- Mascaramento de bases de vetor. Todas as sequências nucleotídicas foram mascaradas contra a sequência do vetor. Utilizou-se o componente do EGene mask_cross_match.pl acoplado ao programa Cross_match do pacote Phred/Phrap/Consed.

- Filtragem por qualidade. Foi utilizado o componente filter_quality.pl do EGene.

- Aparamento das pontas de baixa qualidade. Após a filtragem por qualidade, as pontas das sequências tiveram as suas bases de qualidade ruim aparadas. Para isso, usou-se o componente trimming.pl.

- Filtragem por tamanho. Após o processamento e aparamento de pontas, todas as sequências foram submetidas ao programa filter_size.pl do EGene. Somente foram aceitas sequências acima de 100 pb.

- Agrupamento e montagem de sequências. Foi utilizado o programa CAP3 para realizar a etapa de montagem, resultando em arquivos em formato FASTA para os *contigs* e *singlets* obtidos.

Sequências obtidas por NGS

No caso de sequências geradas por tecnologias NGS, esse protocolo é diferente, pois precisam ser considerados outros fatores como tamanho das sequências geradas e tipo de tecnologia. Assim, para uso de Illumina ou SOLID, em que são gerados fragmentos curtos, são usados programas para montagem como

Velvet/Oases, SOAPdenovo, ABySS entre outros. No caso de 454, que gera sequências maiores, pode-se usar para a montagem Newbler, MIRA, entre outros.

Como não há etapa de clonagem, não é preciso retirar sequências de *primers* e de vetores, No entanto, pode ser necessário retirar os adaptadores usados na construção da biblioteca.

4.4 Desenvolvimento de Protocolo para Anotação de EST

Havia inicialmente uma proposta de protocolo de anotação baseada em anotações realizadas para outro organismo (Ferro, 2008). Durante a instalação de sistemas de anotação existentes, baseado nos resultados obtidos e nas ferramentas utilizadas, decidiu-se pela manutenção de alguns tópicos do protocolo inicial proposto.

Os principais componentes deste protocolo de anotação e que foram incorporados no *pipeline* estão apresentados a seguir:

- Busca e tradução conceitual de ORFs**.** Esta ferramenta foi desenvolvida inteiramente neste projeto como um serviço *web*, e realiza a busca de ORFs e sua tradução, fundamental para a execução de outras ferramentas que utilizam sequências nucleotídicas codificadoras de proteínas ou sequências de aminoácidos como entrada no *pipeline*,

- Busca de similaridade. Para as buscas de similaridade foi utilizado o BLAST, especificamente o programa blastp, pois as ORFs já estão traduzidas,

- Busca de domínios proteicos. As sequências proteicas das ORFs preditas são utilizadas para se realizar uma busca por motivos proteicos, usando a ferramenta InterproScan,

- Busca de sequências de peptídeo-sinal. Para essa busca foi usado o programa de terceiro SignalP (BENDTSEN *et al.*, 2004),

- Busca de domínios transmembranares. Foi utilizado o programa TMHMM (KROGH *et al*., 2001).

- Mapeamento de termos Gene Ontology (GO). Os dados de ontologia gênica foram extraídos do programa InterProScan, onde foram mapeados os termos de ontologia para cada sequência de entrada do *pipeline*.

- Geração dos resultados da anotação. Os resultados com a anotação foram apresentados nos formatos: tabular, XML e nos formatos de saída das ferramentas.

4.5 Implementação do *Pipeline* para EST

Quando são geradas ESTs, objetiva-se descobrir as regiões codificadoras de proteína, sendo que uma estratégia utilizada é a busca de ORFs (*Open Reading Frames*). Nessa etapa devem ser encontradas todas as possíveis regiões codificadoras (CDs) para cada sequência. Dessa forma, conforme consta no protocolo de anotação proposto, a anotação inicia-se com a busca e tradução de ORFs.

Componente para a Busca e Tradução de ORFs

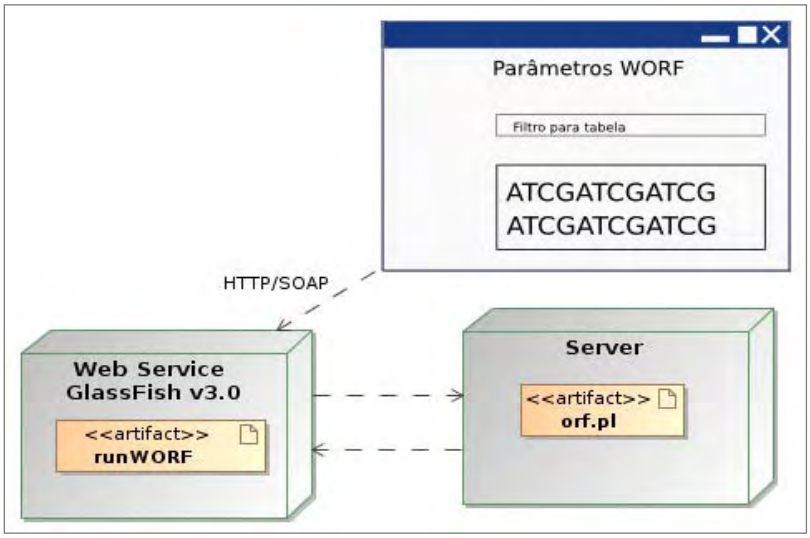
Inicialmente foi analisada a ferramenta ORF Finder do NCBI (*National Center for Biotechnology Information*) e foram verificadas algumas limitações como: incapacidade de analisar grandes quantidades de sequências em um único processamento, não permissão para fazer *upload* de entrada e *download* dos resultados.

Pensando nisso, foi desenvolvido um componente denominado WORF que realiza a busca e tradução conceitual de ORFs como um *web service*. Esse é o primeiro componente do *pipeline* para anotação de ESTs e que processa grande número de sequências de uma vez e utiliza filtros para geração de ORFs.

Para auxiliar na descrição dos elementos de *software* existentes, neste trabalho foi usada a *Linguagem de Modelagem Unificada* (UML). A UML é um padrão OMG - *Object Management Group* (http://www.omg.org/) que atualmente está na versão 2.0 e possui um conjunto de treze diferentes tipos de diagramas e cinco visões (Booch *et al.*, 2006).

A Figura 15 apresenta um diagrama de implantação do WORF, onde se observa o componente runWORF. Nota-se que existe uma anotação UML na forma de um estereótipo <<*artifact*>>, que representa um grupo de componentes (objetos) com características similares, neste caso o *artifact*, que auxilia na identificação de um componente que é a interface do *web service* SOAP, que fornece um ponto de entrada para a aplicação.

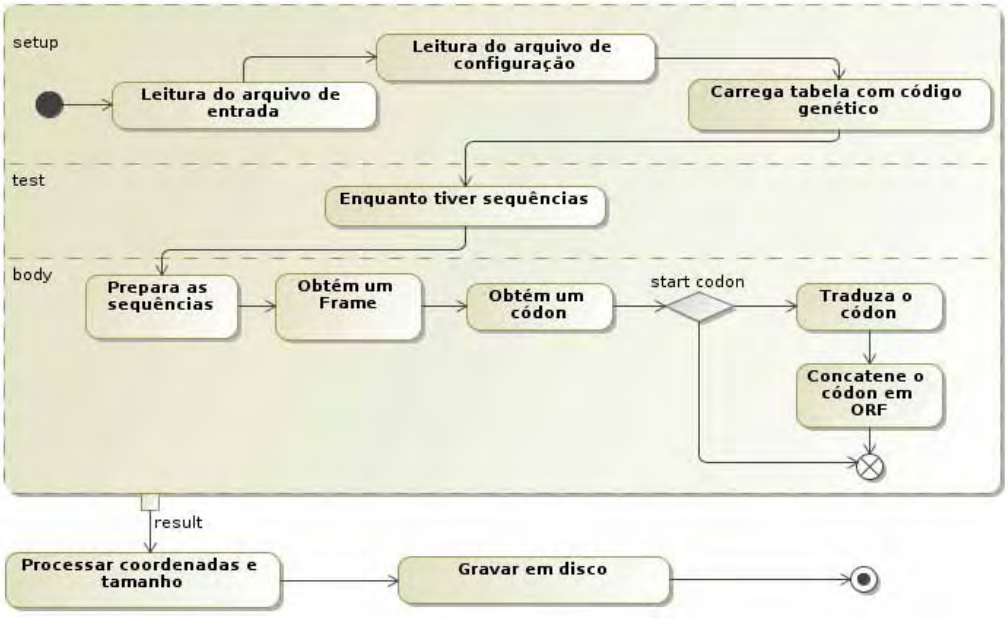
Figura 15 - Diagrama de implementação do WORF.



Observa-se também que existe um componente *orf.pl* que é um programa externo escrito em Perl e está contido dentro do *Server* e responde a chamadas apenas do *web service*. Dessa forma o único ponto de acesso da aplicação do WORF é através do *web service*, que fornece um método global para consulta via o protocolo SOAP/HTTP em uma interface Java.

WORF pode receber as sequências por *upload* do arquivo e, diferentemente do ORF Finder do NCBI, os resultados também ficam disponíveis para *download*. Além disso, WORF apresenta filtros para seleção: do tamanho mínimo das ORFs; tipo de tabela de código genético; uso de *start_codon* padrão e uso de *start_codon* alternativo. As atividades do WORF estão apresentadas no fluxograma (Figura 16).

Figura 16 - Diagrama de Atividades do componente `orf.pl`.

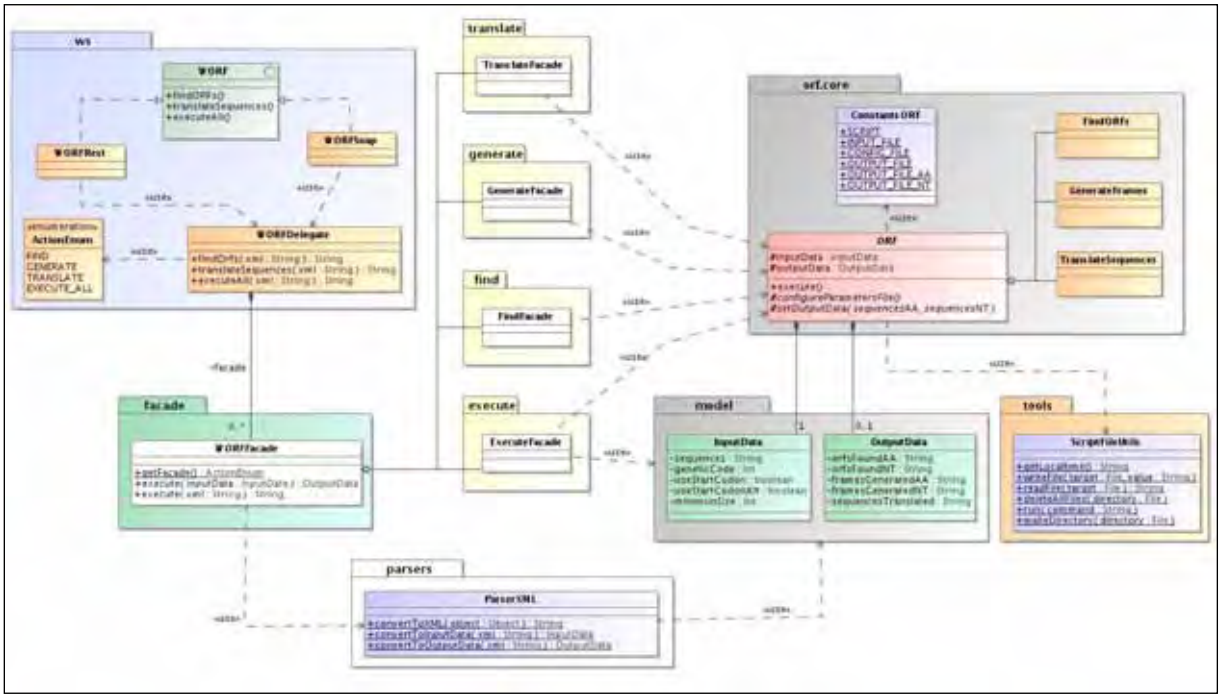


Assim, essa etapa de desenvolvimento criou um modelo preliminar do core (núcleo) da Arquitetura do WORF. Essa arquitetura usa princípios de Orientação a Objetos, Padrões de Projeto e Processos sucessivos de Refatoração que foram

realizados através de um processo de inspeção em pares a fim de que o código tivesse permanentemente qualidades positivas em relação a manutenção do *software*.

Como se pode observar na Figura 17, a Arquitetura do WORF é uma adaptação de uma Arquitetura de 3 camadas (*Model-View-Controller*), de modo que são introduzidas camadas intermediárias para separar a abstração de mais alto nível de elementos de mais baixo nível. Os seguintes pacotes principais constituem a arquitetura do WORF: (1) *web service – ws*; (2) *facades*; (3) *parsers*; (4) *model*; (5) *core*; e (6) *tools*.

Figura 17 - Arquitetura do WORF mostrando as três camadas e seus respectivos pacotes.



O ponto de entrada da arquitetura é o pacote (1) *web service*, que serve de base para implementação do *web service* em SOAP ou REST. Assim, existe uma interface comum entre ambos os serviços que separa a implementação da abstração

e permite que sejam implementados serviços independentemente. Essa camada da WORF usa polimorfismo para escolher dinamicamente qual tipo de implementação usar. Em (2), temos os objetos *facades* que estão associados com objetos *delegate* de mais alto nível de abstração para redirecionar as requisições de um tipo específico de *web service* para um processamento de mais baixo nível, estes representados por objetos como tradução (*translate*), geração (*generate*), busca (*find*) e execução (*execute*).

É importante notar que, embora exista a possibilidade de duas implementações distintas de *web services* (REST e SOAP), esta versão atual contempla apenas a versão em REST.

Em (4) temos um pacote que contém definições básicas dos dados que serão enviados para o *web service*, ou seja, é a camada de modelo de aplicação e em (5) temos o Core da aplicação. Em (6), *tools*, há um pacote para auxiliar na representação de classes acessórias para a execução dos *scripts* em linguagem Perl.

Em pesquisas recentes realizadas no PubMed, nenhuma arquitetura de referência para o desenvolvimento em Bioinformática para aplicações com *Web Services* foi encontrada. Dessa forma, nossa proposta tem um provável impacto na geração de novas publicações e poderá servir como modelo preliminar para desenvolvimento de novos componentes e sistemas.

Testes de Desempenho WORF

Com o intuito de se aumentar o desempenho computacional e sabendo que outros programas seriam desenvolvidos com o mesmo propósito ao longo deste projeto de doutorado, o programa WORF foi elaborado para prover acesso via *web*

como um *web service* usando duas tecnologias distintas, REST e SOAP. Além disso, o programa do ORF foi escrito em duas versões diferentes — usando duas linguagens de programação, Perl e Java.

Após o desenvolvimento, foram aplicadas atividades de teste para caracterizar o desempenho desses dois programas em relação à REST e SOAP. No contexto da Engenharia de Software, as atividades de Verificação e Validação (V&V) têm como objetivo detectar falhas no *software*. Uma abordagem de V&V é a atividade de teste não-funcional, ou seja, a avaliação e detecção de falhas do ponto de vista do desempenho ou carga do sistema (PRESSMAN, 2002).

Para realizar a atividade de teste não-funcional e caracterização do WORF, foi utilizado o JMeter 2.4[2] que é um *software* livre que executa testes de carga e medidas de desempenho em aplicações *web* para diferentes tipos de parâmetros e protocolos de rede. No JMeter com um plano de teste, o usuário seleciona diferentes parâmetros e os tipos de resultados gerados.

Foram desenvolvidos dois planos de testes (Tabela 3), com o objetivo de determinar como o WORF se comporta para os *web services* SOAP e REST, através da variação do número de usuários e do tamanho do arquivo de sequências FASTA.

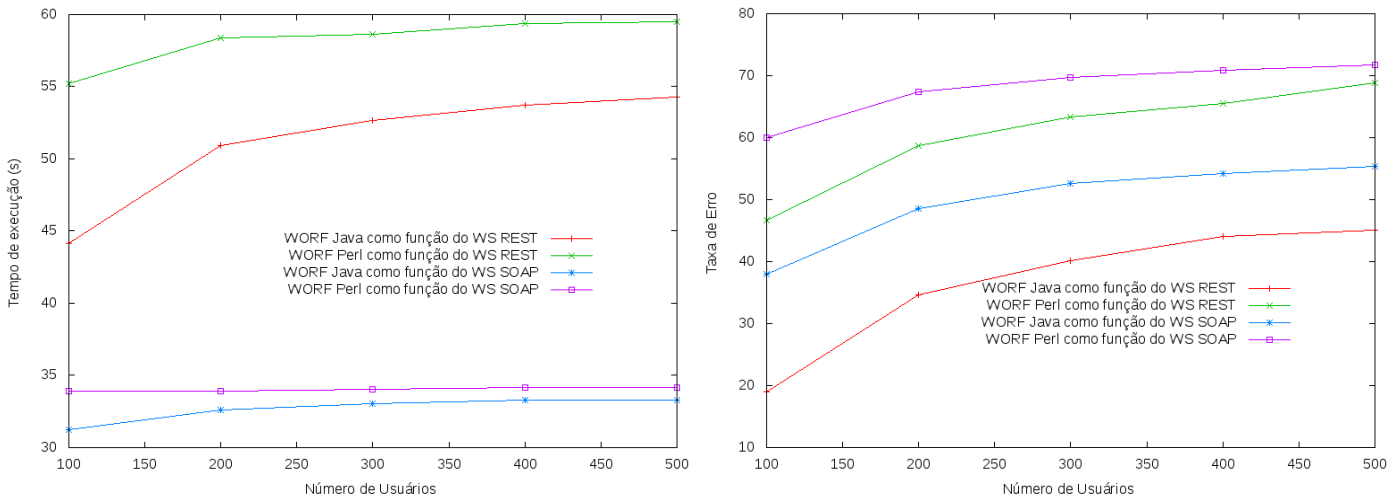
Tabela 3 - Parâmetros para os planos de teste de desempenho realizados com o WORF.

Variação do Número de Usuários	Variação do Tamanho do Arquivo
Tamanho arquivo de EST: 1,3 MBytes	**Número de usuários:** 500
Tempo de criação *threads* (usuários): 2s	**Tempo de criação *threads* (usuários):** 2s
Número de usuários: [200 - 1.000]	**Tamanho arquivo de EST:** [~200 KB - ~3,2 MB]
Amostras: 3	**Amostras:** 3
Tipo de *web service*: [REST|SOAP]	**Tipo de *web service*:** [REST|SOAP]
Linguagem: [Java|Perl]	**Linguagem:** [Java|Perl]

[2]Disponível em http://jakarta.apache.org/jmeter/

Para o primeiro plano de teste, foram realizadas medidas em função do número de usuários para as linguagens Java e Perl. Além destes parâmetros, também foram feitas amostras em função dos tipos de *web services* REST e SOAP. O número de usuários variou de 200 a 1.000 com passo de 200. O intervalo na criação de cada grupo de usuários e o tamanho do arquivo foi constante para todas as medidas (Figura 18).

Figura 18 - Resultados dos testes de desempenho para WORF. (A) Comportamento do WS REST/SOAP e Java/Perl para a variação do número de usuários em relação ao tempo; (B) comportamento do WS REST/SOAP e Java/Perl para a variação do número de usuários em relação à taxa de erro.



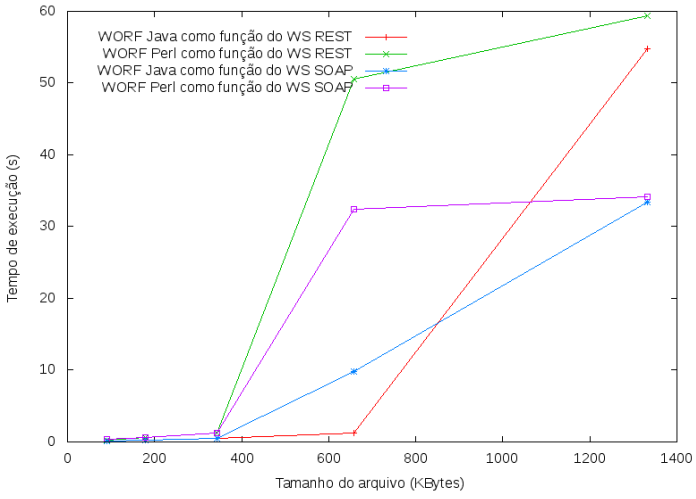
(A) **(B)**

A Figura 18A mostra claramente a separação entre o tempo de resposta para os dois tipos de *web services* do WORF. Pode-se notar que a troca de mensagens em SOAP teve um tempo de resposta aproximadamente constante (< 35s) para ambas as linguagens. Por outro lado, REST apresentou um maior tempo de resposta, chegando a um pico máximo de 59,53 segundos.

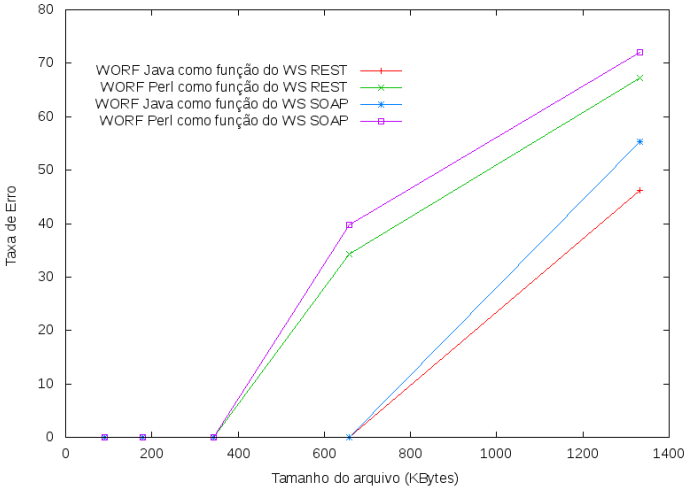
Embora o REST tenha apresentado um tempo de resposta maior, a Figura 18B mostra que, à medida que se aumenta o número de usuários, a taxa de erro (razão entre o número de falhas pelo número de usuários) com a linguagem Perl em SOAP ou REST aumenta, chegando ao pior caso, com cerca de 70% de erro para 500 usuários. Isso ocorre devido ao tipo de integração empregada entre o *core* do WORF e o componente escrito em Perl. Este exige ser executado como um aplicativo em Linux, o que leva o WORF a exigir recursos do sistema operacional como alocação de memória e escalonamento.

No segundo plano de teste, foram feitas medidas de desempenho considerando diferentes tamanhos de arquivos de sequências, de cerca de 200 Kbytes até 3,2 Mbytes (Figura 19), arquivos esses que continham sequências geradas em nosso laboratório.

Figura 19 - Resultados dos testes de desempenho para WORF. (A) Comportamento do WS REST/SOAP e Java/Perl para a variação do tamanho do arquivo em relação ao tempo; (B) comportamento do WS REST/SOAP e Java/Perl para a variação do tamanho do arquivo em relação à taxa de erro.



(A)

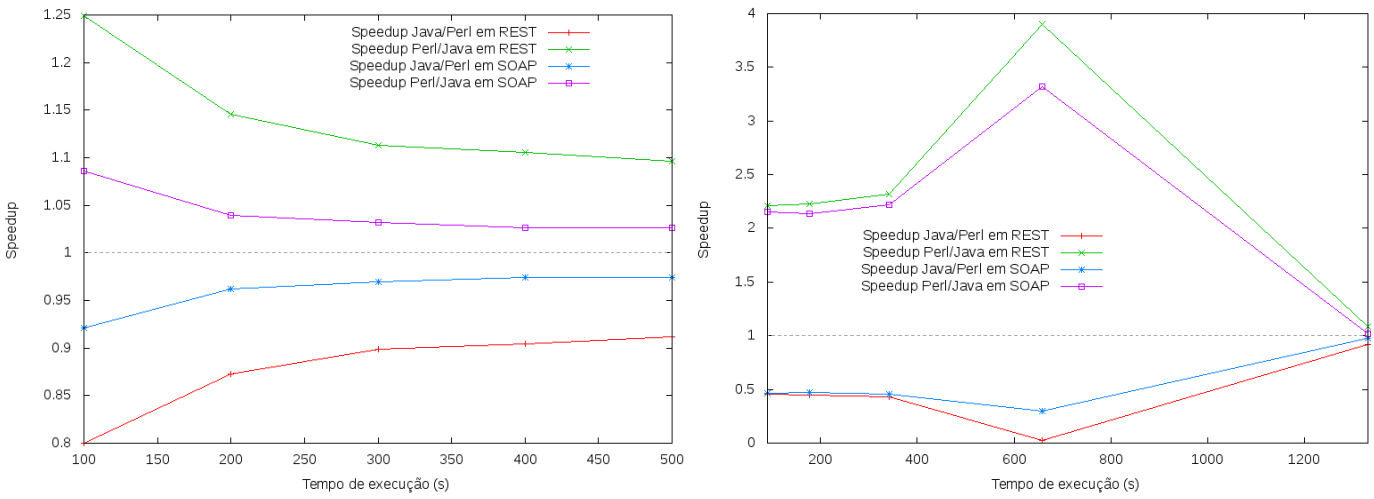


(B)

A partir das Figuras 19A e B, é possível notar um limiar no intervalo entre 600-800. Neste caso, até este limiar não havia taxa de erro e o tempo de execução para os arquivos foi de menos de 1 segundo. Porém, com arquivos maiores de 600 Kbytes notou-se um considerável aumento no tempo de processamento e na taxa de erro variando o tamanho do arquivo.

Em relação ao cálculo de *speedup*, valores maiores que 1 indicam que uma linguagem é melhor do que a outra, para *speedup* igual a 1 não há diferença entre uma linguagem e a outra. A Figura 20 apresenta os gráficos de *speedup*, variando o número de usuários e o tamanho do arquivo de sequências.

Figura 20 - Resultados dos testes de desempenho para WORF. (A) *Speedup* para a variação do número de usuários e o tipo de WS; (B) *Speedup* para variação do tamanho do arquivo de entrada e o tipo de WS.



Na Figura 20B, o resultado mostra que para arquivos de tamanho médio de 600 bytes há uma melhora significativa da linguagem Java em relação ao Perl, chegando a um *speedup* de quase 4, o que significa que a linguagem Java executou ~ 40% mais rápido que Perl.

Com esse estudo, pode-se notar que a escolha da linguagem e do tipo de *web service* utilizado dependem da aplicação de Bioinformática e da quantidade de sequências em análise. No caso do WORF, para os testes de desempenho com variação no número de usuários e de tamanho do arquivo, Java apresentou menor tempo de processamento e também menor taxa de erro.

Em relação ao *speedup*, pode-se observar que o aumento no número de usuários não gera uma variação significativa para as duas linguagens e SOAP/REST. Já para o aumento no tamanho do arquivo, Java com SOAP ou REST, apresenta *speedup* melhor chegando a quase 40% de aumento no desempenho. Embora esses dados mostrem o comportamento de uma aplicação para a busca e tradução de ORFs, estes podem servir como base para o desenvolvimento de planos de testes futuros para outras aplicações de Bioinformática.

Escolha da Técnica de Programação Concorrente para o WORF

A escolha da linguagem Java como plataforma de desenvolvimento e publicação do serviço foi feita pela facilidade em se trabalhar com serviços *web* e o ganho de desempenho comparado com a versão do programa escrito em Perl, conforme verificado nos testes de desempenho apresentados na Tabela 3. Outra facilidade no uso do Java refere-se a nova implementação da biblioteca API Java para concorrência. Essa API fornece novas formas de lidar com paralelismo/concorrência, junto com as boas práticas de programação, sem a necessidade de se preocupar com problemas como Impasses, Condições de Corrida e não lidar com *threads* explicitamente.

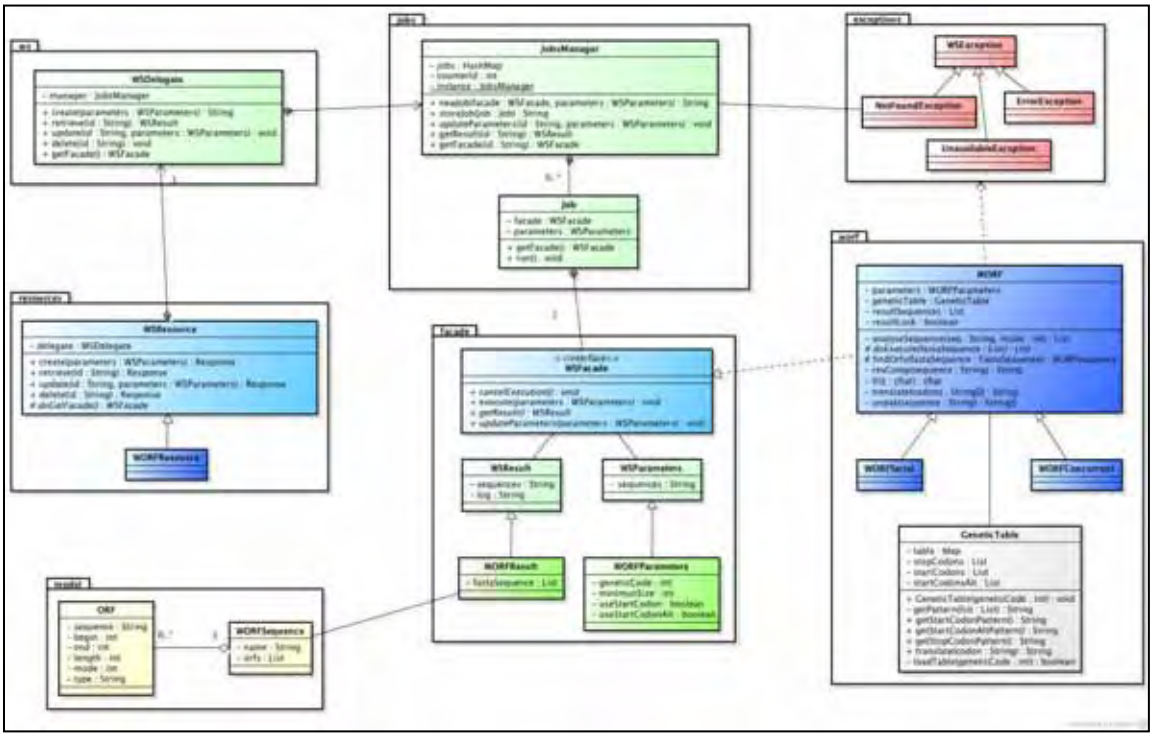
Foi definido também com os testes apresentados acima, que os próximos componentes seriam desenvolvidos usando o protocolo REST no lugar de SOAP.

Essa escolha foi devido a (1) existência de bibliotecas prontas, o que facilita seu uso, (2) facilidade de conversão do formato XML através de JAXB e (3) facilidade na representação de formatos como XML, JSON e texto.

Implementação da Técnica Escolhida para o WORF

O código fonte do WORF foi dividido em uma classe abstrata que realiza todo o processo de busca e tradução de ORFs e delega a função de gerenciamento da execução para as classes filhas. Uma classe filha realiza o gerenciamento da execução de forma serial enquanto outra classe implementa o gerenciamento paralelo. O tipo de gerenciamento é escolhido no momento em que a aplicação é executada (Figura 21).

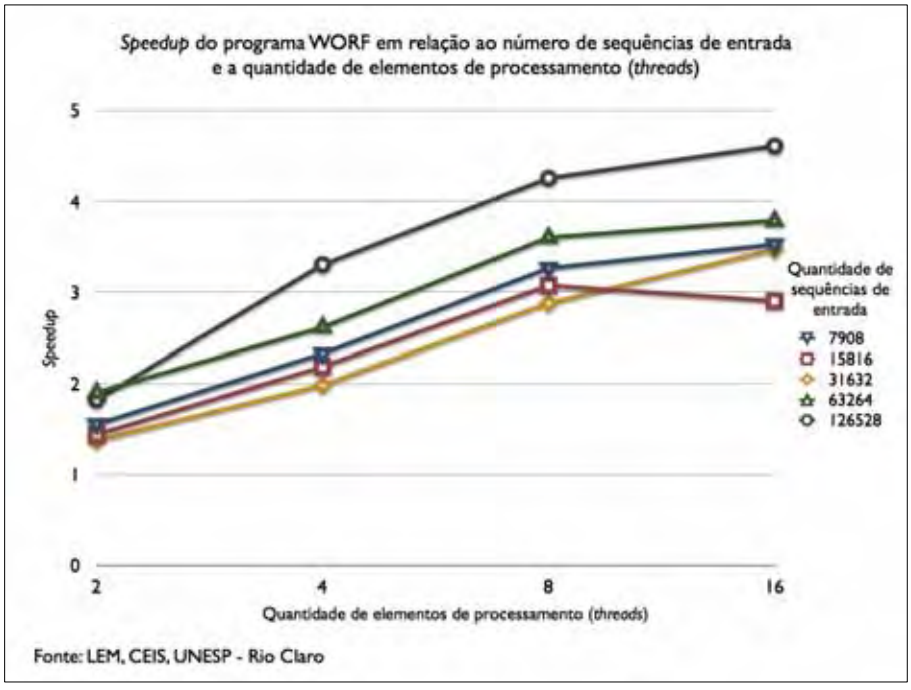
Figura 21 – Diagrama de Classes do WORF.



No caso da ferramenta WORF na versão concorrente, a execução das sequências foi dividida em lotes, sendo utilizado o número de núcleos por processador vezes o número de processadores (SUBRAMANIAM, 2011). Para assegurar a equivalência dos resultados entre a versão serial e a versão concorrente do programa, foram realizados mais alguns testes agora pensando em quantidade de elementos de processamento (*threads* do Java) e quantidade de sequências de entrada.

Os resultados para esse cálculo de *Speedup* (Figura 22) indicam que, para as configurações utilizadas, dobrar a quantidade de *threads* aumenta o desempenho do programa WORF. No entanto, devido ao aumento na quantidade de sequências de entrada, o *Speedup* é maior de forma que o tempo de processamento das sequências é maior que o tempo necessário para sincronizar os resultados produzidos por cada *thread*.

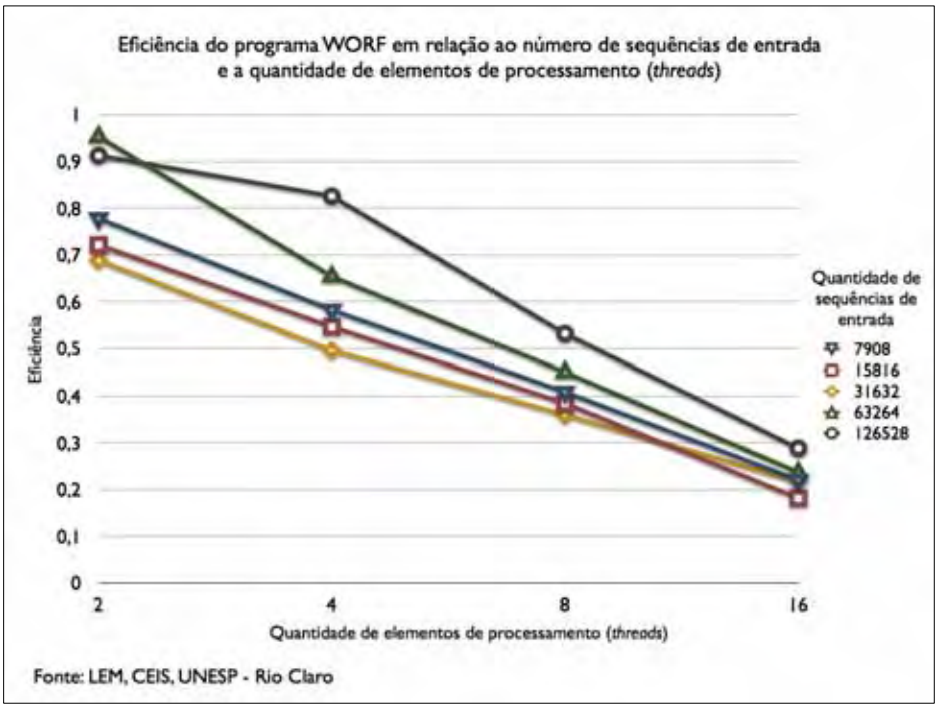
Figura 22 - Gráfico com os resultados do cálculo de *Speedup*, variando a quantidade de sequências de entrada e de *threads*.



Ao utilizar 16 elementos de processamento, o aumento de desempenho é menor devido a maior concorrência entre as *threads*, que aumenta o tempo de espera de cada *thread* para utilizar os recursos de processamento.

Assim, WORF torna-se menos eficiente ao aumentar a quantidade de *threads*. A diminuição da eficiência ocorre porque a *thread* principal deve obter os resultados de todas as outras *threads* na mesma ordem que foram criadas. Como o término do processamento de cada *thread* não pode ser pré-determinado, a *thread* principal deve aguardar até que o processamento do próximo conjunto de sequências esteja concluído, para então tratar os resultados e iterar para o grupo seguinte. Sem essa sincronização, os resultados produzidos não seriam gravados na mesma ordem em que os dados de entrada foram lidos. A Figura 23 apresenta o gráfico com os resultados do cálculo de Eficiência.

Figura 23 - Gráfico com os resultados para o cálculo de Eficiência, variando a quantidade de sequências de entrada e de *threads*.



Comparação do WORF com Outras Ferramentas

WORF foi comparado com outras duas ferramentas de Bioinformática já bem conhecidas e amplamente utilizadas. A Tabela 4 apresenta os programas comparados com suas respectivas potencialidades e endereços eletrônicos de acesso.

Tabela 4 - Programas de Bioinformática que identificam regiões codificadoras de proteínas em sequências de DNA e utilizados em testes comparativos.

Programa Bioinformática	**Potencialidades/Endereço eletrônico**
WORF	Software para a busca e tradução de ORFs nos seis frames possíveis (fita *forward* e *reverse*) desenvolvido em nosso laboratório. É um serviço *web* disponibilizado em: http://evol.rc.unesp.br:8083/worf.
ORF Finder	Software para a busca e tradução de ORFs nos seis frames possíveis desenvolvido pelo NCBI e disponibilizado em: http://www.ncbi.nlm.nih.gov/gorf/gorf.html
Genscan	Preditor de genes baseado na localização e estruturas de éxons e íntrons completas de genes em uma sequência genômica. Desenvolvido pela Universidade de Stanford e disponibilizado em http://genes.mit.edu/GENSCAN.html.

É importante ressaltar que as três ferramentas buscam por regiões codificadoras de proteínas, porém usando algoritmos diferentes, o WORF e ORF Finder buscam por todas as possíveis ORFs em todos os seis frames (para as fitas *forward* e *reverse*) considerando o Código Genético para o organismo em análise. Já o programa Genscan realiza a busca baseado em estruturas completas de genes pré-definidas, apresentando modelos de estruturas preditas para sequências de Vertebrados, *Arabdopsis* e milho. Essa busca é limitada por um modelo de gene

predito e torna a ferramenta Genscan mais limitada em relação aos buscadores de ORFs, pelo fato de ser específica para um determinado grupo ou organismo. Assim, para cada grupo de organismos se faz necessário um preditor de genes com modelos específicos, como GLIMMER (SALZBERG *et al*., 1998) que se baseia em Modelos Ocultos de Markov (HMM) para fazer a pesquisa para Bactéria, Archaea e Vírus.

Os programas apresentados na Tabela 4 foram comparados inicialmente em relação aos parâmetros de execução e suas funcionalidades, que são apresentados na Tabela 5.

Tabela 5 - Características em relação aos parâmetros e funcionalidades das ferramentas comparadas.

Ferramenta/Funcionalidade	**WORF**	**ORF Finder**	**Genscan**
Upload de arquivo FASTA de entrada	X	-	X
Entrada com GI ou número de acesso de uma sequência do GenBank	-	X	-
Seleção de código genético	X	X	-
Seleção de modelo de estrutura gênica	-	-	X
Tamanho mínimo da sequência em a.a.	X	-	-
Seleção de uso de *start_codon*	X	-	-
Seleção de uso de *start_codon* alternativo	X	-	-
Download dos resultados	X	-	-

Considere: (X) presença e (-) ausência

Conforme pode ser visualizado na Tabela 5, o programa WORF apresenta mais funcionalidades que permitem filtrar, por exemplo, o tamanho mínimo da ORF e seleção de códon de iniciação e códon de iniciação alternativa antes de fazer o processamento, diferentemente do que ocorre no ORF Finder.

Foi realizado também um teste comparativo com as três ferramentas, comparando os resultados gerados para uma mesma sequência de entrada (Tabela 6).

Tabela 6 - Apresentação dos resultados obtidos para as três ferramentas comparadas (WORF, ORF Finder e Genscan).

ORFs	WORF	ORF Finder	Genscan
Frame +2 71..3010 980 aa	X	X	X
Frame +3 2760..2972 70 aa	X	X	-
Frame -2 2536..2793 85 aa	X	X	-
Frame -2 1894..2091 65 aa	X	X	-
Frame -2 1138..1347 69 aa	X	X	-
Frame -2 817..972 51 aa	X	X	-
Frame -2 364..573 69 aa	X	X	-
Frame -3 2772..2924 50 aa	X	X	-
Frame -3 2565..2729 54 aa	X	X	-

Considere: (X) presença e (-) ausência

O programa ORF Finder gerou mais ORFs do que o WORF, porém estas não estão apresentadas na Tabela 6, por apresentarem tamanho mínimo menor do que 50 aminoácidos. Genscan gerou apenas um resultado, por trabalhar com modelos preditos, sendo, portanto, mais específico do que os buscadores de ORFs. Para o Genscan, foi encontrada apenas uma região codificadora (CDS), que corresponde a maior ORF encontrada pelas ferramentas WORF e ORF Finder.

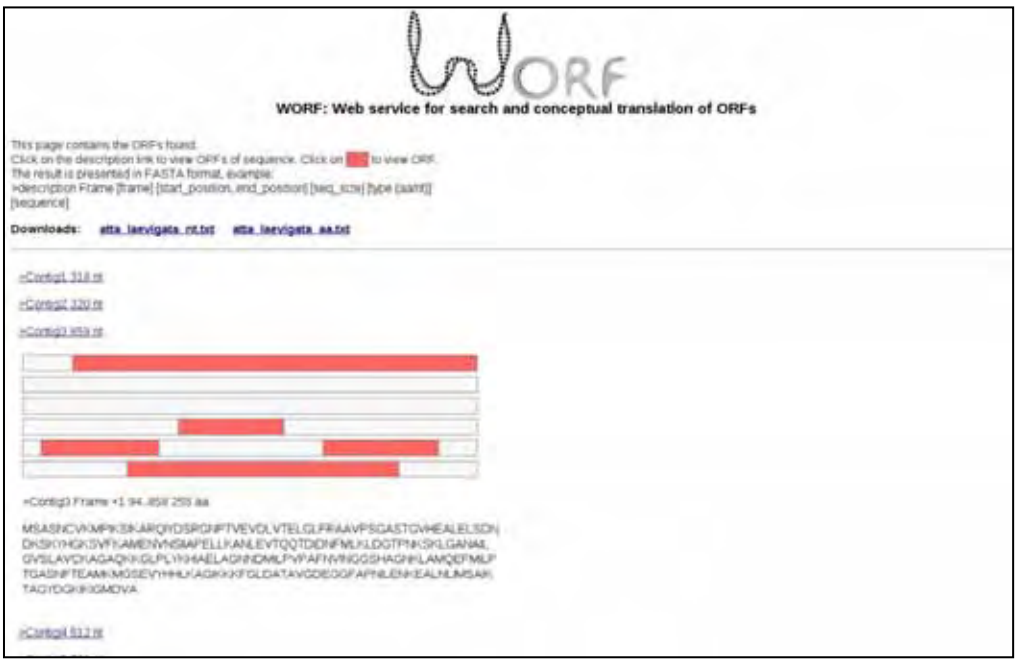
Interface Gráfica do WORF

Foi elaborada uma interface gráfica para o WORF e disponibilizada no endereço: http://omega.rc.unesp.br:8083/worfsite/. Nessa ferramenta é possível executar grande volume de sequências pré-processadas. A Figura 24 apresenta a interface gráfica de acesso ao WORF.

Figura 24 - Tela inicial para execução do WORF.

Conforme mostra a Figura 24, o usuário tem a opção de fazer a busca e tradução de ORFs ou apenas a tradução completa das sequências, usando o código genético de interesse. Os resultados ficam disponíveis para *download* e, além disso, cada sequência pode ser visualizada graficamente com todas as suas ORFs, sequências em formato FASTA com seus respectivos *frames*, coordenadas e tamanho em aminoácidos (Figura 25).

Figura 25 - Tela com os resultados do WORF.



Desenvolvimento de Componentes do *Pipeline* de ESTs

Usando os mesmos testes desenvolvidos com a ferramenta WORF, foram desenvolvidos mais quatro componentes do *pipeline* para análise de ESTs. Estes componentes executam as seguintes tarefas:

1. Alinhamento das ORFs traduzidas no banco de dados NR do GenBank, usando o programa blastp da ferramenta de terceiro BLAST;
2. Para busca de domínios de proteínas em uma série de bases de dados diferentes, usando o InterProScan;
3. Busca de peptídeo sinal usando a ferramenta de terceiro Signalp;
4. Busca de domínio transmembranar usando TMHMM.

Esses componentes foram elaborados usando a linguagem Java e o protocolo REST de serviço *web.*

4.6 Ferramentas Existentes para Análise de Bibliotecas 16S

Existem descritas na literatura algumas ferramentas para análise de dados provenientes de bibliotecas 16S, como DOTUR, UniFrac, TreeClimber, SONS, FastGroup e uma versão mais recente FastGroupII.

Embora os programas FastGroup e FastGroupII realizem a etapa de pré-processamento para pequeno volume de sequências, optou-se pelo uso do sistema EGene, por já ser utilizado em rotinas em nosso laboratório. No caso de grande volume de sequências, conforme explicado na página 66 deverá ser verificada a tecnologia usada para decidir pela forma de realizar o pré-processamento.

As outras ferramentas de anotação foram verificadas e analisadas para selecionar o que seria melhor para compor o *pipeline*. Muitos trabalhos, principalmente na área de Microbiologia, têm sido feitos usando essas ferramentas. O programa DOTUR gera OTUs (Unidade Operacionais Taxonômicas), constroi curvas de rarefação que permitem a comparação de amostras, mesmo que com intensidades amostrais diferentes e gera índices de riqueza e diversidade. A ferramenta UniFrac compara a distância filogenética entre as comunidades para detectar as diferenças de estrutura dessa comunidade. Já TreeClimber utiliza um método baseado em parcimônia para determinar se duas ou mais comunidades apresentam a mesma estrutura. O programa SONS calcula a fração e riqueza de OTUs compartilhadas entre as comunidades. A ferramenta FastGroup realiza um agrupamento de sequências e retira sequências duplicadas das bibliotecas de 16S rDNA; o FastGroupII realiza as mesmas tarefas deixando o arquivo pronto para ser analisado por banco de dados que classificam os organismos.

Em 2009, Schloss e colaboradores reuniram todas as ferramentas apresentadas acima, exceto FastGroup e FastGroupII, em um único programa

denominado MOTHUR. Isso facilitou muito a análise dos dados pelos usuários, pois diminuiu a quantidade de *softwares* a serem compreendidos e instalados individualmente. Há uma série de trabalhos que realizam análises de diversidade de micro-organismos, utilizando o programa MOTHUR. Dentre eles podemos destacar: estudo de bactérias presentes na superfície ocular de humanos (DONG *et al*., 2011) e diversidade de Archaea halófilas cultiváveis em Rambla Salada, Espanha, uma área de grande interesse ecológico (LUQUE *et al*., 2012).

Dessa forma, foi utilizado o programa MOTHUR para compor o *pipeline* para análise de bibliotecas 16S. Como a entrada do MOTHUR é um multifasta de sequências alinhadas, foi usado o programa ClustalW para realizar o alinhamento múltiplo de sequências.

4.7 Elaboração do Protocolo para Análise de 16S

Com a concatenação das tarefas pelo MOTHUR, antes executadas por várias ferramentas, as ferramentas e as atividades que compõem o *pipeline* denominado 16Scan foram:

1. Checagem das sequências de entrada (FASTA) verificando se o formato está correto e se não há sequências repetidas,

2. Verificação de sequências quimeras usando o programa UCHIME v 4.2.40 (EDGAR *et al*., 2011),

3. Alinhamento múltiplo das sequências não quimeras usando o programa ClustalW2,

4. Corte das extremidades das sequências alinhadas, visando retirar excesso de *gaps* das extremidades e manter mesmo tamanho das sequências,

5. Execução do MOTHUR para geração dos índices de diversidade e riqueza de espécies,

6. Geração dos gráficos usando a linguagem R para os índices de diversidade, riqueza e curva de rarefação,

7. Identificação das sequências representativas de cada OTU pelo banco de dados RDP - *Ribosomal Database Project* (http://rdp.cme.msu.edu/).

É importante ressaltar que a execução do programa MOTHUR é inteiramente via linha de comando, o que não é muito amigável ao usuário. Dessa forma, a implementação de um *web service*, torna acessível ao usuário através de uma interface gráfica o uso da ferramenta.

4.8 Implementação do *Pipeline* para Análise de 16S

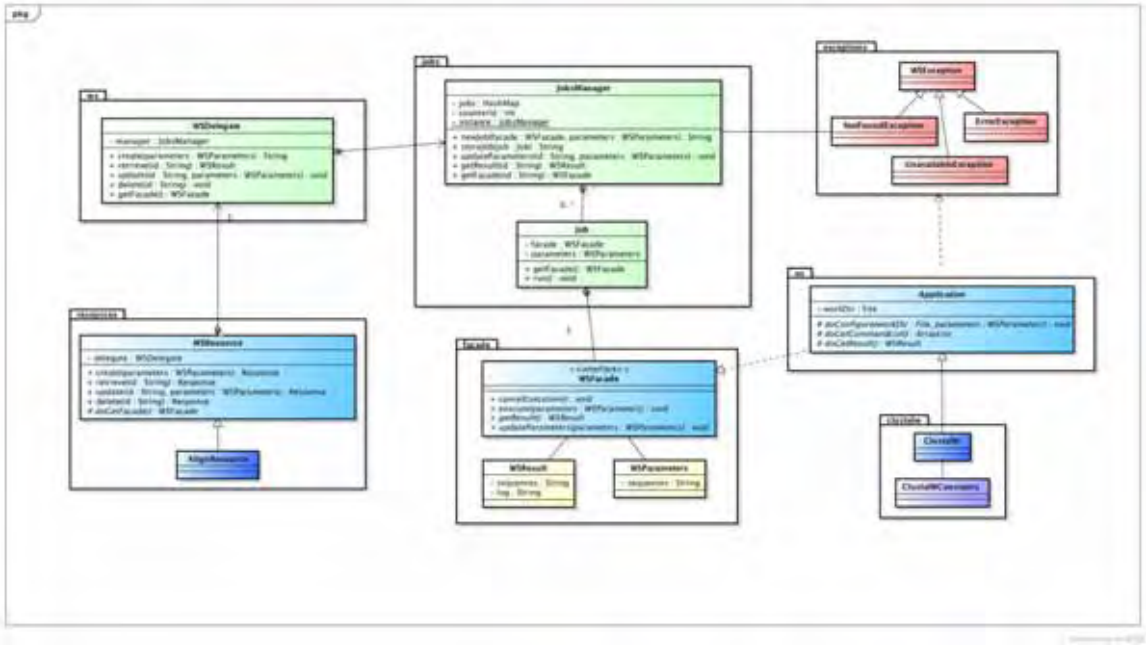
Com o protocolo definido, foi desenvolvido um *pipeline* que integrou os programas ClustalW2 (versão mais recente do ClustalW) e posteriormente o MOTHUR, bem como os *scripts* requeridos na análise. Inicialmente, os princípios de arquitetura REST foram utilizados para a criação de um serviço *web* que encapsula o programa ClustalW.

O serviço *web* disponibilizado pelo EBI - *European Bioinformatics Institute* (http://www.ebi.ac.uk/Tools/webservices/) serviu como modelo para o desenvolvimento desses componentes.

Devido às características do componente desenvolvido para alinhamento múltiplo, este recebeu o nome de MSA-WS, que significa *Multiple Sequence*

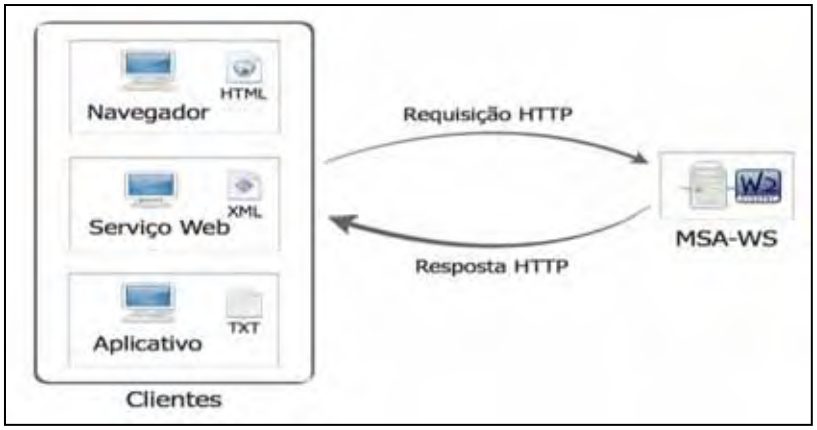
Alignment Web Service ou serviço web para alinhamento múltiplo de sequências. A Figura 26 apresenta as classes Java que compõem esse serviço.

Figura 26 - Diagrama de Classes do MSA-WS.



O MSA-WS disponibiliza o programa ClustalW2 como um recurso que pode ser consumido por diversos tipos de cliente, de acordo com a representação solicitada. Para cada nova requisição é gerado um identificador que verifica o estado da solicitação e obtém os resultados disponíveis. Todas as requisições são executadas em paralelo utilizando *threads* (Figura 27).

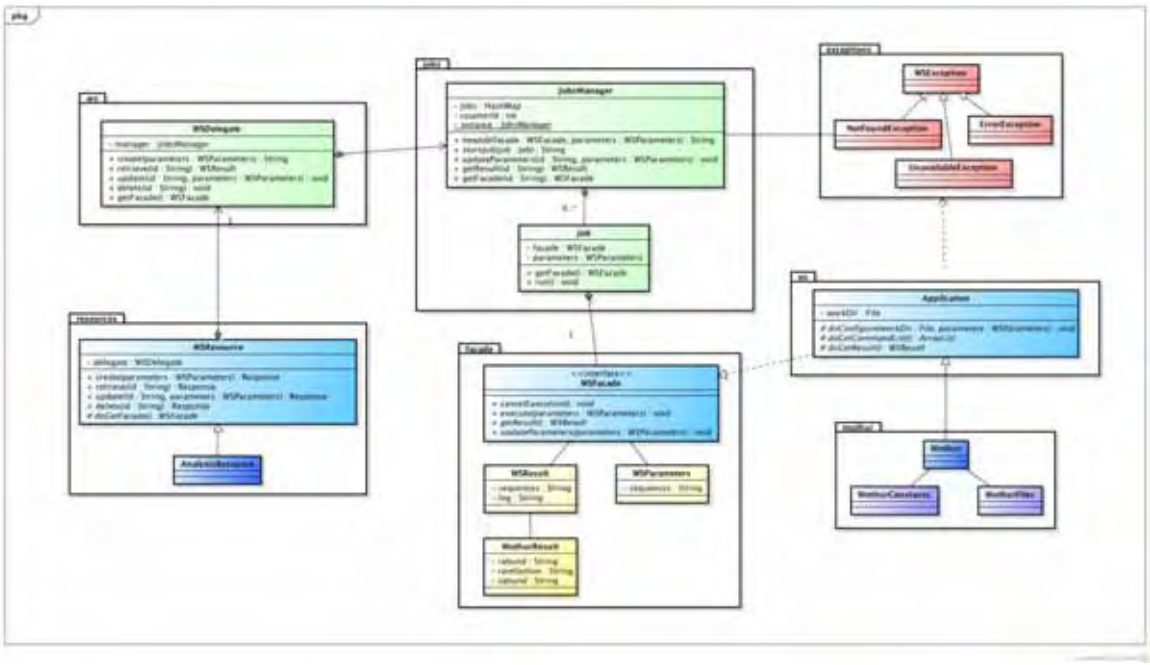
Figura 27 - Arquitetura do MSA-WS.



Foram realizados testes empíricos com o MSA-WS, que mostraram ser possível processar grande volume de sequências, sem problemas de conexão.

Usando o mesmo princípio do desenvolvimento do ClustalW2, foi possível criar o *web service* usando a ferramenta de terceiro MOTHUR. Com o framework desenvolvido, foi possível criar esse componente de forma mais rápida. A Figura 28 apresenta o diagrama de classes Java para esse componente.

Figura 28 - Digrama de classes para o componente que executa o MOTHUR.



Com os dois componentes elaborados, foi possível realizar as conexões necessárias para criar o *pipeline* para análise de sequências 16S. Para facilitar o uso do *pipeline*, foi desenvolvida uma interface gráfica, amigável ao usuário, que facilita a escolha de parâmetros e execução dos programas. Assim o usuário não se preocupa com a execução via linha de comando. 16Scan foi validado com o uso de sequências reais 16S rDNA geradas em projetos no nosso laboratório.

4.9 Elaboração do Protocolo para Análise de Bibliotecas ITS

O protocolo elaborado para análise de sequências ITS seguiu o mesmo padrão do protocolo de ITS, exceto pela alteração no programa para busca de quimeras e identificação das sequências. Os passos são:

1. Checagem das sequências de entrada (FASTA) verificando se o formato está correto e se não há sequências repetidas no programa,

2. Verificação de sequências quimeras usando o programa ChimeraChecker (NILSSON *et al*., 2010),

3. Alinhamento múltiplo das sequências não quimeras usando o programa ClustalW2,

4. Corte das extremidades das sequências alinhadas, visando retirar excesso de *gaps* das extremidades e manter mesmo tamanho das sequências,

5. Execução do MOTHUR para geração dos índices de diversidade e riqueza de espécies usando o programa MOTHUR,

6. Geração dos gráficos usando a linguagem R para os índices de diversidade, riqueza e curva de rarefação,

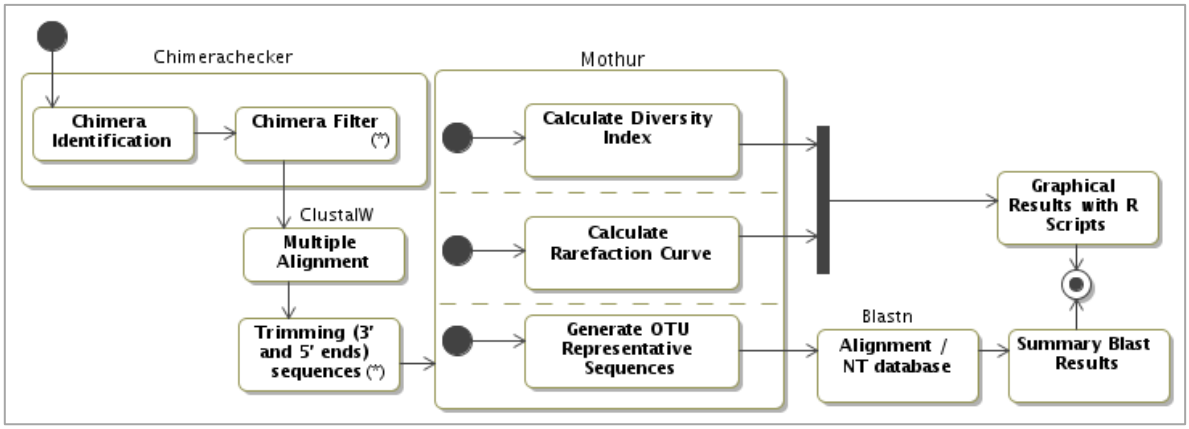
7. Identificação das sequências representativas de cada OTU pelo banco de dados NR do Genbank.

4.10 Implementação do *Pipeline* para Análise de ITS

Um novo *pipeline* denominado ITScan, agora usando como entrada sequências ITS de fungos, foi construído aproveitando alguns componentes como ClustalW2 e MOTHUR previamente desenvolvidos para análise se sequências 16S. ITScan foi desenvolvido seguindo o mesmo padrão do bESTscan e 16Scan, estando acoplado no mesmo modelo de arquitetura.

ITScan possui uma interface gráfica amigável ao usuário e foi validado com sequências reais ITS geradas em projetos no nosso laboratório. A Figura 29 apresenta um diagrama UML/Statecharts mostrando todas as etapas, com seus respectivos programas utilizados, entrada e saída de arquivos.

Figura 29 - Diagrama UML/Statecharts do *pipeline* ITScan, onde (*) indica pontos que admitem inspeção manual e intervenção.



Para realizar a identificação das sequências ITS, foi usado o programa blastn, com os parâmetros *evalue* igual a e-03 e o banco de dados nr (GenBank) de nucleotídeos, onde a entrada, assim como no 16Scan, são as sequências representativas de cada OTU gerada pelo MOTHUR.

Conclusões

5. CONCLUSÕES

- Foi desenvolvido um modelo de arquitetura baseado no padrão MVC já bem estudado e conhecido;

- Foram elaborados protocolos para a anotação de sequências EST, bibliotecas 16S e ITS;

- Os protocolos foram implementados na forma de *pipelines* e acoplados no modelo de arquitetura elaborado;

- Foi desenvolvida uma ferramenta para a busca e tradução conceitual de ORFs, denominada WORF, e nesta ferramenta foram aplicados testes de desempenho computacional, servindo como modelo para desenvolver todos os componentes dos *pipelines*;

- Com os testes de desempenho do WORF, definiu-se pelo uso da linguagem de programação Java e serviço *web* em REST;

- O *pipeline* para análise de ESTs recebeu o nome de bESTscan;

- O *pipeline* para análise de 16S recebeu o nome de 16Scan;

- O *pipeline* para análise de ITS recebeu o nome de ITScan;

- Os *pipelines* apresentam interface gráfica amigável, estão disponíveis para uso via *web* e possuem Manual de uso (ANEXOS);

- Os sistemas foram validados com sequências reais geradas por nosso grupo de pesquisa em nosso laboratório;

- Os *pipelines* já estão sendo utilizados em projetos em vigência no laboratório.

Aplicações

APLICAÇÕES

Os scripts e *pipelines* desenvolvidos no presente trabalho estão sendo utilizados por alunos e pesquisadores, tanto da UNESP-Rio Claro como de outras instituições. Alguns produtos dessa utilização são listados abaixo:

1. Artigo publicado:

RODOVALHO, C.M.; FERRO, M.; FONSECA, F.P.P.; ANTONIO, E.A.; GUILHERME, I.R.; HENRIQUE-SILVA, F.; BACCI, M. Expressed sequence tags from Atta laevigata and identification of candidate genes for the control of pest leaf-cutting ants. BMC Research Notes, v. 4, p. 203, 2011.

2. Artigo aceito:

RODRIGUES, A.; PASSARINI, M.R.Z.; FERRO, M.; NAGAMOTO, N.S.; FORTI, L.C.; BACCI, M.; SETTE, L.D.; PAGNOCCA, F.C. Fungal communities in the garden chamber soils of leaf-cutting ants. Journal of Basic Microbiology, 2013.

3. Projetos usuários:

- Tássio Brito, Mestrado e Doutorado.
- Joana D. Mantovani, Mestrado, Processo FAPESP 2011/04053-6.
- Ana Carolina Marchiori, Doutorado, Processo FAPESP 2009/09258-5.
- Cynara M. Rodovalho, Pós-doutorado, Processo FAPESP 2011/06367-8.

Referências

REFERÊNCIAS

ADAMS, M. D.; KELLEY, J. M.; GOCAYNE, J. D.; DUBNICK, M.; POLYMEROPOULOS, M. H.; XIAO, H.; MERRIL, C. R.; WU, A.; OLDE, B.; MORENO, R. F.; KERVALAGE, A. R.; MCCOMBIE, W. R.; VENTER, J. C. Complementary DNA sequencing: expressed sequence tags and human genome project. *Science*, Washington, v. 252, n. 5013, p. 1651-1656, jun. 1991.

ADAMS, M.D. *et al*. The genome sequence of Drosophila melanogaster. Science, Washington, v. 287, n. 5461, p. 2185-2195, mar. 2000.

ALTSCHUL, S.F.; MADDEN, T.L.; SCHÄFFER, A.A.; ZHANG, J.; ZHANG, Z.; MILLER, W.; LIPMAN, D.J. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*., 25:3389-3402, 1997.

ANJOS, N.; DELLA LÚCIA, T.M.; MAYHÉ-NUNES, A.J. Guia prático sobre formigas cortadeiras em reflorestamentos. Ponte Nova: Editora GarffCor. 1998. 97 p.

BARNEY, B. Introduction to Parallel Computing. Lawrence Livermore National Laboratory. Disponível em <https://computing.llnl.gov/tutorials/mpi/>. Acessado em 20 de fev. 2013

BASS, M.; CHERRETT, J.M. Fungal hyphae as a source of nutrients for the leafcutting ant Atta sexdens. *Physiol. Entomol*., v. 20, n. 1, p. 1-6, mar. 1995.

BELLEMAIN, E.; CARLSEN, T.; BROCHMANN, C.; COISSAC, E.; TABERLET, P.; KAUSERUD, H. ITS as an environmental DNA barcode for fungi: an in silico approach reveals PCR biases. *BMC Microbiology*, 10:189, 2010.

BENDTSEN, J.D.; NIELSEN, H.; VON HEIJNE, G. AND BRUNAK S. Improved prediction of signal peptides: SignalP 3.0. *J Mol Biol*. 340(4):783-95, 2004.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. UML - Guia do Usuário. Editora Campus. 2ª Edição, 2006.

BOLTON, B.; ALPERT, G.; WARD, P. S.; NASKRECKI, P. Bolton's catalogue of ants of the world: 1758-2005. Cambridge: Harvard Unversity Press, 2006.

BOUCK, A.; VISION, T. The molecular ecologist's guide to expressed sequence tags. *Mol. Ecol*., v. 16, n.5, p. 907-924, mar. 2007.

BRUNS, T.D.; WHITE, T.J.; TAYLOR, J.W. Fungal Molecular Systematics. *Annu. Rev. Ecol. Syst.,* v. 22, p.525-564, 1991.

CAMACHO, C. *et al*. BLAST+: architecture and applications. *BMC Bioinformatics*, 10:421, 2009.

CAMERON, R.S. Distribution, impact and control of the Texas leaf-cutting ant: 1983 survey results. *Texas Forest Service Publication*, 1985.

CHEN, T.W.; GAN, R.C.R.; WU, T.H.; HUANG, P.J.; LEE, C.Y.; CHEN, Y.Y.M.; CHEN, C.C.; TANG, P. FastAnnotator: an efficient transcript annotation web tool. *BMC Genomics*, 13:Suppl 7, S9, 2012.

CONESA, A.; GOTZ, S.; GARCIA-GOMEZ, J.M.; TEROL, J.; TALON, M.; ROBLES, M. Blast2GO: a universal tool for annotation, visualization and analysis in functional genomics research. *Bioinformatics*, 21:3674-3676, 2005.

CURRIE, C.R.; SCOTT, J.A.; SUMMERBELL, R.C.; MALLOCH, D. Fungus-growing ants use antibiotic-producing bacteria to control garden parasites. *Nature*, London, v. 398, p. 701-704, abr. 1999.

CHEVREUX, B.; PFISTERER, T.; DRESCHER, B.; DRIESEL, A. J.; MÜLLER, W. E.; WETTER, T.; SUHAI, S. Using the miraEST Assembler for Reliable and Automated mRNA Transcript Assembly and SNP Detection in Sequenced ESTs. *Genome Research*, 14(6), 2004.

DÁVILA, A.M.R.; LORENZINI, D.M.; MENDES, P.N.; SATAKE, T.S.; SOUSA, G.R.; CAMPOS, L.M. *et al*. GARSA: genomic analysis resources for sequence annotation. *Bioinformatics* 21(23): 4302-4303, 2005.

DELCHER, A.L. *et al*. Alignment of whole genomes. *Nucleic Acids Research*, v. 27, 11, 1999.

DONG, Q.; BRULC, J.M.; IOVIENO, A.; BATES, B.; GAROUTTE, A.; MILLER, D.; REVANNA, K.V.; GAO, X.; ANTONOPOULOS, D.A.; SLEPAK, V.Z.; SHESTOPALOV, V.I. Diversity of Bacteria at Healthy Human Conjunctiva. *IOVS*, v. 52, n. 8, p. 5408-5413, 2011.

DURHAM, A.M.; KASHIWABARA, A.Y.; MATSUNAGA, F.T.; AHAGON, P.H.; RAINONE, F.; VARUZZA, L.; GRUBER, A. EGene: a configurable pipeline generation system for automated sequence analysis. *Bioinformatics* 21(12):2812-3, 2005.

EDGAR, R.C.; HAAS, B.J.; CLEMENTE, J.C.; QUINCE, C.; KNIGHT, R. UCHIME improves sensitivity and speed of chimera detection, *Bioinformatics*, 2001.

EWING, B.; GREEN, P. Basecalling of automated sequencer traces using phred. II. Error probabilities. *Genome Research*, 8:186-194, 1998.

FERRO, M. Desenvolvimento e validação de protocolos para a anotação automática de sequências ESTs de Eimeria spp. de galinha doméstica. 125 f. Dissertação (Mestrado em Ciências) – Instituto de Ciências Biomédicas, Universidade de São Paulo, São Paulo, 2008.

FONTANA, S.; KESHAVMURTHY, S.; HSIEH, H.J.; DENIS, V.; KUO, C.Y.; HSU, C.M.; LEUNG, J.K.L.; TSAI, W.S.; WALLACE, C.C.; Chen, C.A. Molecular Evidence Shows Low Species Diversity of Coral-Associated Hydroids in Acropora Corals. *Plos One*, 7(11): e50130, 2012.

FOWLER, M. **Patterns of Enterprise Application Architecture**. Addison-Wesley. ISBN 978-0-321-12742-6, 2002.

FOWLER, H. G.; SILVA, V. P.; SAES, N. B. Population dynamics of leaf-cutting ants: a brief review. In: LOFGREN, C. S.; VANDER MEER, R. K. (Eds.). Fire ants and leaf-cutting ants: biology and management, Boulder, Colorado: *West-View Press*, p. 123-145, 1986.

GARRETTSON, M.; STETZEL, J.F.; HALPERN, B.S.; HEARN, D.J.; LUCEY, B.T.; MCKONE, M.J. Diversity and abundance of understorey plants on active and abandoned nests of leaf-cutting ants (Atta cephalotes) in a Costa Rican rain forest. *J. Trop. Ecol*., Cambridge, v. 14, n. 1, p. 17-26, jan. 1998.

GENE ONTOLOGY CONSORTIUM. The Gene Ontology (GO) project in 2006. *Nucleic Acids Res*., 1;34(Database issue):D322-6, 2006.

GOODISMAN, M.A.D.; ISOE, J.; WHEELER, D.E.; WELLS, M.A. Evolution of insect metamorphosis: a microarray-based study of larval and adult gene expression in the ant Camponotus festinatus. *Evolution*, v. 59, n. 4, p. 858-870, abr. 2005.

GORDON, D.; ABAJIAN, C. AND GREEN, P. Consed: a graphical tool for sequence finishing. *Genome Research*, 8:195-202, 1998.

GRAMA, A.; KARYPIS, G.; KUMAR, V.; GUPTA, A. Introduction to Parallel Computing (2nd Edition). Addison Wesley, 2003.

GRIFFITHS, A. J. F.; MILLER, J. H.; SUZUKI, D. T.; LEWONTIN, R. C.; GELBART, W. M. **Introdução à genetica**. 6 ed. Tradução de P.A. Motta. Rio de Janeiro: Guanabara Koogan, 1998. 856 p.

GRUBER, A. **Expressed Sequence Tags**. In: Dear PH, editor. Bioinformatics. USA, Scion Publishing; 2007. p. 141-167.

GUO, T., GUIYANG, L.; DEATON, R. (2008). “Accelerating Computation of DNA Sequence in Distributed Environment”. In: Advanced Intelligent Computing Theories and Applications. With Aspects of Contemporary Intelligent Computing Techniques. 4th International Conference on Intelligent Computing, ICIC 2008 Shanghai, China, September 15-18, Proceedings.

GUO, Y.; RIBEIRO, J.M.; ANDERSON, J.M.; BOUR, S. dCAS: a desktop application for cDNA sequence annotation. *Bioinformatics*. 25(9):1195-6, 2009.

HÖLLDOBLER, B.; WILSON, E.O. **The Ants**. Massachusetts: Belknap Press of Harvard University, 1990. 732p.

HUANG X.; MADAN, A. CAP3: A DNA sequence assembly program. *Genome Res*., 9:868-877, 1999.

KALIN, M. **Java Web Services: Up and Running**. Sebastopol: O’Reilly Media, 2009.

KATAYAMA, T.; NAKAO, M.; TAKAGI, T. TogoWS: integrated SOAP and REST APIs for interoperable bioinformatics Web services. *Nucleic Acids Research*, 38:W706–W711, 2010.

KELLER, L. Queen lifespan and colony characteristics in ants and termites. Insectes Soc., Basel, v. 45, n. 3, p. 235–246, 1998.

KORF, I.; YANDELL, M.; BEDELL, J. **BLAST**. Ed. O'Reilly Media, 2003, p. 368.

KROGH, A.; LARSSON, B.; VON HEIJNE, G.; SONNHAMMER, E.L. Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes. *J Mol Biol*., (3):567-80, 2001.

LI, K. ClustalW-MPI: ClustalW analysis using distributed and parallel computing. *Bioinformatics,* v. 9, 12, p.1585, 2003.

LOZUPONE, C.; LLADSER, M.E.; KNIGHTS, D.; STOMBAUGH, J.; KNIGHT, R. UniFrac: an effective distance metric for microbial community comparison. *ISME J*, 5(2): 169–172, 2011.

LUO *et al*. SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. *GigaScience*, 1:18, 2012.

LUQUE, R.; GONZÁLEZ-DOMENECH, C.M.; LLAMAS, I.; QUESADA, E. AND BÉJAR, V. Diversity of culturable halophilic archaea isolated from Rambla Salada, Murcia (Spain). Extremophiles. 2012 Jan 5. [Epub ahead of print]

METZKER, M.L. Sequencing technologies — the next generation. *Nature Reviews Genetics*, 11, 31-46, 2010.

MITA, K. *et al*. The genome sequence of silkworm Bombyx mori. *DNA Res*., Oxford, v. 11, n. 1, p. 27-35, fev. 2004.

MUELLER, U.G.; SCHULTZ, T.R.; CURRIE, C.R.; ADAMS, R.M.M. The origin of the attine ant-fungus mutualism. *Q. Rev. Biol*., v. 76, n. 2, p. 169-197, jun. 2001.

NILSSON, R.H.; ABARENKOV, K.; VELDRE, V.; NYLINDER, S.; DE WIT, P.; BROSCHÉ, S.; ALFREDSSON, J.F.; RYBERG, M. AND KRISTIANSSON, E. An open source chimera checker for the fungal ITS region. Molecular Ecology Resources 10: 1076–1081, 2010.

ORT, B.S.; BANTAY, R.M.; PANTOJA, N.A.; O'GRADY, P.M. Fungal diversity associated with hawaiian Drosophila host plants. *PLoS One*, 7(7): E40550, 2012.

PEARSON, W. Finding protein and nucleotide similarities with FASTA. Curr Protoc Bioinformatics, Chapter 3:Unit3.9, 2004.

PETTI, C.A.; POLAGE, C.R; SCHRECKENBERGER, P. The Role of 16S rRNA Gene Sequencing in Identification of Microorganisms Misidentified by Conventional Methods. *J Clin Microbiol*. 43(12): 6123–6125, 2005.

PRESSMAN, R.S. **Engenharia de software**. Rio de Janeiro. McGraw-Hill, 2002.

QUEVILLON, E.; SILVENTOINEN, V.; PILLAI, S.; HARTE, N.; MULDER, N.; APWEILER, R.; LOPEZ, R. InterProScan: protein domains identifier. *Nucleic Acids Res*., 33:W116-120, 2005.

RICHARDSON, L.; RUBY, S. **RESTful Web Services:** Web Services for the Real World. Sebastopol: O'Reilly Media, 2007.

SANDERS, J.; KANDROT, E. **CUDA by example:** an introduction to general-purpose GPU programming. Ed. Addison-Wesley, 2010.

SALZBERG, S.L.; DELCHER, A.L.; KASIF, S.; WHITE, O. Microbial gene identification using interpolated Markov models. Nucleic acids research 26 (2): 544–548, 1998.

SEGURITAN, V.; ROHWER, F. FastGroup: A program to dereplicate libraries of 16S rDNA sequences. BMC Bioinformatics, 2:9, 2001.

SCHATZ, M. C. *et al*. High-throughput sequence alignment using Graphics Processing Units. *BMC Bioinformatics*, 8:474, 2007.

SCHWARTZ, R.L.; PHOENIX, T. Learning Perl. Sebastopol: O'Reilly Media, 2011.

SCHOCH, C.L.; SEIFERT, K.A.; HUHNDORF, S.; ROBERT, V.; SPOUGE, J.L., *et al*. Nuclear ribosomal internal transcribed spacer (ITS) region as a universal DNA barcode marker for Fungi. *PNAS* 109: 6241–6246, 2012.

SCHMID, R.; BLAXTER, M.L. annot8r: GO, EC and KEGG annotation of EST datasets. *BMC Bioinformatics,* 9: 180, 2008.

SCHLOSS, D.; HANDELSMAN, J. Introducing DOTUR, a Computer Program for Defining Operational Taxonomic Units and Estimating Species Richness. *Applied and Environmental Microbiology*, p. 1501-1506, 2004.

SCHLOSS, D.; HANDELSMAN, J. Introducing TreeClimber, a Test to Compare Microbial Community Structures. *Applied and Environmental Microbiology*, p. 2379-2384, 2006a.

SCHLOSS, D.; HANDELSMAN, J. Introducing SONS, a Tool for Operational Taxonomic Unit-Based Comparisons of Microbial Community Memberships and structures. *Applied and Environmental Microbiology*, p. 6773-6779, 2206b.

SIMPSON, J.T.; WONG, K.; JACKMAN, S.D.; SCHEIN, J.E.; JONES, S.J.; BIROL, I. ABySS: A parallel assembler for short read sequence data. *Genome Research*, 2009.

SMITH, R.P.; BUCHSER, W.J.; LEMMON, M.B.; PARDINAS, J.R.; BIXBY, J.L. & LEMMON, V.P. EST Express: PHP/MySQL based automated annotation ESTs from expression libraries. *BMC Bioinformatics,* 9:186, 2008.

SMITH, T.F.; WATERMAN, M.S. Identification of Common Molecular Subsequences. *Journal of Molecular Biology*, 1981, 147: 195–197.

SNELL, J.; TIDWELL, D.; KULCHENKO, P. **Programming Web Services with SOAP**. Ed. O'Reilly Media, 2001, p. 264.

STEIN, L. Genome annotation: from sequence to biology. *Nat Rev Genet*. 2(7):493-503, 2001.

SUBRAMANIAM, V. **Programming Concurrency on the JVM**: Mastering Synchronization, STM, and Actors. Ed. The Pragmatic Bookshelf, 2011, p. 282.

SCHULTZ, T.R.; BRADY, S.G. Major evolutionary transitions in ant agriculture. *Proceedings of the National Academy of Sciences*, U.S.A, 2008, 105:5435–5440.

THE HONEYBEE GENOME SEQUENCING CONSORTIUM. Insights into social insects from the genome of the honeybee Apis mellifera. *Nature*, London, v. 443, n. 7114, p. 931-949, out. 2006.

THE Nasonia GENOME WORKING GROUP. Functional and Evolutionary Insights from the Genomes of Three Parasitoid Nasonia Species. *Science*, Washington, v. 327, n. 5963, p. 343-348, jan. 2010.

TIAN, H.; VINSON, S.B.; COATES, C.J. Differential gene expression between alate and dealate queens in the red imported fire ant, Solenopsis invicta Buren (Hymenoptera: Formicidae). *Insect Biochem. Mol. Biol.*, v. 34, n. 9, p. 937-949, set. 2004.

THOMPSON, J.D. et al. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res., 22, 4673–4680, 1994.

Tribolium GENOME SEQUENCING CONSORTIUM. The genome of the model beetle and pest Tribolium castaneum. *Nature*, London, v. 452, n. 7190, p. 949-955, abr. 2008.

UML. (2011), Unified Modelling Language. Disponível em <http://www.uml.org/>. Acesso em 05 de fev. de 2012.

VASCONCELOS, H.L. Foraging activity of two species of leaf-cutting ants (Atta) in a primary Forest of the Central Amazon. *Insectes Soc*., Basel, v. 37, n. 2, p. 131-145, 1990.

VAN DE PEER, Y.; CHAPELLE, S.; DE WACHTER, R. A quantitative map of nucleotide substitution rates in bacterial rRNA. *Nucleic Acids Res*. 24:3381–3391, 1996.

YU, Y.; BREITBART, M.; MCNAIRNIE, P.; ROHWER, F. FastGroupII: a web-based bioinformatics platform for analyses of large 16S rDNA libraries. BMC Bioinformatics, 7;7:57, 2006.

ZERBINO, D.R.; BIRNEY, E. Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*, 18 (5): 821–829, 2008.

WANG, J.; JEMIELITY, S.; PAOLO, U.; WURM, Y.; GRÄFF, J.; KELLER, L. An annotated cDNA library and microarray for large-scale gene-expression studies in the ant Solenopsis invicta. *Genome Biol*., v. 8, n. 1, R9, 2007.

WEBER, N.A. Gardening ants: the Attines. Philadelphia: The American *Philosophical Society*, 1972, 146 p.

WESTERMAN, L.J.; STEL, H.V.; SCHIPPER, M.E.I.; BAKKER, L.J.; NEEFJES-BORST, E.A.; VAN DEN BRANDE, J.H.M.; BOEL, E.C.H.; SELDENRIJK, K.A.; SIERSEMA, P.D.; BONTEN, M.J.M.; KUSTER, J.G. Development of a Real-Time PCR for Identification of Brachyspira Species in Human Colonic Biopsies. *Plos One*, 7(12): e52281, 2012.

WETTERER, J.K.; SCHULTZ, T.R.; MEIER, R. Phylogeny of fungus-growing ants (Tribe Attini) based on mtDNA sequence and morphology. *Mol. Phylogenet. Evol*., v. 9, n. 1, p. 42-47, fev. 1998.

WILSON, E.O. Caste and division of labor in leaf-cutter ants (Hymenoptera: Formicidae: Atta) I. The overall pattern in A. sexdens. *Behav. Ecol. Sociobiol*., v. 7, n. 2, p. 143–156, 1980.

WIRTH, R.; HERZ, H.; RYEL, R.J.; BEYSCHLAG, W.; HÖLLDOBLER, B. Herbivory of leaf-cutting ants. A case study on Atta colombica in the tropical rain forest of Panama. Berlin: Springer, 2003.

WHITE, J.R.; MADDOX, C.; WHITE, O.; ANGIUOLI, S.V.; FRICKE, W.F. CloVR-ITS: Automated internal transcribed spacer amplicon sequence analysis pipeline for the characterization of fungal microbiota. *Microbiome*, **1**:6, 2013.

Glossário

GLOSSÁRIO

BLAST

Basic Local Alignment Search Tools - *software* desenvolvido por Altschul *et al*. (1997) para buscas de similaridade de sequencias de DNA e proteínas através de alinhamentos locais.

CDD (*Conserved Domain Database*)

Base de dados (Marchler-Bauer *et al*., 2007) que contém uma coleção de matrizes de escores de posição específica (PSSMs) derivadas a partir de alinhamentos múltiplos de sequências protéicas, em regiões de domínios evolutivamente conservados.

Contigs

Sequencias resultantes da montagem de leituras de DNA contendo sobreposição entre si e que a partir de seu consenso, formam uma região “contígua”. No contexto de ESTs, são os as sequencias que foram reconstruídas a partir de várias leituras, ao contrário dos *singlets*, sequencias derivadas a partir de apenas uma leitura.

EMBL (*European Molecular Biology Laboratory*)

Originalmente o nome de um laboratório sediado em Heidelberg, na Alemanha. Atualmente, o laboratório europeu é o EBI (*European Bioinformatics Institute* - http://www.ebi.ac.uk/), sediado no Genome Campus em Hinxton, Inglaterra. A velha denominação é, contudo, ainda utilizada para designar um banco público de sequencias de DNA e proteínas, a versão européia do NCBI/GenBank americano.

ESTscan

Software que realiza predição de genes em sequencias de DNA (Iseli *et al*., 1999).

EST (*Expressed Sequence Tags*)

São etiquetas de sequências expressas, ou seja, sequencias obtidas a partir de bibliotecas de cDNAs de um tecido ou de um estágio evolutivo de um organismo, e que permitem caracterizar um subconjunto de genes expressos. São caracterizadas por serem resultantes de uma única leitura de DNA (*single-pass*).

Feature Table

Formato padrão de anotação de sequencias em bancos de dados internacionais como GenBank, DDBJ e EMBL (http://www.ncbi.nlm.nih.gov/collab/FT/).

GenBank

Banco de dados público de sequencias nucleotídicas e suas traduções protéicas. É hospedado no NCBI (*National Center for Biotechnology Information* - http://www.ncbi.nlm.nih.gov) em Bethesda, Estados Unidos.

***Gene Ontology* (GO)**

Ontologia gênica. Refere-se a um projeto dinâmico envolvendo um consórcio de laboratórios (http://www.geneontology.org/) que desenvolvem um vocabulário controlado de termos para produtos protéicos e suas interrelações. Os termos GO estão distribuídos em três ontologias: processo biológico, componente celular e função molecular (Blake e Harris, 2008; Ashburner *et al*., 2000).

InterProScan

Software (Zdobnov e Apweiler, 2001; Quevillon *et al.*, 2005) que realiza buscas de domínios protéicos, através da identificação de assinaturas a partir do InterPro (http://www.ebi.ac.uk/interpro/), uma coleção contém as seguintes bases de dados: Pfam, PROSITE, PRINTS, ProDom, SMART, TIGRFAMs, PIRSF, SUPERFAMILY, Gene3D, e PANTHER.

ORF (*Open Reading Frame*)

Quadro aberto de leitura. Refere-se a uma região de uma sequencia nucleotídica livre de códons de terminação e, portanto, potencialmente codificadora de proteína.

OTU

Refere-se a um nível definido e utilizado por taxonomistas para discutir ou comparar diferentes organismos. Podemos considerar que em uma OTU um grupo de sequências podem ser agrupadas por possuírem uma mesma porcentagem de identidade de suas bases.

Peptídeo-sinal

Sequencia sinalizadora canônica, presente na porção amino-terminal de uma proteína, que possui pelo menos um aminoácido carregado positivamente e uma região altamente hidrofóbica que forma o centro da sequencia sinalizadora. Apresenta os sinais que irão dirigir a proteína sintetizada para seu destino final e sofre uma clivagem após o transporte da proteína.

Pipeline

Fluxo de processos encadeados através das suas saídas padrão, de forma que a saída de um processo é utilizada como entrada do processo seguinte (Fonte: http://www.babylon.com/).

Reconstrução de transcritos

Processo que compreende a geração de sequencias reconstruídas a partir de dados fragmentados de leituras de ESTs. O processo pode ser realizado em duas etapas: o agrupamento de sequencias de cDNA que compartilham similaridade entre si (também chamada de *clustering*); a montagem dos fragmentos de cada grupo com programas como CAP3 ou Phrap. Em alguns casos, especialmente quando o número total de leituras é relativamente baixo, o processo pode ser feito diretamente através da montagem.

Regiões hipervariáveis

Correspondem aos trechos de sequências de um gene que são mais tolerantes de sofrer mutações ao longo do tempo e, portanto, acumulam mais alterações na sequência de nucleotídeos.

Regiões repetitivas seriadas (*tandem repeats*)

Duas ou mais cópias contínuas de um padrão de sequencia que constitui a unidade repetitiva. O tamanho da unidade repetitiva também é chamada de período. Ex: CAGCAGCAGCAG (unidade repetitiva = CAG, período = 3, número de cópias = 4)

Domínio transmembranar

Região de uma proteína constituída de uma □-hélice predominantemente hidrofóbica, contendo cerca de 20 a 30 resíduos de aminoácidos, e que atravessa a bicamada fosfolipídica de uma membrana. Proteínas transmembranares apresentam três tipos de regiões: citossólica, extracelular e transmembranar.

SignalP

Software para a busca de regiões de peptídeo sinal em sequencias de proteínas (Bendtsen *et al.*, 2004).

Singlets

São sequencias resultantes de um processo de montagem que não apresentaram sobreposição com nenhuma outra sequencia em um processo de montagem. No contexto de ESTs, são os *clusters* (grupos) compostos por apenas uma leitura.

TMHMM

Software baseado no uso de modelos ocultos de Markov (HMM) para a busca de domínios transmembranares em sequencias protéicas (Krogh *et al.*, 2001).

XML (eXtensible Markup Language)

Formato padrão de linguagem capaz de descrever diversos tipos de dados com o uso de etiquetas (*tags*) e uma estrutura hierárquica de dados. Seu propósito principal é a facilidade de compartilhamento de informações através da Internet.

Anexos

ANEXO 1

Manual bESTscan

Manual do bESTscan

O *pipeline* bESTcan foi desenvolvido visando a anotação automatizada de sequências transcriptômicas EST. Está disponível para uso em: http://evol.rc.unesp.br:8083/bestscan. Possui uma interface gráfica (Figura 1) que facilita a inserção dos arquivos, a escolha de parâmetros de análise, bem como a execução de cada etapa do processamento dos dados.

Figura 1 - Tela inicial do bESTscan.

Conforme mostra a Figura 1, podemos observar que a primeira etapa é o *upload* do arquivo de sequências pré-processadas no formato FASTA.

O *pipeline* bESTscan é composto por:

1. Achador de ORFs (WORF),
2. Busca de identidade de sequências usando o programa BLASTN (ALTSCHUL et al., 1997) contra a base de dados NR do GenBank,
3. Busca de domínio protéico usando o InterProScan (QUEVILLON *et al*., 2005),
4. Busca de peptídeo sinal utilizando SignalP (BENDTSEN *et al.*, 2004),
5. Busca de domínio transmembranar usando TMHMM (KROGH *et al*., 2001).

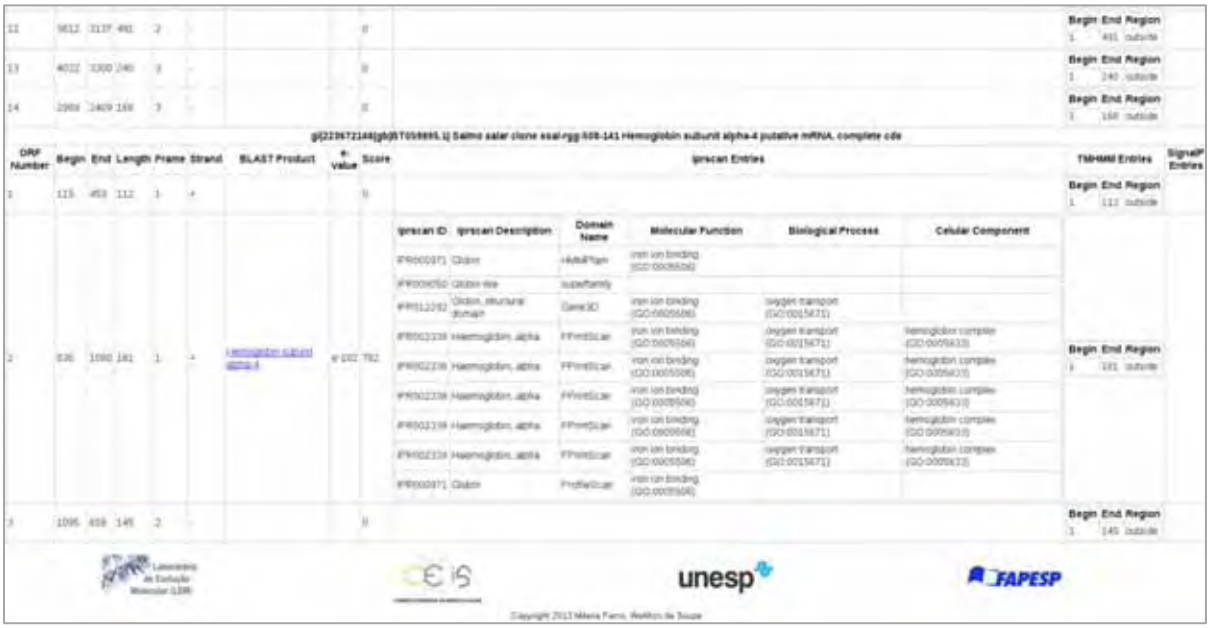
Como nesse *pipeline* só há duas telas sendo uma inicial e a final com os resultados, o usuário escolhe todos os parâmetros na tela inicial.

O achador de ORFs (WORF) permite selecionar o código genético sendo que há um link que apresenta todas as tabelas de código genético para que o usuário possa escolher a correta. Pode selecionar se a ORF se iniciará com Códon de Iniciação padrão ou com um alternativo e também o tamanho mínimo da ORF para ser apresentada, sendo o *default* 100 pb.

No caso do BLAST o único parâmetro que pode ser alterado é o Evalue sendo que o valor *default* é 1e-06.

Para o InterProscan são usados todos os banco de dados disponíveis (BlastProDom, HMMTigr, FPrintScan, ProfileScan, HMMPIR, HAMAP, HMMPanther, HMMPfam, PatternScan, Gene3D, HMMSmart e SuperFamily) e os dois aplicativos (TMHMM e SignalPHMM).

Figura 2 - Tela do *bESTscan - Results* apresentando uma tabela que integra todos os resultados de cada etapa da anotação.



Observando a Figura 2, podemos notar que todos os resultados de cada componente do bESTscan estão integrados e apresentados em uma única tabela, o que facilita a análise e curagem manual. O nome do produto do BLAST e os termos GO são links que remetem ao GenBank e Gene Ontology.

REFERÊNCIAS

ALTSCHUL, S.F.; MADDEN, T.L.; SCHAFFER, A.A.; ZHANG, J.; ZHANG, Z. et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* 25: 3389-3402, 1997.

BENDTSEN, J.D.; NIELSEN, H.; VON HEIJNE, G. AND BRUNAK S. Improved prediction of signal peptides: SignalP 3.0. J Mol Biol. 340(4):783-95, 2004.

KROGH, A.; LARSSON, B.; VON HEIJNE, G.; SONNHAMMER, E.L. Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes. J Mol Biol., (3):567-80, 2001.

QUEVILLON, E.; SILVENTOINEN, V.; PILLAI, S.; HARTE, N.; MULDER, N.; APWEILER, R.; LOPEZ, R. InterProScan: protein domains identifier. Nucleic Acids Res., 33:W116-120, 2005.

ANEXO 2

Manual ITScan

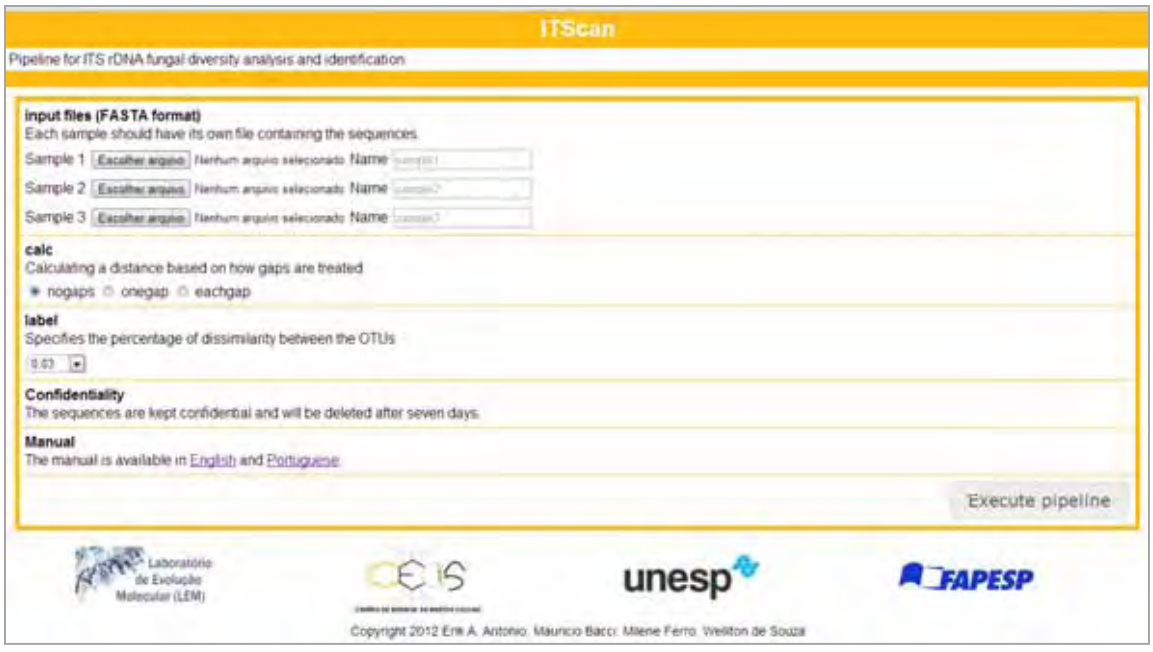
Manual do ITScan

O *pipeline* ITScan foi construído para estudos de diversidade fúngica, utilizando sequências padrões da região ITS (BELLEMAIN *et al*., 2010). Está disponível em: http://evol.rc.unesp.br:8083/itscan (Figura 1). Possui uma interface gráfica que facilita a inserção dos arquivos, a escolha de parâmetros de análise, bem como a execução de cada etapa do processamento dos dados.

ITScan recebe arquivos de entrada no formato FASTA e utiliza o programa Mothur (SCHLOSS *et al*., 2009) para gerar curvas de rarefação, índices de diversidade e índices de riqueza. Além disso, utiliza sequências representativas das Unidades Taxonômicas Operacionais (OTUs) para a identificação de fungos através do programa BLAST (ALTSCHUL *et al*., 1997) e as sequências disponíveis no banco de dados *Nucleotide Collection (nr/nt)* do NCBI (*National Center for Biotechnology Information*).

ITScan pode trabalhar com um volume pequeno de sequências, geralmente provenientes de sequenciamento Sanger, ou com um grande volume de dados, normalmente proveniente de sequenciamentos de Nova Geração (Pirosequenciamento).

Figura 1 - Tela inicial do ITScan.



Upload de arquivos e parâmetros

Inicialmente o usuário deve selecionar o arquivo, em formato FASTA, contendo as sequências pré-processadas, ou seja, que já passaram pelas etapas de *trimming*, para eliminar bases com baixa qualidade e possíveis contaminantes. São aceitos arquivos de entrada no somente no formato FASTA.

Conforme mostra a Figura 1, a ferramenta aceita processar de uma até três amostras distintas (*Sample1*, *Sample2* e *Sample3*) ao mesmo tempo, sendo que o usuário deve fornecer um arquivo para cada amostra contendo todas as sequências ITS e deve preencher o campo *Name* com o nome da amostra. Caso as amostras não sejam nomeadas pelo usuário, elas serão chamadas de *Sample1*, *Sample2* e *Sample3.*

Quando se insere mais de um arquivo, ou seja, se estiver trabalhando com mais de uma amostra, a ferramenta gera um arquivo com extensão .*groups*, que é utilizado pelo programa Mothur.

Além da escolha dos arquivos de entrada, o usuário deve escolher os valores dos parâmetros **calc** e **label**.

O parâmetro **calc** refere-se às formas considerar os *gaps* no cálculo de distâncias pelo programa Mothur. Há três opções para esse parâmetro:

- *nogaps* (*default* no ITScan): ignora os *gaps*, tal como ocorre no programa DNADIST;
- *onegap*: considera um conjunto de *gaps* adjacentes como sendo um único *gap*;
- *eachgap*: considera cada *gap* encontrado.

O parâmetro **label** é utilizado em vários comandos diferentes no Mothur responsáveis por gerar os índices de diversidade e curva de rarefação. Os valores de **label** representam a porcentagem de dissimilaridade (diferença) entre as sequências na mesma OTU. As opções para **label** são:

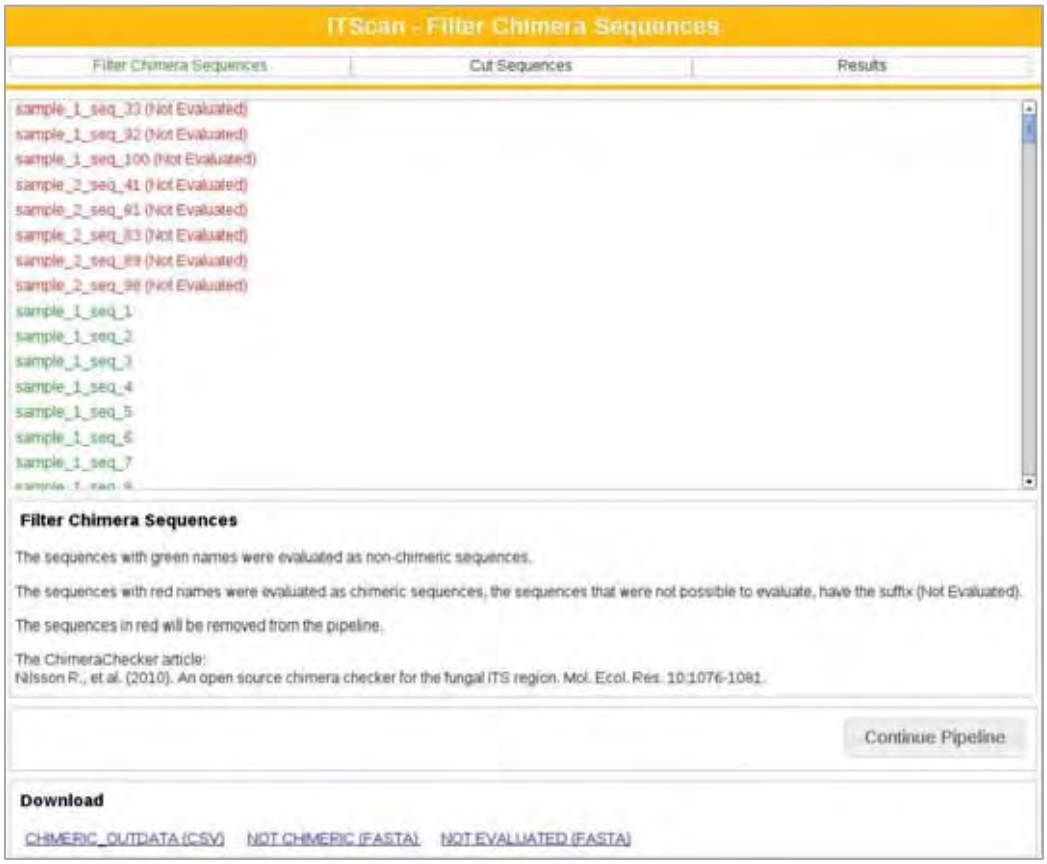
- *unique*: sequências únicas;
- *0.03*: sequências com 97% de identidade ou 3% de dissimilaridade (*default* no ITScan);
- *0.05*: sequências com 95% de identidade ou 5% de dissimilaridade.

Após carregar os arquivos e selecionar os parâmetros, basta clicar no botão Execute pipeline, para iniciar uma verificação do formato das sequências, seguida da busca por quimeras.

Análise de sequências quiméricas

O programa ChimeraChecker (NILSSON *et al*., 2010) classifica todas as sequências analisadas em três grupos: quiméricas (CHIMERIC), não quiméricas (NOT CHIMERIC) ou não avaliadas (Not Evaluated). Uma lista contendo as sequências nos últimos dois grupos é exibida na tela do ITScan (Figura 2).

Figura 2 - Tela do *ITScan - Filter Chimera Sequences*, com os resultados do programa ChimeraChecker.



Esta tela contém links que habilitam o usuário a fazer *download* dos arquivos gerados pelo ChimeraChecker na forma de arquivos FASTA contendo as sequências quiméricas, não-quiméricas ou não avaliadas, bem como um arquivo tabular (CSV) com os dados de alinhamento, apenas para as sequências consideradas quiméricas.

Os nomes das sequências em vermelho sinalizam que estas não foram avaliadas. O usuário pode inspecionar cada sequência não avaliada e, caso verifique que esta realmente não é quimérica, pode alterar manualmente o status desta sequência, clicando sobre seu nome, que passará de vermelho para verde. Todas as sequências com nome em verde passarão para a próxima etapa do *pipeline.* Quando clicar em 

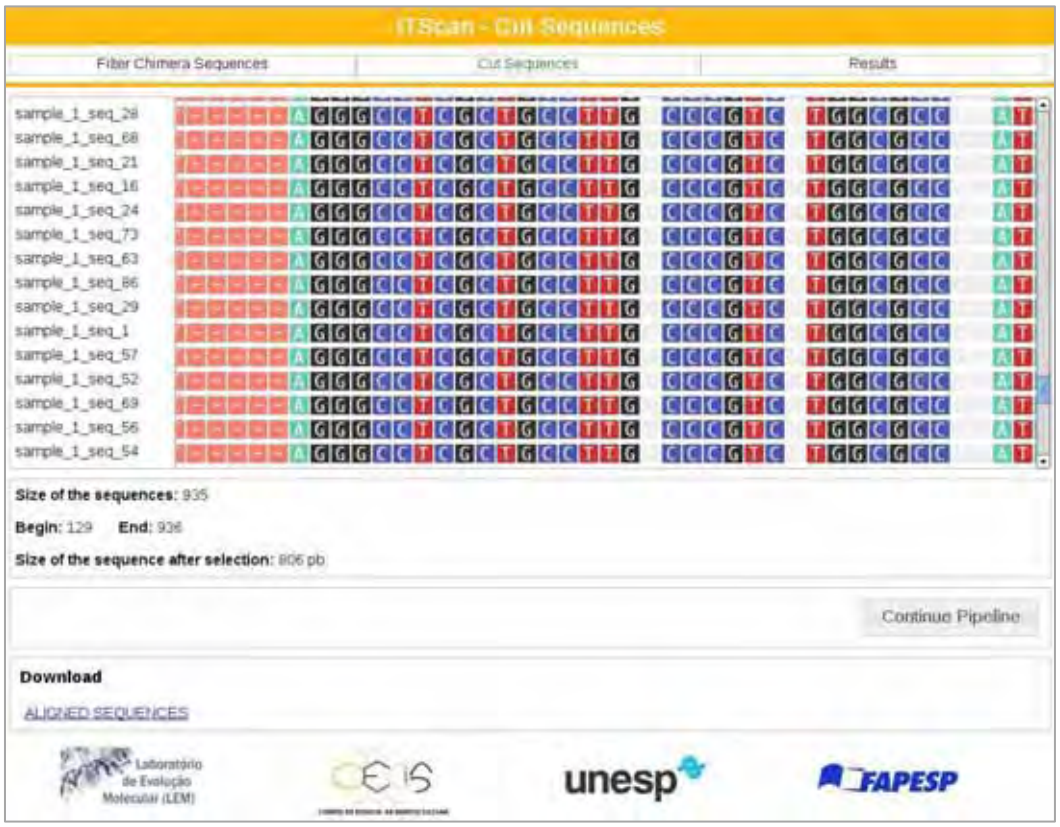
as sequências serão alinhadas.

Alinhamento e corte das extremidades das sequências alinhadas

Todas as sequências não quiméricas serão alinhadas no programa ClustalW (THOMPSON *et al*., 1994). O alinhamento será apresentado na interface gráfica do ITScan denominada *Cut Sequences*, que habilita o usuário a fazer *download* de um arquivo contendo o alinhamento (Figura 3). Nessa interface, o usuário pode também verificar o tamanho das sequências após o alinhamento incluindo os *gaps*, definir as coordenadas de corte (início e fim) e o tamanho das sequências após os cortes das extremidades. O corte é necessário, pois esse arquivo entra no programa Mothur, que aceita apenas alinhamentos contendo sequências de mesmo tamanho. Para selecionar o corte, basta clicar na base desejada, que as sequências numa mesma coluna serão selecionadas.

É importante ressaltar que, independentemente da quantidade de sequências de entrada, sempre será apresentada, na interface de corte das sequências alinhadas, apenas uma amostragem (as maiores sequências), devido a uma limitação da quantidade de sequências carregadas no navegador. Esse problema será resolvido nas próximas versões.

Figura 3 - Tela do *ITScan - Cut Sequences*, com uma amostragem das sequências alinhadas pelo ClustalW para corte das extremidades.



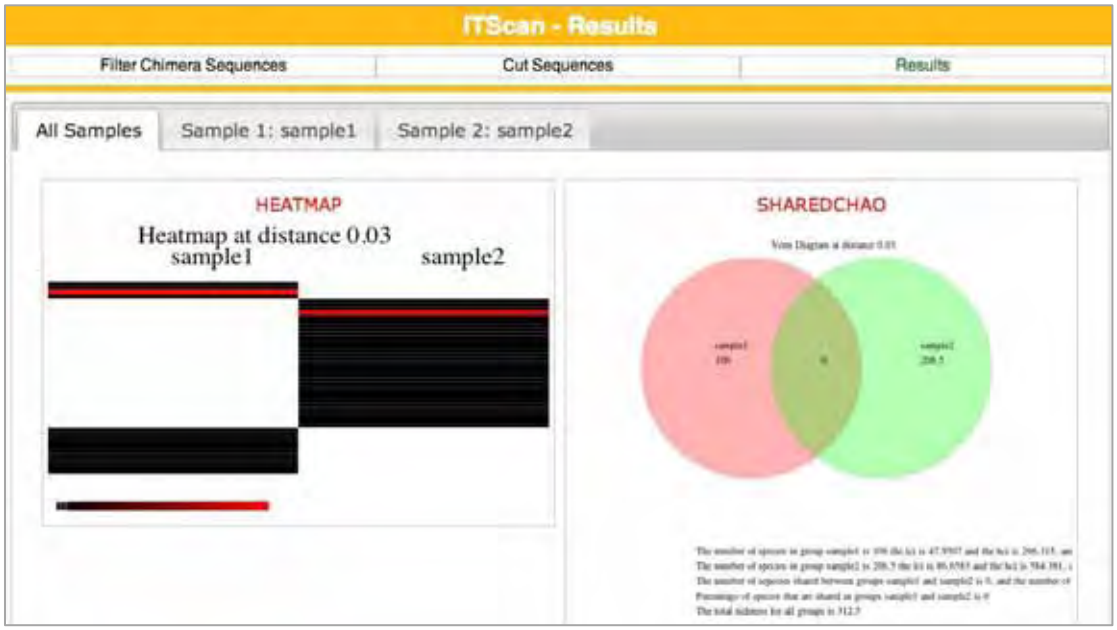
Resultados do Mothur

As sequências alinhadas e cortadas passam para o programa Mothur e os resultados do Mothur (Figura 4) ficam disponíveis da mesma forma que no *pipeline* 16Scan. Os gráficos também são gerados usando a linguagem R.

Figura 4 - Tela do *ITScan - Results* apresentando os resultados gráficos para curva de rarefação e para os índices de diversidade e links para download.

Conforme mostra a Figura 4, os resultados de cada amostra ficam disponíveis em uma aba separada. Há uma aba chamada *All Samples* onde ficam os resultados da comparação entre as amostras (Diagramas de Venn, etc) (Figura 5).

Figura 5 - Tela do *ITScan - Results All Samples*, apresentando os resultados gráficos (diagrama de Venn e Heatmap) compartilhados entre as amostras analisadas.

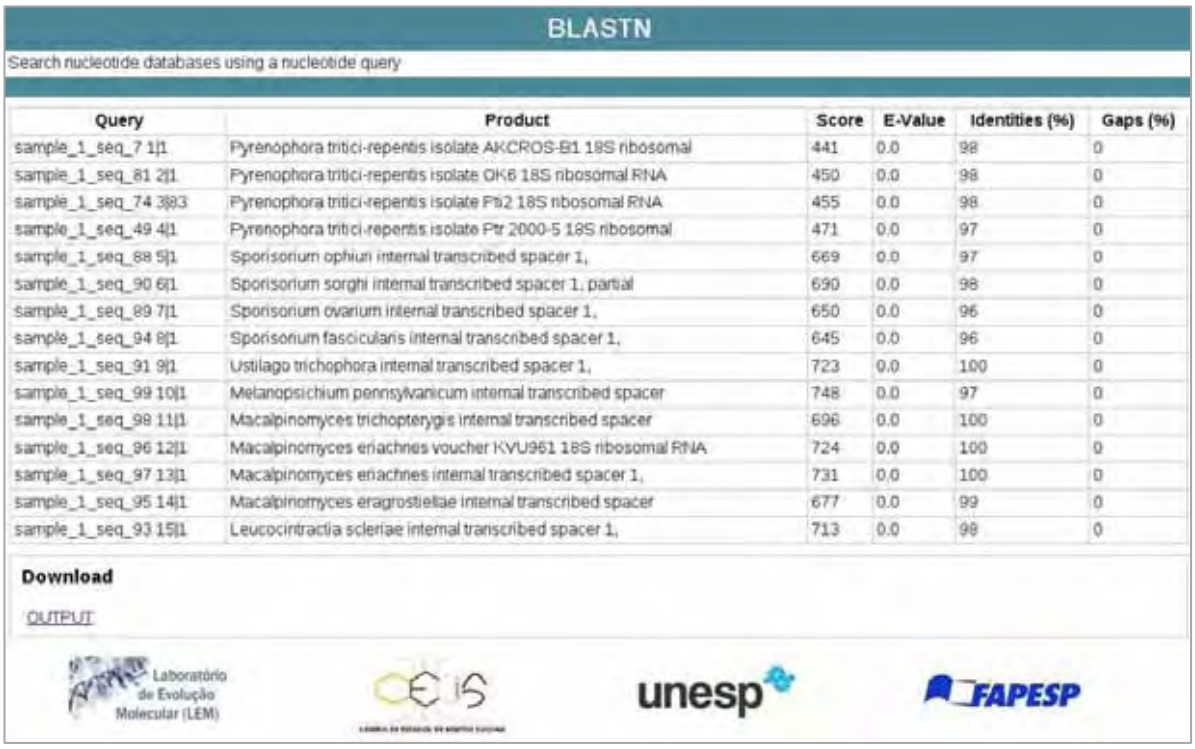


Há ainda um *link* Execute BLASTN para executar o programa BLASTN usando como entrada o arquivo FASTA com as sequências representativas das OTUs (ver explicação no próximo item).

Identificação das sequências das OTUs

A identificação de sequências é feita usando o programa BLASTN, o parâmetro *evalue* e-03 e o banco de dados *Nucleotide Collection (nr/nt)*. Após rodar o BLASTN, será exibida uma tabela com o resultado do alinhamento e também o arquivo texto com o alinhamento do BLASTN para *download* (Figura 6).

Figura 6 - Tela com os resultados da identificação usando o programa BLASTN.



BLASTN

Search nucleotide databases using a nucleotide query

Query	Product	Score	E-Value	Identities (%)	Gaps (%)
sample_1_seq_7 1|1	Pyrenophora tritici-repentis isolate AKCROS-B1 18S ribosomal	441	0.0	98	0
sample_1_seq_81 2|1	Pyrenophora tritici-repentis isolate QK6 18S ribosomal RNA	450	0.0	98	0
sample_1_seq_74 3|3	Pyrenophora tritici-repentis isolate Pti2 18S ribosomal RNA	455	0.0	98	0
sample_1_seq_49 4|1	Pyrenophora tritici-repentis isolate Ptr 2000-5 18S ribosomal	471	0.0	97	0
sample_1_seq_88 5|1	Sporisorium ophiuri internal transcribed spacer 1,	669	0.0	97	0
sample_1_seq_90 6|1	Sporisorium sorghi internal transcribed spacer 1, partial	690	0.0	98	0
sample_1_seq_89 7|1	Sporisorium ovarium internal transcribed spacer 1,	650	0.0	96	0
sample_1_seq_94 8|1	Sporisorium fascicularis internal transcribed spacer 1,	645	0.0	96	0
sample_1_seq_91 9|1	Ustilago trichophora internal transcribed spacer 1,	723	0.0	100	0
sample_1_seq_99 10|1	Melanopsichium pennsylvanicum internal transcribed spacer	748	0.0	97	0
sample_1_seq_98 11|1	Macalpinomyces trichopterygis internal transcribed spacer	696	0.0	100	0
sample_1_seq_96 12|1	Macalpinomyces eriachnes voucher KVU961 18S ribosomal RNA	724	0.0	100	0
sample_1_seq_97 13|1	Macalpinomyces eriachnes internal transcribed spacer 1,	731	0.0	100	0
sample_1_seq_95 14|1	Macalpinomyces eragrostidiae internal transcribed spacer	677	0.0	99	0
sample_1_seq_93 15|1	Leucocintractia scleriae internal transcribed spacer 1,	713	0.0	98	0

Download

OUTPUT

O ITScan gera ainda uma tabela (Figura 6) contendo os seguintes dados do BLASTN:

- *Sequence name*: nome da sequência de entrada,
- *Blast Description*: descrição do resultado para o primeiro *hit*,
- *Evalue*: resultado do *evalue* para o primeiro *hit*,
- *Score*: resultado do *score* para o primeiro *hit*.

REFERÊNCIAS

ALTSCHUL, S.F.; MADDEN, T.L.; SCHAFFER, A.A.; ZHANG, J.; ZHANG, Z. et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* 25: 3389-3402, 1997.

BELLEMAIN, E., CARLSEN, T., BROCHMANN, C., COISSAC, E., TABERLET, P. AND KAUSERUD, H. ITS as an environmental DNA barcode for fungi: an in silico approach reveals PCR biases. BMC Microbiology, 10:189, 2010.

NILSSON, R.H.; ABARENKOV, K.; VELDRE, V.; NYLINDER, S.;, DE WIT, P.; BROSCHÉ, S.; ALFREDSSON, J.F.; RYBERG, M. AND KRISTIANSSON, E. An open source chimera checker for the fungal ITS region. *Molecular Ecology Resources* 10: 1076–1081, 2010.

SCHLOSS, P.D.; WESTCOTT, S.L., RYABIN, T.; HALL, J.R.; HARTMANN, M. ET AL. Introducing mothur: Open source, platform-independent, community-supported software for describing and comparing microbial communities. *Appl Environ Microbiol*. doi:10.1128/AEM.01541-09, 2009.

THOMPSON, J.D.; HIGGINS, D.G.; GIBSON, T.J. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. *Nucleic Acids Res*. 22(22):4673-80, 1994.

ANEXO 3

Manual 16Scan

Manual do 16Scan

O *pipeline* 16Scan foi construído para estudos de diversidade bacteriana, utilizando sequências 16S. Está disponível em: http://evol.rc.unesp.br:8083/16scan (Figura 1). Possui uma interface gráfica que facilita a inserção dos arquivos, a escolha de parâmetros de análise, bem como a execução de cada etapa do processamento dos dados

16Scan recebe arquivos de entrada no formato FASTA já pré-processados e utiliza o programa Mothur (SCHLOSS *et al*., 2009) para gerar dados referentes a curva de rarefação e índices de diversidade e riqueza. Além disso, utiliza sequências representativas das Unidades Taxonômicas Operacionais (OTUs) para a identificação de bactérias usando o banco de dados RDP (Ribosomal Database Project).

16Scan pode trabalhar com um volume pequeno de sequências, geralmente provenientes de sequenciamento Sanger, ou com um grande volume de dados, normalmente proveniente de sequenciamentos de Nova Geração, como, por exemplo, 454, SOLID ou Illumina.

Figura 1 - Tela inicial do 16Scan.

Upload de arquivos e parâmetros

As etapas de *upload* de arquivo e escolha de parâmetros funciona exatamente como o *pipeline* ITScan, conforme mostra o Anexo 2 (Manual do ITScan).

Análise de sequências quiméricas

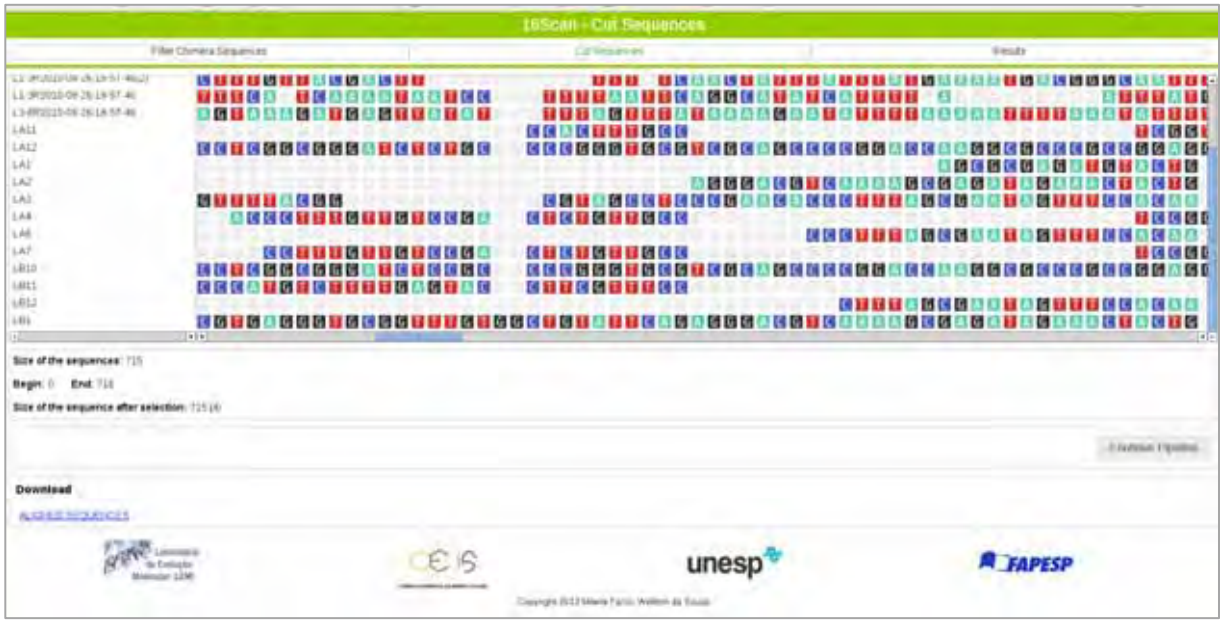
No caso do 16Scan, o programa UCHIME (EDGAR *et al*., 2001) faz a verificação de sequências quiméricas, sendo que apenas as sequências não quiméricas são passadas para a próxima etapa do *pipeline.* A apresentação dos resultados e a forma de uso é a mesma que para o ITScan. Assim, o usuário pode fazer *download* do arquivo gerado pelo UCHIME na forma de arquivos FASTA contendo as sequências quiméricas e as não-quiméricas em formato FASTA.

Alinhamento e corte das extremidades das sequências alinhadas

Todas as sequências não quiméricas são alinhadas no programa ClustalW (THOMPSON *et al*., 1994). O alinhamento é apresentado na interface gráfica do 16Scan denominada *Cut Sequences*, que habilita o usuário a fazer *download* de um arquivo contendo o alinhamento (Figura 2). Nessa interface, o usuário pode também verificar o tamanho das sequências após o alinhamento incluindo os *gaps*, definir as coordenadas de corte (início e fim) e o tamanho das sequências após os cortes das extremidades. O corte é necessário, pois esse arquivo entra no programa Mothur, que aceita apenas alinhamentos contendo sequências de mesmo tamanho. Para selecionar o corte, basta clicar na base desejada, que as sequências numa mesma coluna serão selecionadas.

É importante ressaltar que, independentemente da quantidade de sequências de entrada, sempre será apresentada, na interface de corte das sequências alinhadas, apenas uma amostragem (as maiores sequências), devido a uma limitação da quantidade de sequências carregadas no navegador. Esse problema será resolvido nas próximas versões.

Figura 2 - Tela do *16Scan - Cut Sequences*, com uma amostragem das sequências alinhadas pelo ClustalW para corte das extremidades.



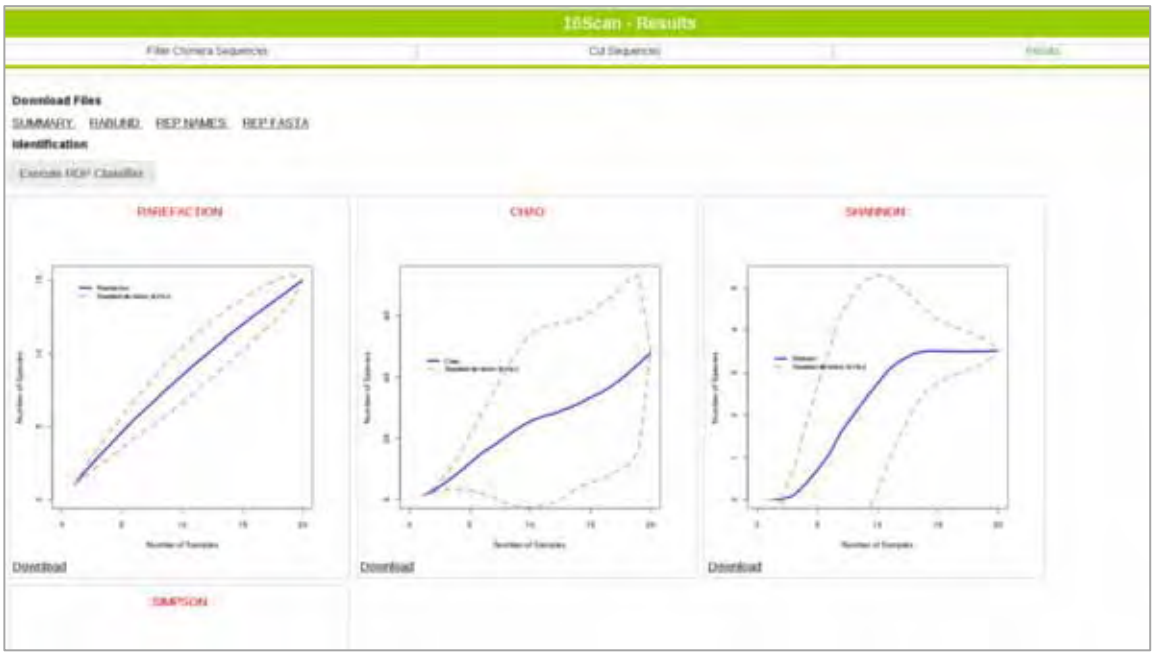
Resultados do Mothur

As sequências alinhadas e cortadas passam para o programa Mothur, que gera índices de diversidade e riqueza e curva de rarefação para cada amostra. Os resultados do Mothur ficam disponíveis para *download* (Figura 3) através dos links:

- *Summary*: arquivo texto com um resumo de todos os resultados obtidos;
- *Rabund*: arquivo texto contendo a abundância das sequências para cada OTU;
- *Rep Names*: arquivo texto com os nomes das sequências representativas de cada OTU;
- *Rep Fasta*: arquivo FASTA com as sequências representativas de cada OTU;

A ferramenta 16Scan utiliza a linguagem R para gerar os gráficos de curva de rarefação e para os índices de diversidade (Chao, Shannon e Simpson).

Figura 3 - Tela do *16Scan - Results* apresentando os resultados gráficos para curva de rarefação e para os índices de diversidade e links para download.



Conforme mostra a Figura 3, os resultados de cada amostra ficam disponíveis em uma aba separada e da mesmo forma que ITScan há uma aba chamada *All Samples* onde ficam os resultados da comparação entre as amostras (Diagramas de Venn, e HeatMap).

Identificação das sequências das OTUs

A identificação de sequências diferentemente do ITScan é feita usando o programa RDP Classifier (http://rdp.cme.msu.edu/classifier/classifier.jsp), usando o banco de dados RDP 16S rRNA que é uma base específica para sequências 16S. Como resultado é exibida a própria tela do classificador.

REFERÊNCIAS

SCHLOSS, P.D.; WESTCOTT, S.L., RYABIN, T.; HALL, J.R.; HARTMANN, M. ET AL. Introducing mothur: Open source, platform-independent, community-supported software for describing and comparing microbial communities. *Appl Environ Microbiol*. doi:10.1128/AEM.01541-09, 2009.

THOMPSON, J.D.; HIGGINS, D.G.; GIBSON, T.J. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. *Nucleic Acids Res*. 22(22):4673-80, 1994.

EDGAR, R.C.; HAAS, B.J.; CLEMENTE, J.C.; QUINCE, C.; KNIGHT, R. UCHIME improves sensitivity and speed of chimera detection, Bioinformatics, 2001.

ANEXO 4

Artigo enviado para publicação na revista Bioinformatics

Ferro, M.; Antonio, E.A.; Souza, W.; Bacci, M. ITScan: a web-based analysis pipeline for fungal Internal Transcribed Spacer (ITS) sequences.

Sequence analysis

ITScan: a web-based analysis pipeline for fungal Internal Transcribed Spacer (ITS) sequences

Milene Ferro[1], Erik A. Antonio[3], Wélliton Souza[1] and Maurício Bacci[1,2*]

[1]Centro de Estudos de Insetos Sociais e [2]Departamento de Bioquímica e Microbiologia, Instituto de Biociências, UNESP – Univ Estadual Paulista, Rio Claro SP, 13506-900, Brazil.

[3]Departamento de Ciência da Computação, Universidade Federal de São Carlos, São Carlos SP, 13565-905, Brazil.

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: Dr. Alex Bateman

ABSTRACT

Summary: Studies on fungal diversity and ecology aim to identify fungi and to investigate their interaction with each other and with the environment. DNA sequence-based tools are essential for these studies because the majority of microorganisms cannot be evaluated by standard cultivation techniques. The internal transcribed spacer (ITS) has recently been proposed as a standard marker for molecular identification of fungi and evaluation of fungal diversity. However, the analysis of large sets of ITS sequences involves many programs and steps, which makes this task intensive and laborious. We developed the web-based pipeline ITScan, which automates the analysis of fungal ITS sequences generated either by Sanger or Next Generation Sequencing (NGS) technologies. Validation was performed using datasets containing ca. 400 to 8000 sequences each. ITScan fully automates all the steps currently used in fungal diversity analysis based on ITS sequences. This speeds up a process which would otherwise be repetitive and time-consuming for users.

Availability: ITScan is open source under the GNU GPL license and is available at http://evol.rc.unesp.br:8083/itscan.

Contact: milenef@gmail.com; mbacci@rc.unesp.br

Supplementary Information: available at Bioinformatics online.

INTRODUCTION

Studies on fungal biodiversity use DNA sequence-based tools to generate molecular markers to identify rare species and determine associations in a microbial community (Schoch *et al*., 2012). The technique is powerful in characterizing fungal diversity in environmental samples containing many fungal species which do not grow in laboratory cultures. Many biodiversity studies are based on the Internal Transcribed Spacer (ITS) rDNA region (Fontana *et al*., 2012; Ort *et al*., 2012), which is a small region occurring in multiple copies in the fungal genome and shows a high degree of variation even between closely related species (Bellemain *et al*., 2010). The ITS region has been recently proposed as a universal marker for molecular barcoding of fungi (Schoch *et al*., 2012). To determine the microbial diversity in environmental samples, generated ITS sequences are grouped in operational taxonomic units (OTUs), often using the MOTHUR program (Schloss *et al*., 2009) and other thirty-party programs for fungal diversity analysis. The use of multiple programs and stages of analysis make the process laborious and time-consuming. Here, we describe a pipeline that automates fungal diversity analysis based on ITS sequences.

IMPLEMENTATION

1.1 Architecture Design

We developed an architectural model (illustrated in Supplementary Material) based on MVC (Model-View-Controller) and J2EE design patterns (Fowler, 2002). The architectural depicts two base formats for data interchange (dashed lines): JavaScript Object Notation (JSON) and Extensible Markup Language (XML). These formats represent concerns for each of the pipeline components. The architecture model was tailored to represent two main viewpoints:

- Client Mode — aims at dealing with client-side concerns;
- Request-Response Mode — performs a set of server-side and business logic concerns using coupled third-party programs and their business rules. The Pipeline Manager provides Representation State Transfer - REST (Richardson & Ruby, 2007) service.

1.2 Pipeline for fungal ITS analysis

The third-party programs ClustalW (Thompson *et al*., 1994), ChimeraChecker (Nilsson *et al*., 2010), MOTHUR (Schloss *et al*., 2009) and BLAST (Altschul *et al*., 1997) were

* To whom correspondence should be addressed.

integrated in the pipeline as shown by the state machine diagram using UML (Booch *et al*., 2005) (Figure 1). Each program in ITScan is a web service developed using REST technology, which was shown to improve client usability (Katayama *et al*., 2010; Medina *et al*., 2012). Manual inspection and intervention at given points (Figure 1) facilitate the user to control the analysis process. Aiming to speed up processing, ITScan runs BLASTN on a representative sequence of a given OTU instead of on all sequences composing the OTU.

RESULTS

The architectural model enables the user to develop web service components and couple them in a new customized pipeline. R language scripts provide graphic results representing rarefaction curves as well as Shannon, Simpson and Chao diversity indexes. ITScan has a user-friendly interface and can process up to three samples simultaneously. The pipeline was validated using Sanger sequences (Mantovani *et al*., in preparation) or a large dataset (8000 sequences) simulating results from Next Generation Sequencing (NGS) retrieved from the documentation of the Emerencia pipeline (Nilsson *et al*., 2005).

CONCLUSION

This work describes an architectural model that can be used with bioinformatic third-party programs. All components follow the same framework, which facilitates construction of new components. ITScan works with sequences derived from Sanger and NGS technologies. The pipeline can process single or as many as three datasets to compare distinct biological samples. Output data include graphs that are automatically generated to represent fungal diversity.

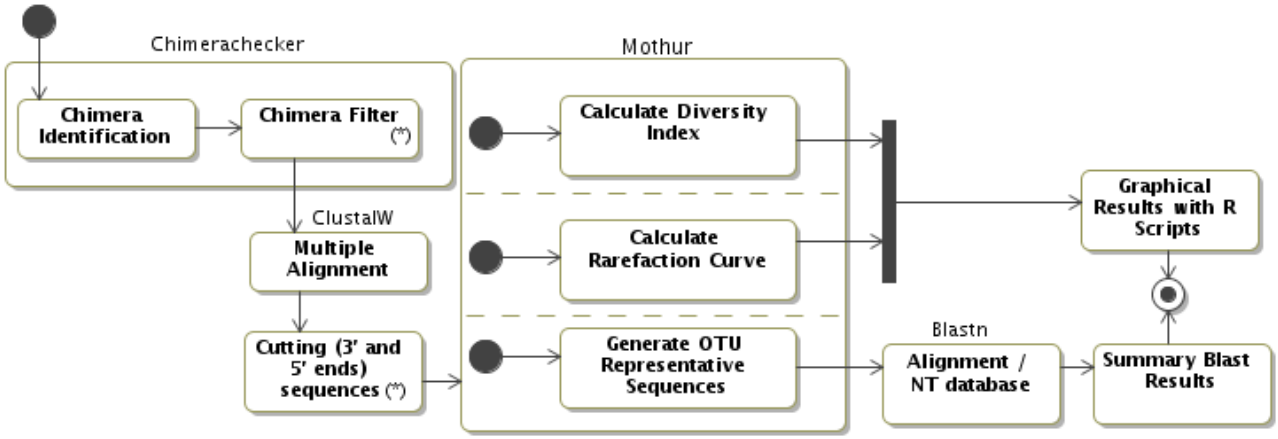


Figure 1: State machine diagram describing the ITScan pipeline for metagenomic analysis based on ITS. Unnamed states were written specifically for ITScan. (*): points admitting manual inspection and intervention.

ACKNOWLEDGEMENTS

This work was supported by Fundação de Amparo à Pesquisa do Estado de São Paulo (Proc. 2011/50226-0). MF receives a doctoral fellowship (Proc. 2009/52289-9).

Conflict of Interests: none declared.

REFERENCES

Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J., Zhang, Z., Miller, W. and Lipman, D.J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*. Sep 1;25(17):3389-402.

Bellemain, E., Carlsen, T., Brochmann, C., Coissac, E., Taberlet, P. and Kauserud, H. (2010). ITS as an environmental DNA barcode for fungi: an *in silico* approach reveals PCR biases. *BMC Microbiology*, 10:189.

Booch, G., Rumbaugh, J. and Jacobson, I. **Unified Modeling Language User Guide**. Addison-Wesley Professional, 2005.

Ort, B.S., Bantay, R.M., Pantoja, N.A. and O'Grady, P.M. (2012). Fungal diversity associated with hawaiian Drosophila host plants. *PLoS ONE*, 7(7): E40550.

Fontana, S., Keshavmurthy, S., Hsieh, H.J., Denis, V., Kuo, C.Y., Hsu, C.M., Leung, J.K.L., Tsai, W.S., Wallace, C.C. and Chen, C.A. (2012). Molecular Evidence Shows Low Species Diversity of Coral-Associated Hydroids in Acropora Corals. Plos One, 7(11): e50130.

Fowler, M. **Patterns of Enterprise Application Architecture**. Addison-Wesley, 2002.

Katayama, T., Nakao, M and Takagi, T. (2010). TogoWS: integrated SOAP and REST APIs for interoperable bioinformatics Web services. *Nucleic Acids Research*, 38, W706-W711.

Medina, I., De Maria, A., Bleda, M., Salavert, F., Alonso, R., Gonzalez, C.Y. and Dopazo, J. (2012). VARIANT: Command Line, Web service and Web interface for fast and accurate functional characterization of variants found by Next-Generation Sequencing. *Nucleic Acids Research*, 40, W40-W58.

Nilsson, R.H.. Abarenkov, K., Veldre, V., Nylinder S, Wit, P., Brosché, S., Alfredsson, J.F., Ryberg, M., Kristiansson, E. (2010). An open source chimera checker for the fungal ITS region. *MolEcol Res* 10:1076-1081.

Nilsson, R.H., Kristiansson, E., Ryberg, M. and Larsson, K.H. (2005). Approaching the taxonomic affiliation of unidentified sequences in public databases – an example from the mycorrhizal fungi. *BMC Bioinformatics*, **6**:178.

Richardson, L. & Ruby, S**. RESTful Web Services: Web Services for the Real World**. O'Reilly Media, 2007.

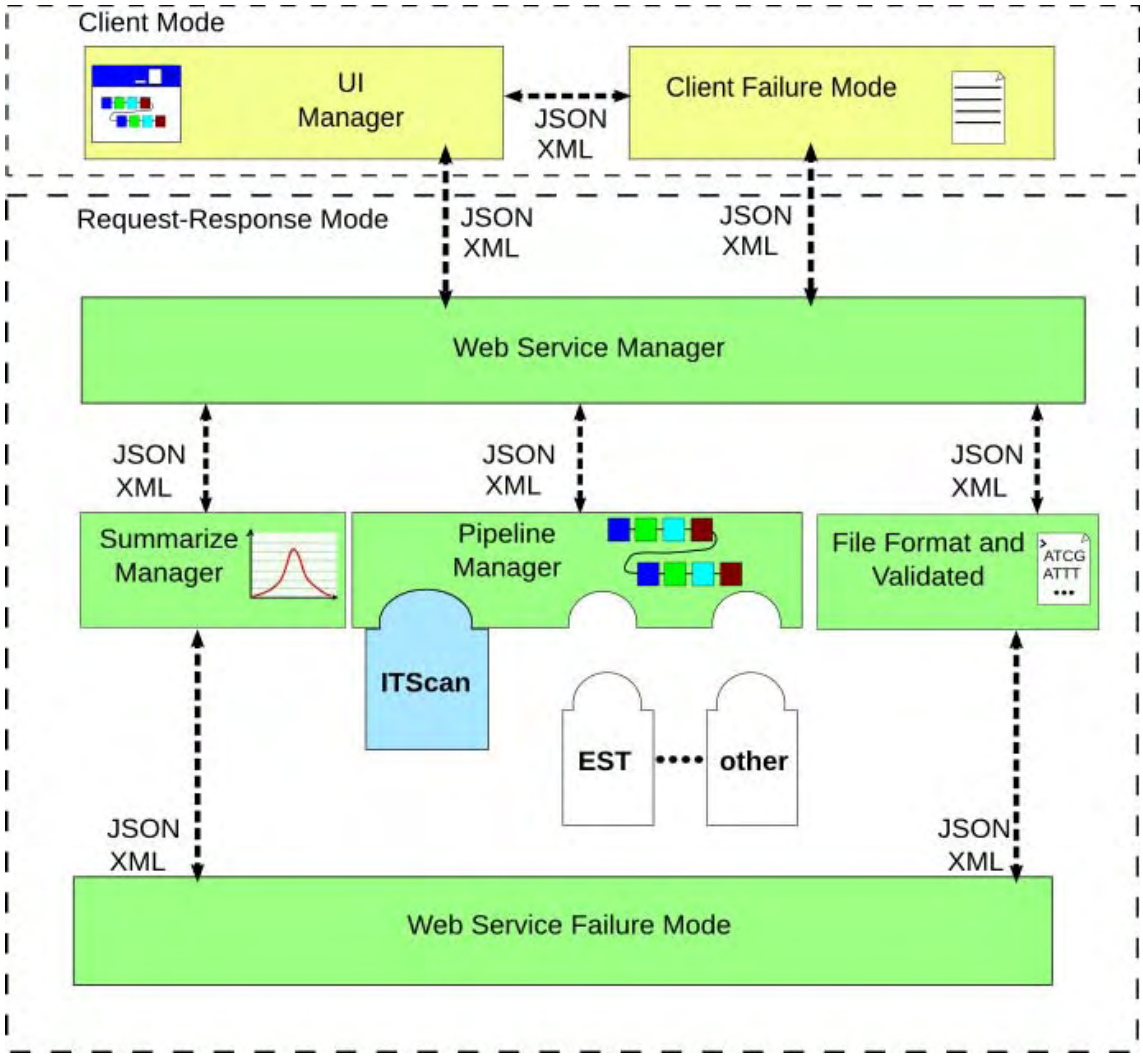
Schloss, P. D., et al. (2009). Introducing mothur: Open-source, platform-independent, community-supported software for describing and comparing microbial communities. *Appl Environ Microbiol*, 75(23):7537-41.

Schoch, C.L., Seifert, K.A., Huhndorf, S., Robert, V., Spouge, J.L., et al. (2012). Nuclear ribosomal internal transcribed spacer (ITS) region as a universal DNA barcode marker for Fungi. PNAS 109: 6241–6246.

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

Supplementary Information

Representation of the client-server architecture.



Dashed lines shows JSON and XML data interchange formats. Client Mode is represented in yellow boxes and Request-Response Mode in green boxes.