Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Método de Aprendizagem por Reforço no Sistema BioAgents

Hugo Wruck Schneider

Monografia apresentada como requisito parcial
para conclusão do Mestrado em Informática

Orientadora
Prof.ª Dr.ª Célia Ghedini Ralha

Coorientadora
Prof.ª Dr.ª Maria Emília Machado Telles Walter

Brasília
2010

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado em Informática

Coordenadora: Prof.ª Dr.ª Alba Cristina Magalhães de Melo

Banca examinadora composta por:

Prof.ª Dr.ª Célia Ghedini Ralha (Orientadora) — CIC/UnB
Prof.ª Dr.ª Ana Lúcia Cetertich Bazzan — Instituto de Informática - UFRGS
Dr.ª Natália Florêncio Martins — Embrapa/Recursos Genéticos e Biotecnologia

CIP — Catalogação Internacional na Publicação

Schneider, Hugo Wruck.

Método de Aprendizagem por Reforço no Sistema BioAgents / Hugo Wruck Schneider. Brasília : UnB, 2010.
94 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2010.

1. Aprendizagem por Reforço, 2. Anotação Manual, 3. Sistema Multiagente, 4. BioAgents

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Método de Aprendizagem por Reforço no Sistema BioAgents

Hugo Wruck Schneider

Monografia apresentada como requisito parcial
para conclusão do Mestrado em Informática

Prof.ª Dr.ª Célia Ghedini Ralha (Orientadora)
CIC/UnB

Prof.ª Dr.ª Ana Lúcia Cetertich Bazzan
Instituto de Informática - UFRGS

Dr.ª Natália Florêncio Martins
Embrapa/Recursos Genéticos e Biotecnologia

Prof.ª Dr.ª Alba Cristina Magalhães de Melo
Coordenadora do Mestrado em Informática

Brasília, 29 de Março de 2010

Dedicatória

Todo este trabalho é dedicado a minha família, especialmente aos meus pais.

Hugo Wruck Scnheider

Agradecimentos

Agradeço primeiramente as minhas orientadoras, Célia e Maria Emília, por todo o conhecimento compartilhado, toda ajuda que me deram e paciência que tiveram durante os períodos de improdutividade. Também agradeço aos colegas Carla e Jaime pelo interesse demonstrado pelo projeto e pela ajuda na implementação de partes do projeto. E Agradeço ao professor Ralha pela ajuda nos ajustes finais neste texto.

Agradeço aos colegas "VipSigners" e ex-"VipSigners", Alessandro, Luiz, Augusto, Rommel, Lu, Germano, Alexandre, Leo e Júlio, que entenderam a minha ausência em dias de trabalho durante o meu mestrado e além disso demonstraram interesse pelo progresso do meu projeto.

Também agradeço a Nátalia, Roberto, Marcos e Georgios, que disponibilizaram um ambiente dentro do laboratório de bioinformática da Embrapa/Cenargen para que eu pudesse fazer alguns testes.

Agradeço a todo apoio da minha família, que sempre me deram incentivo para concluir o projeto. E agradeço a Julia, que sempre esteve comigo, nunca deixou de me apoiar e de me ajudar, e me ajudou muito na elaboração e nas últimas correções deste texto.

Por fim, mas não menos importante, agradeço a Deus.

Hugo Wruck Schneider

Resumo

Neste trabalho, propusemos e implementamos um método de aprendizagem por reforço no sistema *BioAgents*, que tem a finalidade de auxiliar os biólogos na fase de anotação manual de sequências biológicas em projetos de sequenciamento de genomas. O sistema foi reimplementado de modo a permitir a incorporação do mecanismo de aprendizagem, por meio de uma camada adicional no *BioAgents*. Para validar o método e a implementação, realizamos dois experimentos, com dados reais dos Projetos Genoma do fungo *Paracoccidioides brasiliensis* e da planta *Paullinia cupana*. Utilizamos genomas de referência para validar as anotações sugeridas, atribuindo recompensas aos bancos de dados e as ferramentas utilizadas pelos agentes. Os resultados obtidos com a camada de aprendizagem, quando comparados com o sistema sem o método proposto, mostram uma melhora de desempenho.

Palavras-chave: Aprendizagem por Reforço, Anotação Manual, Sistema Multiagente, BioAgents

Abstract

In this work, we proposed and implemented a reiforcement learning method at the *BioAgents* system which has the objective of assisting biologists in the process of manual annotation of biological sequences in genome sequencing projects. The system was reimplemented to allow the incorporation of a learning mechanism, based in an additional layer in *BioAgents*. To validate the method and implementation, we conducted two experiments, with real data of the Projeto Genoma of fungus *Paracoccidioides brasiliensis* and of plant *Paullinia cupana*. We used reference genomes to validate the suggested annotations, giving rewards to the databases and tools handled by the agents. The results obtained with the learning layer, when compared with the system without the proposed method, showed a performance improvement.

Keywords: Reinforcement Learning, Manual Annotation, Multiagent System, BioAgents

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Desde que a estrutura de dupla hélice de uma molécula de DNA foi descoberta por Watson e Crick em 1953 (Watson and Crick 1953), pesquisadores de todo o mundo têm realizado grandes esforços para melhor compreender a estrutura e o funcionamento da genética dos seres vivos. Desde o início dos anos 90, os avanços nos métodos e nas técnicas relacionadas à Biologia Molecular e a Bioinformática possibilitaram acelerar muito o processo de descoberta e descrição da estrutura e das funcionalidades dos genes.

Em particular, o Projeto Genoma Humano (PGH), iniciado em 1990 e finalizado em 2001 (Lander et al. 2001, Venter et al. 2001), foi de essencial importância para o atual desenvolvimento da Biologia Molecular e da Bioinformática. Muitos países tiveram a oportunidade de participar desse projeto, possibilitando que métodos e tecnologias fossem desenvolvidos e compartilhados entre diversas instituições de ensino e pesquisa.

Antes do PGH, foram desenvolvidos outros projetos de sequenciamento de organismos mais simples, como *Saccharomyces cerevisiae*, *Caenorhabditis elegans*, *Drosophilamelanogaster* e *Arabidopsis thaliana*, entre outros. Estes projetos visavam a geração de dados nos laboratórios de Biologia Molecular e o aperfeiçoamento das técnicas de análise computacional das sequências biológicas para que depois fossem usados no sequenciamento do genoma humano. Então, o grande investimento de recursos no PGH proporcionaram o surgimento de uma nova era, que pode ser chamada de Era Genômica da Biologia (Green 2001). Desde então, centenas de projetos de sequenciamento de genomas surgiram em todo o mundo.

Esses projetos geraram um grande volume de sequências biológicas e informações relacionadas, implicando na necessidade de se criar novas metodologias de pesquisa nas áreas de Biologia Molecular e Computação. Surgiram máquinas de sequenciamento de alta precisão, com custos relativamente baixos, denominadas de sequenciamento *Sanger*, além de inúmeras ferramentas computacionais de análise e automação de todo o processo. Esses recursos possibilitaram aos pesquisadores explorarem cada vez mais rapidamente os padrões de expressão gênica dos genomas, tanto considerando sequências expressas quanto o próprio DNA dos organismos.

No atual contexto da geração de milhões de sequências pelos novos sequenciadores de alto desempenho, a análise e anotação dos genes deve ser realizada de forma automática, pois não é mais possível incluir uma etapa de anotação manual, como nos projetos *Sanger*.

Um grande volume de recursos financeiros e humanos vem sendo investido em Bioinformática, como se pode notar pelo grande número de centros privados e governamentais especializados em sequenciamento genômico. Citamos o TIGR (*The Institute for Genome Research*) ((http://www.tigr.org/)), um dos principais centros de pesquisa e geração de dados de sequências genômicas, o Instituto *Broad* (`http://www.broadinstitute.org/`), o Instituto *Sanger* (`http://www.sanger.ac.uk/`), o DOE *Joint Genome Institute* (`http://www.jgi.doe.gov/`), a Universidade de Washington em St. Louis (`http://www.wustl.edu/`) e o *European Bioinformatics Institute* (EBI) (`http://www.ebi.ac.uk/`), entre outros. Uma lista completa dos projetos de sequenciamento de genomas e dos centros de pesquisa pode ser acessada no GOLD (*Genomes OnLine Database*) (`http://wit.integratedgenomics.com/GOLD/`).

Em âmbito nacional, os esforços pioneiros foram realizadas pela FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) e pelo CNPq, que foram as primeiras instituições a apoiar projetos na área de sequenciamento de genomas no Brasil. Inicialmente, a FAPESP e o CNPq formaram um consórcio de laboratórios e criaram um instituto virtual responsável pelo sequenciamento e análise de nucleotídeos, denominado ONSA (*Organization for Nucleotide Sequencing and Analysis*) (`http://watson.fapesp.br/onsa/Genoma3.htm`). O primeiro resultado importante da ONSA e o seu reconhecimento internacional ocorreram com a publicação do genoma do fitopatógeno *Xylella fastidiosa*, agente etiológico da *Citrus Variegated Chlorosis* (CVC), mais conhecida como "praga do amarelinho", essa doença destrói lavouras de laranja, principalmente no Estado de São Paulo, ocasionando prejuízos econômicos de grandes proporções (Simpson et al. 2000).

O sucesso desse projeto foi tão significativo que motivou os órgãos governamentais, particularmente a FAPESP, a investir em outros projetos (`http://www.fapesp.br/`), como o do mapeamento do genoma da cana-de-açúcar, do câncer humano, que teve a participação do Instituto Ludwig para Pesquisa do Câncer, do café e também de vários organismos e pragas como o *Xylella fastidiosa* de videira, o *Xanthomonas campestris*, o *Xanthomonas axonopodis* e o *Leifsonia xyli*, além de subsidiar outros projetos como o do mapeamento do genoma funcional do *Schistosoma mansoni*.

Outro aspecto de destaque foi a criação das redes de sequenciamento, estimulado pelo governo federal, que ocorreu tanto no âmbito nacional como no regional. No âmbito nacional, foram criados dois projetos: o Projeto Genoma Brasileiro, que sequenciou a bactéria *Chromobacterium violaceum* (Vasconcelos 2003), que apresenta resultados de potencial aplicabilidade no controle da doença de chagas e da leishmaniose, e o projeto Genolyptus (`http://ftp.mct.gov.br/especial/genolyptus3.htm`), responsável pelo sequenciamento do eucalipto (Fundo Verde-Amarelo/MCT). No âmbito regional surgiram várias redes para executar projetos de sequenciamento de organismos importantes, especialmente para o controle de pragas e doenças. Dentre elas podemos destacar:

- Rede Genoma do Estado de Minas Gerais (*Schistosoma mansoni*);

- Rede Genoma Nordeste (*Leishmania chagasi*);
- Rede Genômica do Estado da Bahia e São Paulo (*Crinipellis perniciosa*);
- Rede Genoma integrante do Consórcio do Instituto de Biologia Molecular do Paraná, FIOCRUZ e Universidade de Mogi das Cruzes (*Trypanossoma cruzi*);
- Programa Genoma do Estado do Paraná (*Herbaspirillum seropedicae*);
- Rede Genoma do Rio de Janeiro (*Gluconacetobacter diazotrophicus*);
- Rede Sul de Análise de Genomas e Biologia Estrutural (*Mycoplasma hyopneumoniae*);
- Rede Genoma Centro-Oeste (*Paracoccidioides brasiliensis*).

Estes projetos inseriram o Brasil no grupo dos países com tecnologia e infra-estrutura suficientes para viabilizar e conduzir pesquisas nas áreas de Biotecnologia e Bioinformática. Esse fato tem importância estratégica, pois permite que o país desenvolva tecnologia própria para resolver problemas específicos que afetam nossa população e/ou produção agropecuária, independente da boa vontade dos governos e laboratórios estrangeiros. Além disso, esses projetos estimulam o desenvolvimento de tecnologias e a capacitação de profissionais especializados, o que contribui para colocar o Brasil em posição de igualdade perante a comunidade científica internacional nessa área.

Os dois tipos de sequenciamento, o *Sanger* e de alto desempenho, diferem na quantidade e no tamanho das sequências obtidas no projeto. O tamanho da sequências obtidas no sequenciamento *Sanger* são maiores que no sequenciamento de alto desempenho, porém nesse último são obtidas milhões de sequências, enquanto no *Sanger* são obtidas poucos milhares.

Como uma área importante da computação, e dentro da visão moderna de Inteligência Artificial (IA) encontramos na literatura a abordagem de agentes inteligêntes (Russell and Norvig 2002). Um Agente Inteligente pode ser definido como uma entidade de *software* capaz de perceber o ambiente onde se encontra por meio de sensores tendo capacidade de interagir com tal ambiente por meio de atuadores. Um Sistema Multiagente (SMA) consiste em um grupo de agentes inteligentes que interagem entre si, por meio de troca de mensagens através de uma infra-estrutura de rede computacional ou por um *software* que viabilize tal comunicação (Weiss 2000, Wooldridge 2009).

Ainda na área de IA, temos a subárea de Aprendizagem de Máquina, a qual tem a função de melhorar o conhecimento manipulado pelos agentes racionais, na qual os algoritmos de aprendizagem de máquina podem ser classificados em três tipos distintos: (i) supervisionados, que consistem na aprendizagem de uma função a partir de exemplos de entradas e saídas desta função; (ii) não supervisionados, que consiste no reconhecimento de padrões nas entradas sem nenhuma informação sobre a saída desejada; (iii) e por reforço, que consiste na aprendizagem a partir de recompensas obtidas a cada ação tomada (Mitchell 1997, Russell and Norvig 2002).

O *BioAgents* é um SMA desenvolvido com a finalidade de auxiliar os biólogos na fase de anotação manual, sugerindo funções às sequências analisadas (Lima et al. 2007,

Ralha et al. 2008). O *BioAgents* utiliza a plataforma *JADE* (Bellifemine et al. 2003; 2005), como *framework* de desenvolvimento, e o motor de inferência *JESS* (Friedman-Hill 2003) para processar as regras. Para melhorar o processo de recomendação de anotação no *BioAgents*, utilizaremos algoritmos de aprendizagem de máquina para aumentar a racionalidade e autonomia dos agentes existentes no sistema, o que aumenta a complexidade de desenvolvimento do projeto.

Dentro do contexto dos projetos de sequenciamento, existe uma grande dificuldade na atribuição de funções às sequências genéticas devido à quantidade de sequências obtidas nos projetos de sequenciamento, na ordem de milhões no sequenciamento de alto desempenho. Hoje, grande parte dessa atribuição é feita automaticamente por programas, que não são especializados na anotação de sequências biológicas, não permitindo a acurácia desejada devido a redução do tamanho das sequências, bem como o grande volume de sequências. Tendo isso em vista viu-se a necessidade de melhorar a qualidade e a acurácia das sugestões feitas pelo *BioAgents* e uma forma de se alcançar essa meta é a aplicação de algoritmos de aprendizagem de máquina para melhorar o conhecimento e a autonomia dos agentes inteligentes durante o processo de anotação. Neste contexto, os objetivos deste trabalho são:

- Propor um método de aprendizagem automática, baseado na abordagem de aprendizagem por reforço, a ser implementado no sistema *BioAgents*;
- Incorporar uma camada de aprendizagem ao sistema *BioAgents*;
- Realizar experimentos utilizando as anotações realizadas no Projeto Genoma Pb da Rede Genoma Centro-Oeste (`https://helix.biomol.unb.br/Pb/`) e no Projeto Genoma Guaraná da Rede Genoma Norte (`https://www.biomol.unb.br/GR/`);
- Discutir e comparar os resultados obtidos pelo *BioAgents* com e sem a camada de aprendizagem.

Esse trabalho foi dividido em sete capítulos. O Capítulo 2 apresenta conceitos básicos de Biologia Molecular e Bioinformática utilizados neste trabalho. O Capítulo 3 apresenta definições de Inteligência Artificial os quais foram utilizados tantos na modelagem quanto na implementação da solução proposta. O Capítulo 4 descreve o sistema *BioAgents*, as ferramentas utilizadas e todas as alterações introduzidas no presente trabalho para alcançar os objetivos propostos. O método proposto para a aprendizagem por reforço e o funcionamento do sistema, com o método incluído, são descritos no Capítulo 5. Os experimentos e resultados obtidos são descritos e discutidos no Capítulo 6. Por fim, o Capítulo 7 aprensenta as conclusões e propõe trabalhos futuros.

Capítulo 2

Fundamentos de Biologia Molecular e Bioinformática

Neste capítulo serão apresentados conceitos básicos de Biologia Molecular necessários para o entendimento do contexto do trabalho.

Na Seção 2.1, serão apresentados algumas técnicas biológicas utilizadas no estudo de genomas, pelos dois métodos, *Sanger* e alto desempenho. Na Seçao 2.2 apresentaremos conceitos de Bioinformática. E por último, na Seção 2.3 descreveremos métodos utilizados nos projetos de sequenciamento.

2.1 Estudando o Genoma

A informação básica que se deseja extrair de uma molécula de DNA é sua sequência de pares de bases. O processo através do qual essa informação é obtida denomina-se **sequenciamento**.

2.1.1 Sequenciamento *Sanger*

Os projetos de sequenciamento de genomas utilizando as técnicas *Sanger* permitiram o sequenciamento de fragmentos com cerca de 1000 pares de base, enquanto, por exemplo, um cromossomo humano tem em torno de 10^9 pares de bases (Setúbal and Meidanis 1997). Assim, torna-se necessário quebrar o DNA em fragmentos menores, sequenciá-los e remontá-los de modo a obter a sequência original. A montagem dos fragmentos, em especial, é um problema bastante complexo e requer a utilização de técnicas da área de Matemática e Computação. Atualmente são utilizadas duas técnicas para quebrar as moléculas de DNA, são elas:

- **nucleases de restrição**: faz o reconhecimento de sítios específicos dentro da molécula, determinados por uma curta sequência de bases, e quebra o DNA nesses pontos, sendo que cada nuclease é específica para uma dada sequência de bases;

- ***shotgun***: consiste em uma solução contendo DNA purificado — uma grande quantidade de moléculas idênticas — que é submetida a uma alta carga de vibrações (ou algum procedimento que induza à quebra desordenada das moléculas). Cada molécula é quebrada aleatoriamente em vários locais, sendo os fragmentos resultantes filtrados e separados para posterior processamento.

Para realizar qualquer experimento laboratorial com DNA é necessário uma porção mínima de material. Assim, é essencial que se possa produzir esse material em quantidade, de modo a permitir que o experimento seja repetido ou que novos experimentos sejam conduzidos. Um modo de se obter cópias de uma amostra de DNA é utilizar o mecanismo de reprodução natural de certos organismos. Primeiramente, insere-se a amostra no DNA do organismo, que é chamado de **hospedeiro** ou **vetor**. Quando o vetor se multiplica por sua reprodução natural, a amostra é replicada juntamente com o restante de seu DNA. O DNA obtido dessa forma é chamado de **DNA-recombinante**. Como o genoma do vetor já é conhecido, basta retirar essas sequências do DNA resultante para conseguir as cópias das amostras.

Uma outra maneira de produzir cópias de moléculas de DNA é utilizar a enzima DNA polimerase. Essa enzima catalisa o processo de construção da segunda fita do DNA a partir de uma fita inicial solta, esse método é conhecido como ***Polymerase Chain Reaction*** (PCR).

Sequenciamento

O sequenciamento, isto é, a identificação das bases que compõem uma molécula ou porção de DNA, é efetuado por meio de uma técnica denominada **eletroforese em gel**. Primeiramente, através de uma replicação controlada, são gerados separadamente para cada base todos os fragmentos possíveis que terminam com aquela base (Figura 2.1).

TGAACTGCCACAGT

TGA	T	TG	TGAAC
TGAA	TGAACT	TGAACTG	TGAACTGC
TGAACTGCCA	TGAACTGCCACAGT	TGAACTGCCACAG	TGAACTGCC
TGAACTGCCACA			TGAACTGCCAC

Figura 2.1: Cada coluna indica todos os fragmentos da sequência original terminados em uma base particular.

A seguir, são preparados quatro blocos de gel fino — um para cada base — onde serão depositados os fragmentos correspondentes (por exemplo, os fragmentos terminados em G serão depositados no bloco da base G). Aplica-se então um campo elétrico nos blocos. Uma vez que o DNA é carregado negativamente, os fragmentos migram em direção ao eletrodo positivo, de tal forma que os fragmentos maiores migram mais lentamente pois o seu avanço é mais lento no gel. No decurso de várias horas, os fragmentos de DNA ficam espalhados ao longo do bloco de acordo com o tamanho, formando uma escada de faixas discretas, cada uma composta de uma coleção de moléculas de DNA de comprimento idêntico. Utilizando-se algum método para tornar possível a visualização dos quatro blocos com as faixas lado a lado, pode-se identificar a sequência de bases que compõem

o fragmento original a partir dessas faixas (Figura 2.2). Esse processo é chamado de **corrida**.

Figura 2.2: Resultado da eletroforese em gel em filme fotográfico.

Às vezes, essa técnica apresenta algumas falhas, pois as marcas podem estar borradas ou não muito claras, especialmente nas proximidades das bordas. Além disso, o tamanho dos fragmentos que podem ser sequenciados dessa maneira está limitado a cerca de 1000 bases.

Os sequenciadores automáticos capazes de sequenciar uma quantidade da ordem de centenas de amostras de DNA simultaneamente. O *MegaBace*, por exemplo, é capaz de sequenciar 96 amostras em uma corrida.

A técnica utilizada pelos sequenciadores modernos utiliza o mesmo princípio da eletroforese em gel, mas com algumas diferenças importantes. A primeira é que os fragmentos são depositados em um único bloco, chamado de capilar, ao invés de um bloco para cada base. A segunda é que a visualização dos resultados é feita através de um **eletroferograma**. O eletroferograma gerado pelo sequenciador apresenta quatro gráficos coloridos — cada um correspondendo a uma das quatro bases. Quando uma base é identificada em uma dada posição do fragmento, o gráfico daquela base apresentará um pico na posição correspondente. Assim, a identificação no eletroferograma da sequência dos picos e suas respectivas cores revela a sequência de bases do fragmento sequenciado (Figura 2.3).

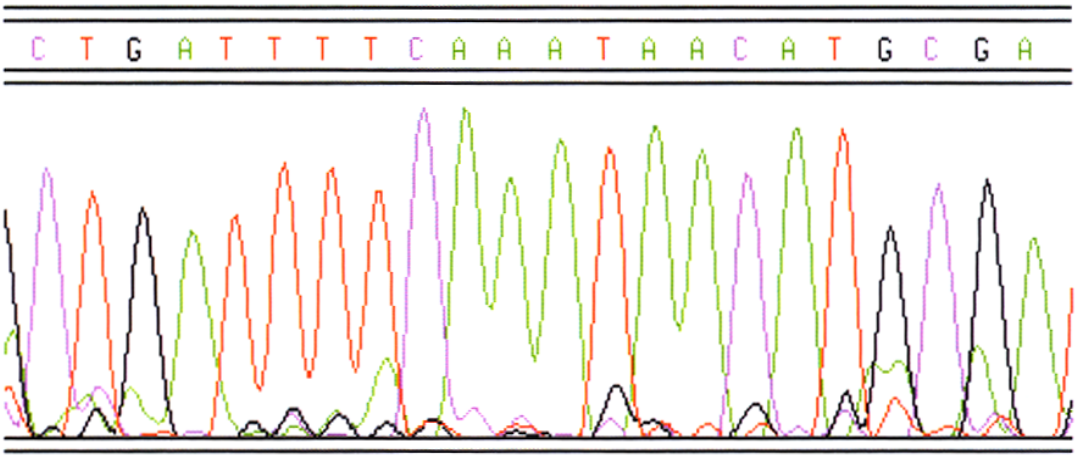


Figura 2.3: Exemplo de eletroferograma gerado por um sequenciador automático.

2.1.2 Sequenciamento de Alto desempenho

Embora o sequenciamento *Sanger* tenha sido a técnica de sequenciamento dominante durante os últimos anos, novas técnicas de sequenciamento massivamente paralelo estão no mercado e modificaram a forma como se realiza o sequenciamento de DNA no mundo. Ao permitirem o sequenciamento de milhões de sequências a um custo muito baixo, em comparação com o método *Sanger*, esses métodos tiveram um grande impacto nas áreas de pesquisa onde se realizam sequenciamentos de DNA, e abriram novas frentes de pesquisa, tais como o estudo de DNAs antigos, como mamutes, e a caracterização da diversidade ecológica por meio do sequenciamento de DNA de amostras ambientais (Mardis 2008).

Existem hoje no mercado alguns sequenciadores de alto desempenho, o 454-FLX Roche (Mardis 2008) e o Illumina Solexa (Mardis 2008). Esses sequenciadores funcionam diferentemente, o que dificulta a elaboração de um pipeline e de *softwares* que apóiam os projetos de sequenciamento, modificando o *pipeline* realizado na bioinformática desses projetos.

O sequenciador 454-FLX Roche foi o primeiro a aparecer no mercado, em 2004, e utiliza uma técnica de sequenciamento conhecida como pirosequenciamento. No pirosequenciamento a incorporação de cada nucleotídeo a uma fita de DNA por meio da enzima DNA polimerase acarreta a liberação de pirofosfato. Esta molécula por sua vez, inicia uma séria de reações químicas cujo produto final é a liberação de luz. A identificação das bases é feita através da detecção dessa liberação de luz. Uma característica desse processo é o aumento da liberação de luz quando um mesmo nucleotídeo é anexado a sequência. Isso dificulta a correta identificação da quantidade de bases de um monômero[1], gerando erros durante o sequenciamento.

O primeiro passo do processo de sequenciamento feito pelo 454-FLX Roche é a amplificação do DNA. Essa amplificação é feita misturando fragmentos de DNA com estruturas

[1]Monômero é uma molécula formada pela repetição de uma mesma estrutura, como por exemplo a sequência $AAAAAAAAAAA$

de agarose[2] contendo sequências de DNA complementares às sequências adaptadoras. Com isso cada estrutura de agarose fica ligada a um único fragmento de DNA. Essas estrutura contendo um fragmento de DNA são isoladas em micélios óleo/água contendo reagentes para a enzima DNA polimerase. Através de um ciclo térmico, produzem-se milhões de cópias dos fragmentos de DNA contidos na superfície da estrutura de agarose.

O próximo passo feito no 454-FLX Roche é o sequenciamento. Cada estrutura de agarose é colocada em um recipiente de estrutura sílica capilar. Esses recipientes fornecem uma localização fixa para o monitoramento das reações de sequenciamento. Em cada recipiente, enzimas catalizam a reação de pirosequenciamento são adicionadas e a mistura é centrifugada com o objetivo de cobrir as agaroses.

Por fim, cada nucleotídeo é anexado na sequência e durante esse processo um senso CCD registra a luz emitida, assim determinando a sequência de DNA. Todo esse processo é ilustrado na Figura 2.4. Esse sequenciador provê em média 100 milhões de sequências de boa qualidade com aproximadamente 250 bases de comprimento após um período de 8 horas.

O sequenciador Illumina Solexa utiliza técnicas de sequenciamento por síntese, onde todos os quatro nucleotídeos são anexados simultaneamente a uma célula. Essa célula é um dispositivo micro-fabricado de vidro com oito canais que permite uma ampliação *bridge* dos fragmentos em sua superfície e utiliza DNA polimerase para produzir cópias de DNA, o que é chamado de amplificação.

Após esse processo são às moléculas de DNA nucleotídeos fosforescentes. Como o grupo 3'-OH é quimicamente bloqueado, a DNA polimerase é capaz de anexar somente um fragmento por vez às sequências amplificadas. Durante esse processo são processadas imagens das luzes oriundas dos nucleotídeos fosforescentes. Por fim esses nucleotídeos são quimicamente removidos. Esse processo é repetido uma quantidade de vezes determinada pelo operador. Esse processo por completo é descrito nas Figuras 2.5 e 2.6. O Illumina Solexa é capaz de determinar mais de um bilhão de sequências com 25 a 35 bases de comprimento.

2.2 Conceitos de Bioinformática

Até o final dos anos 80, as metodologias manuais utilizadas para gerar sequências de DNA exigiam um tempo muito maior que os sequenciadores automáticos usados atualmente. O crescente avanço da tecnologia tem permitido que os laboratórios de Biologia Molecular forneçam detalhes cada vez mais precisos sobre as estruturas estudadas. O enorme volume de informações acumulado desde então e a necessidade de trabalhar esses dados eficientemente criou uma série de problemas que são, por natureza, interdisciplinares. Em particular, as teorias da matemática e da computação tornaram-se fundamentais no processo de manipulação de dados científicos dentro da Biologia Molecular (Setúbal and Meidanis 1997).

[2]A agarose é um polímero composto de subunidades de galactose. Quando dissolvida em água quente e seguidamente arrefecida, a agarose toma uma consistência gelatinosa. Este gel é muito utilizado na biologia molecular para processos como sequenciamento.

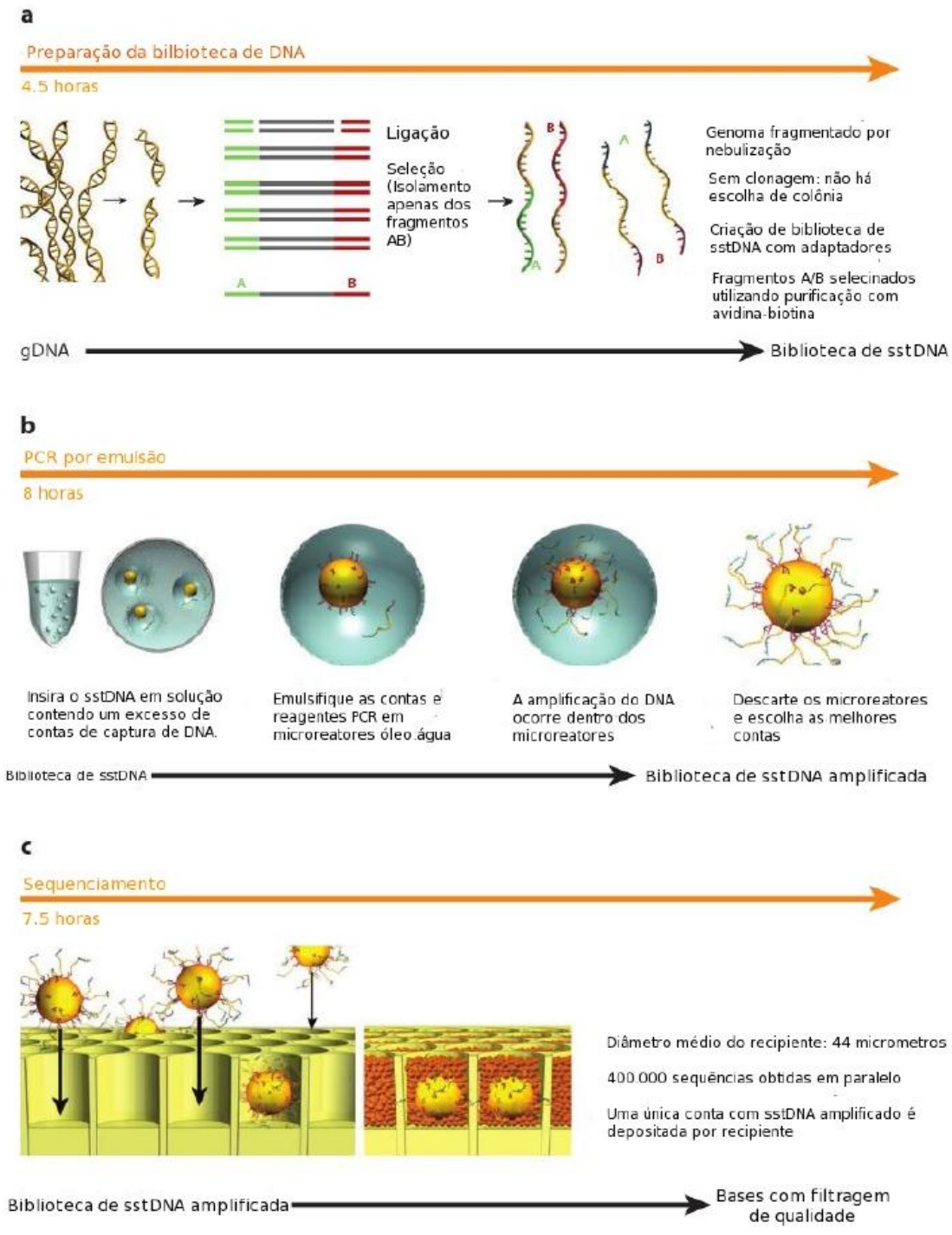


Figura 2.4: Processo de sequenciamento do 454-FLX Roche (Mardis 2008).

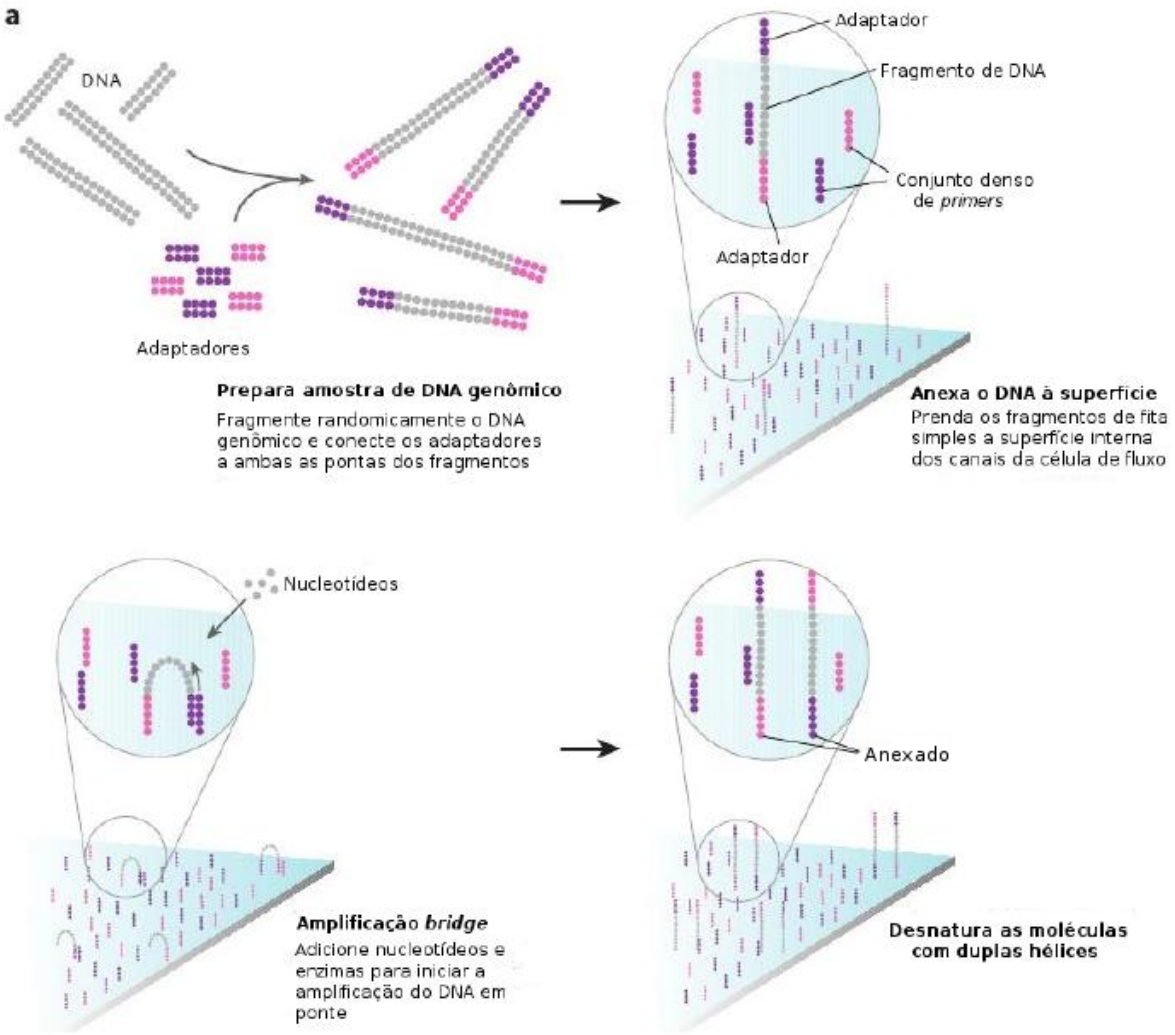


Figura 2.5: Processo de amplificação do Illumina Solexa (Mardis 2008).

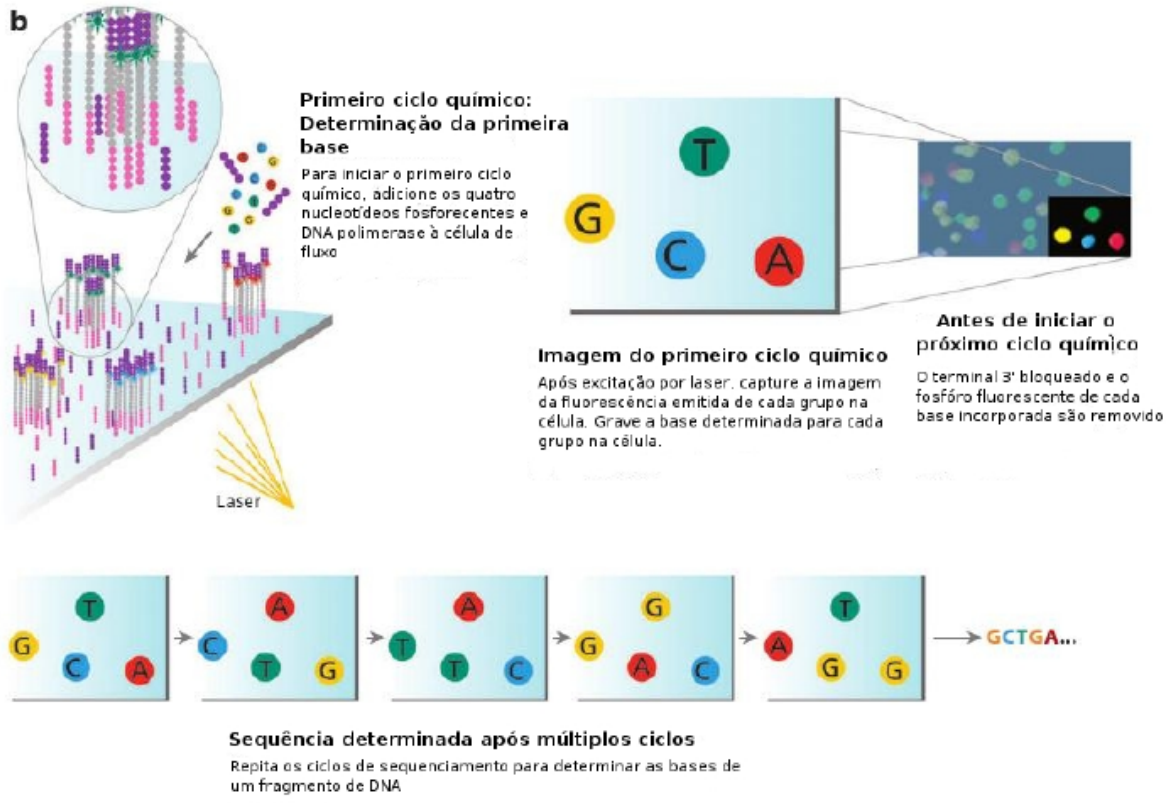


Figura 2.6: Processo de sequenciamento do Illumina Solexa (Mardis 2008).

É nesse contexto que surge a **Bioinformática**, uma nova área do conhecimento que visa estudar e aplicar técnicas e ferramentas computacionais na organização e interpretação das informações associadas às estruturas estudadas na Biologia Molecular (Luscombe et al. 2001). A Bioinformática, apesar de ser um campo relativamente novo, tem evoluído dramaticamente nos últimos anos e, hoje, é fundamental para compreender vários aspectos dentro da Biologia Molecular (Gibas and Jambeck 2001).

Há três objetivos principais dentro da Bioinformática. O primeiro é prover um meio de organizar os dados biológicos de forma a tornar fácil o acesso às informações e a submissão de novos dados à medida que estes são gerados. Diversos bancos de dados biológicos foram e continuam sendo criados com esse objetivo. O segundo objetivo é o desenvolvimento de ferramentas para ajudar a analisar esses dados armazenados. Por exemplo, depois de sequenciar uma determinada proteína é interessante compará-la com outras sequências já produzidas em busca de alguma similaridade estrutural ou funcional. O terceiro objetivo é o processamento dos resultados gerados por essas ferramentas para analisar e interpretar as informações de uma maneira que seja biologicamente consistente e relevante (Luscombe et al. 2001).

Tradicionalmente, pesquisas em Biologia investigavam um sistema isoladamente e o comparava com outros poucos sistemas relacionados. A Bioinformática permitiu conduzir análises globais envolvendo um volume muito maior de dados, de forma a descobrir princípios comuns e características importantes mais facilmente (Luscombe et al. 2001).

2.2.1 *Pipeline* de um projeto de sequenciamento de genoma *Sanger*

O *Pipeline* computacional para um projeto de sequenciamento de genoma refere-se a uma sequência de passos a serem seguidos para a realização do processamento de sequências biológicas. Em projetos de sequenciamento de genoma baseado na técnica *Sanger*, o *pipeline* é constituído de três grandes fases: Submissão, Montagem e Anotação, que são executadas em sequência, sendo que os dados resultantes de cada fase são utilizados como dados de entrada da fase seguinte.

Na Figura 2.7, apresentamos as três fases que formam o pipeline.

Figura 2.7: As três grande fases de um *Pipeline* para um projeto de sequenciamento de genoma baseado na técnica *Sanger*.

Submissão

Cada fragmento do DNA (também chamado *read*) é colocado em um espaço específico e marcado de uma placa contendo vários fragmentos, de tal forma que esta placa será pro-

cessada por algum sequenciador automático (por exemplo, o MegaBace). O sequenciador automático gera um arquivo de eletroferograma, que contém gráficos que identificarão cada uma das bases nitrogenadas da *read* este arquivo de probabilidade de error é gerado pelo programa *phred*.

O início da etapa de Submissão ocorre quando os biólogos enviam arquivos compactados com os eletroferogramas. A Figura 2.8 mostra um exemplo da fase de submissão, para o Projeto Genoma *Paracoccidioides brasiliensis* - Pb.

Para submeter os arquivos com todas as sequências de uma placa, o pesquisador deve informar seu nome (usuário) e senha, para que o sistema possa realizar a autenticação. Caso a autenticação seja confirmada, o sistema criará uma sessão e o pesquisador terá acesso à todas as informações do sistema. Na submissão, o pesquisador envia o arquivo compactado (formato zip, por exemplo). Quando a transferência é concluída, inicia-se a análise. O sistema descompacta o arquivo e executa o programa *Phred* que transforma os gráficos em *strings* e calcula a qualidade das bases, gerando arquivos com extensão *phd*. O sistema executa em seguida um programa chamado *Phd2fasta*, que converte os arquivos *phd* em arquivos no formato *fasta*.Observamos que o *fasta* é um padrão utilizado em projetos de sequenciamento e possui um formato como a apresentada na Figura 2.9.

Após a execução do *Phd2fasta*, o sistema executa o *Crossmatch* para retirar os vetores da sequência. Neste ponto termina a etapa de submissão, sendo gerados dois arquivos, um contendo a sequência de caracteres correspondentes às bases do fragmento sequenciado e outro contendo as qualidades de cada base dessa sequência. São gerados também relatórios de acordo com as necessidades do projeto.

Montagem

O processo de montagem consiste em tentar agrupar as sequências que foram aceitas na etapa de submissão, de tal forma que sequências contendo extremidades similares são unidas no mesmo conjunto. São processados dois arquivos de entrada, um contendo as sequências e outro contendo as qualidades e o programa de montagem das sequências gera como saída um grupo de *contigs* e outro de *singlets*. *Contigs* são sequências formadas a partir de um grupo contendo duas ou mais sequências e *singlets* são formados por um única sequência que não foi agrupada com nenhuma outra. A Figura 2.10 mostra um exemplo da etapa de montagem, para o Projeto Genoma Pb.

Anotação

Esta etapa é dividida em anotação automática e anotação manual.

O processo de anotação automática executa programas que comparam as sequências geradas no projeto com sequências armazenadas em bancos de dados que já tiveram suas funções determinadas.

A anotação manual realizada pelo biólogo, por interface web amigável, que decide a função associada a cada gene. A Figura 2.11 mostra um exemplo do processo de anotação para o Projeto Genoma Pb.

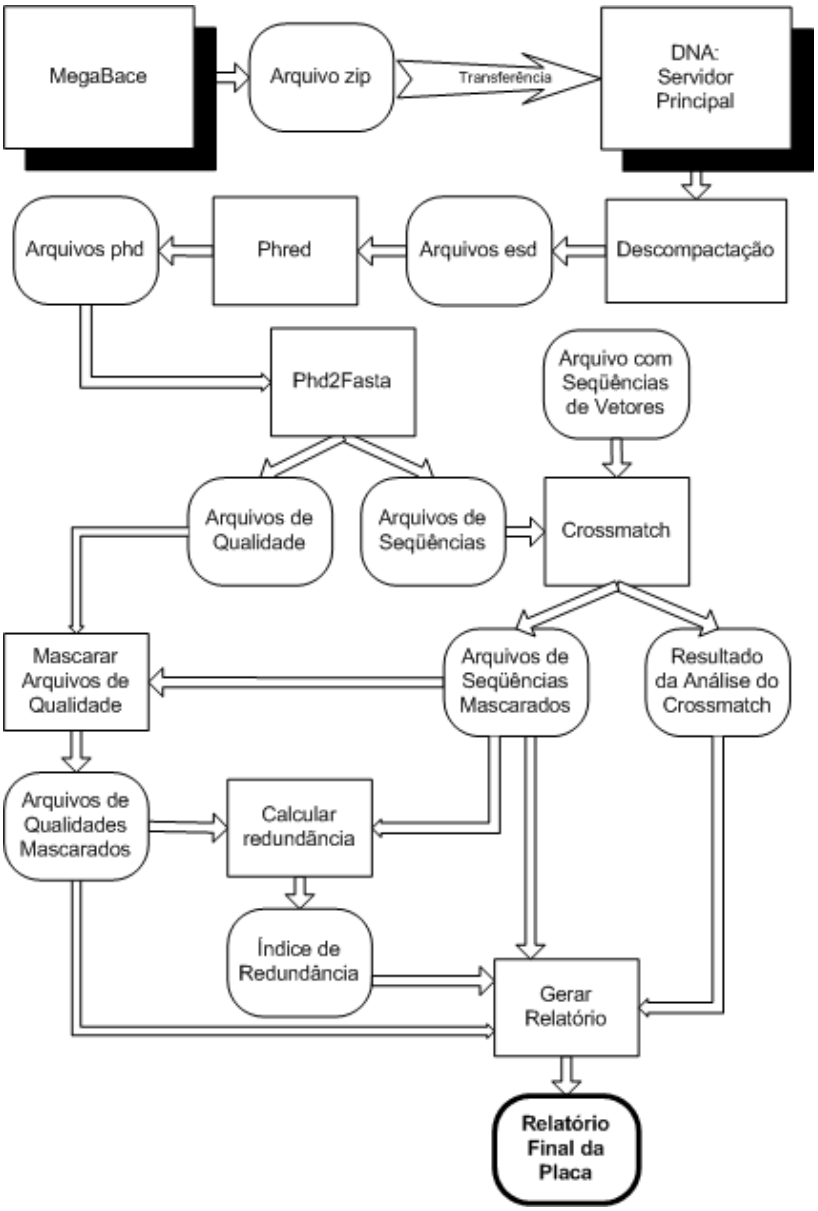


Figura 2.8: Exemplo de *Pipeline* da etapa de Submissão do Projeto Pb.

```
TGATCCCCGGCTGCAGGAATCGGCACGAGGCCCCCGATATAATATATACAGGCTTTTCAT
TCATAATTCTATCATTCATTCCATAAGACCATTCCTGTTAACACTTCTATCCAACCGCGA
CGGCCACGCCCATTGTCACAAACATTTTGGCGTCAGAGAGAACCTTAGGCGTGCATTGTG
TTCTGGCAATTGCAGTCTTCTCCCTCCTGCTTCACCACTTTAACAACATAAAATTATCCT
TTCAATATCCGTCATTATATAACCAAATGCCCCGCTTACCTATATTCATATCCTTGAAGA
```

Figura 2.9: Exemplo de um arquivo no formato Fasta.

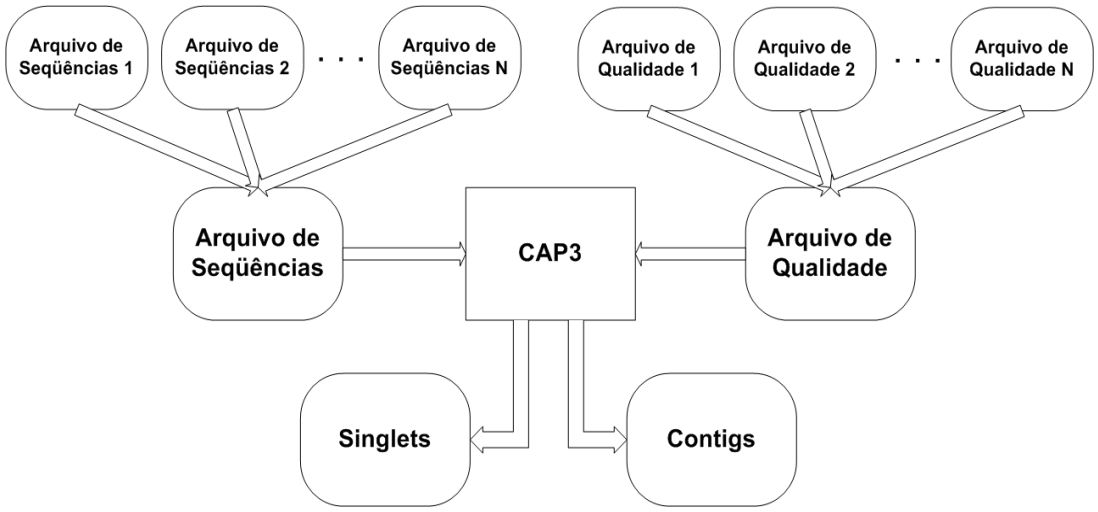


Figura 2.10: Exemplo de *pipeline* da etapa de Montagem do Projeto Genoma Pb.

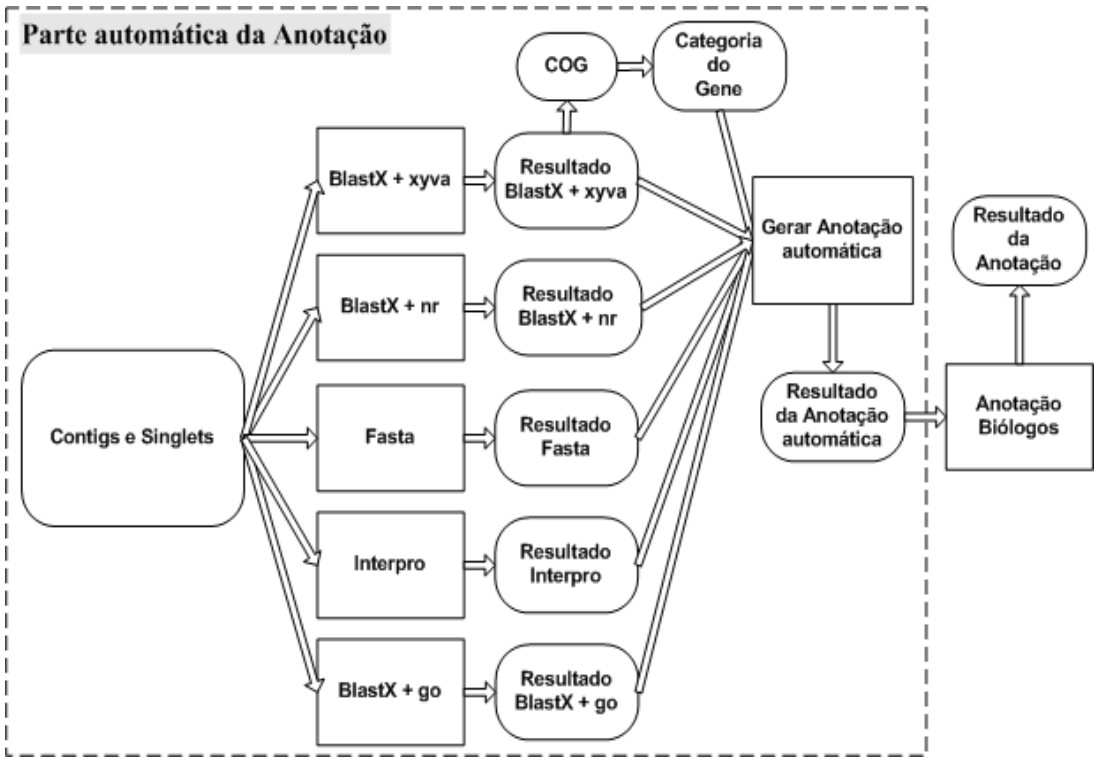


Figura 2.11: Exemplo de *pipeline* da etapa de Anotação do Projeto Genoma Pb.

2.2.2 *Pipeline* de um projeto de sequenciamento de genoma de alto desempenho

Devido as características das sequências obtidas pelos novos sequenciadores, dois *pipelines* são possíveis. A escolha de qual *pipeline* adotar dependerá, entre outros, do sequenciador utilizado, das características dos dados gerados, da disponibilidade de outros dados.

O primeiro tipo de pipeline possível e constituído de três fases, a saber mapeamento, montagem e anotação. É mais indicado para sequenciadores como o Illumina Solexa, que produzem sequências muito pequenas, impedindo a montagem direta de todos os fragmentos. A submissão, onde os dados são obtidos e armazenados, é feita pelos próprios sequênciadores automáticos.

A etapa chamada de mapeamento, consiste no alinhamento das sequências a um genoma de referência, formando grupos de sequências próximas. A seguir, cada um desses grupos é montado por meio de um software de montagem, obtendo assim os *singlets* e os *contigs*. Por fim, os grupos são anotados utilizando-se algoritmos de alinhamento, como por exemplo o BLAST (Altschul et al. 1990; 1997).

O outro pipeline possível contém apenas as etapas de montagem e anotação. Esse *pipeline* é indicado quando as sequências produzidas pelo sequenciador permitem a montagem dos fragmentos diretamente ou quando ainda não há um genoma de referência que permita a execução da fase de mapeamento. Nesse pipeline, as sequências obtidas do sequenciador são montadas, sendo obtidos os *singlets* e os *contigs*. Por fim, na fase de anotação, a função de cada um dos grupos é anotada utilizando os algoritmos já conhecidos desde o sequenciamento *Sanger*.

2.2.3 Anotação Genômica

Conforme mencionado, o processo de interpretar os dados gerados em um projeto de sequenciamento, tornando-os informações biologicamente relevantes, é chamado de anotação (Lewis et al. 2000). Com o extraordinário progresso das técnicas biológicas de sequenciamento, as etapas de anotação, correção e verificação da relevância das informações têm sido um fator limitante na maioria dos projetos de sequenciamento. Embora a anotação de um genoma completo forneça uma perspectiva geral sobre este, não descreve detalhadamente todos os genes. Para o aprimoramento do processo de descrição dos genes, inúmeros programas foram e vêm sendo desenvolvidos, visando, por exemplo, a predição de genes, elementos regulatórios, sítios de iniciação da tradução, localização de motivos em sequências, entre outros (Pertsemlidis and Fodon 2001). É importante mencionar que, além da automatização da análise dos dados, uma visão biológica é essencial na interpretação e correção das informações (Andrade 2002).

Como dito anteriormente, essa etapa dos projetos de sequenciamento é feita no computador, por meio de programas de análise e de consulta a bancos de sequências destinados à anotação genômica. Observamos que nos projetos de sequenciamento *Sanger* esta etapa era parcialmente feita pelos biólogos. Sendo assim, essa etapa é denominada **anotação semi-automática**.

Os programas e bancos de sequências utilizados na anotação variam de acordo com as necessidades de cada projeto de sequenciamento. Nesta seção descreveremos algumas ferramentas comumente utilizadas.

Ferramentas de Análise

Nesta seção, apresentaremos inicialmente programas para comparar sequências geradas nos projetos com sequências armazenadas nos banco de dados com informações biologicamente relevantes.

BLAST

BLAST (*Basic Local Alignment Search Tool*) é uma família de ferramentas para análise de similaridade entre sequências e está entre os programas mais usados em pesquisa de bancos de sequências no mundo (Setúbal and Meidanis 1997).

BLAST utiliza uma heurística que tenta otimizar uma medida de similaridade específica. A complexidade espacial do algoritmo utilizado é expressa pelo produto do comprimento da sequência a ser consultada (m) e comprimento das sequências pertencentes a base de sequências(n), ou seja, $O(mn)$ (Altschul et al. 1990; 1997).

BLAST pesquisa pequenos pares de subsequências, uma da sequência a ser consultada e outra de uma das sequências do banco, cujo alinhamento tem qualidade mínima de acordo com informações do usuário. Alguns desses alinhamentos são ligados para produzir alinhamentos maiores, porém, mantendo a qualidade mínima. BLAST utiliza então programação dinâmica e procedimentos heurísticos para produzir os alinhamentos finais com espaços. No caso de pesquisa de sequências de proteínas, o BLAST utiliza matrizes de substituição para pontuar os pares de resíduos alinhados (Altschul et al. 1990; 1997).

BLAT

O BLAT é uma ferramenta de alinhamento de sequências genéticas mais rápida que as ferramentas mais populares (Kent 2002), similar ao BLAST. Esta ferramenta foi utilizado para resolver dois problemas durante o sequenciamento do genoma humano, alinhar milhões de genes humanos e alinhar milhões de genes inteiros e escolhidos aleatoriamente de ratos contra genes humanos.

FASTA

FAST é uma outra família de programas para consulta a bancos de sequências. O primeiro programa a ser disponibilizado publicamente, o FASTP, foi projetado para pesquisar apenas sequências de proteínas. O algoritmo básico usado pelo FASTP era comparar cada sequência do banco com a sequência a ser pesquisada e retornar aquelas mais similares junto com os alinhamentos e outras informações importantes. A velocidade do FASTP era devida à sua eficiência em comparar duas sequências muito rapidamente.

Além das ferramentas de pesquisa, a família FAST inclui um programa bastante útil para estabelecer o rigor estatístico de uma pontuação. Melhorias na detecção de similaridades no programa FASTP levaram à criação do FASTA. Uma característica incorporada foi a capacidade de processar DNA assim como sequências de proteínas. FASTA pode ser visto como uma combinação de FASTP com FASTN, um programa projetado especificamente para analisar sequências de nucleotídeos. Um outro programa dessa família, TFASTA, compara uma sequência de proteínas com um banco de sequências de DNA, fazendo as traduções nas seis fases de leitura (`http://fasta.bioch.virginia.edu/`).

Uma outra observação relaciona-se com a computação das pontuações iniciais. FASTA executa um passo adicional depois que as melhores regiões foram selecionadas: tenta juntar sequências que estejam em regiões vizinhas, mesmo que elas não pertençam à mesma diagonal. Com isso, as pontuações iniciais melhoram significativamente para sequências relacionadas. Além disso, as 10 melhores regiões são escolhidas em FASTA, enquanto que no FASTP eram apenas 5. Outros programas que usam as mesmas técnicas também fazem parte do pacote. LFASTA é uma ferramenta para similaridades locais, reportando mais do que um bom alinhamento entre um par de sequências.

PFAM

O **Pfam** (`http://pfam.wustl.edu/`) é banco de alinhamentos de famílias de domínios protéicos (região de uma proteína com uma função específica). Um exemplo é a DNA polimerase, que possui um domínio responsável exclusivamente pela ligação com a fita de DNA. O Pfam é dividido em dois bancos menores: o Pfam-A e o Pfam-B . O banco Pfam-A contém mais de 2700 perfis de alinhamento (a maioria destes perfis abrange domínios protéicos inteiros), sendo que esses alinhamentos foram montados permitindo-se a inserção de espaços. Por sua vez, o banco Pfam-B é gerado automaticamente a partir do agrupamento das sequências que sobraram durante o processo de construção do Pfam-A.

PSORT

O ***PSORT*** é um pacote de programas, desenvolvido por Nakai em 1991, para realizar a predição da localização de proteínas nas células. De acordo com o tipo de organismo ao qual a proteína pertence, um conjunto de localizações possíveis é testado. O *PSORT* verifica se a proteína pertence ao citoplasma, à membrana interna, à membrana externa ou ao periplasma da célula, um espaço situado entre a membrana externa e membrana citoplasmática (`http://www.psort.org/`).

INFERNAL

O ***INFERNAL*** é um pacote de ferramentas que permite ao usuário fazer *perfis*[3] em estruturas secundárias de RNA similares ou criar novas estruturas baseadas em alinha-

[3] *Perfis* de alinhamento são representações numéricas, normalmente matriciais, de um alinhamento múltiplo. Esse perfil representa as características comuns àquele particular conjunto de sequências, que é, em geral, uma família de proteínas.

mentos múltiplos de sequências (Eddy 2005). O INFERNAL é também capaz de fazer a detecção de RNAs não codificadores.

O *INFERNAL* é comparável ao *HMMER* (Vide http://hmmer.wustl.edu/). O pacote de ferramentas *HMMER* utiliza Modelos de Markov - de sequências lineares - para produzir *perfis* de alinhamentos múltiplos de sequências, chamados de *perfis* HMMs. Os *perfis* HMMs capturam apenas características de estruturas primárias similares. Os *perfis* produzidos pelo *INFERNAL* incluem informações sobre sequências e estruturas secundárias de RNAs.

O INFERNAL é a implementação de modelos probabilísticos para a análise de estruturas de RNA, chamados de Modelos de Covariância (MC). Uma vantagem considerável desta implementação foi o ganho na complexidade do algoritmo de programação dinâmica para MC em estruturas de RNA (em relação aos baseados nos Modelos de Markov). O algoritmo base do INFERNAL é análogo ao algoritmo de Myers/Miller (para alinhamentos de sequências lineares), porém tem uma complexidade da ordem de $O(N^2 \log N)$ enquanto outros possuem complexidade da ordem de $O(N^3)$ (Eddy 2002).

A ferramenta gera saídas robustas, porém com falta de muitos dados acerca de características relevantes para os especialistas. É muito importante ressaltar que os algoritmos do *INFERNAL* requerem um enorme tempo de processamento, tornando-o uma ferramenta muito lenta e que os projetos que venham a necessitar de seu uso provavelmente vão requerer o uso de vários processadores através de um ambiente distribuído, tal como computação em grade por exemplo.

Portrait

O Portrait é uma ferramenta baseada em Máquinas de Vetores de Suporte, apresentada e desenvolvida por Arrial e co-autores (Arrial et al. 2009). Essa ferramenta possibilita a análise de ncRNA's de transcriptomas de espécies pouco caracterizadas. O resultado dado pelo Portrait é a probabilidade de uma sequencia não codificar uma proteína.

2.3 Genoma Estrutural e Funcional

Nos projetos de sequenciamento existem dois possíveis caminhos para analisar o DNA de um organismo: projeto **genoma estrutural** ou projeto **genoma funcional**. Os projetos de genoma estruturais visam sequenciar todo o DNA do organismo, incluindo as partes que não codificam proteínas (íntrons e regiões intergênicas). Já nos projetos de genoma funcional, o enfoque está no sequenciamento das regiões gênicas cujos transcritos efetivamente farão parte do mRNA final, que será traduzido nos ribossomos. O objetivo principal nos projetos genoma funcionais é conhecer os genes que são expressos do organismo estudado.

Tanto os projetos genoma estruturais como os funcionais têm dedicado uma parte do esforço ao sequenciamento de **ESTs** (*Expressed Sequence Tags*). O conceito de EST foi proposto no final dos anos 80 (Wolfberg and Landsman 2001). Uma EST é uma sequência obtida do mapeamento de uma porção do cDNA, gerado pela transcrição reversa do

mRNA citoplasmático, e pode ser feito a partir da extremidade 5′ (EST 5′) e/ou da extremidade 3′ (EST 3′) (Adams et al. 1991). O esquema de produção de ESTs pode ser observado na Figura 2.12.

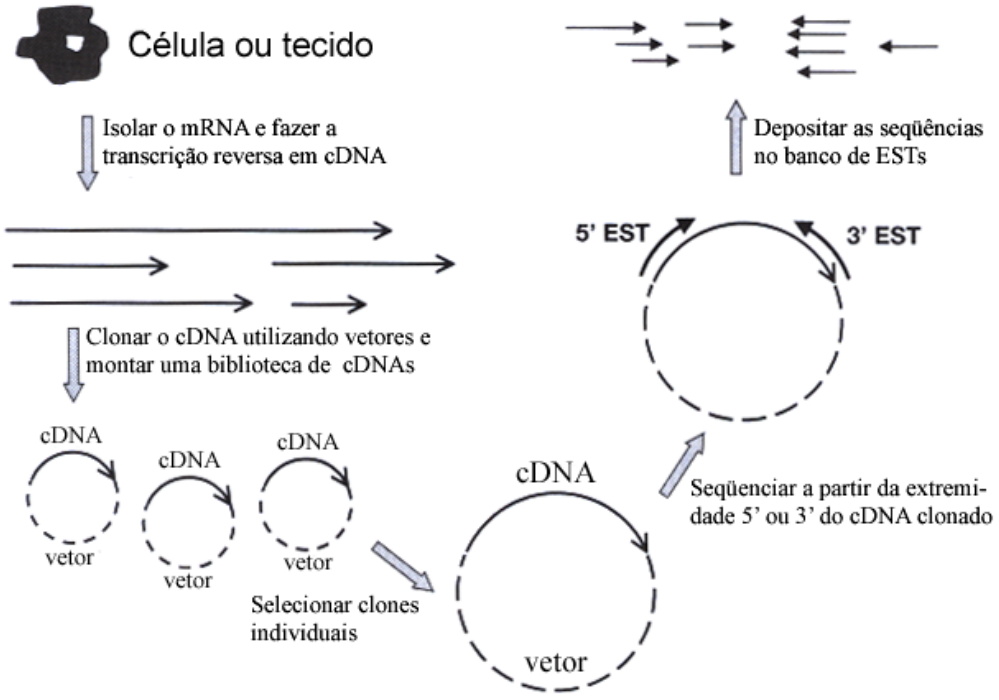


Figura 2.12: Ciclo de produção de ESTs.

Geralmente ESTs têm tamanho pequeno, variando em média de 200 a 800 pares de bases e representam fragmentos de genes e não sequências completas. Geralmente ESTs são sequenciadas uma única vez. Por isso, as sequências podem apresentar uma maior frequência de erros do tipo remoções, substituições e inserções de bases quando comparadas ao mRNA original (Wolfberg and Landsman 1997). Além disso, essas sequências comumente apresentam contaminações com material genético do vetor, que precisa ser detectado e retirado antes, de prosseguir na análise final dessas sequências, mas isso pode ser feito por métodos computacionais.

O sequenciamento usando ESTs é uma estratégia rápida e eficiente na prospecção de novos genes, pois permite a análise da informação efetivamente expressa por estes, abordando principalmente o estudo de proteínas sem a necessidade de enfocar elementos estruturais ou de regulação gênica. Nesta estratégia, é possível trabalhar com pequenas sequências, dispensando o esforço e os custos de efetuar a montagem completa do genoma. Entretanto, o conhecimento de uma sequência genômica completa às vezes torna-se necessário, pois provê informações específicas, que nem sempre podem ser obtidas apenas com o sequenciamento de ESTs.

Desde a descrição original do projeto de sequenciamento de 609 ESTs, em 1991 (Adams et al. 1991), tem havido um acentuado crescimento no número de ESTs depositadas nos bancos públicos de sequências. Em meados de 1995 o número de ESTs no *GenBank*

(http://www.ncbi.nlm.nih.gov/Genbank/) ultrapassou o número de não-ESTs. Já em junho de 2000 os registros chegaram a 4,6 milhões de ESTs, constituindo 62% do total de sequências depositadas no GenBank. Apesar do fato das ESTs originais serem de origem humana, o banco dbEST do NCBI (http://www.ncbi.nlm.nih.gov/) contém atualmente ESTs de mais de 250 organismos (Wolfberg and Landsman 2001).

Em sequenciamento de alto desempenho, a produção de sequências expressas, denominadas de cDNA, é realizada por um conjunto de técnicas descritas de forma breve na Seção 2.1.2.

2.3.1 Genoma Funcional

Atualmente, novas sequências biológicas são anotadas funcionalmente simplesmente da comparação com sequências existentes, armazenadas em bancos de dados como, por exemplo, o GenBank (Rigden and de Mello 2002). A anotação feita nessa abordagem compara a nova sequência com outras sequências bem caracterizadas, verifica o grau de similaridade entre essas sequências e transfere a função da sequência bem caracterizada mais similar para a nova sequência.

A grande vantagem do genoma funcional é a sua eficiência em anotar novas sequências. Essa eficiência se mostra muito importante hoje devido ao grande volume de sequências obtidas nos projetos de sequenciamento. Porém existem duas falhas nesse método. Uma não muito grave é a incapacidade de anotar novas sequências que não apresentam similaridades com nenhuma outra sequência previamente conhecida. A falha mais grave nesse método é a grande probabilidade de se introduzir e propagar um erro de anotação nas bases de dados. Isso se deve ao fato que as anotações transferidas por métodos de comparação são muito mais frequentes do que as anotações produzidas por experimentos laboratoriais. Uma grande fonte de erros é a má interpretação dos resultados de alinhamento, feitos, por exemplo, pelo BLAST (Altschul et al. 1990; 1997). Esse alinhamento mede a similaridade local de duas sequências o que em alguns casos não é uma característica suficiente para caracterizar uma transferência da função biológica.

Existem cinco métodos que são classificados como genoma funcional. Três deles são dependentes das sequências obtidas em projetos de genomas completos [4], que são os perfis filogenéticos, contexto genômicos e genoma diferencial. Os outros dois métodos restantes são similaridade de árvores filogenéticas e o método baseado nas consequências de eventos de fusão de genes.

2.3.2 Genoma Estrutural

Sabe-se que tanto a estrutura de uma proteína quanto sua estrutura tridimensional podem determinar sua função. Os métodos tradicionais funcionam bem devido às conhecidas relações entre sequência, estrutura e função de proteínas (Rigden and de Mello 2002).

[4] O termo genoma completo se refere aos projetos de genoma que sequenciam todos o conteúdo genético de um organismo.

Verifica-se que, de modo geral, proteínas de uma mesma família não possuem similaridades em suas sequências, mas conservam mesma estrutura tridimensional.

A parte que caracteriza a função de uma proteína são os resíduos, que orientados em suas corretas posições, servem de interfaces para a ligação da proteína a outras moléculas. Como esses resíduos podem não ser conservados mesmo com grande conservação da sequência ou pequenas mudanças na sequência mudaram o posicionamento desses resíduos, impedindo assim o acesso a eles, a simples análise de uma sequência pode levar a conclusões erradas sobre a função, introduzindo assim erros aos bancos de dados. Esse problema pode ser evitado através de uma extrapolação da sequência em estrutura, denominado de modelagem protéica.

Existem duas categorias de ferramentas de bioinformática estrutural utilizadas para inferência de funções a partir da estrutura protéica. A primeira categoria de ferramentas busca por possíveis sítios de ligação e a segunda tenta localizar possíveis sítios de catálise, que só são aplicáveis a enzimas.

2.3.3 Anotação dos Projetos Genoma Pb e Genoma Guaraná

O Projetos Genoma Pb e Genoma Guaraná, foco dos experimentos desse trabalho, são genomas funcionais. Ambos anotados com base na similaridade das sequências obtidas nos projetos. Nos dois projetos foram utilizados as seguintes classificações quanto a anotação da sequências:

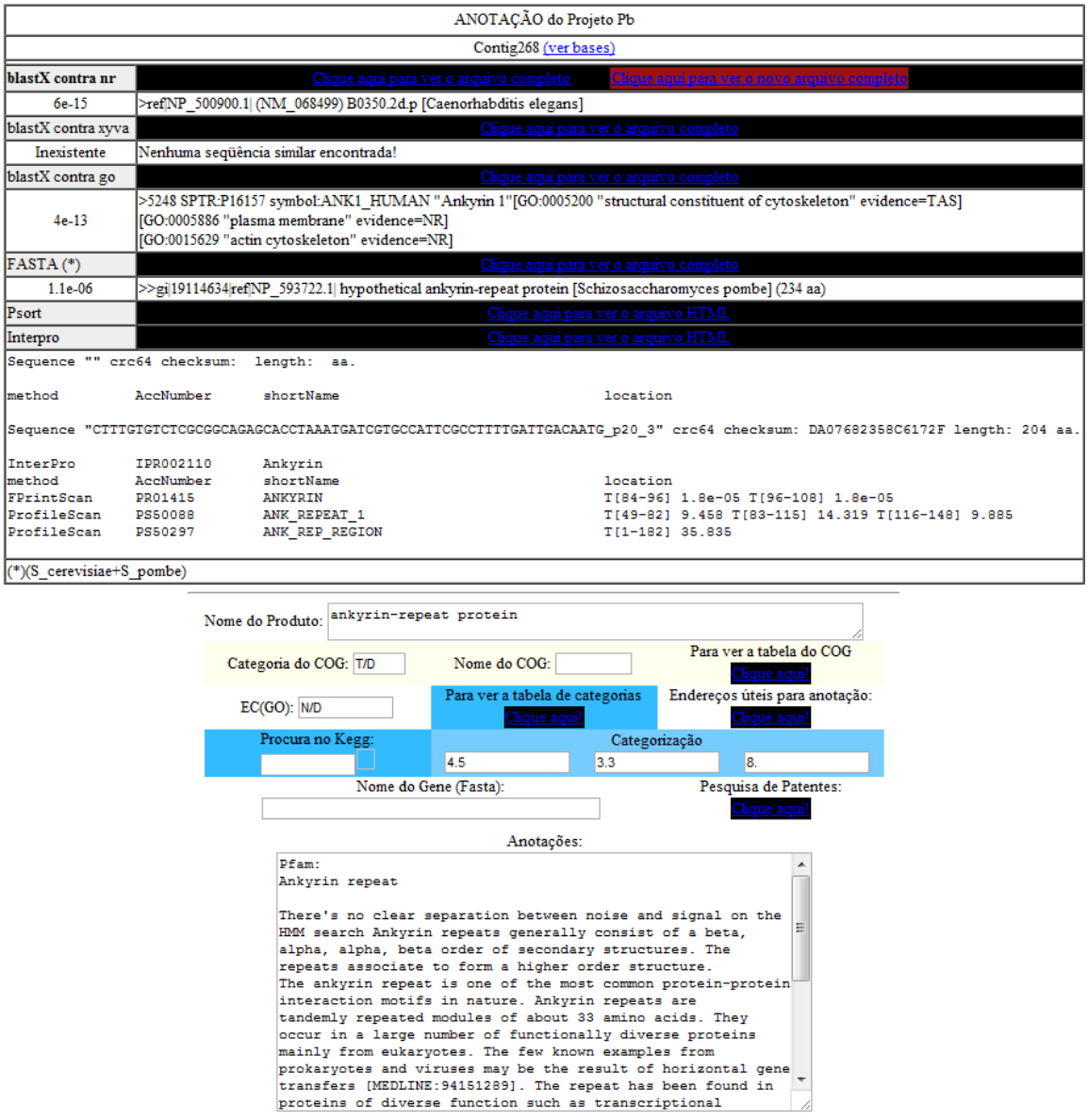
- Anotação com sucesso - a anotação automática reportou bons alinhamentos que deram subsídios a anotação do campo;
- Não-conclusivo (nc) - a anotação automática reportou alinhamentos com valores de *e-value*; menores que 10^{-5} e baixos *scores*
- Não-identificado (nid) - a anotação automátia não reportou alinhamentos.

O Projeto Genoma Pb foi anotado com base nos resultados da anotação automática feita com cinco bancos de dados, *nr*, *COG*, *GO*, *Saccharomyces cerevisiae* e *Schizosaccharomyces pombe*, e dois algoritmos, BLAST e FASTA. Os bancos de dados *nr*, *COG* e *GO* foram utilizados com o BLAST e os bancos de dados *S. cerevisiae* e *S. pombe* com o FASTA, perfazendo um total de 2820 genes anotados manualmente, como mostrado a Figura 6.2 da página 69.

A Figura 2.13 mostra uma anotação do Projeto Genoma Pb. A seção superior da tela mostra os resultados da anotação automática que foram utilizados na anotação manual. A função da sequência é descrita no campo Nome do Produto e esse dado é o foco da sugestão dada pelo *BioAgents*.

As anotações manuais do Projeto Genoma Guaraná foram feitas com base nos bancos de dados *nr*, *KOG*, *GO* e *SwissProt* utilizando a ferramenta BLAST, perfazendo um total de 7.725 genes anotados, como mostra a Figura 6.6 da página 73.

A Figura 2.14 mostra uma anotação do Projeto Genoma Guaraná. Análogo ao Projeto Genoma Pb, a seção superior da tela mostra os resultados da anotação automática que



ANOTAÇÃO do Projeto Pb

Contig268 (ver bases)

blastX contra nr	Clique aqui para ver o arquivo completo — Clique aqui para ver o novo arquivo completo
6e-15	>ref|NP_500900.1| (NM_068499) B0350.2d.p [Caenorhabditis elegans]
blastX contra xyva	Clique aqui para ver o arquivo completo
Inexistente	Nenhuma seqüência similar encontrada!
blastX contra go	Clique aqui para ver o arquivo completo
4e-13	>5248 SPTR:P16157 symbol:ANK1_HUMAN "Ankyrin 1"[GO:0005200 "structural constituent of cytoskeleton" evidence=TAS] [GO:0005886 "plasma membrane" evidence=NR] [GO:0015629 "actin cytoskeleton" evidence=NR]
FASTA (*)	Clique aqui para ver o arquivo completo
1.1e-06	>>gi|19114634|ref|NP_593722.1| hypothetical ankyrin-repeat protein [Schizosaccharomyces pombe] (234 aa)
Psort	Clique aqui para ver o arquivo HTML
Interpro	Clique aqui para ver o arquivo HTML

```
Sequence "" crc64 checksum:  length:  aa.

method          AccNumber       shortName                       location

Sequence "CTTTGTGTCTCGCGGCAGAGCACCTAAATGATCGTGCCATTCGCCTTTTGATTGACAATG_p20_3" crc64 checksum: DA07682358C6172F length: 204 aa.

InterPro        IPR002110       Ankyrin
method          AccNumber       shortName                       location
FPrintScan      PR01415         ANKYRIN                         T[84-96] 1.8e-05 T[96-108] 1.8e-05
ProfileScan     PS50088         ANK_REPEAT_1                    T[49-82] 9.458 T[83-115] 14.319 T[116-148] 9.885
ProfileScan     PS50297         ANK_REP_REGION                  T[1-182] 35.835
```

(*)(S_cerevisiae+S_pombe)

Nome do Produto: ankyrin-repeat protein

Categoria do COG: T/D — Nome do COG: — Para ver a tabela do COG: Clique aqui!

EC(GO): N/D — Para ver a tabela de categorias: Clique aqui! — Endereços úteis para anotação: Clique aqui!

Procura no Kegg: — Categorização: 4.5 3.3 8.

Nome do Gene (Fasta): — Pesquisa de Patentes: Clique aqui!

Anotações:

```
Pfam:
Ankyrin repeat

There's no clear separation between noise and signal on the
HMM search Ankyrin repeats generally consist of a beta,
alpha, alpha, beta order of secondary structures. The
repeats associate to form a higher order structure.
The ankyrin repeat is one of the most common protein-protein
interaction motifs in nature. Ankyrin repeats are
tandemly repeated modules of about 33 amino acids. They
occur in a large number of functionally diverse proteins
mainly from eukaryotes. The few known examples from
prokaryotes and viruses may be the result of horizontal gene
transfers [MEDLINE:94151289]. The repeat has been found in
proteins of diverse function such as transcriptional
```

Figura 2.13: Tela de anotação do Projeto Genoma Pb.

foram utilizados na anotação manual. A função da sequência também é descrita no campo Nome do Produto.

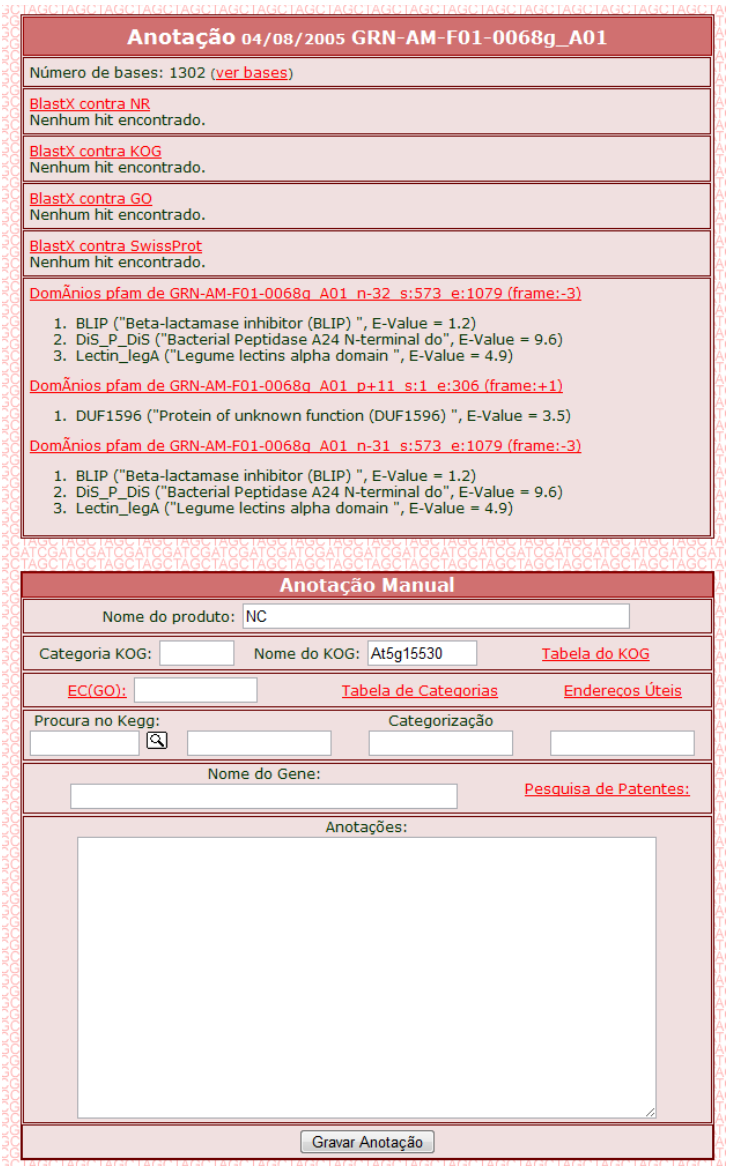


Figura 2.14: Tela de anotação do Projeto Genoma Guaraná.

Capítulo 3

Fundamentos de Inteligência Artificial

A área de IA possui várias definições, não existindo um consenso entre os pesquisadores. Segundo (Russell and Norvig 2002), as definiçõe podem ser separadas em quatro categorias: sistemas que pensam como humanos, sistemas que pensam racionalmente, sistemas que agem como humanos e sistemas que agem racionalmente. Para esse trabalho serão adotados as definições que se encaixam na quarta categoria, ou seja, sistemas que agem racionalmente, como definido em (Winston 1992):

> "The study of the computations that make it possible to perceive, reason, and act."

3.1 Agentes Inteligentes e Sistemas Multiagentes

Nesta seção, discutimos conceitos relacionados a IA, com foco no desenvolvimento de soluções com agentes inteligentes de software, bem como no agrupamento de agentes formado em um SMA, necessários para o entendimento deste trabalho.

3.1.1 Agentes Inteligentes

Podemos dizer que o conceito de **agente** segundo (Russell and Norvig 2002) é uma entidade capaz de perceber seu ambiente através de **sensores** e de agir sobre esse ambiente através de **atuadores**, conforme mostrado na Figura 3.1. Note que na figura, o módulo de raciocínio "?" formaliza a **função do agente**, que mapeia sequências de percepções específicas para determinadas ações.

Para um melhor entendimento da figura e dos termos mencionados acima, consideremos, por exemplo, um agente robótico. Neste agente poderíamos ter como sensores câmeras e detectores da faixa de infravermelho e seus vários motores, por exemplo, braços robóticos, esteiras de movimentação, funcionando como atuadores. Um outro exemplo poderia ser um agente de *software*, que recebe sequências de teclas digitadas, conteúdo de arquivos e pacotes de rede como entradas sensoriais e atua sobre o ambiente exibindo algo na tela, gravando arquivos e/ou enviando pacotes de rede.

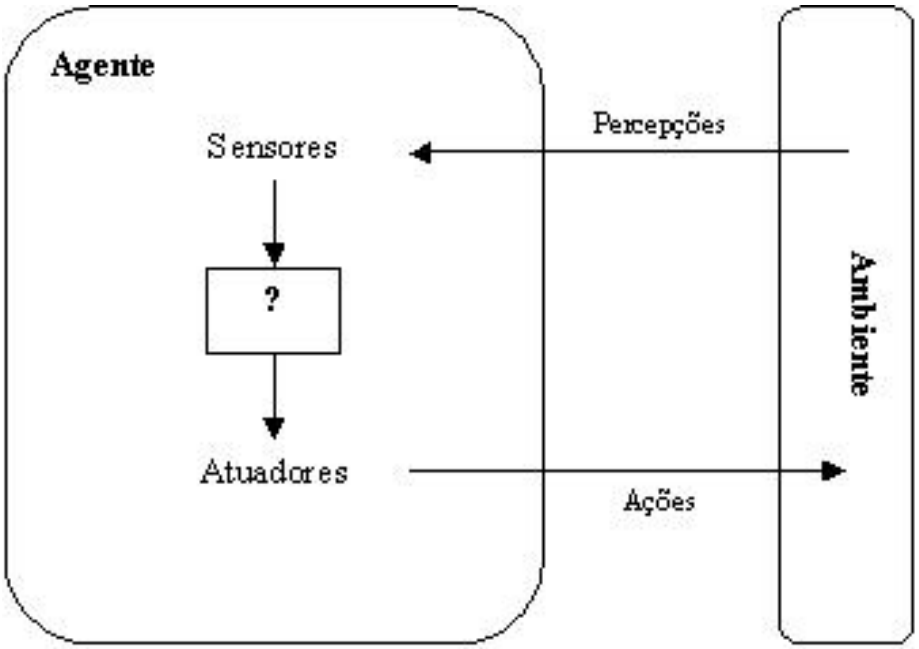


Figura 3.1: Características de um agente inteligente.

Segundo (Wooldridge 2009), um agente inteligente deve possuir como características desejáveis: autonomia, proatividade, sociabilidade (noção de sociedade) e reatividade. Um agente inteligente deve ser autônomo e agir sem intervenção humana. Por proatividade podemos entender que os agentes inteligentes devem ter iniciativas para que possam atingir seus objetivos. A noção de sociedade deve-se ao fato de que os agentes podem e devem se relacionar com os outros agentes, ou seres humanos no ambiente, por exemplo, a comunicação dentro de um SMA para alcançar seus objetivos. Por fim, o conceito de reatividade está ligado à percepção do ambiente pelo agente, e sua resposta pode ser entendida em tempo hábil às alterações ocorridas, sempre agindo de forma a alcançar seus objetivos (Wooldridge 2009).

Passaremos a relacionar as características desejáveis de um agente inteligente conforme (Sycara et al. 1996):

- Execução de tarefas: o agente pode executar tarefas delegadas tanto por humanos como por outros agentes que pertençam ao ambiente relacionado;
- Confiabilidade: o agente deve realizar com segurança e eficácia as tarefas delegadas pelo usuário;
- Adaptabilidade: o agente deve se adaptar bem as alterações das necessidades dos usuários e do ambiente;
- Colaboração: o agente deve colaborar tanto com os humanos quanto com os outros agentes. Esta característica permite que os agentes incrementem seus conhecimentos, resolvam conflitos e inconsistências nas informações recebidas e nas tarefas realizadas, assim melhorando a capacidade de suporte à decisão;

- Pro-atividade: o agente deve iniciar suas atividades para solução de problemas, antecipar informações e alertar os usuários sobre as informações relevantes, além de decidir quando mesclar informações para prover um melhor entendimento.

3.1.2 Sistema Multiagente

Há vertentes, direcionadas por diversos grupos de pesquisa, em relação ao paradigma de Agentes Inteligentes, na qual acredita-se que a inteligência não pode ser separada do contexto social, ou seja, não se deve considerar os agentes apenas como entidades inteligentes isoladas. A partir deste contexto surge a idéia de SMA (Ferber 1999, Wooldridge 2009).

Segundo (Wooldridge 2009), um SMA consiste de agentes que interagem entre si, tipicamente por envio de mensagens através de alguma infra-estrutura de rede computacional; ou através de *softwares* capazes de viabilizar esta comunicação, por exemplo, o *framework* JADE que será explanado na Seção 4.1.1 (Bellifemine et al. 2003). Na Figura 3.2 é apresentada a estrutura de um agente e suas características desejáveis dentro de um SMA.

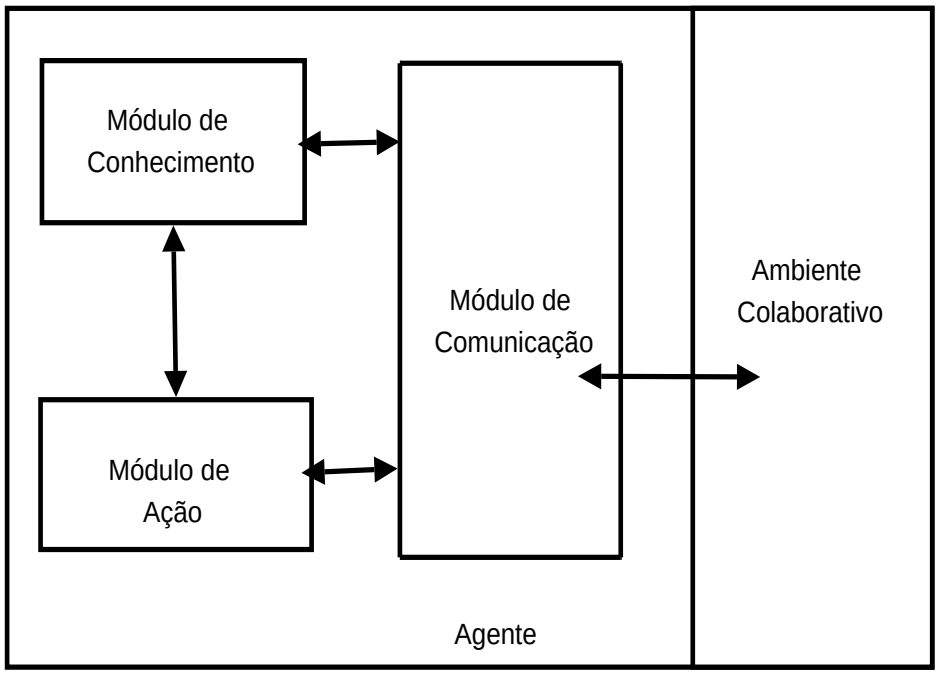


Figura 3.2: Estrutura de um agente na abordagem de um SMA.

Analisando a Figura 3.2, podemos notar o Módulo de Conhecimento, o qual contém a base de conhecimento utilizada pelo agente bem como o conhecimento alterado através de mecanismos de aprendizagem. As ações são executadas pelo Módulo de Ação que processa a execução das requisições através das percepções dos agentes. Essas requisições são enviadas periodicamente pelo sistema. Estas ações podem ser, por exemplo, ações *BLAST* (que executa a ferramenta NCBI *BLAST*) ou ações de detecção de genes (*Genemark, Glimmer*). O Módulo de Comunicação recebe as mensagens enviadas pelos outros agentes

e pelo sistema e as envia para o destino apropriado. As requisições para execução de ações são enviadas para o Módulo de Ação enquanto as mensagens de dados são enviadas para o Módulo de Conhecimento.

Os agentes trocam mensagens utilizando determinadas linguagens de comunicação, como a ACL (*Agent Communication Language*), padrão FIPA (FIPA 2006), ou a KQML (*Knowledge and Query Manipulation Language*). Os agentes que compõem um SMA podem interagir entre si ou com os usuários, de acordo com seus objetivos e motivações.

Segundo (Sycara 1998), as características de SMAs são tais que: (i) cada agente possui uma visão limitada; (ii) não há controle global do sistema; (iii) trabalham com dados descentralizados; e (iv) a computação é assíncrona (permitir comunicação entre entidades heterogêneas).

Para que os agentes sejam capazes de interagir, através de uma linguagem comum, é desejável que exista uma ontologia conceitual entre eles, a qual é definida e declarada em uma linguagem, como em ACL, por exemplo.

3.2 Ontologia

Segundo (Gruber 1993), ontologia[1] é uma especificação explícita de uma conceitualização. Neste contexto, conceitualização é a organização do conhecimento sobre o mundo em forma de entidades e a especificação é a representação da conceitualização em uma forma concreta.

Para (Almeida and Bax 2003) existem características e componentes básicos presentes em grande parte das ontologias, a saber:

- Relações: representam o tipo de interação entre os conceitos de um domínio;
- Axiomas: usados para modelar sentenças sempre verdadeiras;
- Instâncias: utilizadas para representar elementos específicos, ou seja, os próprios dados;
- Classes: organizadas em uma taxonomia[2].

No contexto deste trabalho a Ontologia é utilizada na comunicação entre os agentes, definindo assim, uma linguagem comum entre os agentes nos sistema *BioAgents*.

3.3 Regras de Produção

Inferência é um processo pelo qual se chega a uma proposição, afirmada com base em uma ou mais proposições aceitas, como ponto de partida do processo. Existem dois tipos

[1] O termo ontologia conforme a Filosofia, significa uma sistemática da existência. Segundo (Gruber 1993), para a IA, o que existe é o que pode ser representado.

[2] Refere-se a classificação das coisas e aos princípios subjacentes da classificação. Quase tudo - objetos animados, inanimados, lugares e eventos - pode ser classificado de acordo com algum esquema taxonômico.

de raciocínio inferencial neste escopo: o encadeamento progressivo (forward channing) e encadeamento regressivo (backward chaining).

No encadeamento progressivo, a parte esquerda da regra é comparada com os fatos contidos atualmente na base de fatos. As regras que satisfazem a esta descrição têm sua parte direita executada, o que, em geral, significa a introdução de novos fatos na base de fatos. Por exemplo, temos um conjunto de condições (X, Y e Z) que compõem a parte esquerda da regra, quando essas condições forem satisfeitas então a parte direita da regra será executada (W), sendo que W é uma ação que pode vir a ser um novo fato, ou seja pode vir a ser inserida na base de fatos.

X,Y,Z -> W

No encadeamento regressivo, ocorre um processo relativamento contrário ao encadeamento progressivo, da direita pra esquerda. O comportamento do sistema é controlado por uma lista de objetivos. Um objetivo pode ser satisfeito diretamente por um elemento da base de fatos, ou podem existir regras que permitam inferir algum dos objetivos correntes, isto é, que contenham uma descrição deste objetivo em suas partes direitas. As regras que satisfazem esta condição têm as instâncias correspondentes às suas partes esquerdas adicionadas à lista de objetivos correntes. Caso uma dessas regras tenha todas as suas condições satisfeitas diretamente pela base de fatos, o objetivo em sua parte direita é também adicionado à base de fatos. Um objetivo que não possa ser satisfeito diretamente pela base de fatos, nem inferido através de uma regra, este objetivo é inválido. Quando o objetivo inicial é satisfeito, ou não há mais objetivos, o processamento termina.

Neste trabalho foi utilizado o encadeamento progressivo das regras de produção utilizados pelos agentes inteligentes, tendo em vista a característica dos conjuntos de condições das regras (parte esquerda das regras), como condições necessárias a serem satisfeitas para uma recomendação de anotação (parte direita da regra).

3.4 Aprendizagem de Máquina

A aprendizagem de máquina trouxe um progresso significativo em diversas áreas, inclusive na Biologia (Mitchell 1997). Diz-se que um programa aprende a partir de um conjunto de experiências, em relação a um conjunto de tarefas, com uma medida de desempenho, se seu desempenho na execução de suas tarefas melhora com o aumento de sua experiência (Mitchell 1997).

Como citado em (Russell and Norvig 2002), os algoritmos de aprendizagem de máquina são classificados em três tipos distintos: supervisionados, não supervisionados e por reforço. A aprendizagem supervisionada consiste na descoberta de uma função a partir de exemplos de entradas e saídas desta função. Em ambientes totalmente observáveis, o agente sempre pode observar o resultado da sua ação, o que favorece o uso da aprendizagem supervisionada. Existem algumas técnicas de aprendizagem supervisionado, como por exemplo, Redes Neurais do tipo *Multilayer Perceptron*, Máquina de Vetores de Suporte (MVS), Algoritmos Genéticos e Árvores de Decisão.

A aprendizagem não supervisionada consiste no reconhecimento de padrões nas entradas sem nenhuma informação sobre a saída desejada. Um agente que usa somente aprendizagem não supervisionada não consegue aprender o que fazer, pois ele não possui informações sobre qual seria uma ação correta e um estado desejável a ser alcançado. Redes neurais do tipo mapas auto-organizáveis, algoritmo k-médias e os algoritmos de agrupamento hierárquico são exemplos de técnicas de aprendizagem não supervisionada.

A aprendizagem por reforço utilizada nesse trabalho será abordada de forma mais extensa e apresentada na Seção 3.5.

Segundo (Russell and Norvig 2002), alguns conceitos básicos são necessários para que se possa utilizar aprendizagem de máquina:

- Exemplo: é um par $(x, f(x))$, onde x é a entrada e $f(x)$ é a saída esperada para a entrada x. A inferência pura é a tarefa de achar uma função h que se aproxima de f dado, um conjunto de exemplos da função f.
- Atributo: é uma característica de um objeto, com isso, dizemos que um conjunto de atributos define um objeto.
- Classe: é a classificação dada a um objeto.
- Conjunto de Exemplos: Os conjuntos de exemplos são divididos em conjunto de treinamento e conjunto de teste. O conjunto de treinamento, como o próprio nome sugere, treina o algoritmo e o conjunto de teste valida o treinamento feito.
- *Overfitting*: ocorre quando o algoritmo se ajusta para um conjunto de dados muito específico, sendo assim ineficaz para um conjunto de dados mais geral.

3.5 Aprendizagem por Reforço

Segundo (Sutton and Barto 1998), a aprendizagem por reforço é uma abordagem de IA que permite a uma entidade aprender, a partir de sua interação com o ambiente, por meio do conhecimento sobre o estado do indivíduo, das ações efetuadas e das mudanças de estado que ocorreram devido a essas ações.

A aprendizagem por reforço pode ser utilizado quando existem agentes, que necessitam de uma maior autonomia, inserido em um ambiente, no qual não existem exemplos de ações, que sirvam de parâmetros úteis para determinação das próximas ações. Neste caso, o agente, após executar qualquer ação, poderá obter um retorno, o qual podemos chamar de recompensa. Essa recompensa pode ser boa ou ruim dependendo da ação tomada pelo agente. Tendo isso em vista, o papel da aprendizagem por reforço é utilizar as recompensas obtidas para aprender as ações ótimas, ou quase ótimas, dentro deste ambiente (Mitchell 1997).

Segundo (Dietterich et al. 2008), existe a necessidade de se aprender simultaneamente e em diferentes escalas de tempo. Isto, apesar de complexo, ajuda a tornar a aprendizagem por reforço mais eficiente, já que as recompensas podem ser instantâneas, o que pode aumentar a velocidade da aprendizagem. Existem algumas técnicas de aprendizagem por

reforço, como por exemplo, o agente baseado na utilidade, que utiliza a função de utilidade nos estados para selecionar uma ação que maximize a utilidade esperada. Também podemos citar o algoritmo *Q-learning*, que utiliza a função Q para calcular a utilidade de uma ação em um determinado estado.

3.5.1 Características da Aprendizagem por Reforço

Conforme (Sutton and Barto 1998), grande parte dos sistemas que utilizam a aprendizagem por reforço, possuem quatro características em comum: aprendizagem pela interação, retorno atrasado, orientação ao objetivo e usufruto *versus* exploração. Esses quatro conceitos serão explanados:

- Aprendizagem pela Interação - Um agente de aprendizagem por reforço atua no ambiente e aguarda pelo valor de reforço que o ambiente ou uma "entidade crítica" deve informá-lo como resposta perante a ação tomada. Ele deve assimilar, através da aprendizagem e durante várias interações, o valor de reforço obtido para tomar decisões posteriores.
- Retorno atrasado - O valor de reforço obtido, seja ele máximo até o momento, não significa que o agente tomou a melhor ação possível, porém até o momento essa ação foi a melhor.
- Orientação ao Objetivo - Na aprendizagem por reforço, os agentes tentam alcançar sempre um objetivo e esperam um retorno do ambiente ou de uma "entidade crítica".
- Usufruto *versus* Exploração - Os agentes durante as interação no ambiente, devem decidir se deve agir segundo a melhor informação já obtida até o momento ou agir tomando outra informação qualquer para tentar coletar novas informações sobre o ambiente.

3.5.2 O Problema da Aprendizagem por Reforço

O problema da aprendizagem por reforço é um problema de aprendizagem através de interações para se alcançar um objetivo (Sutton and Barto 1998). O agente é o elemento tomador de decisões e também é a entidade que aprende dentro do sistema. O agente percebe e interage com o ambiente, que é tudo além do próprio agente, através das percepções de estados(s_t) e ações(a_t), como mostra a Figura 3.3.

O agente e o ambiente interagem durante uma sequência discreta de tempo, $t = 1, 2, 3, ...$ Em cada momento t, o agente recebe uma representação do estado do ambiente, $s_t \in S$, onde S é o conjunto de possíveis estados e com isso seleciona uma ação, $a_t \in A(s_t)$, onde $A(s_t)$ é o conjunto de ações para o estado s_t.

Toda ação $a_t \in A(s)$ tem uma recompensa imediata $r_{t+1}(s_t, a_t) \in R$, onde R é o conjunto de possiveis recompensas, ou seja, no momento $t+1$ o agente recebe a recompensa r_{t+1} pela ação a_t que foi tomada no estado s_t e após essa interação o ambiente se encontrará no estado s_{t+1}.

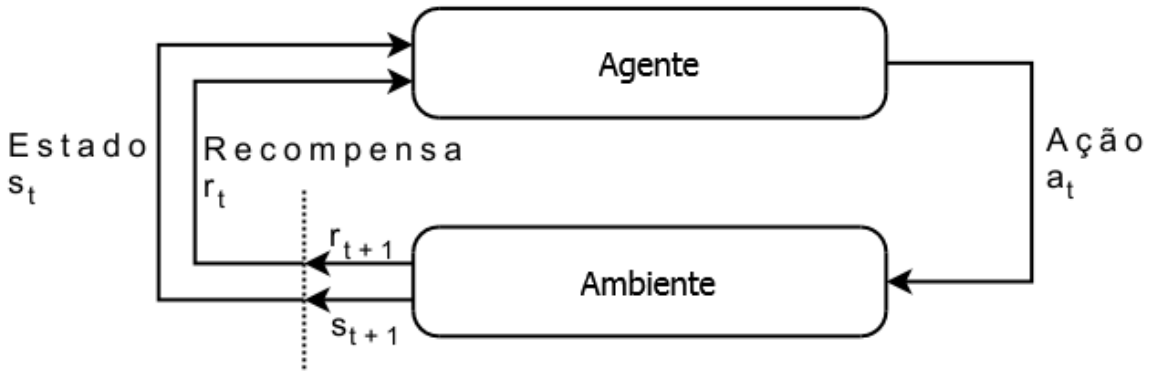


Figura 3.3: Modelo clássico de aprendizagem por reforço (Sutton and Barto 1998).

Assim temos que a recompensa total que um agente recebe em seu tempo de vida, partindo de um momento t é:

$$R(t) = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{3.1}$$

onde $\gamma \in (0, 1]$ é chamado de fator de desconto ou taxa de amortização. Caso k possua um valor final, ou seja, k seja limitado superiormente ao invés de tender ao infinito como mostrado na Equação 3.1, a interação entre o agente e o ambiente chega a um estado final e termina. Porém normalmente esse limite superior não existe.

Se $\gamma < 1$ dizemos que o desconto é utilizado. Caso $\gamma \longrightarrow 0$, significa que o agente maximiza somente os reforços imediatos e caso $\gamma \longrightarrow 1$ o agente da maior importância ao estado final, desde que a interação chega a um fim.

Resumindo, um algoritmo de aprendizagem por reforço procura maximizar $R(t)$ para todos os $s \in N_t$, onde N_t é o conjunto de estados no momento t.

As funções de retorno ou de reforço, são responsáveis por calcular a recompensa $r_{t+1}(s_t, a_t)$ de cada ação a_t no momento s_t. Existem pelo menos três classes para essas funções e devem ser escolhidas de acordo com o problema abordado:

- Reforço só no estado final - O agente nesse caso só recebe uma recompensa ou uma penalidade quando alcança o estado final. Assim o agente aprenderá que os estados correspondentes a uma recompensa são bons e os que levam a uma penalidade devem ser evitados.
- Tempo mínimo ao objetivo - Essa funções sempre retornam uma penalidade para qualquer estado exceto o estado final, que tem recompensa 0. Assim o agente aprende escolher ações que minimizam o tempo para alcançar o estado final.
- Minimizar reforços - Nem sempre, o agente precisa ou deve tentar maximizar a função de reforço, ao invés, ele precisa minimizá-la. Isso é necessário quando o

reforço é uma função de recursos limitados e o agente precisa aprender a conservá-los.

Na aprendizagem por reforço o agente normalmente deve aprender uma política para alcançar seu objetivo, essa política é denotada por π e não é nada além de um mapeamento de estados e ações em um valor $\pi(s, a)$. Um agente de aprendizagem pode mudar sua política e assim a probabilidade da escolha de ações muda, o que acarreta numa variação do funcionamento do sistema. Assim podemos dizer que um problema de aprendizagem por reforço pode ser expresso em termos da convergência até uma política $\pi^*(s, a)$ que conduz a uma solução ótima.

Outras duas funções importantes usadas na maioria dos algoritmos de aprendizagem por reforço são as funções valor-estado e valor-ação, denotadas por $V(s)$ e $Q(s, a)$ respectivamente. Essas funções definem se estar em um estado ou estar em um estado e executar uma ação é bom para o agente. Ser bom para o agente quer dizer que o agente pode esperar bons retornos.

Como a escolha das ações é baseada na política π, essas funções também são dependentes dessa política. Com isso pode-se definir em um processo de decisão markoviano as funções valor-estado $V^\pi(s)$ e valor-ação $Q^\pi(s, a)$ com sendo:

$$V^\pi(s) = E_\pi\{R_t|s_t = s\} = E_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s\}$$
$$Q^\pi(s, a) = E_\pi\{R_t|s_t = s, a_t = a\} = E_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a\}$$

Onde E_π denota o valor esperado se o agente seguir a política π.

3.5.3 Propriedade de Markov

Quando a probabilidade de transição no momento t de uma estado s_t para um estado s' após a execução de uma ação a_t, depender pelo menos do estado s_t e da ação a_t, mas nunca depender de mais que o histórico completo de estados e ações até o momento, ou seja, do conjuntos $s_0, s_1, s_2, ..., s_t$ e $a_0, a_1, a_2, ..., a_t$, diz-se que satisfaz a propriedade de Markov (Sutton and Barto 1998).

Com isso podemos definir a equação que expressa a probabilidade de alcançar o estado s_{t+1} e a recompensa r_{t+1} no momento $t + 1$ após uma ação a_t executada no momento t, como sendo:

$$Pr\{s_{t+1} = s', r_{t+1} = r'|s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, ..., s_1, a_1, r_1, s_0, a_0\}$$

Caso essa probabilidade somente dependa do estado e da ação anterior, a equação é a seguinte:

$$Pr\{s_{t+1} = s', r_{t+1} = r'|s_t, a_t\}$$

onde Pr é o operador de probabilidade.

3.5.4 Processo de Decisão de Markov

Um problema de aprendizagem por reforço que satisfaz a propriedade de Markov é chamado de processo de decisão de Markov. Se o conjunto de estados e ações forem finitos, então este é um processo de decisão de Markov finito.

Em um processo de decisão de Markov finito, podemos definir a probabilidade de um estado s passar a um estado s' dada uma ação a, é:

$$P^a_{s,s'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$$

De forma análoga podemos definir a recompensa esperada, dados os estados s e s' e a ação a, podemos definir a expressão como:

$$R^a_{s,s'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\}$$

Os valores de $P^a_{s,s'}$ e $R^a_{s,s'}$ determinam os aspectos mais importantes da dinâmica de uma problema de decisão de Markov finito. Ele pode ser caracterizado como:

- um ambiente que evolui probabilisticamente baseado em um conjunto finito e discreto de estados;
- para cada estado do ambiente, existe um conjunto finito de ações possíveis;
- para cada passo em que o sistema de aprendizagem executar uma ação, é verificado um custo positivo ou negativo para o ambiente em relação à ação;
- estados são observados, ações executadas e reforços, relacionados.

Assim, para quase todos os problemas de aprendizagem por reforço é suposto que tenha a forma de um processo de decisão de Markov, desde que seja satisfeita a propriedade de Markov no ambiente. Nem todos os algoritmos de aprendizagem por reforço necessitam de uma modelagem como processo decisório de Markov inteiro do ambiente, mas é necessário ter pelo menos a visão do ambiente como um conjunto finito de estados e ações.

3.5.5 Métodos de Solução

Segundo (Sutton and Barto 1998), para solucionar problemas de aprendizagem por reforço existem três classes de métodos fundamentais: programação dinâmica, Monte Carlo e diferença temporal.

A programação dinâmica refere-se ao conjunto de algoritmos que podem ser usados para computar políticas ótimas, dado que se tem um modelo perfeito do ambiente como um processo decisório de Markov. Esses algoritmos, portanto, são considerados limitados, já que, necessitam do modelo perfeito do ambiente.

Os algoritmos clássicos de programação dinâmica são usados de forma limitada, já que uma modelagem perfeita do ambiente exige um grande esforço computacional. Apesar

disso eles fornecem um bom fundamento para o conhecimento dos outros métodos usados na solução de problemas de aprendizagem por reforço e também fornecem um padrão de comparação.

A idéia na qual a programação dinâmica se baseia é usar valores de funções para organizar e estruturar as buscas pelas melhores políticas. E com um ambiente bem modelado como um processo de decisão de Markov, toda dinâmica do sistema é baseada nas funções $P^a_{s,s'}$ e $R^a_{s,s'}$, que foram descritas na Seção 3.5.4.

O método de Monte Carlo, por sua vez, não necessita de uma modelagem do ambiente e se baseia simplesmente no cálculo da média dos retornos obtidos em sequência. Para garantir que existam valores de retorno bem definidos, o método é definido somente para tarefas temporais, ou seja, a experiência é dividida em episódios e todos os episódios eventualmente terminam independentemente das ações tomadas. Com isso, somente após o termino de um episódio que os valores são estimados e as políticas são alteradas.

Apesar das diferenças do método Monte Carlo e da programação dinâmica, as idéias mais importantes do método da programação dinâmica são utilizados no Monte Carlo. Os dois métodos computam a mesma função de valor e também interagem da mesma forma para alcançar a resultados ótimos.

Já o método de diferença temporal utiliza idéias dos dois métodos anteriores, assim como o Monte Carlo, esse método pode aprender diretamente da experiência pura sem uma modelagem do ambiente, onde aplica técnicas da programação dinâmica, quando faz estimativas da função de valor a partir de outras estimativas já aprendidas.

O método que será proposto nesse trabalho no Capítulo 5 pode ser classificado como um método de diferença temporal e faz uso de suas vantagens sobre os métodos Monte Carlo e programação dinâmica. Como vantagens podemos citar: a não necessidade de um modelo do ambiente bem definido e a experiência não precisa ser dividida em episódios, o que se adequa ao contexto de um sistema que recomenda anotações em projetos de sequenciamento de genomas.

3.5.6 Algoritmos de Aprendizagem por Reforço

Nesta seção será apresentado um algoritmo clássico de aprendizagem por reforço, o *1-step Q-Learning*.

1-step Q-Learning

Segundo (Whitehead 1991), o algoritmo *1-step Q-learning* é descrito como segue:

```
One-Step-QLearning(){
1.  s = Estado atual
2.  se (s \in G) então pare
3.  escolha a \in A(s)
4.  execute ação a
  /* Como consequência, o agente recebe
```

```
     uma recompensa r(s,a) e move para o
     estado succ(s,a), incrementando o
     número de passos t = t + 1
5.  Q(s, a) := r(s, a) + yU (succ(s, a))
6.  Volta para 1.
}
```

Deve-se notar que $U(s) = max_{a \in A(s)} Q(s, a)$ em todos os instantes.

Esse algoritmo armazena informação sobre seus "acertos" em ações dentro dos estados. Isso é feito mantendo o valor de $Q(s, a)$ dentro do estado s para cada ação a. $Q(s, a)$ se aproxima ao total ótimo de recompensas recebidas por um agente que inicia em s, executa a, e age de forma ótima.

A linha 3 implementa a seleção de uma ação, porém aqui ela não é definida, já que podem ser utilizadas quaisquer heurísticas para essa seleção. As linhas 4 e 5 implementam respectivamente, a execução da ação e a atualização do valor $Q(s, a)$.

Para analisar o *1-step Q-learning*, vamos considerar um caso geral e determinar que o conjunto G é o conjunto de todos os s que são classificados como estados objetivos, ou seja, estados que o agente quer alcançar. E que todos os valores Q de cada estado são inicializados com 0.

Definimos que:

1. O valor Q é consistente se e somente se para todo $s \in G$ e $a \in A(s)$, $Q(s, a) = 0$ e para todo $s \in S - G$ e $a \in A(s)$, $-1 + U(succ(s, a)) \leq Q(s, a) <= 0$
2. Um Q-Learning sem desconto com representação de penalidades por ação é chamado de admissível se somente se os valores Q são consistentes e sua seleção de ação é dado por: $a = max_{a' \in A(s)} Q(s, a')$.

Se um algoritmo *Q-Learning* é adminissível, então seus valores Q são consistentes e são decrescentes.

Lema 1: Para todo passos t de uma algoritmo sem desconto com representação de penalidades por ação e admissível temos:

$$U^t(s^t) + \sum_{s \in S} \sum_{a \in A(s)} Q^0(s, a) - t \leq \sum_{s \in S} \sum_{a \in A(s)} Q^t(s, a) + U^0(s^0) - loop^t$$

e

$$loop^t \leq \sum_{s \in S} \sum_{a \in A(s)} (Q^0(s, a) - Q^t(s, a))$$

onde $loop^t$ é o número de ação antes do passo t que não alteraram um estado.

O Lema 1 mostra que todos os passos dependem somente do estado inicial e do valor Q atual, e que todos os valores Q diminuem ao longo da execução.

Lema 2: Um algoritmo *Q-Learning* sem desconto com representação de penalidades por ação e admissível termina em até:

$$\sum_{s \in S-G} \sum_{a \in A(s)} (Q^0(s,a) + gd(succ(s,a)) + 1) - U^0(s^0)$$

Onde $-gd(s)$ é o total de recompensas ótimo sem descontos.

Teorema 1: Um algoritmo *Q-Learning* sem desconto com representação de penalidades por ação e admissível alcança um estado objetivo e termina em no máximo $O(en)$ passos.

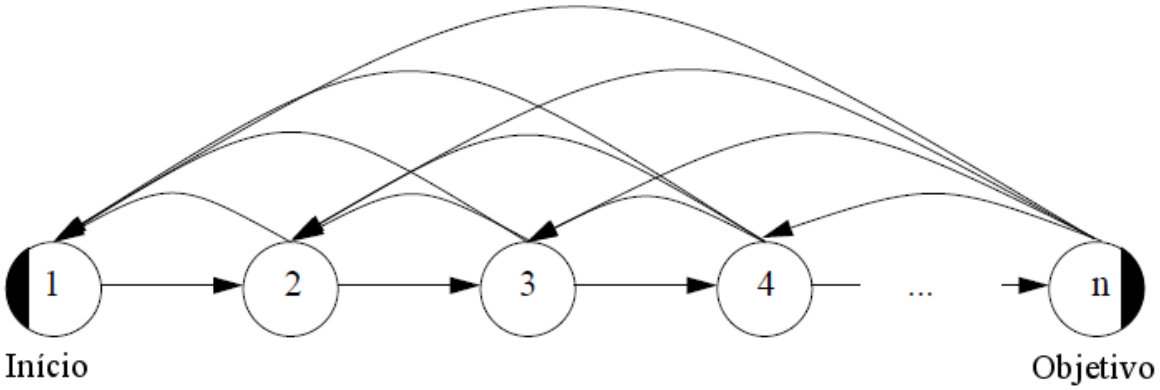


Figura 3.4: Exemplo de pior caso para o *Q-Learning*.

Se o conjunto de estados não possuir duplicatas, então $e \leq n^2$, então a complexidade de pior caso se torna $O(n^3)$. A Figura 3.4 mostra um conjunto de estados onde no mínimo $\frac{1}{6}n^3 - \frac{1}{6}n$ passos podem ser necessários para chegar ao estado objetivo.

Como o conjunto de estados e seus conjuntos de ações para o uso do *Q-Learning* dentro do *BioAgents* não foi definido, não temos como calcular o valor exato de e. Porém pode-se afirmar que o *Q-Learning* é polinomial em n em seu pior caso, já que é possível modelar um conjunto de estados e ações que não faz com que e se torne exponencial e ainda mais, se o conjunto de estados não possuir duplicatas podemos dizer que $e \leq n^2$.

3.6 Trabalhos Correlatos

Vários trabalhos na área de Bioinformática utilizam técnicas de IA, através do uso de abordagens distintas como a de SMA, Mineração de Dados, Aprendizagem de Máquina, Raciocínio Automático e Tratamento Semântico. Essas abordagens têm sido aplicadas em diferentes processos envolvidos no *pipeline* de execução que inclui desde a comparação e análise de genomas até a inferência das funções dos genes dos organismos. Apesar de um grande volume de trabalhos nesta área, não foi encontrado nenhuma trabalho que utiliza SMA e aprendizagem de máquina para anotar funções genômicas como proposto neste trabalho.

Passaremos a apresentar alguns trabalhos que consideramos importantes por se relacionarem a Bioinformática com utilização de técnicas de IA, através de diversas abordagens, no entanto, não pretendemos ser exaustivos.

O sistema BioMAS utiliza a abordagemde SMA para anotação automática do vírus da herpes (Decker et al. 2001). O foco do trabalho está na extração da informação contida nos bancos de dados públicos e no processo de anotação automática.

O Eletronic Annotation-EAnnot é uma ferramenta originalmente desenvolvida para a anotação manual do genoma humano (Ding et al. 2004). O software combina ferramentas para extrair e analisar grandes volumes de dados em bancos públicos, gerando anotações automáticas e predições de genes de forma rápida. EAnnot usa informações contidas em RNA mensageiros, Expressed Sequence Tags-ESTs e alinhamentos de proteínas, além de identificar pseudogenes, entre outras características.

O software Ambiente para Anotação Automática e Comparação de Genomas-A3C (Santos and Bazzan 2004) é baseado em uma aquitetura de SMA e tem como propósito a integração de tarefas relacionadas a anotação denominada pelos autores como nível 1 e a comparação de genomas considerada como nível 2. O nível 1 é composto por ferramentas para a anotação automática de proteínas, enquanto o nível 2 é composto por algoritmos para comparação de genomas que visam a extração de informações úteis aos resultados do nível 1. O objetivo do A3C é descobrir a relação entre diversos organismos, obtendo então informações específicas sobre um dado genoma através do conhecimento sobre outros genomas que já se encontram sequenciados.

A ferramenta denominada *Agent-based environmenT for aUtomatiC annotation of Genomes-ATUCG* é baseada em uma aquitetura de agentes, tendo como objetivo reduzir o trabalho manual dos biólogos através da re-anotação (Bazzan et al. 2003). No processo de re-anotação as informações adquiridas das sequências originalmente anotadas são revisadas e comparadas com novos modelos e dados para obter características e informações sobre as sequências e refazer a anotação manual, caso seja necessário.

Foi apresentado por (Tetko et al. 2008) um método que utiliza aprendizagem de máquina aplicado na anotação, que utiliza o melhor valor de *score* de alinhamentos para determinar a função da sequência. Neste trabalho, foi utilizada a técnica de *5-fold cross-validation* em cada algoritmo de aprendizagem para predizer o *score* de cada par de proteínas. Após a validação em diferentes bases de dados, é escolhido o melhor *score* calculado por esses algoritmos.

Com o intuito de melhorar a anotação de *Caenorhabditis elegans* utilizando técnicas de aprendizagem de máquina, foi proposto por (Rätsch et al. 2007) um sistema de aprendizagem, chamado de *mSplicer*, que foi treinado para reconhecer éxons e íntrons dentro de um mRNA. Foi utilizada a técnica de Máquina de Vetores de Suporte e *Label Sequence Learning*. Este trabalho conseguiu identificar corretamente todos os éxons e íntrons do genoma do *C. elegans*.

Capítulo 4

BioAgents

Nesse capítulo será apresentada o sistema *BioAgents*. Sua concepção inicial será descrita na Seção 4.1. Na Seção 4.2 serão apresentadas todas as modificações que foram realizadas para melhoria do sistema e para o suporte à nova arquitetura que será proposta para o uso do aprendizado por reforço.

O sistema *BioAgents* possui código fonte aberto utilizando licença GPL e esta hospedado no SourceForge (`http://bioagents.sourceforge.net/`), sendo que seu código fonte esta disponibilizado no endereço `http://sourceforge.net/projects/bioagents/`.

4.1 Concepção Inicial do *BioAgents*

O sistema *BioAgents*, proposto em (Lima 2007, Lima et al. 2007) e depois de aprimorado em (Ralha et al. 2008), é um SMA, no qual diferentes agentes cooperam entre si e interagem com o ambiente, com o objetivo de auxiliar os biólogos no processo de anotação manual. Sua arquitetura inicial foi proposta como mostra a Figura 4.1.

Nessa arquitetura, a camada de apresentação é responsável por receber as requisições submetidas ao sistema e retornar o resultado do processamento ao usuário. Essa requisição consiste na submissão de sequências a serem analisadas.

A camada colaborativa é responsável pela consolidação dos resultados provenientes das análises feitas sobre os bancos de dados da Camada Física e por retorná-los à Camada de Apresentação. Essa camada é composta pelos seguintes agentes:

- Agente Resolvedor de Conflitos - Esse agente tem por objetivo submeter as requisições enviadas pela Camada de Apresentação aos agentes gerentes especializados. Ao receber os resultados dos agentes gerentes, esse agente também decide qual a sugestão mais apropriada para ser enviada à Camada de Apresentação.
- Agentes Gerentes - Os agentes gerentes recebem do agente resolvedor de conflitos as requisições de acordo com sua especialidade. Um particular agente deste tipo verifica quais são os bancos de dados e saídas dos programas que devem ter sido previamente executados. Esses agentes alocam os Agentes Analisadores para fazer a análise

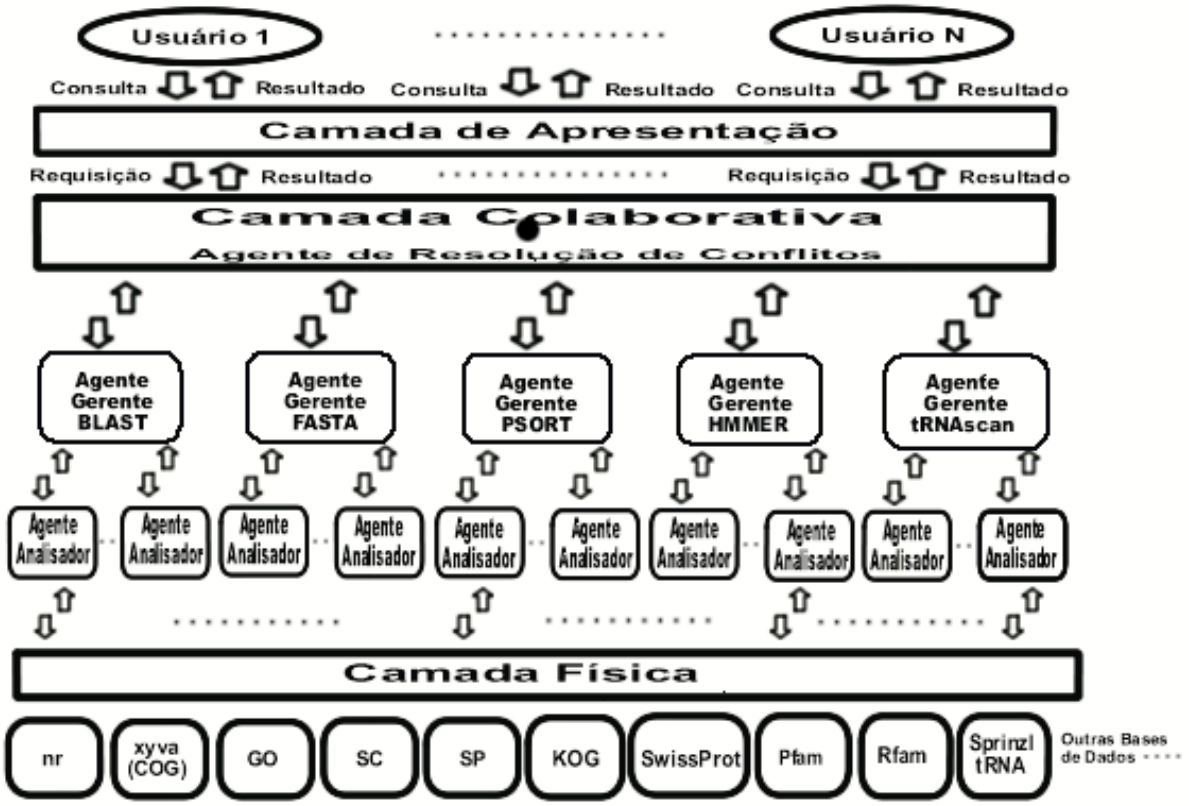


Figura 4.1: Arquitetura proposta para o *BioAgents* (Lima et al. 2007).

individual dessas saídas juntamente com os bancos de dados. Os Agentes Gerentes aguardam as sugestões de todos agentes analisadores, consolidando-as através do uso de regras de produção previamente definidas.

- Agentes Analisadores - Cada agente analisador avalia os resultados apresentados em um arquivo de saída gerado pela ferramenta específica. Ele é criado por solicitação de um agente gerente e esses agentes utilizam analisadores para extrair informações dos arquivos de saída, gerando uma estrutura contendo dados específicos das ferramentas. O resultado desse processamento é uma sugestão, que é retornada a um agente gerente.

A camada física é responsável pelos bancos de dados utilizados pelo *BioAgents*.

Na implementação inicial (Lima et al. 2007), foram utilizados duas ferramentas para fazer o alinhamento das sequências, o BLAST (Altschul et al. 1997) e o FASTA (`http://fasta.bioch.virginia.edu/`). Os Agentes Analisadores, por sua vez, não executavam as ferramentas para obter a saída. mas necessitavam da saída como parte de sua requisição.

Para a comunicação entre os agentes, foi utilizado um único protocolo de comunicação entre todos os agentes, o *Request Interaction Protocol* definido pela FIPA (FIPA 2006) como mostra a Figura 4.2. Utilizando esse protocolo, os agentes faziam suas requisições, normalmente aceitas, exceto quando um agente recebia uma requisição inválida, ou não a entendia.

4.1.1 Ferramentas Utilizadas

Passaremos a apresentar as ferramentas utilizadas para o desenvolvimento do *BioAgents*:

JADE

O JADE (*Java Agent DEvelopment Framework*), que segue os padrões da organização FIPA (FIPA 2006), tem sido amplamente utilizado em vários projetos como: projetos que trabalham com ontologias; projetos de telecomunicações e projetos voltados para computação em grade.

Segundo (Bellifemine et al. 2003; 2005), o *framework* JADE, desenvolvido pela TILAB (Telecom Italia LAB), é um *middleware* utilizado para desenvolvimento e execução de aplicações *peer-to-peer* baseadas em agentes inteligentes, que pode operar tanto em ambientes computacionais *wired* (com fio) como *wireless* (sem fio). JADE é um *framework* completamente desenvolvido em Java e está fundamentado de acordo com os princípios seguintes:

- Interoperabilidade - Conformidade com a especificação FIPA (FIPA 2006). Isto significa que os agentes construídos com o JADE podem interoperar com outros agentes desde que estes obedeçam a especificação FIPA.

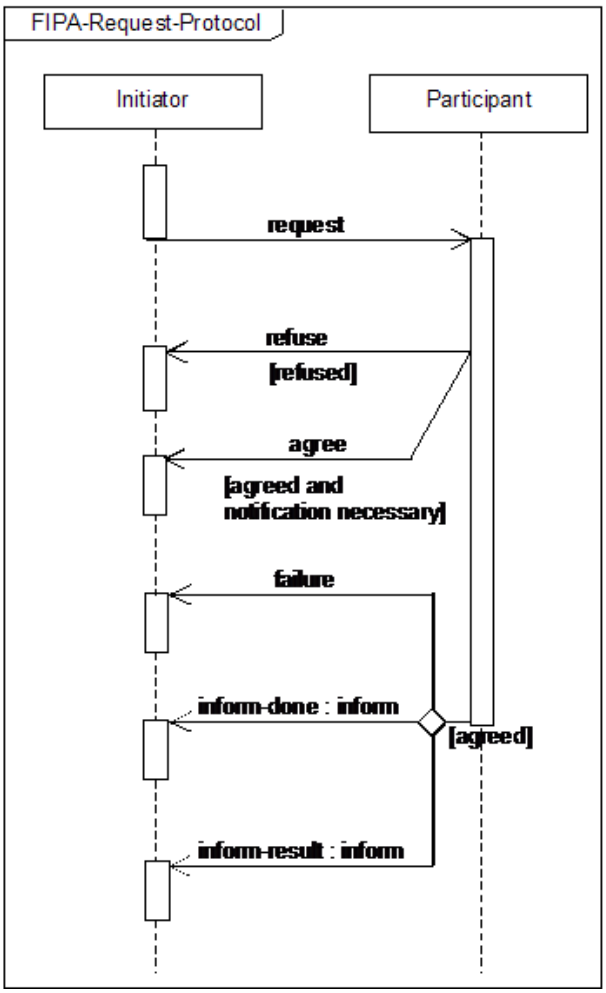


Figura 4.2: FIPA *Request Interaction Protocol* (FIPA 2006).

- Uniformidade e Portabilidade - JADE provê um conjunto homogêneo de APIs[1] que é independente da rede e da versão Java utilizada. Em relação a linguagem Java, as APIs JADE fornecem recursos para desenvolvedores em ambientes J2EE[2], J2SE[3] e J2ME[4].
- Facilidade de Uso - Apesar do *framework* possuir uma determinada complexidade, esta fica transparente ao usuário devido a simplicidade provida pelo seu conjunto de APIs.
- Filosofia *"Pay-as-you-go"* - Os desenvolvedores não precisam usar todos os recursos fornecidos pelo *middleware*. Os recursos que não estão sendo utilizados não trazem *overhead* aos recursos computacionais.

O JADE possui uma vasta biblioteca, através de um conjunto de classes Java, para o desenvolvimento de agentes inteligentes, bem como um ambiente de execução que provê serviços básicos, os quais devem ser ativados no dispositivo[5] antes dos agentes serem executados. Cada instância, em tempo de execução do JADE, é chamada de *container* (desde que esta contenha agentes). O conjunto de *containers* é chamado de plataforma e provê uma camada homogênea, a qual não é transparente aos agentes, pois o *container* determina a localização de um agente na plataforma (Bellifemine et al. 2003).

O *framework* JADE é extremamente versátil, com bom desempenho tanto em conjunto com aplicações construídas em arquiteturas complexas, tais como .NET or J2EE, ou em conjunto com aplicações executadas em ambientes com recursos limitados, tais como dispositivos móveis, como telefones celulares. Na Figura 4.3 temos a arquitetura do *framework* JADE.

O *framework* pode ser observado a partir de duas visões: funcional e da aplicação. Do ponto de vista funcional, o JADE provê os serviços básicos necessários para aplicações *peer-to-peer* em ambientes *wired* ou *wireless*. O JADE permite que cada agente, dentro do paradigma *peer-to-peer*, descubra dinamicamente os outros agentes, que fazem parte do sitema multiagente, e estabeleça comunicação entre eles. Do ponto de vista da aplicação, cada agente é identificado por um nome (identificador único) e provê um conjunto de serviços. O agente pode registrar e modificar seus serviços e/ou buscar por agentes, dentro do SMA, que forneçam um determinado serviço, pode controlar seu ciclo de vida e, em particular, estabelecer comunicação com todos os outros *peers* (topologia *peer-to-peer*) (Bellifemine et al. 2003).

A comunicação estabelecida entre os agentes, por troca de mensagens, pode ser feita de maneira assíncrona. Segundo (Bellifemine et al. 2005), este é um modelo de comunicação

[1] API, *Application Program Interfaces*, é um conjunto de rotinas, classes, protocolos e ferramentas para construção de *softwares*.

[2] J2EE (***J**ava 2 Platform **E**ntreprise **E**dition*) é um ambiente de desenvolvimento para aplicações em máquinas de grande porte (servidores), por exemplo.

[3] J2SE (***J**ava 2 Platform **S**tandard **E**dition*) é um ambiente de desenvolvimento para aplicações em *desktop*, por exemplo.

[4] J2ME (***J**ava 2 Platform **M**icro **E**dition*) é um ambiente de desenvolvimento para aplicações em redes *wireless* e dispositivos móveis, por exemplo.

[5] Um dispositivo móvel, caso J2ME, ou uma aplicação executando em um PC (no caso do J2SE ou do J2EE).

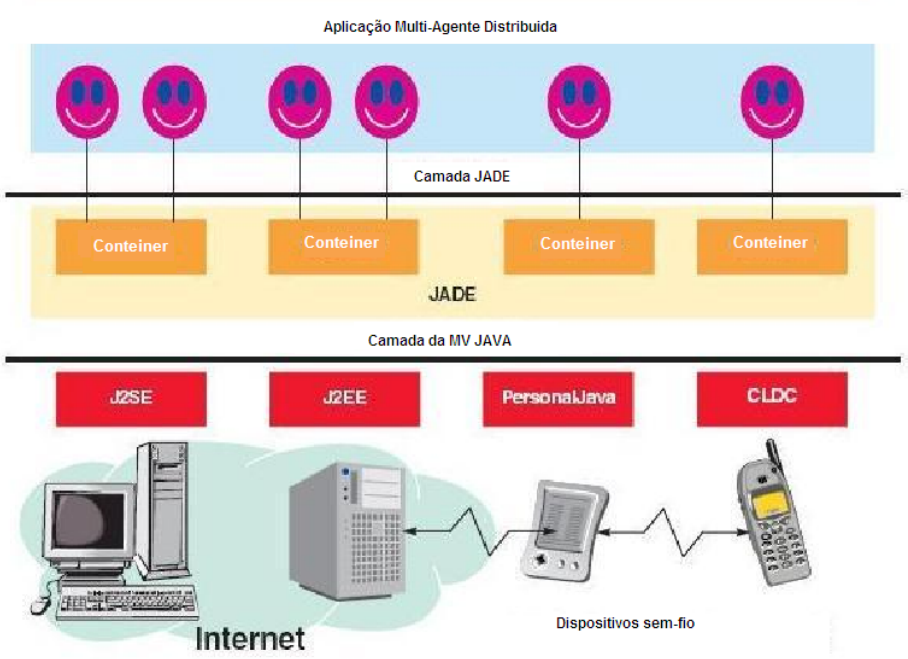


Figura 4.3: Arquitetura do *framework* JADE (Bellifemine et al. 2003).

usado para comunicações entre entidades heterogêneas que não sabem nada sobre as demais entidades com as quais estabelecem comunicação. A estrutura da mensagem está de acordo com a linguagem ACL, definida pela FIPA (FIPA 2006).

BioJava

O Biojava (`http://biojava.org/`) é um projeto de código aberto que visa disponibilizar ferramentas, desenvolvidas em Java, para processamento de dados biológicos. Estas ferramentas incluem objetos para manipulação de sequências, analisadores de arquivos, programação dinâmica, entre outros. O BioJava foi utilizando na trasformação das saídas tipo BLAST em uma estrutura de dados adequada ao *BioAgents*.

JESS

Em relação ao processo de inferência realizado por regras de produção foi utilizado o JESS (*Java Expert System Shell*) (`http://www.jessrules.com/`), O JESS é um motor de inferência para construção de bases de conhecimento e inferência dirigida por padrão. Foi desenvolvido por Friedman-Hill no *Sandia National Laboratories*. Como o próprio nome sugere, o JESS é feito para interagir com Java, o que permite a criação de *software* Java com a capacidade de reagir usando conhecimento vindo das regras declarativas implementadas no JESS. O JESS é considerado leve e rápido sendo projetado para dar acesso a toda API Java. Desta forma, é possível criar objetos Java, invocar métodos dessa linguagem e implementar interfaces Java sem compilar nenhum código Java.

Protégé

Com relação ao apoio ferramental para definição de ontologias, a ferramenta Protégé é uma das mais utilizadas em âmbito mundial. Protégé foi desenvolvida como uma ferramenta de código livre, permitindo assim que possa ser realizada customizações de modo a compatiblizá-la com outras ferramentas para criação de ontologias, bem como permite a geração automática de código. Isso é possível porque podemos definir em Protégé uma ontologia em um formato abstrato, que pode então ser convertido para linguagens específicas, através de *plug-ins*. O Protégé gera código de ontologia em JADE automaticamente, usando como formato abstrato na ferramenta o modelo de ontologias de JADE (`http://protege.stanford.edu/`).

4.2 Modificações no Sistema *BioAgents*

Todas as mudanças feita no sistema *BioAgents* foram feitas baseada na arquitetura original, ou seja, todas as camadas continuam com a mesma estrutura. Porém a camada de apresentação será representada como uma núvem, conforme apresentado na Figura 4.4, pois essa camada pode ser generalizada para que qualquer tipo de sistema, que possa enviar uma mensagem para o agente resolvedor de conflitos, faça uma requisição ao sistema e obtenha uma resposta.

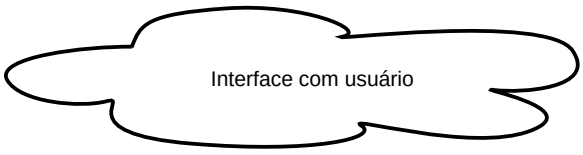


Figura 4.4: Representação da camada de apresentação.

4.2.1 Drools

A primeira mudança feita no *BioAgents*, foi a alteração do motor de inferência. O JESS apesar de ser um bom motor de inferência não apresenta uma linguagem, ferramentas de auxílio para o desenvolvimento muito amigáveis, e a evolução deste motor de inferência está aparentemente desativada, já que a última atualização foi feita em 11 de novembro de 2008. Pelos motivos expostos, o JESS foi substituído pelo Drools (Browne 2009). O Drools é um projeto da *JBoss.org* e também é um motor de inferência para construção de bases de conhecimento e realiza inferência como o JESS. O Drools foi construido em Java e disponibiliza uma API para integração com qualquer aplicação Java. O Drools é dividido em quatro subprojetos:

- Drools Guvnor - É um repositório centralizado de bases de conhecimento drools que possui diversas ferramentas web para gerenciamento.

- Drools Expert - Esse projeto é o motor de inferência propriamente dito.
- Drools Flow - Provê a capacidade ao motor de inferência de organizar os processos e as regras em fluxos.
- Drools Fusion - Provê a capacidade ao motor de inferência de processar eventos temporais.

Para a implementação do *BioAgents* foi utilizado somente os projetos *Drools Expert* e *Drools Flow*. Na Figura 4.5 é apresentado um exemplo de fluxo utilizado no agente resolvedor de conflitos.

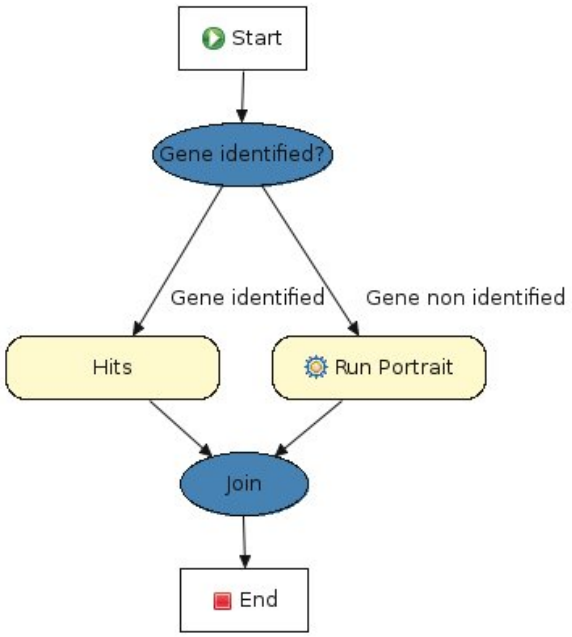


Figura 4.5: Fluxo Drools utilizado no Agente Resolvedor de Conflitos (FIPA 2006).

O fluxo apresentado na Figura 4.5 descreve parte do funcionamento interno do Agente Resolvedor de Conflitos, onde primeiramente é verificado se existem sugestões que identificaram o gene. Caso existam tais sugestões, o agente executa as regras do grupo "'Hits"', caso contrário, o agente solicita a outro agente a execução do *Portrait*, que será discutida posteriormente na Seção 4.2.3.

No código abaixo apresentamos um exemplo de regra utilizado no sistema *BioAgents*.

```
rule "Get Best Hit"
    ruleflow-group "Hits"
    when
        s1 : Sugestion()
        s2 : Sugestion()
        eval(s1.getHit().getEvalue() >= s2.getHit().getEvalue())
```

```
        eval(s1.getHit().getScore() <= s1.getHit().getScore())
        eval(s1 != s2)
    then
        RuleIdentifier ruleIdentifier = RuleIdentifier();
        ruleIdentifier.setPackageName(kcontext.getRule().getPackageName());
        ruleIdentifier.setName(kcontext.getRule().getName());
        if(!s2.getRules().contains(ruleIdentifier)){
            s2.getRules().add(ruleIdentifier);
        }
        retract(s1);
end
```

Nota-se que o Drools possibilita que a parte *"then"* da regra seja escrita em Java, o que facilita o desenvolvimento de regras no Drools. Essa regra verifica se existem duas sugestões distintas que possuem *e-values* distintos e remove a de maior *e-value*.

4.2.2 Problema da Troca de Mensagens

Foi identificado na implementação inicial do *BioAgents* um problema na troca de mensagens entre os agentes. O fluxo de mensagens no qual foi identificado o problema é apresentado na Figura 4.6.

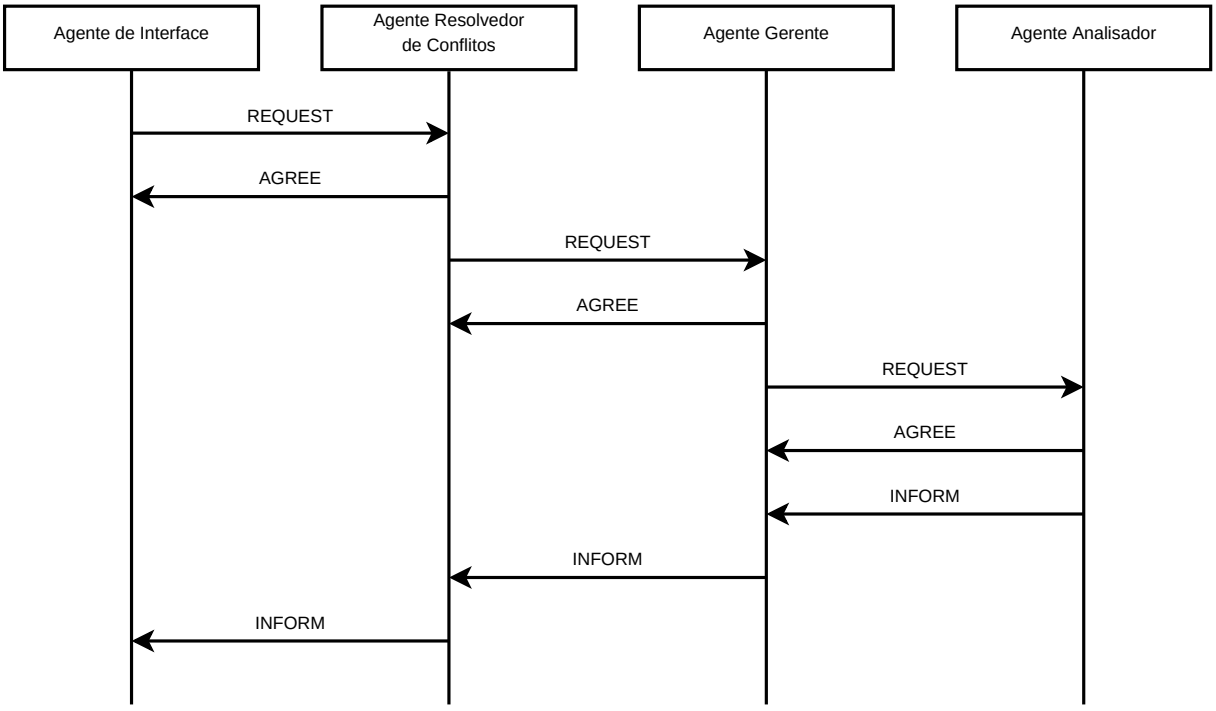


Figura 4.6: Fluxo de Mensagens na implementação inicial do *BioAgents*.

Entende-se por agente de interface qualquer programa que tenha a capaciadade de enviar uma requisição ao Agente Resolvedor de Conflitos e espere uma resposta. Qualquer

programa, refere-se à capacidade do *JADE* de receber informações através de mensagens de qualquer programa, não apenas de agentes, e também de responder essas mensagens.

Todas as mensagens recebidas e enviadas por um agente eram tratadas por um único comportamento (*Behaviour* segundo a nomenclatura do *JADE*). Esse comportamento era implementado utilizando uma máquina de estados finita (*FSMBehaviour*). Com isso, o comportamento não era flexível o suficiente para tratar mensagens fora deste fluxo, ou seja, quando um agente recebia uma mensagem que não estava esperando, o fluxo era interrompido e o agente entrava em um estado de permanente espera.

Primeiro passo para resolver este problema, foi trocar o protocolo de comunicação entre alguns agentes. Entre o Agente Resolvedor de Conflitos e os Agentes Gerentes foi utilizado um protocolo de interação definido pela FIPA como *Contract Net Interaction Protocol* (FIPA 2006) descrito na Figura 4.7. Esse protocolo permite a competição e concorrência entre vários agentes gerentes do mesmo tipo para a execução de uma tarefa.

Entre os Agentes Gerentes e os Agentes Analisadores, foi utilizado o protocolo *Achieve Simple Rational Effect*. Esse protocolo é uma simplificação do protocolo *FIPA Request Interaction Protocol*, que não necessita que um agente concorde (mensagem *AGREE*) ou descorde (mensagem *REFUSE*) da requisição. Esse novo fluxo de mensagens esta apresentado na Figura 4.8.

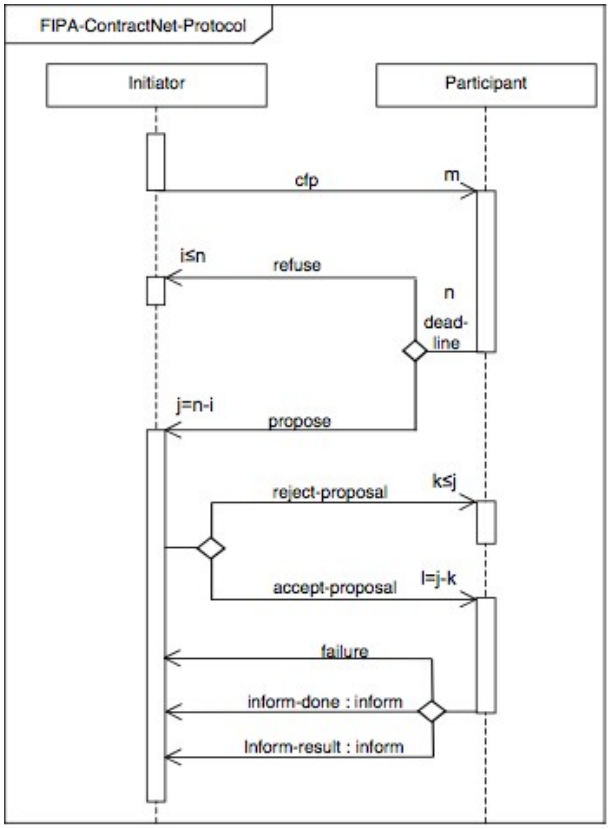


Figura 4.7: FIPA Contract Net Interaction Protocol (FIPA 2006).

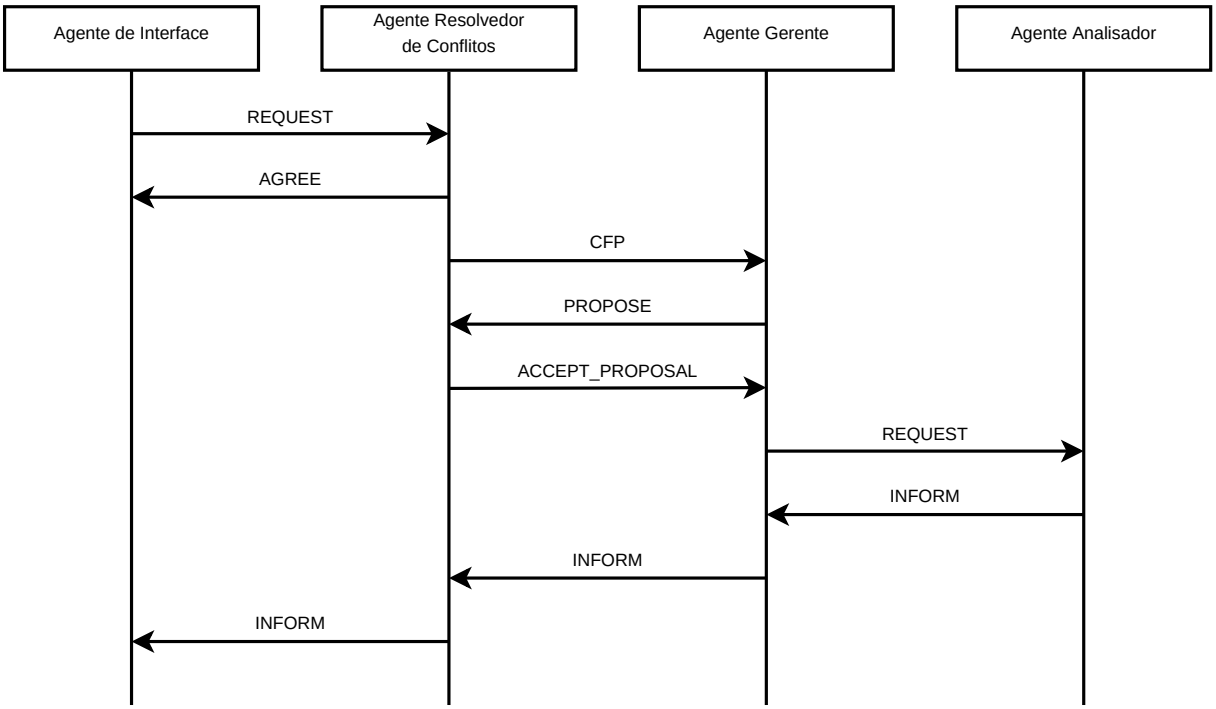


Figura 4.8: Fluxo de Mensagens dentro da nova implementação do *BioAgents*.

A troca de mensagens também foi dividida entre diversos comportamentos, ao invés de um único comportamento, os quais foram agrupados em grupos de trabalho. Para dividir esse trabalho entre os comportamentos de cada agente, definiu-se que cada agente apresenta um único comportamento responsável tanto pelo recebimento de requisições quanto pela criação de grupos de trabalhos responsáveis pelo tratamento de cada mensagem recebida. Dentro do grupo de trabalho, existe um único comportamento responsável por responder à requisição inicial podendo existir diversos outros comportamentos que se responsabilizam por cada parte de trabalho. Assim um agente pode possuir diversos grupos de trabalho tratando, cada um, de uma tarefa diferente. Além disso o agente ainda pode receber outras requisições, sem interromper outro grupo de trabalho, e tratá-las corretamente.

Na Figura 4.9 é apresentada a troca de mensagens de um Agente Resolvedor de Conflitos, onde a mensagem em vermelho (a requisição de um trabalho) é recebida e tratada pelo comportamento responsável pelo recebimento e criação do grupo de trabalho. O grupo de trabalho nessa figura é representado pelos comportamentos que recebem e enviam as mensagens azuis e verdes, onde as mensagens azuis são enviadas pelo comportamento responsável por responder a requisição e as mensagens verdes são enviadas e recebidas por um comportamento auxiliar.

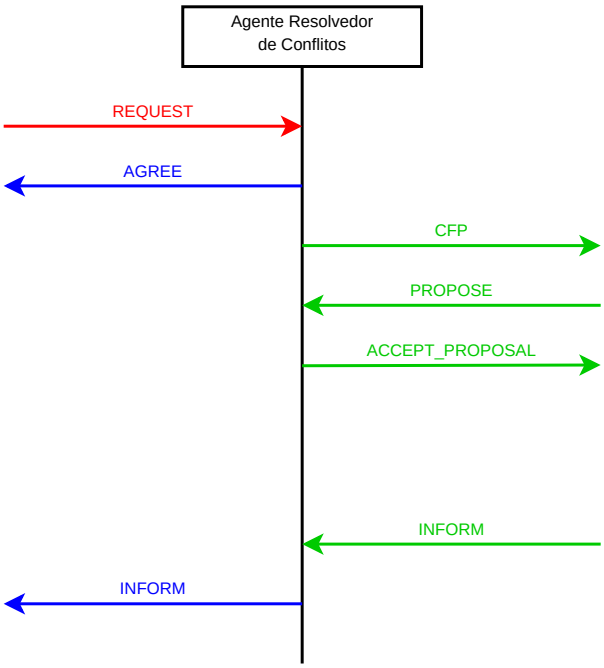


Figura 4.9: Fluxo de Mensagens de um Agente Resolvedor de Conflitos separadas por comportamento.

4.2.3 BLAT e Portrait

O algoritmo FASTA foi substituido pelo algoritmo BLAT (Kent 2002), pois não foi usado FASTA nos projetos de sequenciamento dos quais foram retirados os dados usados nos experimentos.

O impacto de incluir o BLAT no sistema foi mínimo, já que à saída do BLAT pode ser igual a saída do BLAST, portando foram usados analisadores e estruturas de dados já desenvolvidas.

O sistema *BioAgents* não possuia uma forma de identificar RNAs não-codificadores (ncRNAs). Então foi introduzido na camada colaborativa um Agente Analisador Portrait (Figura 4.10).

O Portrait foi utilizado no seguinte contexto dentro do *BioAgents*. Foi criado o Agente Analisador Portrait que recebe uma requisição do Agente Resolvedor de Conflitos toda vez que não existirem sugestões vindas dos agentes gerentes, ou seja, caso o agente resolvedor de conflitos não receba alguma sugestão dos agentes que identificam proteínas, ele solicita ao Agente Analisador Portrait para verificar se a sequência poderia ser um ncRNA.

4.2.4 Execução de Ferramentas de Comparação

Nessa nova implementação do *BioAgents*, os Agentes Analisadores ganharam a capacidade de executar as ferrramentas BLAST e BLAT. Isso possibilitou a execução do *BioAgents* mesmo sem os arquivos de saída do BLAST e BLAT terem sido gerados. Portando, o

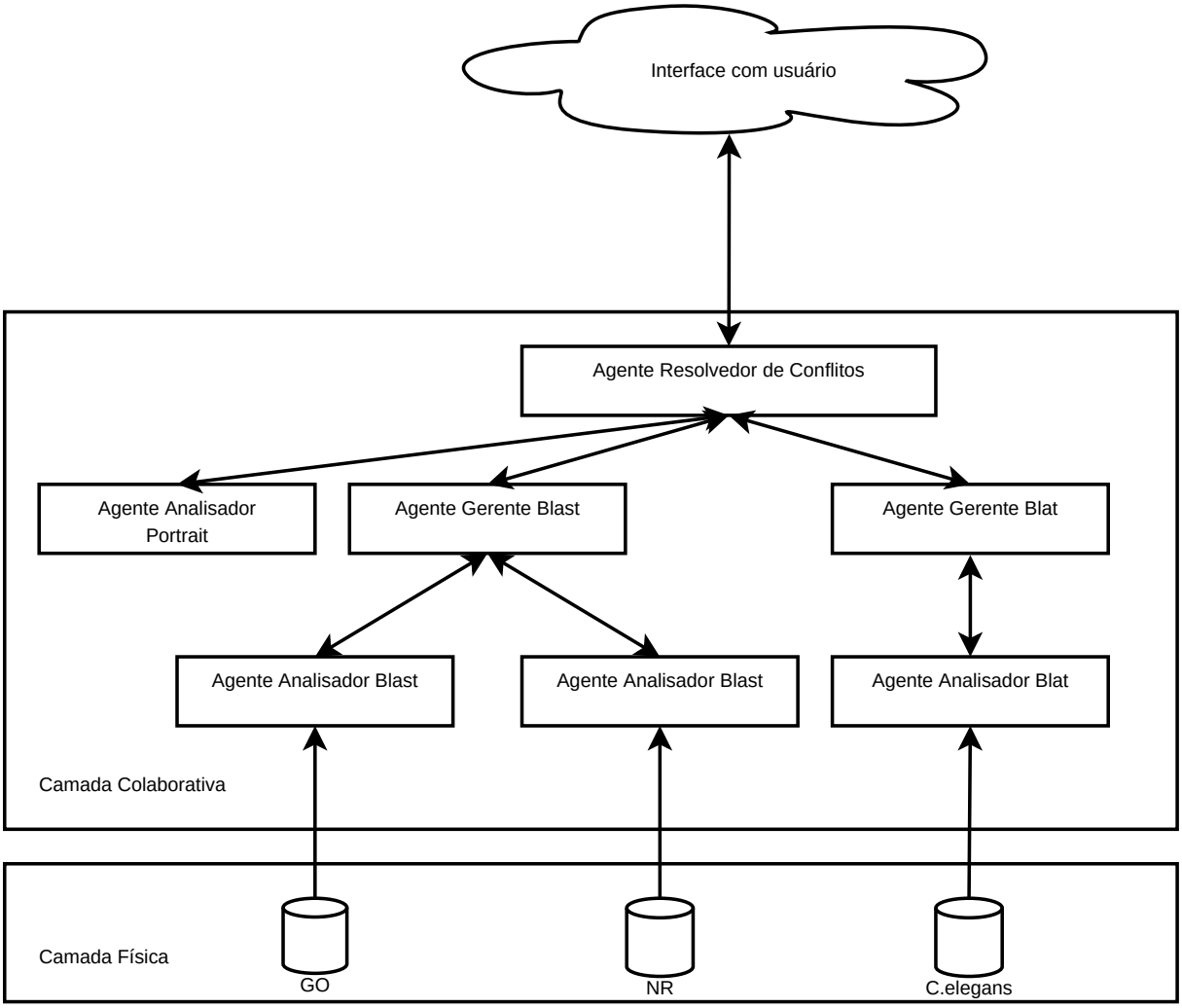


Figura 4.10: Arquitetura do *BioAgents* incluindo o Agente Analisador Portrait.

usuário não precisa mais produzir arquivos de saída compatíveis com os analisadores já implementados.

Isso aumentou um pouco a complexidade das requisições, uma requisição é um conjunto de subrequisições que se diferenciam em subrequisições de análise de arquivo ou de execução de ferramenta de comparação. Um agente gerente pode receber um conjunto mesclado de subrequições, dividindo-as e distribuindo-as entre os agente analisadores.

Capítulo 5

Método de aprendizagem por reforço no *BioAgents*

Nesse capítulo será apresentado o método utilizado para o aprendizagem de máquina incluído no sistema *BioAgents*. A descrição do método será feita na Seção 5.1. A arquitetura baseada em agentes que compõem a camada de aprendizagem é apresentada na Seção 5.2. E os detalhes de funcionamento do sistema serão apresentados na Seção 5.3.

5.1 Descrição do Método

Antes de se estabelecer um método de aprendizagem por reforço, foi analisado o conjunto de estados e ações do *BioAgents* para verificar a aplicabilidade dos métodos tradicionais. Nota-se que os agentes do *BioAgents* não possuem um conjunto S finito de estados e um conjunto $A(s)$ finito de ações predeterminadas para cada estado, pois não se pode prever todas as sequências genéticas que serão submetidas ao *BioAgents*. Isso implica que os métodos tradicionais, que tentam maximizar a provável recompensa através da escolha de $a \in A(s)$, como, por exemplo, o *Q-Learning*, não podem ser aplicados.

Desta forma, o método de aprendizagem por reforço proposto para o sistema *BioAgents* usa os bancos de dados e as ferramentas utilizadas na camada colaborativa. Os bancos de dados utilizados podem variar de acordo com o projeto de sequenciamento. Os algoritmos utilizados foram os BLAST e o BLAT, os quais são suportados pelo *BioAgents*.

A idéia do método é baseada na pontuação das ferramentas e dos bancos de dados utilizados no sistema. Essa pontuação é feita para que o *BioAgents* considere mais fortemente as ferramentas e os bancos de dados com maior pontuação. Assim, os agentes, ao longo de suas interações, vão aprendendo quais os bancos de dados e as ferramentas mais adequadas para as sugestões, o que caracteriza a aprendizagem por interação, uma das características da aprendizagem por reforço.

Essa pontuação não é feita imediatamente, mas após a sugestão ser feita pelo sistema. Isso mostra outra característica da aprendizagem por reforço, o retorno atrasado. Essa pontuação será feita por "entidades críticas" dentro do ambiente, o que é transparente para

os agentes que esperam pela pontuação. Essas "entidades críticas" compõem a camada de aprendizagem, que será detalhada na seção 5.2.

Outra característica do método é a orientação ao objetivo. Todos os agentes dentro do *BioAgents* trabalham de forma cooperativa para alcançar um objetivo e sugerir uma boa anotação para uma sequência analisada. A pontuação feita pela camada de aprendizagem auxilia o cumprimento desse objetivo.

Para calcular se um banco de dados e uma ferramenta devem ser pontuados, a camada de aprendizagem primeiramente faz uma comparação com um banco de dados de referência, o qual não é analisado dentro do projeto de sequenciamento, porém é um organismo similar e com uma anotação muito curada, denominada de organismo modelo. Essa comparação é feita utilizando o algoritmo BLAST que alinha a sequência em questão com esse banco de dados de referência e toma o melhor resultado. A partir dele é obtido o *Gene Ontology Accession Number* (*GO Accession Number*) tanto a sugestão quanto para o resultado obtido a partir do banco de referência. Se os *GO Accession Numbers* forem iguais, a sugestão é considerada correta. Porém para pontuar os bancos e as ferramentas, são analisadas todas as sugestões de cada etapa de execução do *BioAgents*, que serão descritas na seção 5.3, para verificar quais bancos e ferramentas retornaram uma sugestão correta, sendo então pontuados. A pontuação feita aos bancos de dados e às ferramentas é considerada a atribuição de recompesas feita aos agentes. Utilizando a Equação 3.1 apresentada na Seção 3.5.2, na página 33, temos os seguintes valores de recompensa (r_{t+k+1}) dado aos banco de dados e às ferramentas tratadas pelos agentes.

- 1 quando ao menos uma sugestão é considerada correta
- 0 quando nenhuma sugestão é considerada correta

Cada agente, antes de fazer a análise de seu conjunto dados, verificará a pontuação do banco de dados e da ferramenta que estão sendo analisados para ver o conjunto de sugestões a serem retornadas. Isso é feito para que o agente do nível superior tenha mais *hits* de um bom banco de dados e de uma boa ferramenta para analisar e extrair suas sugestões.

A quantidade de sugestões, $Qtd(p)$, a serem retornadas foi empiricamente definida. O critério de escolha é baseado numa função que aumenta o número de sugestões com poucos pontos. Esta decisão também considerou que um projeto de sequenciamento possui milhares de sequências. Então, após um crescimento acentuado dos pontos, evita-se que a quantidade retornada por $Qtd(p)$ seja um número muito grande, considerando que a quantidade de bons *hits* de uma ferramenta de comparação é limitada. A função $Qtd(p)$ foi definida como sendo:

$$Qtd(p) = \begin{cases} 5 & , p=0 \\ \lfloor 5*\log(p) \rfloor + 5 & , p>0 \end{cases}$$

O valor de p para um agente gerente é a pontuação do algoritmo que ele está tratando, e para os agentes analisadores, é a pontuação dos bancos que eles estão tratando. Pode-se afirmar que quando um algoritmo é pontuado, obrigatoriamente pelo menos um banco de

dados é pontuado, e se um banco de dados é pontuado obrigatoriamente um algoritmo também é pontuado. Novamente utilizando a Equação 3.1, temos que:

$$p = R(t) = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

5.2 Arquitetura Proposta

A camada de aprendizagem foi concebida como mostra a Figura 5.1. Essa camada possui um Agente Controlador de Aprendizagem e Agentes de Aprendizagem, que farão a análise e armazenamento dos resultados da análise. Observamos que, para simplificar a Figura 5.1, não foi colocada a base de dados do *BioAgents* que armazena todas as informações de execução, como requisições, sugestões e pontuação das bases de dados e das ferramentas. Todos os agentes fazem acessos a esse banco de dados para executar suas tarefas. Esses acessos serão descritos na Seção 5.3.

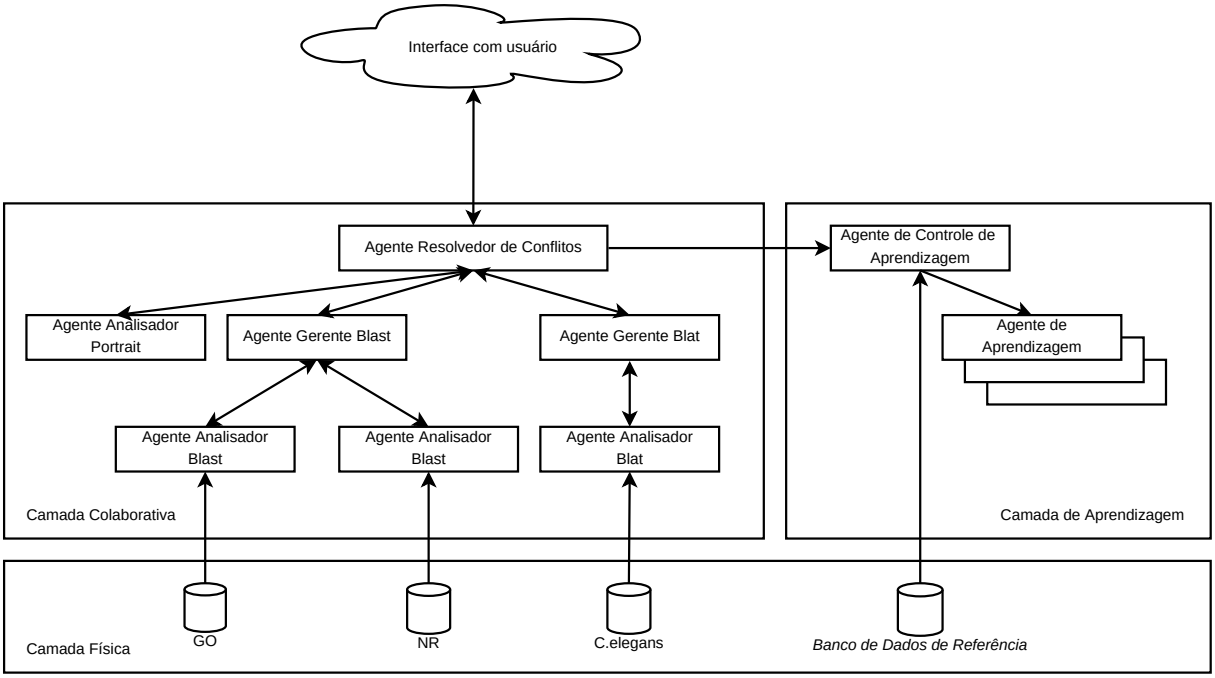


Figura 5.1: Arquitetura com o tratamento de requisições Blast e Blat, um Agente Analisador Portrait, um Agente de Controle de Aprendizagem e um conjunto de 3 Agentes de Aprendizagem.

A função do Agente Controlador de Aprendizagem é fazer a execução do BLAST da sequência requerida com o banco de referência escolhido para o projeto. Os melhores resultados serão enviados para os Agentes de Aprendizagem, junto com o escopo a ser analisado. Entende-se por escopo uma parte da análise feita pelo *BioAgents*, por exemplo, a análise da execução do Agente Resolvedor de Conflitos ou de um Agente Gerente.

A análise de um Agente de Aprendizagem no escopo do Agente Resolvedor de Conflitos vai ser feita considerando todas as sugestões recebidas dos Agentes Gerentes e comparando-as, utilizando o *GO Accession Number*, com as sequências enviadas pelo Agente de Controle de Aprendizagem. Assim o Agente de Aprendizagem irá pontuar as ferramentas que fizeram uma sugestão correta.

O outro escopo de análise, que analisa um Agente Gerente, tem um funcionamento similar. Da mesma forma, serão comparadas as sugestões retornadas pelos Agentes Analisadores, e serão pontuados os bancos de dados que originaram as sugestões corretas.

Os dados para as análises dos agentes da camada de aprendizagem são obtidas do banco de dados do *BioAgents* implementado para suportar essa solução. Notamos que essa camada não existia na proposta original. A camada contém todos os dados de uma requisição e as sugestões que foram feitas por cada agente para essa requisição. Esse dados serão utilizados na camada de aprendizagem para fazer as comparações, mas logo após isso, eles serão removidos por questão de espaço. O único dado preservado é a requisição original e seu resultado, para consultas posteriores.

5.3 Funcionamento

Para exemplificar o funcionamento do sistema *BioAgents*, vamos utilizar a seguinte requisição:

- Requisição:
  - Subrequisição 1:
    - Tipo: Execução de ferramenta
    - Ferramenta: BLAST
    - Sequência: ACGTAGGTCCA...
    - Base de Dados: SwissProt
  - Subrequisição 2:
    - Tipo: Execução de ferramenta
    - Ferramenta: BLAT
    - Sequência: ACGTAGGTCCA...
    - Base de Dados: S.cerevisiae
  - Subrequisição 3:
    - Tipo: Análise de Arquivo
    - Ferramenta: BLAST
    - Caminho: /var/exemplo/saida.blast
  - Subrequisição 4:
    - Tipo: Análise de Arquivo
    - Ferramenta: BLAT
    - Caminho: /var/exemplo/saida.blat

Nota-se que uma requisição é dividida em subrequisições. Uma subrequisição contém o seu tipo, a ferramenta a ser utilizada e uma sequência com uma base de dados, ou uma arquivo de saida produzido pela ferramenta em questão. Existem dois tipos de subrequisições, análise de arquivo e execução de ferramenta. O tipo análise de arquivo pressupõe um arquivo de saída previamente pronto, enquanto o tipo execução de ferramenta deve produzir esse arquivo utilizando a sequência e a base de dados informada.

O funcionamento do sistema *BioAgents* inicia-se com uma mensagem *REQUEST*, contendo uma requisição como descrita previamente, vinda da interface com o usuário, como mostra a Figura 5.2. A interface com usuário não é implementada junto do *BioAgents*, e é representada por uma nuvem pois qualquer sistema pode obter serviços do *BioAgents* através da troca de mensagens.

Em seguida o Agente Resolvedor de Conflitos recebe a requisição com seu comportamento de recebimento, o qual cria um grupo de trabalho para dividir a requisição por especialidade, de acordo com as ferramentas solicitadas, e fazer o protocolo *Contract Net* com os agentes gerentes. Esse grupo de trabalho apresenta um comportamento que responderá a interface e controlará o fluxo do trabalho a ser feito

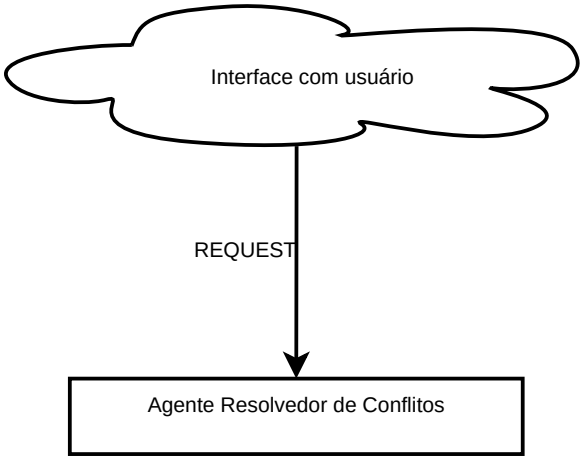


Figura 5.2: Passo 1: Início do ciclo de funcionamento com *BioAgents*.

O segundo passo ainda acontece no Agente Resolvedor de Conflitos, o qual, através do grupo de trabalho, inicia um *Contract Net*, com uma mensagem *CFP* com os Agentes Gerentes, como mostrado na Figura 5.3. O *Contract Net* ocorre entre Agentes Gerentes de mesma especialidade somente. Nesse momento os Agentes Gerentes, através de seus comportamentos de recebimento, criam um grupo de trabalho responsável para tratar esta solicitação. Para responder a solicitação, primeiro os gerentes verificam a disponibilidade de Agentes Analisadores para atenderem à solicitação. A partir daí, os Agentes Gerentes fazem suas propostas ao Agente Resolvedor de Conflitos, como mostra a Figura 5.3. Essa proposta consiste no número de agentes disponíveis para atender à requisição.

Neste momento, o Agente Resolvedor de Conflitos analisa as propostas agrupadas por especialidade, seleciona uma ou descarta todas. Para aceitar uma proposta, o número de agentes disponíveis deve ser igual ou maior que o número de base de dados e arquivos a

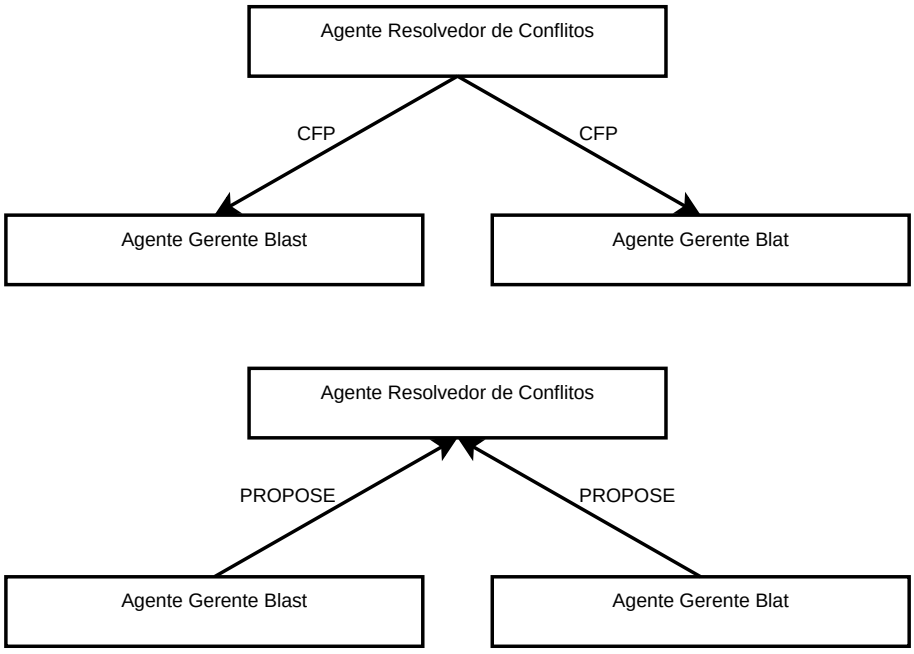


Figura 5.3: Passos 2 e 3: Solicitação de proposta por parte do Agente Resolvedor de Conflitos e retorno das propostas dos Agentes Gerentes.

serem analisados. Aceitando as propostas dos Agentes Gerentes, o Agente Resolvedor de Conflitos envia uma aceitação para cada gerente, como mostrado na Figura 5.4.

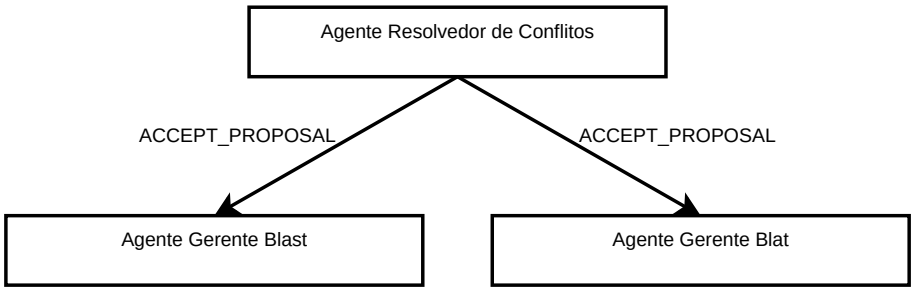


Figura 5.4: Passo 4: Aceitação das propostas dos Agentes Gerentes.

Os Agentes Gerentes, ao receberem a aceitação, solicitam que Agentes Analisadores sejam alocados para analisar cada base de dados e arquivos que foram inicialmente requeridos. A requisição então é dividida e são criados comportamentos dentro do grupo de trabalho para fazer as solicitações aos seus respectivos Agentes Analisadores, como mostrado na Figura 5.5.

Os Agentes Analisadores, recebendo um requisição, não poderão atender outra requisição qualquer, porque nesse momento eles entram em um estado ocupado. Esses agentes tratam requisição de análise de arquivo, ou seja, analisam um arquivo de saída já pronto de uma ferramenta de comparação ou de análise com execução de ferramenta, ou seja, os Agentes Analisadores devem executar a ferramenta solicitada para obter o arquivo de

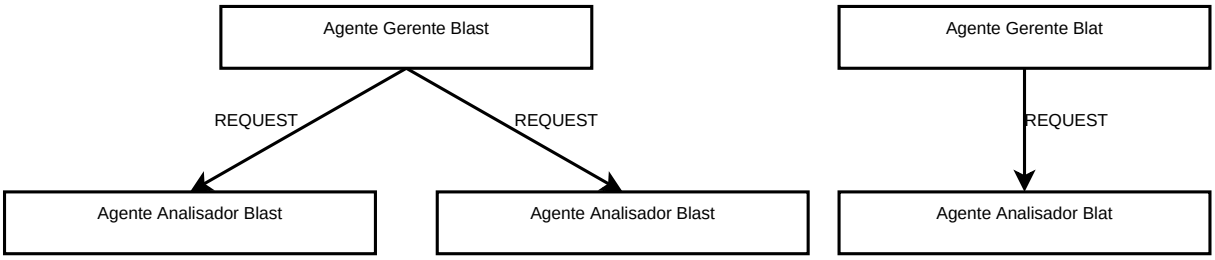


Figura 5.5: Passo 5: Requisição aos Agentes Analisadores.

saída desejado. Um requisição de análise com execução de ferramenta, antes de qualquer ação, requer que o agente execute a ferramenta, *BLAST* ou *BLAT* dependendo da sua especialidade, para obter o resultado da comparação. Após a execução do algoritmo, a analise é igual para os dois tipo de requisição, já que o arquivo é uma saída de uma ferramenta. O agente transforma todo o conteúdo da saída dos algoritmos em objetos definidos na ontologia e depois alimenta a base de fatos do Drools, executando as regras e recuperando os resultados. Cada resultado é uma coleção de sugestões, as quais contém *hits*. A quantidade máxima de sugestões é definida pela função $Qtd(p)$ e a quantidade de pontos é recuperado do banco de dados do *BioAgents*. O arquivo de saída e as sugestões são armazenadas no banco de dados do *BioAgents* para análise posterior na camada de aprendizagem. Por fim, o resultado é enviado para o Agente Gerente que fez a requisição. Esse passos estão descritos na Figura 5.6.

Ao receber todas as sugestões dos agentes analisadores, os agentes gerentes populam a base de fatos do Drools com as sugestões recebidas, executam as regras, recuperam os resultados, salvam todos os dados na base de dados do *BioAgents* e repassam os resultados para o agente resolvedor de conflitos, como descrito na Figura 5.7. Aqui também é utilizado a função $Qtd(p)$ para definir a quantidade máxima de sugestões a serem retornadas.

O Agente Resolvedor de Conflitos, nesse momento, deve fazer uma decisão baseada nas sugestões recebidas. Caso não tenha recebido nenhuma sugestão, deve fazer uma solicitação ao Agente Analisador Portrait, esperando dele uma resposta, que é a probabilidade da sequência analisada ser um ncRNA. Esse resultado então é repassado para a interface de usuário quando então termina uma interação completa do *BioAgents*. O funcionamento do Agente Analisador Portrait inicia-se com uma requisição, o Portrait é executado e extrai o resultado do arquivo de saída, retornando-o para o Agente Resolvedor de Conflitos.

Se o Agente Resolvedor de Conflitos tiver recebido sugestões, faz uma análise das sugestões recebidas utilizando a base de conhecimento e regras Drools, recupera os resultados, armazena todos os dados e resultados no banco de dados do *BioAgents* e envia duas mensagens, uma com o resultado da requisição inicial para a interface com o usuário e outra para o Agente de Controle de Aprendizagem dizendo que uma requisição terminou e deve ser analisada. Esses dois casos são descritos na Figura 5.8.

Nos dois casos a interface com o usuário recebe uma resposta. Não havendo sugestões, a resposta contém somente a probabilidade da sequência ser um ncRNA, como por exemplo:

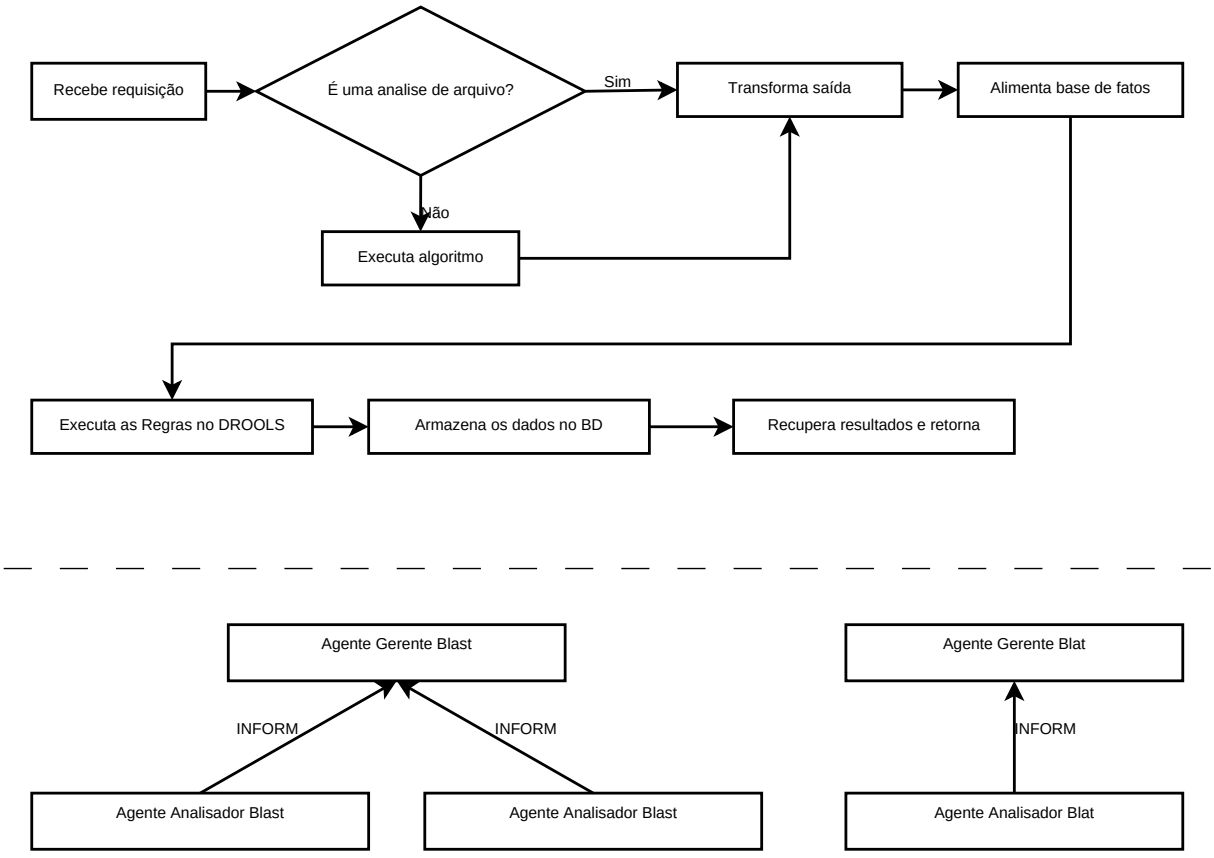


Figura 5.6: Passo 6: Fluxo de execução dos Agentes Analisadores e resposta aos Agentes Gerentes.

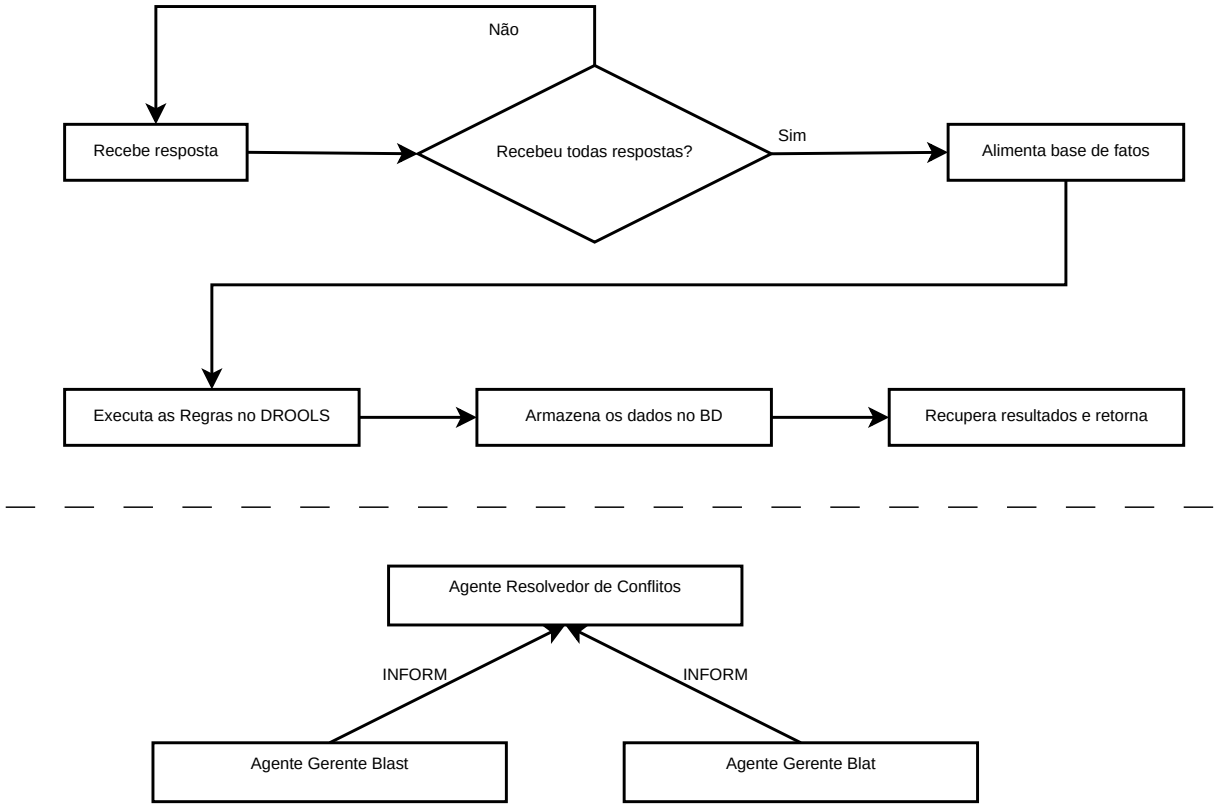


Figura 5.7: Passo 7: Fluxo de execução da análise dos Agentes Gerentes e resposta ao Agente Resolvedor de Conflitos.

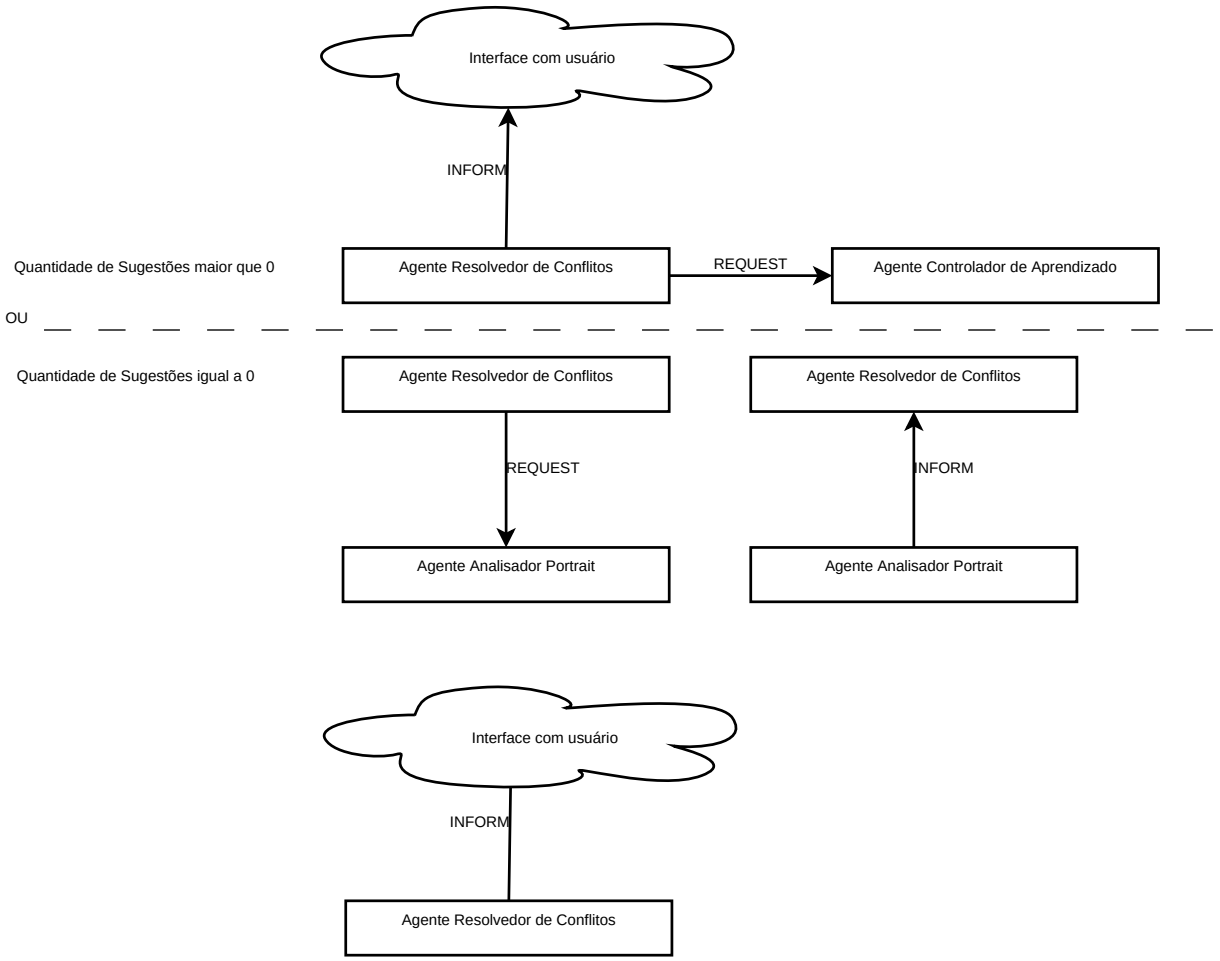


Figura 5.8: Passo 8: Decisão e execução do Agente Resolvedor de Conflitos.

- Resposta:
  - Probabilidade ncRNA: $87,2\%$

Havendo sugestões, a resposta contem todos os dados do alinhamento sugerido recuperados a partir da saída da ferramenta utilizada, como por exemplo:

- Resposta:
  - Sugestão:
    - HidId: gi|166989585|sp|A4WFL5.1|GLGB_ENT38
    - Nome do Produto: RecName: Full=1,4-alpha-glucan-branching enzyme
    - E-Value: $1.0e^{-114}$
    - Score: 410

A partir desse momento, a metodologia de aprendizagem por reforço descrita na Seção 5.1 é aplicada e se inicia quando o Agente Controlador de Aprendizagem, ao receber uma solicitação, recupera a requisição de análise feita pelo Agente Resolvedor de Conflitos e todas as requisições originadas a partir desta, ou seja, requisições feitas aos Agentes Gerentes. Após essa recuperação, o agente alinha a sequência contida na requisição inicial com o banco de dados de referência utilizando o algoritmo BLAST. Em seguida, escolhe o melhor *hit* baseado no menor *e-value*. Por fim, ele envia aos Agente de Aprendizagem esse *hit*, uma requisição, a inicial ou uma das originadas, e um escopo a ser analisado. Deve-se ressaltar que existe sempre um Agente de Aprendizagem para cada Agente Gerente envolvido na requisição e mais um para o Agente Resolvedor de Conflitos. A Figura 5.9 mostra todas essa etapas.

Por fim os Agentes de Aprendizagem, recebendo a requisição, recuperam as sugestões feitas para a requisição a ser analisada, obtem o *GO Accession Number* para cada *hit* das sugestões e para o *hit* que recebeu do Agente Controlador de Aprendizagem e os compara. Caso algum número EC de uma sugestão for igual ao número EC do *hit*, a base de dados ou o algoritmo será pontuado, dependendo do escopo analisado. Porém deve-se ressaltar que se um Agente de Aprendizagem pontua um banco de dados, um outro Agente de Aprendizagem deve pontuar a ferramenta que foi executada utilizando desse banco de dados. Por fim essa pontuação é salva no banco de dados, como mostra a Figura 5.10.

Esse é a descrição completa do funcionamento do *BioAgents* que foi implementada para a solução do aprendizagem por reforço.

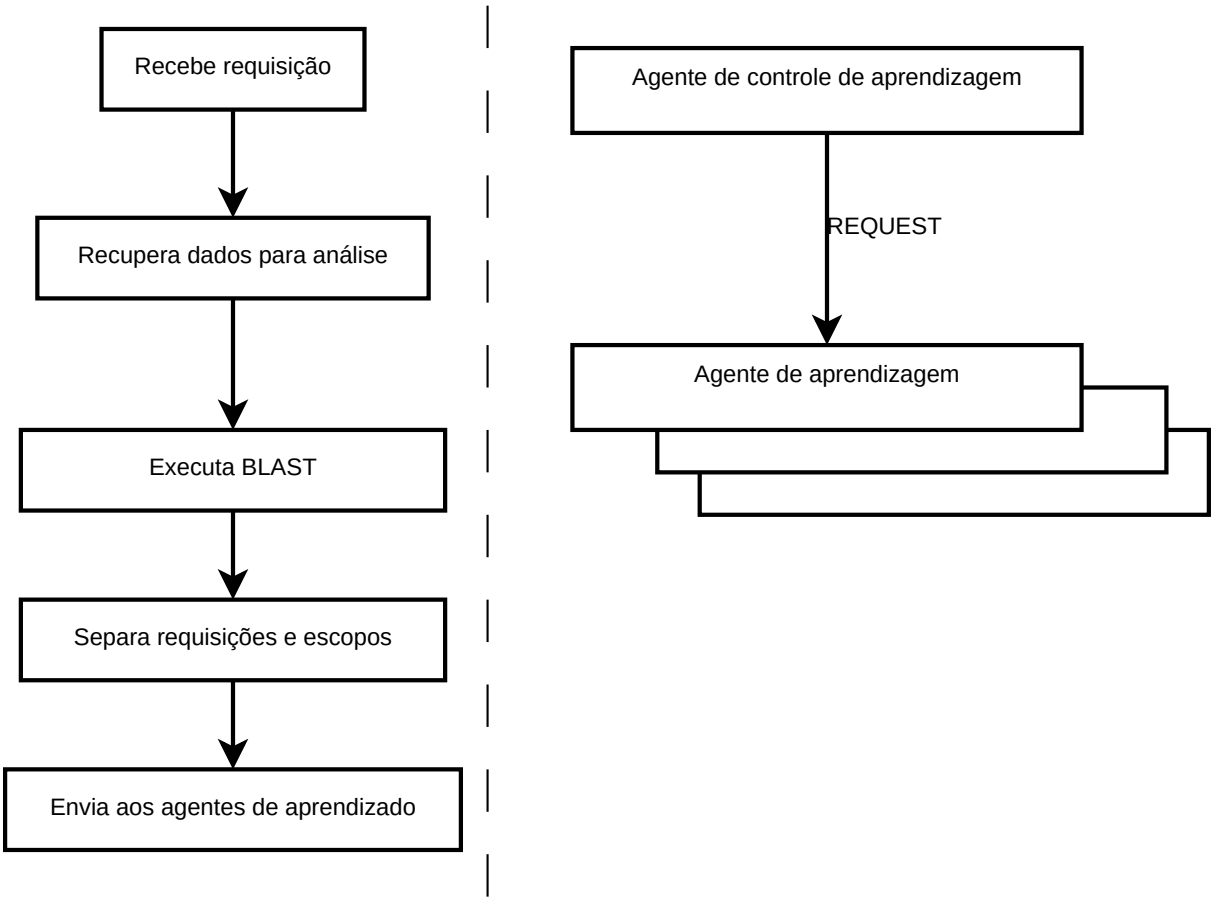


Figura 5.9: Passo 9: Início do fluxo de aprendizado no Agente Controlador.

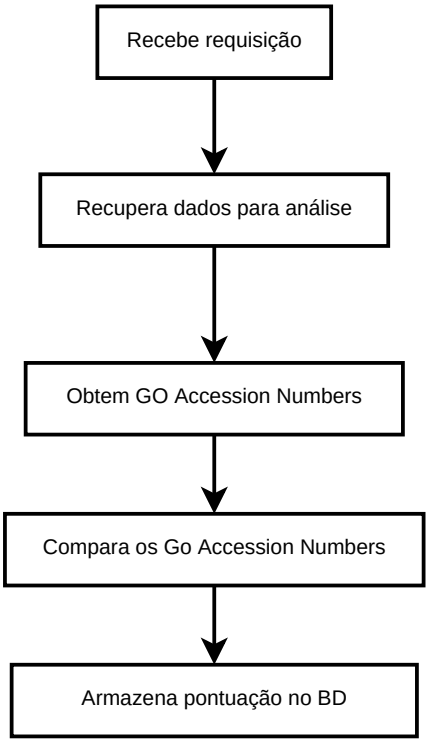


Figura 5.10: Passo 10: Fluxo dos Agentes de Aprendizagem.

Capítulo 6

Experimentos

Para a validação do método proposto neste trabalho e da implementação realizada, foram feitos dois experimentos baseados no Projeto Genoma Funcional e Diferencial do *Paracoccidioides brasilienses* (Projeto Genoma Pb) e no Projeto Genoma Funcional e Genética Genômica de *Paullinia cupana* (Projeto Genoma Guaraná), descritos respectivamente nas Seções 6.1 e 6.2 .

Os dois experimentos foram executados tanto para validar o funcionamento do sistema quanto para verificar a acurácia das sugestões, comparando-as com experimentos previamente realizados e executados com dados desses dois projetos.

Para que os resultados pudessem ser devidamente comparados com a anotação manual, foram utilizadas as mesmas saídas do BLAST usadas na anotação manual realizada pelos biólogos. Isso foi feito para que as novas anotações e correções das bases de dados mais atualizadas não interferissem nos resultados. A condição utilizada para verificar se uma sugestão feita pelo *BioAgents* estava correta foi baseada na comparação com a anotação manual feita pelos biólogos. A recomendação de sugestão era considerada igual à anotação manual se ambas apresentassem pelo menos três palavras iguais.

6.1 Projeto Genoma Pb

Nesta seção inicialmente descreveremos o Projeto Genoma Pb, em seguida o experimento executado, e por fim discutiremos os resultados obtidos.

6.1.1 Descrição do Projeto

O Projeto Genoma Pb tem uma base de dados com 6.107 genes, divididos em 2.662 *contigs* e 3.445 *singlets*, como mostra a Figura 6.1.

A anotação foi baseada nos resultados da anotação automática feita com cinco bancos de dados, *nr*, *COG*, *GO*, *Saccharomyces cerevisiae* e *Schizosaccharomyces pombe*, e dois algoritmos, BLAST e FASTA. Os bancos de dados *nr*, *COG* e *GO* foram utilizados com o BLAST e os bancos de dados *S. cerevisiae* e *S. pombe* com o FASTA.

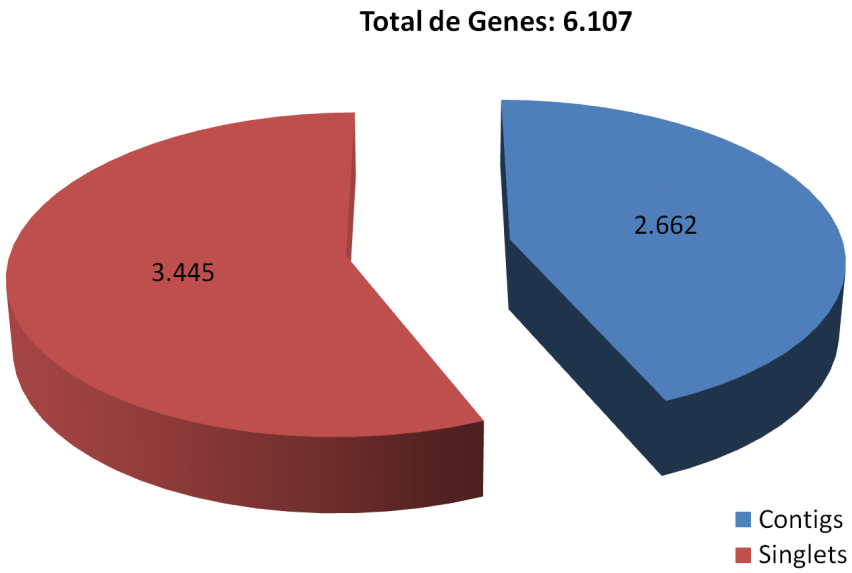


Figura 6.1: Distribuição de *contigs* e *singlets* do Projeto Genoma Pb.

Desses 6.107 genes, 2.820 foram anotados manualmente, como mostra a distribuição na Figura 6.2.

6.1.2 Descrição do Experimento

O sistema *BioAgents* utilizou as ferramentas BLAST com os bancos de dados *nr*, *COG* e *GO* e a ferramenta BLAT com os bancos de dados, *S. cerevisiae* e *S. pombe* usando os dados do Projeto Genoma Pb. Neste experimento utilizou-se a base de dados do *Caenorhabditis elegans* como referência.

6.1.3 Resultados

Os resultados obtidos pelo *BioAgents* com os dados do Projeto Genoma Pb estão descritos na Tabela 6.1 e nas Figuras 6.3 e 6.4.

A linha 3 da Tabela 6.1 mostra uma aumento de 79 sugestões feitas pelo *BioAgents* utilizando a camada de aprendizado, perfazendo um aumento de 1.29%. A quantidade total de sugestões representa 51.07% do total de genes do Projeto Genoma Pb.

Um resultado interessante é o aumento de 164 sugestões consideradas corretas, como indicado na linha 4 da Tabela 6.1, perfazendo um aumento de 3.81%. Isso representa um aumento percentual de 57.43% para 61.24%. Em relação aos dados do Projeto Genoma Pb, esse valor representa 31.28% dos genes do projeto e 67.73% dos genes anotados.

A quantidade de ncRNAs identificadas permaneceu constante, como mostra a linha 5 da Tabela 6.1, pois o mecanismo de aprendizagem não incluiu o Portrait.

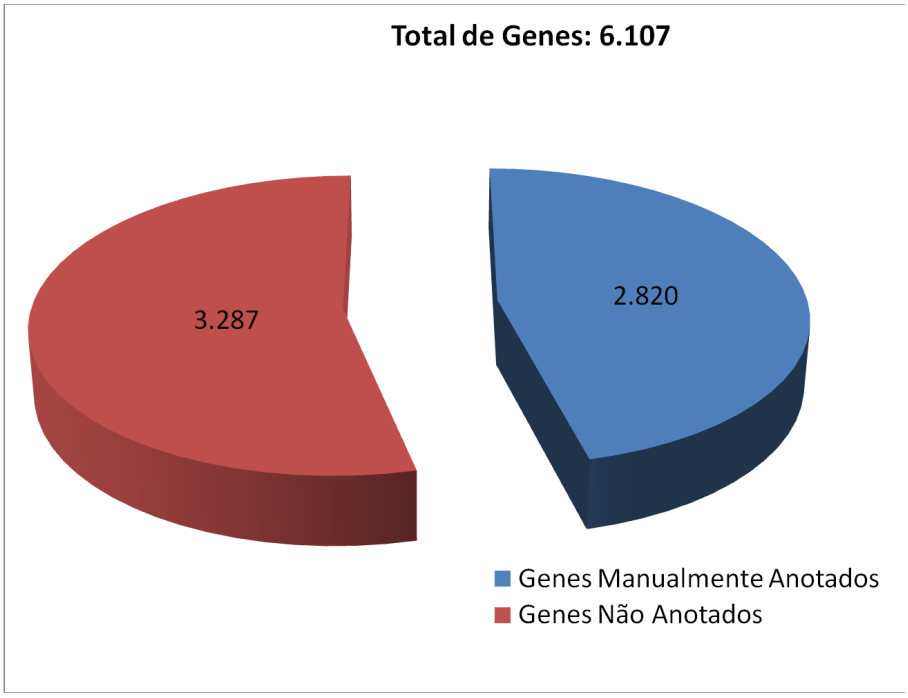
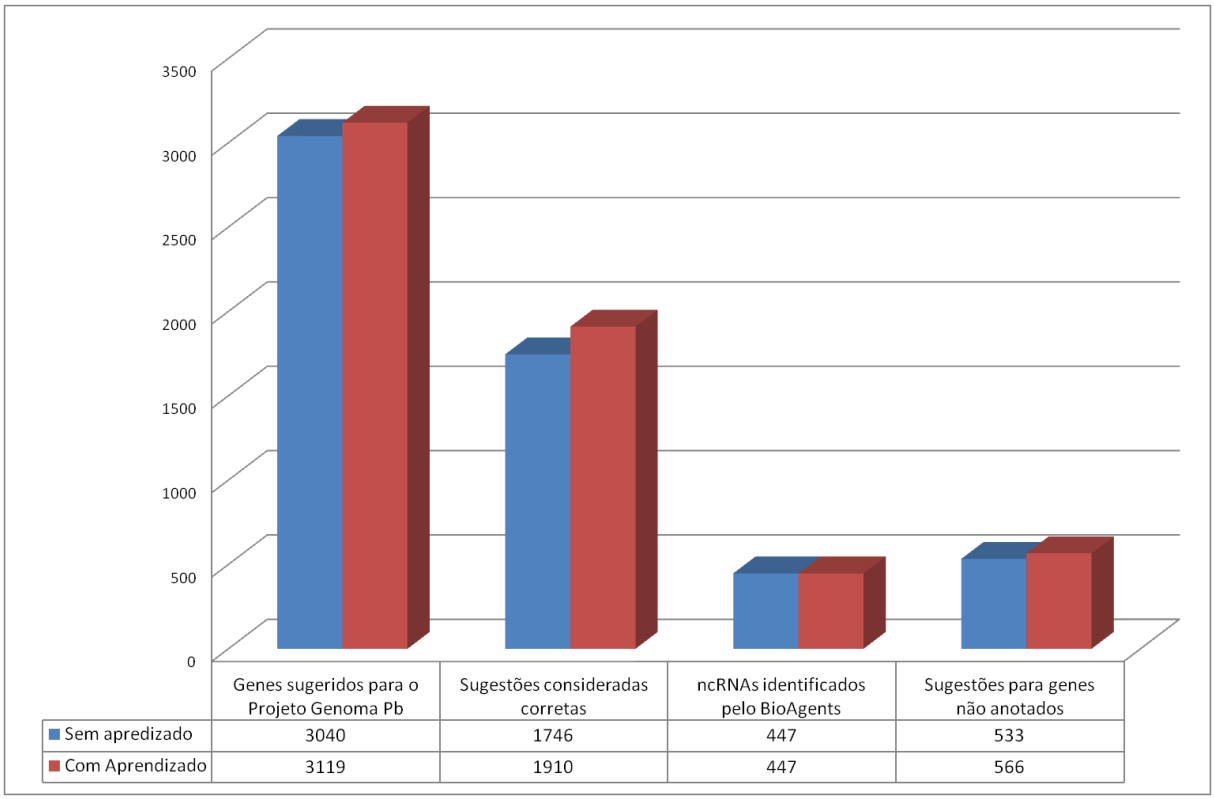


Figura 6.2: Distribuição dos grupos do Projeto Genoma Pb, conforme anotação manual.

		Sem Aprendizado	Com Aprendizado
1	Genes do Projeto Genoma Pb	6.107	
2	Genes manualmente anotados	2.820	
3	Genes sugeridos pelo *BioAgents*	3.040	3.119
4	Sugestões consideradas corretas	1.746	1.910
5	ncRNAs identificados pelo *BioAgents*	447	447
6	Sugestões para genes não anotados	533	566

Tabela 6.1: Tabela de resultados obtidos no experimento com o Projeto Genoma Pb.



	Genes sugeridos para o Projeto Genoma Pb	Sugestões consideradas corretas	ncRNAs identificados pelo BioAgents	Sugestões para genes não anotados
Sem apredizado	3040	1746	447	533
Com Aprendizado	3119	1910	447	566

Figura 6.3: Comparação dos valores obtidos no Projeto Genoma Pb.

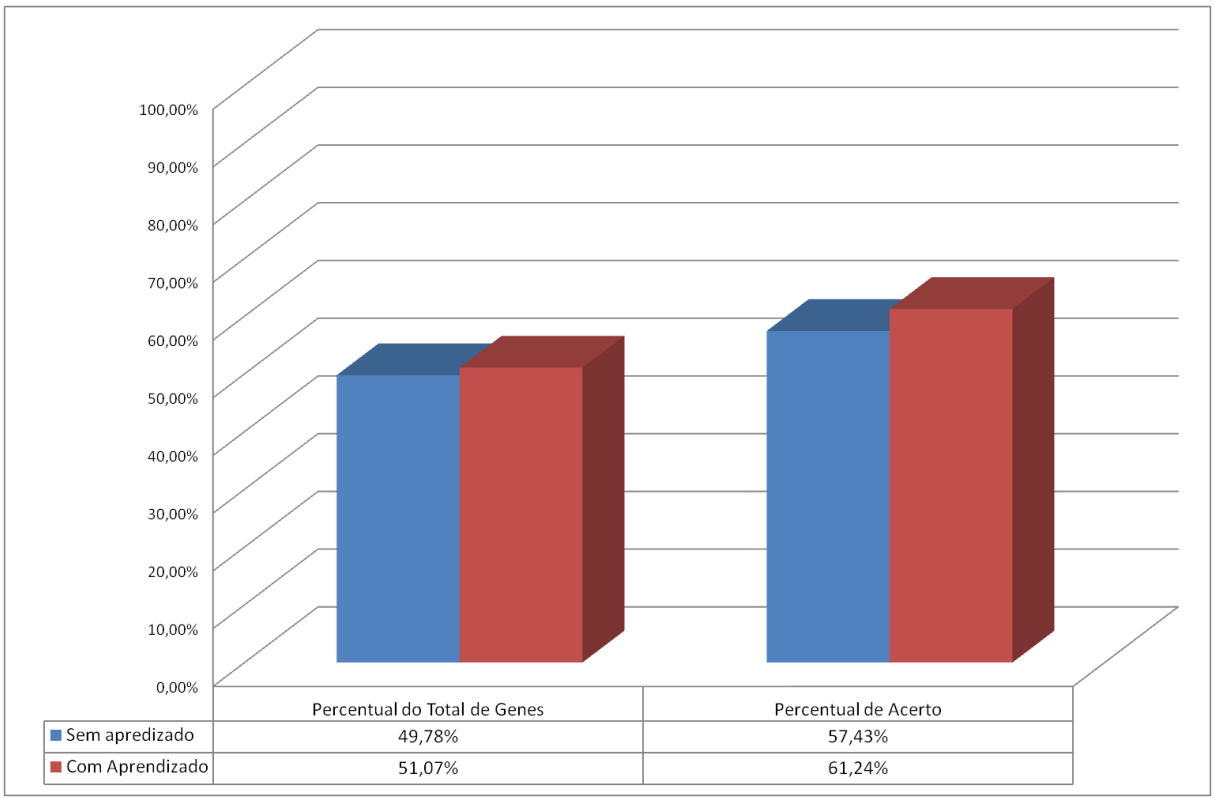


Figura 6.4: Comparação das proporções em relação ao total de genes do Projeto Genoma Pb.

Na linha 6, nota-se um aumento da quantidade de 33 sugestões feitas para genes não anotados, perfazendo um total de 1.00% do genes não anotados manualmente.

6.2 Projeto Genoma Guaraná

Nesta seção inicialmente descreveremos o Projeto Genoma Guaraná, em seguida o experimento executado e por fim discutiremos os resultados obtidos.

6.2.1 Descrição do Projeto

O Projeto Genoma Guaraná possui 8.597 grupos, divididos em 2.638 *contigs* e 5.959 *singlets*, como mostrado na Figura 6.5.

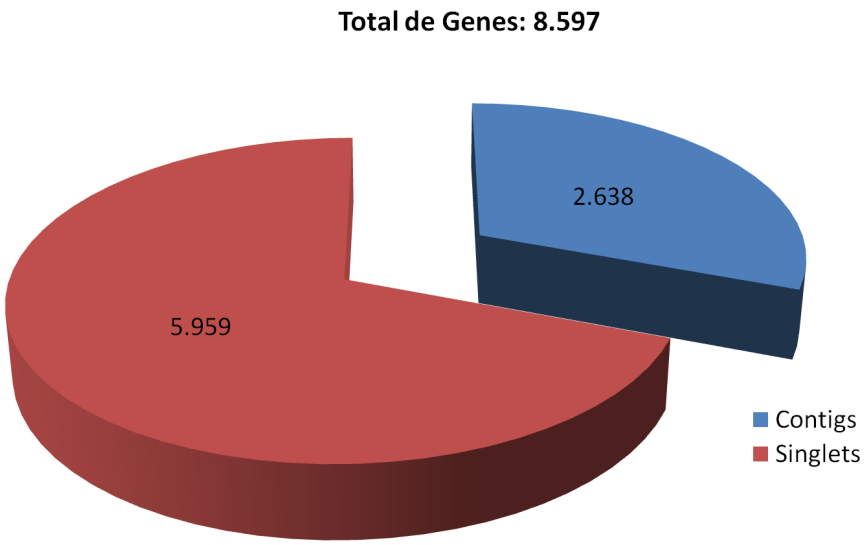


Figura 6.5: Distribuição de *contigs* e *singlets* do Projeto Genoma Guaraná.

As anotações manuais foram feitas com base nos bancos de dados *nr*, *KOG*, *GO* e *SwissProt* utilizando a ferramenta BLAST, perfazendo um total de 7.725 genes anotados, como mostra a Figura 6.6.

6.2.2 Descrição do Experimento

O experimento feito pelo *BioAgents* usou os bancos *nr*, *KOG*, *GO* e *SwissProt* com o BLAST, além do banco de dados da *Oriza sativa* com o BLAT. Neste experimento foi utilizado a base de dados da *Arabidopsis thaliana* como referência para a camada de aprendizado.

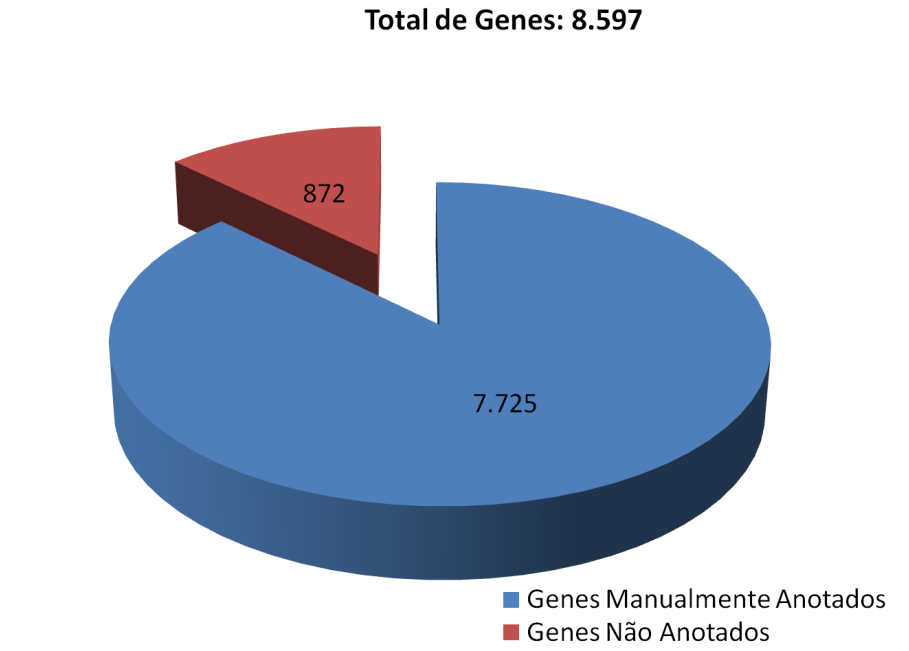


Figura 6.6: Distribuição dos grupos do Projeto Genoma Guaraná, conforme anotação manual.

O resultado na anotação automática do Projeto Genoma Guaraná utilizou a saída HTML disponibilizada pelo BLAST. Isso exigiu que os dados fossem previamente preparados para que a plataforma *BioAgents* pudesse interpretá-las corretamente, já que o pacote *BioJava* não tem a capacidade de transformar esse tipo de saída. A preparação dos dados consistiu na transformação desses resultados em arquivos compatíveis com a saída do tipo texto, também disponibilizada pelo BLAST. Esse transformação utilizou como base a ferramenta *html2text*, disponível para plataformas Linux. Porém, como alguns arquivos HTML do Projeto Genoma Guaraná estavam mal formatados, foi necessário corrigir certas informações para que pudessem ser submetidos ao *html2text*.

6.2.3 Resultados

Os resultados obtidos pelo *BioAgents* no Projeto Genoma Guaraná estão descritos na Tabela 6.2 e nas Figuras 6.7 e 6.8.

A linha 3 da Tabela 6.2 mostra uma aumento de 78 sugestões feitas pelo *BioAgents* utilizando a camada de aprendizado, perfazendo um aumento de 0.91%. A quantidade total de sugestões representa 73.00% do total de genes do Projeto Genoma Guaraná.

Esse resultado apresentou um aumento de 121 sugestões consideradas corretas, indicado na linha 4 da Tabela 6.2. Isso representa um aumento percentual de 56.15% para 57.38%. Em relação aos dados do Projeto Genoma Guaraná, esse valor representa 41.89% dos genes do projeto e 46.61% dos genes anotados.

		Sem Aprendizado	Com Aprendizado
1	Genes do Projeto Genoma Guaraná	8.597	
2	Genes manualmente anotados	7.725	
3	Genes sugeridos pelo *BioAgents*	6.198	6.276
4	Sugestões consideradas corretas	3.480	3.601
5	ncRNAs identificados pelo *BioAgents*	1379	1317
6	Sugestões para genes não anotados	306	367

Tabela 6.2: Tabela de resultados obtidos no experimento com o Projeto Genoma Guaraná.

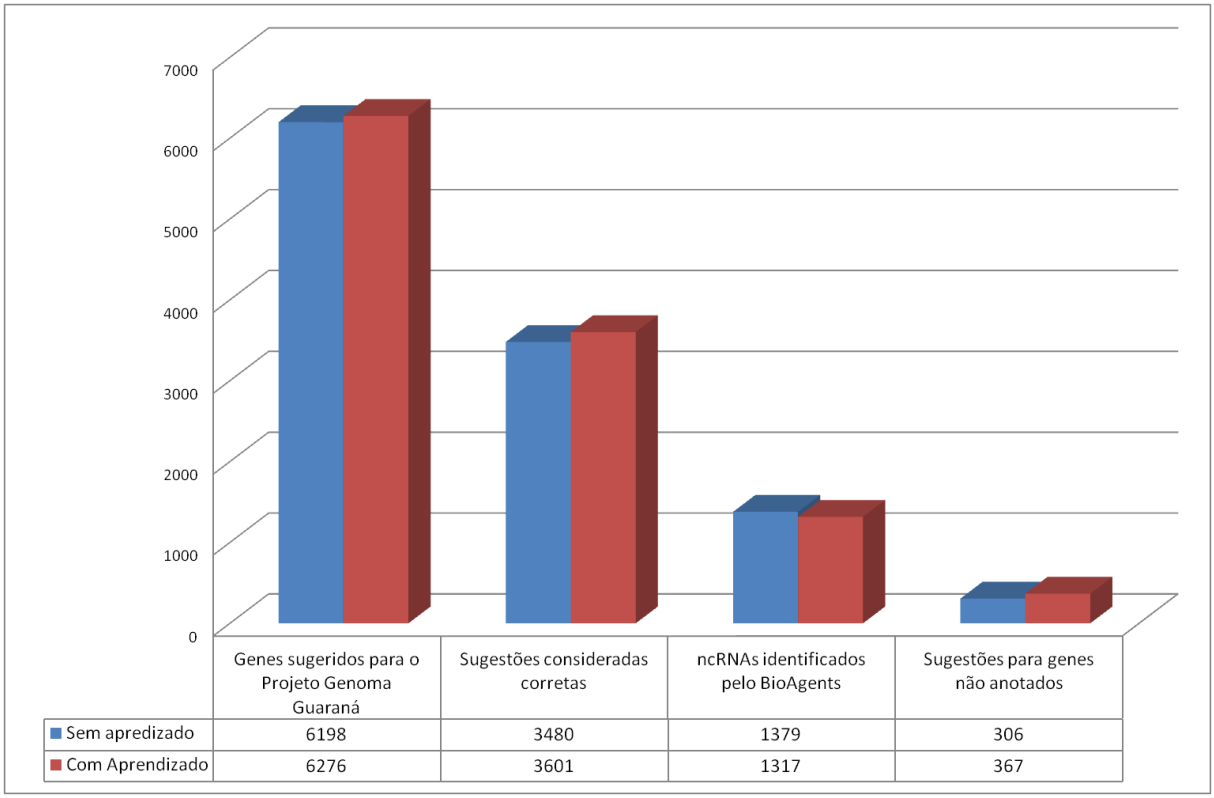


Figura 6.7: Comparação dos valores obtidos no Projeto Genoma Guaraná.

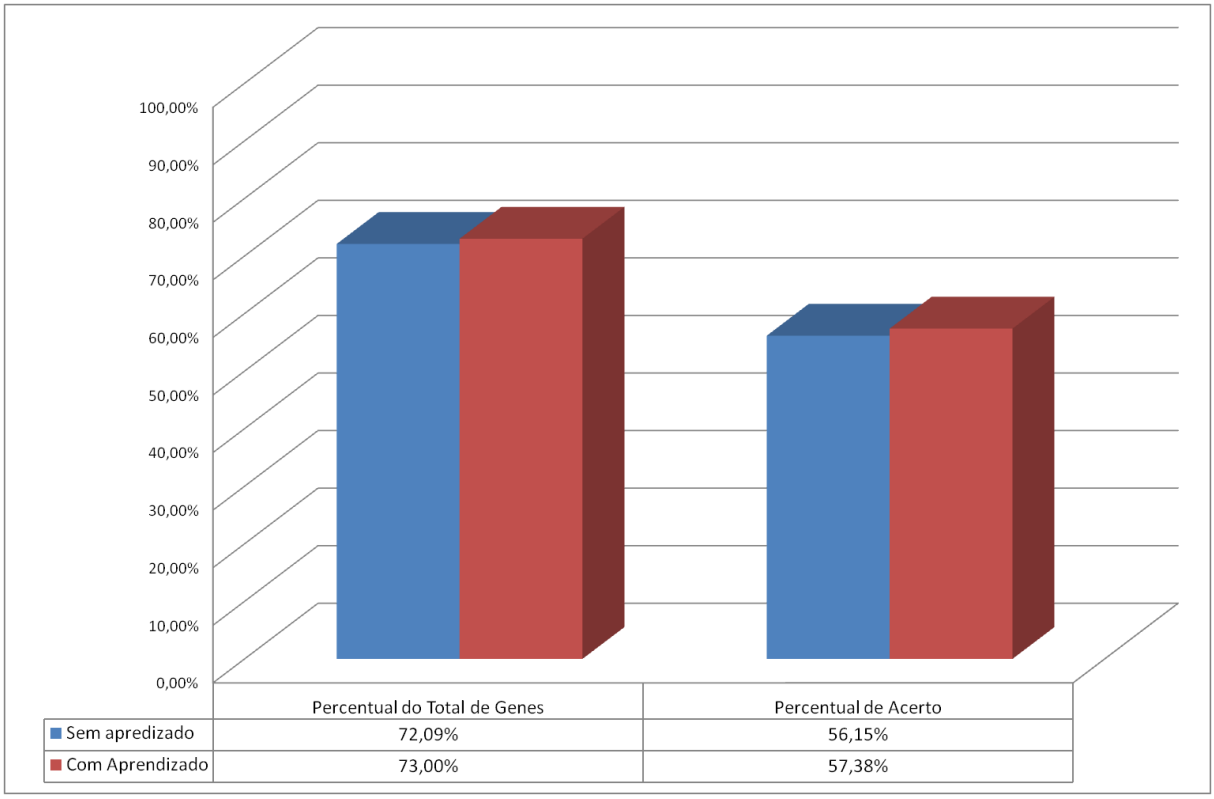


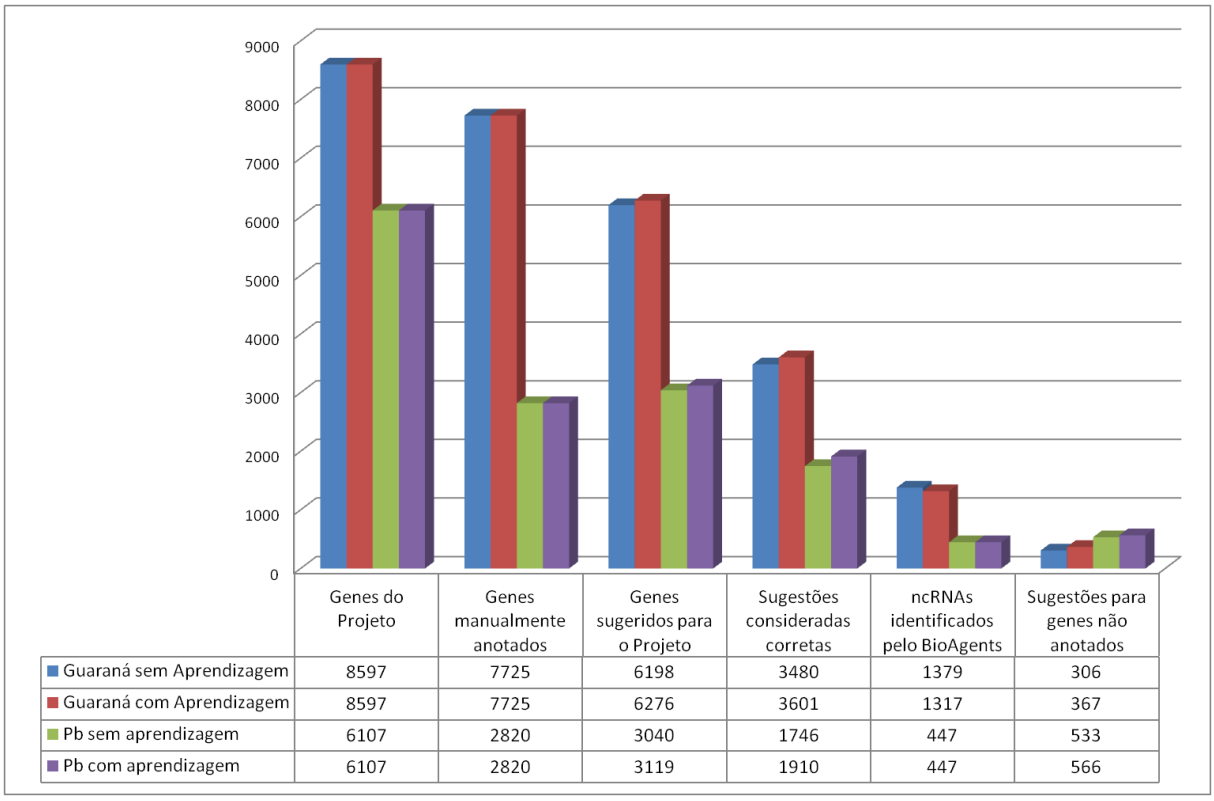
Figura 6.8: Comparação das proporções em relação ao total de genes do Projeto Genoma Guaraná.

A quantidade de ncRNAs identificadas reduziu, como mostra a linha 5 da Tabela 6.2, pois o *BioAgents* com a camada de aprendizagem apresentou sugestões baseadas em BLAST e BLAT para genes identificados como ncRNAs anteriormente.

Na linha 6, nota-se um aumento da quantidade de 61 sugestões feitas para genes não anotados.

6.3 Comparação entre os experimentos e discussão dos resultados

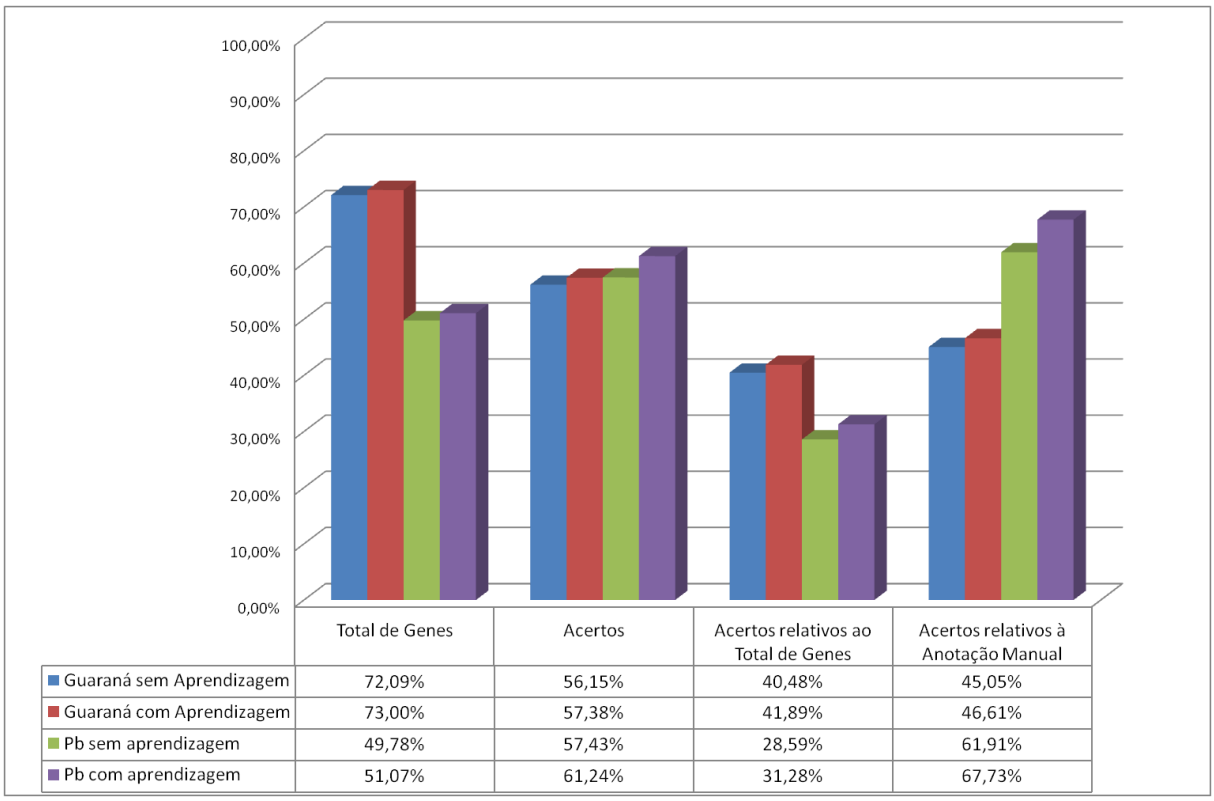
Nessa seção serão mostrados alguns gráficos comparando os resultados dos dois experimentos. Na Figura 6.9, mostramos uma comparação de todos os valores apresentadas nos resultados de ambos os experimentos. Na Figura 6.10, mostramos uma comparação das proporções relativas a quantidade total de genes dos projetos.



	Genes do Projeto	Genes manualmente anotados	Genes sugeridos para o Projeto	Sugestões consideradas corretas	ncRNAs identificados pelo BioAgents	Sugestões para genes não anotados
Guaraná sem Aprendizagem	8597	7725	6198	3480	1379	306
Guaraná com Aprendizagem	8597	7725	6276	3601	1317	367
Pb sem aprendizagem	6107	2820	3040	1746	447	533
Pb com aprendizagem	6107	2820	3119	1910	447	566

Figura 6.9: Comparação dos valores obtidos no Projeto Genoma Pb e no Projeto Genoma Guaraná.

Notamos que as porcentagens no caso do Projeto Genoma Guaraná foram menores quando comparadas ao Projeto Genoma Pb. Isto poderia ser explicado pelo fato da anotação manual do Projeto Genoma Pb ter sido feita antes do Projeto Genoma Guaraná, o que implica que os bancos de dados estavam menos atualizados. Pode-se verificar que o aumento da porcentagem de sugestões consideradas corretas relativa a anotação manual no



	Total de Genes	Acertos	Acertos relativos ao Total de Genes	Acertos relativos à Anotação Manual
Guaraná sem Aprendizagem	72,09%	56,15%	40,48%	45,05%
Guaraná com Aprendizagem	73,00%	57,38%	41,89%	46,61%
Pb sem aprendizagem	49,78%	57,43%	28,59%	61,91%
Pb com aprendizagem	51,07%	61,24%	31,28%	67,73%

Figura 6.10: Comparação das proporções obtidos no Projeto Genoma Pb e no Projeto Genoma Guaraná.

caso do Projeto Genoma Pb foi maior (5.82%), quando comparada a do Projeto Genoma Guaraná (1.56%), o que poderia ser explicado pelo fato do Projeto Genoma Pb ter menos sequências anotadas manualmente. Essa característica é muito interessante no contexto de sequenciamento de alto desempenho, que não possui a etapa de anotação manual.

Por fim, os resultados também melhoraram em comparação com os resultados apresentados por (Ralha et al. 2008). Em relação a esse último, o porcentual de acerto no Projeto Genoma Pb aumentou de 44.25% para 61.24% e em relação ao Projeto Genoma Guaraná, aumentou de 45.35% para 57.38%. Isso foi devido tanto a implementação da camada de aprendizagem quanto das modificações feitas no sistema *BioAgents*.

A comparação utilizada entre as sugestões do *BioAgents* e as anotações, que utilizou três palavras, pode ter produzido falsos positivos ou falsos negativos. Além disso, comparações mais refinadas, como usar certas classes do GO, poderiam reduzir esses falsos positivos e falsos negativos.

Apesar dos aspectos mencionados acima, nossos experimentos mostram que a incorporação de um mecanismo de aprendizagem pode aprimorar a anotação de projetos de sequenciamento de genomas.

Capítulo 7

Conclusões e Trabalhos Futuros

Neste trabalho, foi proposto um método baseado na abordagem de aprendizagem por reforço, que foi implementado no sistema *BioAgents* por meio de uma camada de aprendizagem. Foram realizados experimentos com dados reais dos Projetos Genoma Pb e Guaraná. O método de aprendizagem incorporado ao sistema usou uma pontuação baseada nos algoritmos e nas bases de dados adotados nos projetos genoma que foram usados nos experimentos. A camada de aprendizagem, dentro do *BioAgents*, consistia de dois tipos de agentes, respectivamente, Agente Controlador de Aprendizagem e Agente de Aprendizagem, que implementaram o mecanismo de aprendizagem proposto. A comparação entre os resultados obtidos pelo *BioAgents*, quando comparados com as anotações manuais desses projetos, mostram que a camada de aprendizagem melhora o desempenho.

Sugestões de trabalhos futuros para aprimorar o sistema *BioAgents* e na camada de aprendizagem são:

- Melhorias na camada de aprendizagem, por exemplo, a inclusão do uso da taxa de amortização (γ) no cálculo da pontuação das ferramentas e dos bancos de dados.
- Aplicação de algoritmos genéticos para a geração de novas regras de inferência. Esse trabalho contribuiria com a ampliação do domínio das regras de inferência do *BioAgents* que ainda hoje são limitadas.
- Execução distribuída do sistema *BioAgents*. Apesar da plataforma de agentes *JADE* permitir a execução distribuída dos agentes, essa funcionalidade não foi utilizada no *BioAgents*. Esse trabalho aumentaria o desempenho de execução da plataforma, auxiliando a camada de aprendizagem a obter resultados mais rapidamente.
- Inclusão de novos Agentes Gerentes (ferramentas) e novos Agentes Analisadores (bases de dados), desenvolvendo regras para trabalhar com informações para identificação de homologia em famílias de proteínas, como HMMER/Pfam por exemplo.

A partir dos resultados obtidos nesse trabalho, acreditamos que o sistema *BioAgents* com a camada de aprendizagem apoiará fortemente a fase de anotação no novo contexto de sequenciamento de alto desempenho, e poderá ser bastante melhorado incorporando mais conhecimento dos biólogos com mais informações de novos bancos de dados e programas.

Referências

Adams, M. D., Kelley, J. M., Gocayne, J. D., Dubnick, M., Polmeropolus, M. H., Xiao, H., Merril, C. R., Wu, A., Olde, B., Moreno, R. F., Kerlavage, A. R., Mccombie, W. R., and Venter, J. C. (1991). Complementary DNA sequencing: expressed sequence tags and human genome project. *Science*, 252:1651–1656. 21

Almeida, M. B. and Bax, M. P. (2003). Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção. *Revista Ciência da Informação*, 32(3). 29

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410. 17, 18, 22

Altschul, S. F., Schaffer, T. L. M. A. A., Zhang, J., Zhang, Z., M., W., and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402. 17, 18, 22, 42

Andrade, R. V. (2002). Caracterização parcial do transcriptoma do fungo dimórfico e patogênico *Paracoccidioides brasiliensis*. Master's thesis, Universidade de Brasília, Departamento de Biologia Molecular. 17

Arrial, R. T., Togawa, R. C., and Brigido, M. M. (2009). Screening non-coding rnas in transcriptomes from neglected species using portrait: case study of the pathogenic fungus paracoccidioides brasiliensis. *BMC Bioinformatics*, 10(239). 20

Bazzan, A. L. C., Duarte, R., Pitinga, A. N., Schroeder, L. F., and Souto, F. D. A. (2003). Atucg - an agent-based environment for automatic annotation of genomes. *Int. J. Cooperative Inf. Syst.*, 12(2):241–273. 39

Bellifemine, F., Caire, G., Poggi, A., and Rimassa, G. (2003). Jade - a white paper. White Paper 3, TILAB - Telecom Italia Lab. x, 4, 28, 42, 44, 45

Bellifemine, F., Caire, G., Trucco, T., and Rimassa, G. (2005). *Jade Programmers Guide*. TILAB, formerly CSELT and University of Parma, Italia. 4, 42, 44

Browne, P. (2009). *JBoss Drools Business Rules*. Packt Publishing. 46

Decker, K., Zheng, X., and Schmidt, C. (2001). A multi-agent system for automated genomic annotation. In *AGENTS '01: Proceedings of the* 5^{th} *international conference on Autonomous agents*, New York, NY, USA. ACM Press. 39

Dietterich, T. G., Domingos, P., Getoor, L., Muggleton, S., and Tadepalli, P. (2008). Structured machine learning: the next ten years. *Mach. Learn.*, 73(1):3–23. 31

Ding, L., Sabo, A., Berkowicz, N., Meyer, R. R., Shotland, Y., Johnson, M. R., Pepin, K. H., Wilson, R. K., and Spieth, J. (2004). Eannot: A genome annotation tool using experimental evidence. *Genome Research*, 14(12):2503–2509. 39

Eddy, S. (2002). A memory-efficient dynamic programming algorithm for optimal alignment of a sequence to an rna secondary structure. 20

Eddy, S. (2005). Infernal user's guide: Sequence analysis using profiles of rna secondary structure consensus. User guide, Howard Hughes Medical Institute and Dept. of Genetics. Saint Louis, USA. 20

Ferber, J. (1999). *Multi-Agent Systems*. Addison-Wesley, England. 28

FIPA (2006). Fipa web site. http://www.fipa.org/. x, 29, 42, 43, 45, 47, 49

Friedman-Hill, E. (2003). *Jess in Action: Rule-Based Systems in Java*. Manning Publications Co, Greenwich, CT. 4

Gibas, C. and Jambeck, P. (2001). *Developing Bioinformatics Computer Skills*. O'Reilly, Sebastopol, CA, United States. 13

Green, E. D. (2001). Strategies for the systematic sequencing of complex genomes. *Nature Reviews Genetics*, 2(8):573–583. 1

Gruber, T. R. (1993). Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In Guarino, N. and Poli, R., editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands. Kluwer Academic Publishers. 29

Kent, W. J. (2002). Blat - the blast-like alignment tool. *Genome Research*, 12(4):656–664. 18, 51

Lander, E. S., Linton, L. M., Birren, B., et al. (2001). Initial sequencing and analysis of the human genome. *Nature*, 409:860–921. 1

Lewis, S., Ashburner, M., and Reese, M. G. (2000). Annotating eukaryote genomes. *Current Opinion in Structural Biology*, 10(3):349–354. 17

Lima, R. S. (2007). Sistema multiagente para anotação manual em projetos de seqënciamento de genomas. Master's thesis, Universidade de Brasília. 40

Lima, R. S., Ralha, C. G., Walter, M. E. M. T., Schneider, H. W., Pereira, A. G. F., and Brígido, M. M. (2007). Bioagents: Um sistema multiagente para anotação manual em projetos de seqüenciamento de genomas. *Encontro Nacional de Inteligência Artificial*. x, 3, 40, 41, 42

Luscombe, N. M., Greenbaum, D., and Gerstein, M. (2001). What is bioinformatics? an introduction and overview. In HAUX, R. and KULIKOWSKI, C., editors, *Yearbook Of Medical Informatics* 2001, pages 83–100. Internation Medical Informatics Association. 13

Mardis, E. R. (2008). Next-generation dna sequencing methods. *Annual Review of Genomics and Human Genetics*, 9:387–402. x, 8, 10, 11, 12

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill Education (ISE Editions). 3, 30, 31

Pertsemlidis, A. and Fodon, III, J. W. (2001). Having a BLAST with bioinformatics (and avoiding BLASTphemy). *Genome Biology*, 2(10):1–10. 17

Ralha, C. G., Schneider, H. W., Fonseca, L. O., Walter, M. E., and Brígido, M. M. (2008). Using bioagents for supporting manual annotation on genome sequencing projects. In *BSB '08: Proceedings of the 3rd Brazilian symposium on Bioinformatics*, pages 127–139, Berlin, Heidelberg. Springer-Verlag. 4, 40, 78

Rigden, D. J. and de Mello, L. V. (2002). Computacional de proteínas. *Biotecnologia Ciência & Desenvolvimento*, 4(25):64–70. 22

Russell, S. J. and Norvig, P. (2002). *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition. 3, 26, 30, 31

Rätsch, G., Sonnenburg, S., Srinivasan, J., Witte, H., Müller, K. R., Sommer, R., and Schölkopf, B. (2007). Improving the c. elegans genome annotation using machine learning. *PLoS Computational Biology*, 3(2):e20. See also http://www.msplicer.de. 39

Santos, C. T. and Bazzan, A. L. C. (2004). Using the A3C system for annotation of keywords - a case study. *III Brazilian Workshop on Bioinformatics (WOB). Brasília, DF*. 39

Setúbal, J. and Meidanis, J. (1997). *Introduction to Computational Molecular Biology*. Brooks/Cole Publishing Company, Pacific Grove, CA, United States. 5, 9, 18

Simpson, A. J. G., Reinach, F. C., Arruda, P., and Others (2000). The genome sequence of the plant pathogen *Xylella fastidiosa*. *Nature*, 406(6792):151–157. 2

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, Estados Unidos. x, 31, 32, 33, 34, 35

Sycara, K. (1998). Multiagent systems. Technical Report 2, AI Magazine. 29

Sycara, K., Decker, K., Pannu, A., Williamson, M., and Zeng, D. (1996). Distributed intelligent agents. *IEEE Expert: Intelligent Systems and Their Applications*, 26(11):36–46. 27

Tetko, I. V., Rodchenkov, I. V., Walter, M. C., Rattei, T., and Mewes, H. W. (2008). Beyond the 'best' match: machine learning annotation of protein sequences by integration of different sources of information. *Bioinformatics*, 24(5):621–628. 39

Vasconcelos, A. T. R. (2003). The complete genome sequence of chromobacterium violaceum reveals remarkable and exploitable bacterial adaptability. *PNAS*, 1. 2

Venter, J. C., Adams, M. D., Myers, E. W., Li, P. W., and et al (2001). The sequence of the human project. *Science*, 291(16):1304–1351. 1

Watson, J. D. and Crick, F. H. (1953). Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid. *Nature*, 171:737–738. 1

Weiss, G., editor (2000). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, Cambridge, Massachusetts. 3

Whitehead, S. D. (1991). A complexity analysis of cooperative machanisms in reinforcement learning. In *AAAI-91 Proceedings*. 36

Winston, P. H. (1992). *Artificial Intelligence*. Addison-Wesley, third edition edition. 26

Wolfberg, T. G. and Landsman, D. (1997). A comparison of expressed sequence tags (ESTs) to human genomic sequences. *Nucleic Acids Research*, 25(8):1626–1632. 21

Wolfberg, T. G. and Landsman, D. (2001). Expressed sequence tags (ESTs). In Baxevanis, A. D. and Ouellette, B. F. F., editors, *Bioinformatics*, chapter 12. Wiley–Interscience, New York, NY, United States, second edition. 20, 22

Wooldridge, M. (2009). *An Introduction to Multiagent Systems*. John Wiley & Sons, LTD, England, 2nd edition. 3, 27, 28