UNIVERSIDADE FEDERAL DO PARANÁ

DANHYLO ALMEIDA RAMOS

METODOLOGIA DE BUSCA DE SIMILARIDADE DE GENES POR MATRIZ DE CO-OCORRÊNCIA

CURITIBA

2012

DANHYLO ALMEIDA RAMOS

METODOLOGIA DE BUSCA DE SIMILARIDADE DE GENES POR MATRIZ DE CO-OCORRÊNCIA

Dissertação apresentada ao Curso de Pós-Graduação em Bioinformática da Universidade Federal do Paraná, como requisito parcial para obtenção do título de Mestre em Bioinformática.

Orientadora:
Profª. Dr.ª Jeroniza Nunes Marchaukoski

Co-orientadora:
Profª. Dr.ª Maria Berenice Reynaud Steffens

Co-orientador:
Prof. Dr. Roberto Tadeu Raittz

CURITIBA

2012

Uma nova vida nasce.

AGRADECIMENTOS

A Deus, pelo fortalecimento de fé, crescimento e aprimoramento de vida.

Aos meus orientadores Profª. Dr.ª Jeroniza Nunes Marchaukoski, Profª. Dr.ª Maria Berenice Reynaud Steffens e Prof. Dr. Roberto Tadeu Raittz, pelo apoio e encorajamento.

Aos colegas de mestrado, Juliana, Leviston, Vanely, Lucas, Paula, Sérgio, Willyan, Rosa, Michellly, Rafaela e Dieval, pelas conversas e contribuições.

Ao professor Lucas, por ter sempre uma palavra amiga quando necessário.

Ao professor e amigo, Pedro Luiz de Paula Filho, tudo começou com uma conversa.

Ao professor Saulo José Benvenutti, pelos finais de semana juntos.

Aos amigos, Hector, Joana, Sara e Gabrielle, por me apoarem nos momentos difíceis.

A todos que, de forma direta ou indiretamente, contribuíram para a realização desta pesquisa, meu muito obrigado.

“Existem homens que lutam um dia e são bons;
Existem outros que lutam um ano e são melhores;
Existem aqueles que lutam muitos anos e são muito bons;
Porém, existem os que lutam toda a vida;
Estes são os imprescindíveis."
Bertolt Brechet

RESUMO

O barateamento das tecnologias de sequencimento de DNA têm resultado em um aumento expressivo de novos organismos sequenciados. No estudo destes dados cientistas utilizam o alinhamento de sequência de nucleotídeos para promover a anotação de genes ou de proteínas nos organismos recém sequenciados, comparando com banco de dados públicos de sequências. Alinhamento de sequências é um problema quando se observa o grande volume de organismos sequenciados e sequências depositadas em banco de dados mundiais, fazendo-se necessária uma otimização do processo. Esta pesquisa propõe uma Metodologia de Busca de Similaridade de Genes por Matriz de Co-ocorrência de modo a oferecer um meio que contribua na tarefa de anotação de um novo gene, fornecendo respostas rápidas e precisas. Uma base de dados contendo todos os genes dos genomas completos foi obtida do NCBI em setembro de 2010. As sequências de nucleotídeos foram processadas, avaliando-se a co-ocorrências entre as bases, extraindo valores estatísticos que sejam capazes de representar estas sequências em dados numéricos, permitindo a comparação de sequências através dessas medidas. Um banco de dados relacional é utilizado para armazenar os dados obtidos. Através de consulta SQL as sequências são pesquisadas na base de dados. Os resultados obtidos pela aplicação da metodologia foram validados comparando com as respostas do BLAST referente à mesma base de dados. Dez mil sequências de nucleotídeos foram testadas, e quando aplicada uma linha de corte de 50% de *score* relativo ao *self-score* a metodologia proposta encontra mais de 97% de respostas idênticas ao BLAST, este percentual é maior, quase totalizando 100% quando aplicada uma linha de corte de 70% a 90% de score relativo ao *self-score* nas respostas.

Palavras-chave: Alinhamento; Bioinformática; Características; Co-ocorrênia; Gene; Identidade.

ABSTRACT

The cheapening of DNA sequencing technologies have resulted in a significant increase of new organisms sequenced. In the study of these data, scientists use the alignment of nucleotide sequence to promote the annotation of genes or proteins in newly sequenced organisms, compared with public database sequences. Sequence alignment is a problem when looking at the large volume of sequenced organisms and sequences deposited in the database world, making necessary an optimization process. This research proposes a Methodology for Similarity Search for Genes Co-occurrence Matrix to offer a means to contribute in the task of annotating a new gene, providing quick and accurate answers. A database containing all the genes of complete genomes was obtained from NCBI in September 2010. The nucleotide sequences were processed by assessing the co-occurrences between the bases, extracting statistical values that are able to represent these sequences into numerical data, allowing the comparison of sequences through these measures. A relational database is used to store the data. Through SQL query the sequences are researched in the database. The results obtained by application of the methodology have been validated compared with the responses from BLAST on the same database. Ten thousand nucleotide sequences have been tested, and implemented as a cutting line scoring of 50% on the self-score the proposed method is more than 97% identical to the response BLAST, this percentage is higher, almost total of 100% when applied a line of cut from 70% to 90% of score on the self-score responses.

Keywords: Alignment, Bioinformatics, Features, Co-occurrence ; Gene; Identity.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANSI	-	American National Standards Institute
DNA	-	Ácido Desoxirribonucléico
HSP	-	High-scoring Segment Pairs
INREC	-	Indexação Recursiva
ISO	-	International Strandards Organization
JDK	-	Java Development Kit
JRE	-	Java Runtime Environment
JVM	-	Java Virtual Machine
mpb	-	Mega pares de bases
mRNA	-	Ácido Ribonucléico Mensageiro
MSP	-	Maximum Segment Pair
NCBI	-	National Center for Biotechnology Information
ORF	-	Open Read Frame
PDB	-	Protein Data Bank
RNA	-	Ácido Ribonucléico
SEQUEL	-	Structured English Query Language
SGBD	-	Sistema Gerenciador de Banco de Dados
SMA	-	Segundo momento angular
SQL	-	Structured Query Language
tRNA	-	Ácido Ribonucléico transportador

SUMÁRIO

1 INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS

A Bioinformática consiste na junção da biologia e da ciência da computação e tecnologia da informação formando um novo campo da ciência em uma única disciplina. Esta disciplina tem como objetivo permitir a descoberta de novos conhecimentos biológicos, assim como criar perspectiva global de que princípios unificadores da biologia possam ser entendidos (NCBI, 2011a).

Com o auxílio de ferramentas e técnicas de Bioinformática, um número crescente de genomas microbianos tem sido sequenciados por organizações em todo o mundo, e submetidos a procedimentos de anotação por similaridade e, eventualmente, disponibilizados publicamente. Este processo passa primeiro pela montagem dos *contigs*, pelo fechamento dos *gaps* até o termino da montagem do genoma. Em seguida entra em ação o trabalho de anotação de genes, bem como o seu papel funcional dentro do genoma (MARKOWITZ *et al.*, 2009).

Vários sistemas e recursos *on-line* de anotação de genes são indispensáveis para análise de dados genômicos. Contudo, o acesso a esses recursos *on-line* geralmente dispersos, fazem com que os cientistas muitas vezes visitem dezenas de *sites* a cada gene a ser pesquisado. Em cada *site* obtém-se uma lista de possíveis genes candidatos, geralmente uma lista extensa que necessita de análises posteriores. O processo de anotação genômica permite a obtenção das *ORFs*, e segundo Ochmam (2002), não é viável validar a capacidade de codificação de todas as regiões, de forma experimental, porém é possível realizar comparações de *ORFs* em genomas relacionados, podendo assim exibir no novo genoma as *ORFs* que codificam proteínas funcionais.

Atualmente o grande número de genomas bacterianos completos disponíveis em Banco de Dados públicos facilita a anotação das ORFs, utilizando método comparativo entre diversas espécies, avaliando o estado funcional das mesmas. As ORFs do novo genoma são comparadas com um genoma de referência, ou com um banco de dados de genes utilizando a pesquisa de similaridade do BLAST, aplicando uma linha de corte de 70% de identidade e 80% do comprimento da ORF (OCHMAM 2002).

O alinhamento de sequências biológicas em busca de similaridade é uma operação básica em Bioinformática, por servir de suporte para outros processos como, por exemplo, a determinação da estrutura das proteínas e análises filogenéticas (SETUBAL; MEIDANIS, 1997). A tarefa de alinhamento de sequências de uma forma simples consiste em colocar uma sequência sobre a outra e analisar as correspondências considerando os conceitos de similaridade e homologia. Pelo alinhamento de sequências é indicado o grau de similaridade, já a referência de homologia é uma hipótese do processo evolutivo, e não possui uma medida, assim duas sequências são homólogas somente se elas derivam de um ancestral comum (PRODOCIMI *et al.*, 2002).

Como forma de contribuir com o processo de anotação genômica, oferecendo um mecanismo que permita a busca rápida de possíveis genes candidatos, neste trabalho foi desenvolvida a Metodologia de Busca de Similaridade de Genes por Matriz de Co-ocorrência em Nucleotídeos.

1.2 JUSTIFICATIVA DO TRABALHO

Este trabalho propõe uma metodologia de busca de similaridade de genes por Matriz de Co-ocorrência em Nucleotídeos que contribua na tarefa de anotação de um novo gene, fornecendo respostas rápidas e precisas. Estas respostas poderão auxiliar na ação de priorizar os genes candidatos para anotação, no qual estão em longas listas dispersas em diversos *sites* e necessitam de teste e comparações mais detalhadas, uma vez que são crescentes as ações de sequenciamento de genomas e consequentemente a necessidade de mais agilidade no processo de anotações das regiões funcionais.

1.3 OBJETIVOS

1.3.1 OBJETIVO GERAL

Desenvolver uma estratégia computacional de classificação por identidade alternativa ao alinhamento de sequência tradicional, de modo que seja capaz de minimizar comparação das sequências de nucleotídeo em toda a base de dados e assim dinamizar a busca de genes semelhantes a um gene pesquisado.

1.3.2 OBJETIVOS ESPECÍFICOS

- Extrair características da matriz de co-ocorrência em DNA
- Armazenar as informações sobre o gene e as características extraídas em um banco de dados relacional.
- Recuperar sequências gênicas com alto grau de identidade a partir de extração de características da matriz de co-ocorrência de DNA e consultas SQL;
- Avaliar a estratégia de banco de dados no que diz respeito a tempo de processamento.

2 REVISÃO DE LITERATURA

2.1 BIOINFORMÁTICA

A Bioinformática surgiu para suprir a necessidade de análise do grande volume de dados gerados pelas novas tecnologias da biologia molecular. Para Hughey *et al.*, (2003) a Bioinformática engloba tudo, desde o armazenamento de dados até a recuperação dos testes computacionais de hipóteses biológicas. Os dados e as técnicas podem ser bastante diversificados, incluindo tarefas como encontrar genes em uma sequência de DNA, buscar semelhanças entre as sequências, prever a estrutura de proteínas entre outros. Sistemas de Bioinformática incluem multicamadas de software, hardware e soluções experimentais que reúnem uma variedade de ferramentas e métodos de analise de dados.

Três objetivos da Bioinformática são apontados por Luscombe *et al*., (2001), sendo o primeiro de organizar os dados de tal forma que permita a pesquisadores acessar as informações existentes, e enviar novos dados para armazenamento. Porém para Bioinformática ser somente um armazém de dados é uma função muito básica, e os dados contidos em um banco de dados necessitam de análises, assim o segundo objetivo é desenvolver recursos ou ferramentas capazes de auxiliarem na análise dos dados armazenados. Já o terceiro objetivo é utilizar estas ferramentas, analisar os dados e interpretar os resultados de forma biologicamente significativa.

2.2 CONCEITOS BÁSICOS DE BIOLOGIA MOLECULAR

2.2.1 ÁCIDO DESOXIRRIBONUCLÉICO

A informação referente à como, quando e onde deve ser produzido cada tipo de proteína está contida no material genético, um polímero chamado de ácido desoxirribonucleico (DNA). A estrutura do DNA consiste de duas longas fitas helicoidais que se enrolam em torno de um eixo comum, formando uma dupla hélice (WATSON; CRICK, 1953, apud LODISH *et al.*, 2005).

Em cada carbono 1’ (um linha) encontra-se ligadas bases nitrogenadas. Existem quatro tipos de bases: adenina (A), citosina (C), guanina (G), e timina (T) (FIGURA 1). Ao conjunto formado pelo grupo fosfato, açúcar e base, é dado nome nucleotídeo (GRIFFITHS, 2004).

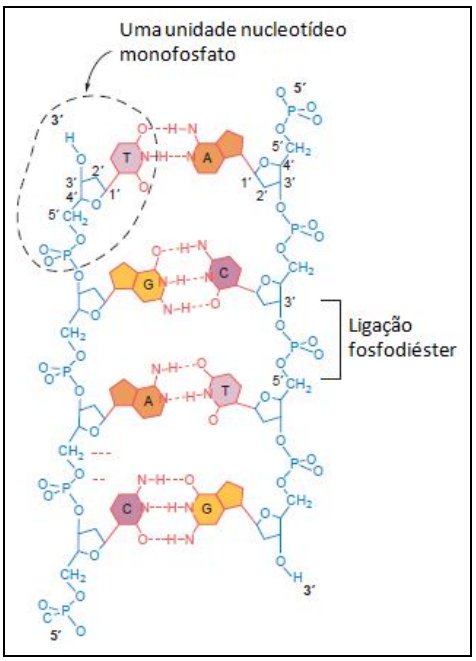


FIGURA 1 – ESTRUTURA DO DNA
Diagrama da ligação química da dupla hélice do DNA, mostrando de forma aberta da ligação de açúcar-fosfato (azul) e pares de bases (vermelho); A ligação entre cada lado da hélice ocorrer em direções opostas; As extremidades 5’ e 3' são nomeadas dessa forma devido à orientação do quinto e terceiro átomos de carbono dos anéis de açúcar; Cada par de bases tem uma base de purina, a adenina (A) ou guanina (G), e uma base de pirimidina, timina (T) ou citosina (C), ligada por pontes de hidrogênio (linhas pontilhadas).
Fonte: Adaptação de Griffiths (2004).

Os diferentes nucleotídeos são unidos por meio de suas extremidades em uma fita de DNA. Cada dupla hélice de DNA é estruturada de forma simples, pois sempre que houver um **A** sobre uma das fitas existirá um **T** sobre a outra, e cada **C** corresponderá a um **G** sobre a fita complementar.

Os nucleotídeos são compostos por uma molécula de açúcar chamada de 2’-desoxirribose ligada a resíduo fosfato. A molécula de açúcar é formada por cinco carbonos numerados de 1’ (um linha) a 5 ’ (cinco linha). A ligação entre as unidades faz-se entre o carbono 3’ (três linha) de um nucleotídeo e o resíduo fosfato e carbono 5’ (cinco linha) do nucleotídeo seguinte (FIGURA 1). Por este fato, a molécula de DNA

tem uma orientação que, por convenção começa no sentido 5’ (cinco linha) e acaba no 3’ (três linha) (LODISH *et al.*, 2005).

2.2.2 DOGMA CENTRAL DA BIOLOGIA MOLECULAR

O dogma central da Biologia Molecular enunciado por Francis Crick em 1950 (Crick, 1970) faz uma relação entre a transferência da informação sequencial do DNA para o RNA e deste para as Proteínas. Estabelece ainda que a mesma informação não possa voltar da proteína para outra proteína ou acido nucléico. De forma simplificada, a sequência de informações ocorre em três eventos: (I) Replicação, aonde a informação genética é passada para as próximas gerações através da replicação do DNA dupla fita; (II) Transcrição, que consiste na produção de uma molécula de RNA a partir de uma das cadeias do DNA dupla-fita e, (III) Tradução, evento que ocorre a partir de um RNA mensageiro e onde cada três bases do polímero compõe um códon que especifica um determinado aminoácido. O conjunto de aminoácidos ordenado de acordo com a informação genética e ligado covalentemente forma a proteína (NELSON; COX 2006).

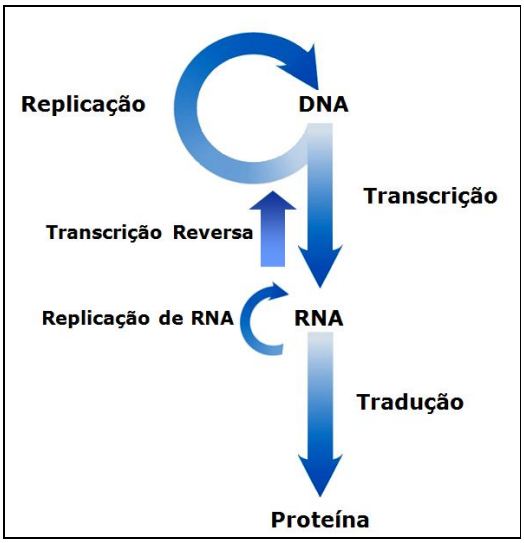


FIGURA 2 – DOGMA CENTRAL DA BIOLOGIA MOLECULAR.
A transcrição é o processo para síntese de todos os RNAs da célula, (mRNA) RNA mensageiro, (tRNA) RNA transportador, (rRNA) RNA ribossômico, (snRNA) RNA nuclear pequeno e (iRNA) RNA de interferência pequeno.
FONTE: Adaptação de Nelson e Cox (2006)

A tradução dos códons ocorre de maneira sequencial, sem sobreposição das bases, e o primeiro códon lido estabelece a pauta de leitura ou *frame*. Cada cadeia de DNA apresenta três *frames* potenciais (FIGURA 3), mas somente uma é responsável por codificar certa proteína (NELSON; COX, 2006).

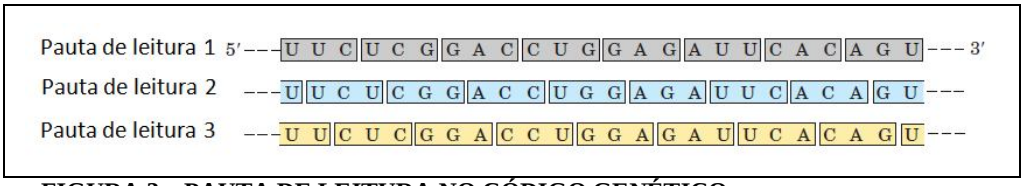


FIGURA 3 – PAUTA DE LEITURA NO CÓDIGO GENÉTICO
Em uma trinca de código não-superposto todos os mRNAs possuem três pautas de leitura potenciais. Os aminoácidos especificados em cada pauta de leitura são diferentes.
FONTE: Adaptação de Nelson e Cox (2006).

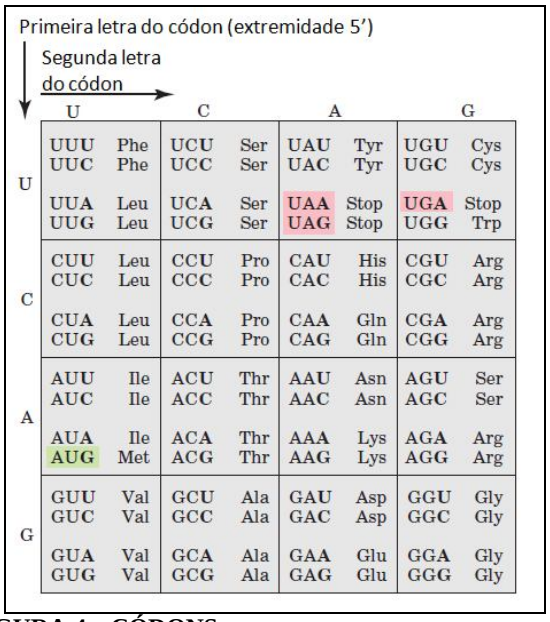


FIGURA 4 – CÓDONS.
DICIONÁRIO DAS PALAVRAS DO CÓDIGO DOS AMINOÁCIDOS NO mRNAs.
FONTE: Adaptação de Nelson e Cox (2006).

A pauta de leitura é estabelecida a partir do códon AUG e é mantida à medida que os componentes para síntese interpretam química e sequencialmente um códon após o outro. O sinal de terminação da síntese protéica é indicado por um de três códons: UAG, UAA, UGA (JUNQUEIRA; CARNEIRO, 2005).

2.2.3 ORFs

Um quadro aberto de leitura, do inglês “open reading frame”, é uma grande sequência genética de códons lida no sentido 5’ e 3’, possuindo um códon de início e um códon de termino de tradução. Uma sequência de DNA possui seis *frames* de leituras (FIGURA 5), três em cada sentido, possibilitando a busca de ORFs através de analise computacional e seleção, de uma sequência que codifique uma proteína que inicie com o códon ATG e possua um códon de terminação após cinquenta ou mais códons. As ORFs encontradas representam sequências que são possíveis genes, e necessitam de confirmação (GRIFFITHS *et al.*, 2004).

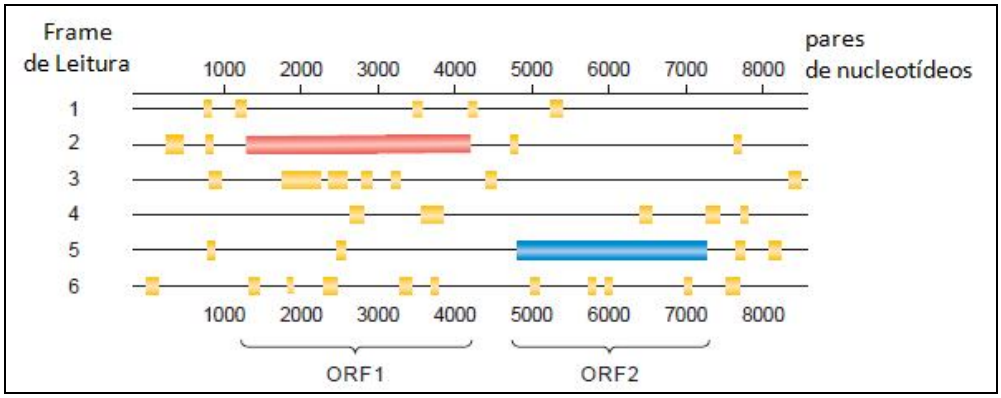


FIGURA 5 – BUSCA QUADRO ABERTO DE LEITURA
Em uma sequência de DNA há seis possíveis quadros de leitura, três em cada direção.
Fonte: Adaptação de Griffiths (2004).

2.2.4 GENOMA

Em 1929 o termo foi criado por Hans Winkler professor da Universidade de Hamburgo. O termo genoma refere-se a todo código genético contido em uma célula ou vírus, este código possui toda a informação hereditária codificada no DNA (LYNCH, 2007). Genoma também pode ser considerado a soma de genes na qual define como um ser vivo vai se desenvolver, sendo transmitido de geração em geração determinando qual é a espécie do individuo (GREGORY, 2005).

O genoma humano dispõe das informações básicas e necessárias para o desenvolvimento físico de um ser humano, e é formado pela sequência de 23 pares de cromossomos, possuindo cerca de 2.900 *mpb* de nucleotídeos, já bactérias atualmente sequenciadas, variam de 1 a 10 *mpb* de nucleotídeos (GRIFFITHS *et al.*, 2004).

2.2.5 GENES

Gene é uma unidade hereditária que possui estrutura, função e localização definidas. O gene é uma sequência particular de bases de DNA que contém informações que possam ser convertidas em produtos funcionais, como a produção de certas proteínas ou moléculas de RNA funcional quando uma celular necessitar. Existem três tipos de genes, aqueles que são apenas transcritos, os que são transcritos e traduzidos e os que não são transcritos e consequentemente não são traduzidos (NELSON; COX, 2006).

Adicionalmente as sequências que são codificadas em um gene, incluem-se outras sequências de nucleotídeos em posições adjacentes, geralmente no inicio e no fim do gene, como a região promotora onde se dá o início da transcrição e a região de término na qual é uma região onde não é traduzida, e que auxiliam para a correta expressão do gene em uma molécula de mRNA, bem como sua quantidade certa e momento correto do ciclo celular (LODISH *et al.*, 2005).

2.3 ANOTAÇÃO

Na visão de Weiss (2010), uma vez obtida as sequências genômicas é necessário agregar informação biológica a ela. Esta caracterização é chamada de anotação, em que são identificadas regiões codificadoras com base na parte estrutural (códon de início, códon de término, quantidade de GC) e regulatória (região promotora, sítio de ligação de ribossomo).

O trabalho de anotação genômica é a atividade que envolve a identificação de padrões de genes e a atribuição de funções das sequências de DNA obtidas de um sequenciamento. Sendo importante ressaltar que todas estas informações poderão estar distribuídas nos seis *frames* de leitura das sequências. Vários algoritmos que utilizam um processo estatístico são utilizados na busca das *ORFs* dentro de um genoma, localizando um códon de inicio de tradução e um códon terminador, que corresponda a uma sequência que possuía uma provável região codificadora com mais de 150 bases (MOUNT, 2004).

Para Almeida *et al.,* (2006), a anotação genômica tem a função de identificar regiões gênicas bem como descrever os seus prováveis produtos proteicos, com o auxílio de consultas realizadas a bancos de dados mundiais como Gene Onthology, no qual traz informações essenciais para a anotação funcional do genoma, tais como função molecular e processo biológico.

Com a anotação contida em um genoma é possível descrever as interações entre proteínas e demais moléculas biológicas, além de realizar inferência sobre vias metabólicas (AUDE *et al.*, 2001).

2.4 ALINHAMENTO DE SEQUÊNCIAS

A comparação de sequências é um dos problemas mais importantes para biologia computacional, devido ao grande número de sequências existentes e o volume com que elas crescem nos banco de dados mundiais. Através da comparação é possível inferir a homologia entre sequências de proteínas e obter informações a respeito de um novo gene. Portanto a atividade de alinhamento de sequências biológicas é uma operação básica para Bioinformática, que serve de suporte pra outros processos como, por exemplo, a determinação da estrutura tridimensional das proteínas e análises filogenéticas (OLAZAR, 2007).

Duas sequências são alinhadas quando colocadas uma sobre a outra de forma que os pontos correspondentes se alinhem. Partes idênticas ou semelhantes são alinhadas na mesma coluna, já partes diferentes são alinhadas com a outra sequência com o valor incompatível ou com um *gap*, com a finalidade de obter um alinhamento com o maior número possível de pontos idênticos. Sequências que podem ser facilmente alinhadas desta forma são ditas similares (MOUNT, 2004).

Existem dois tipos de alinhamento de sequências, o global e o local (FIGURA 6). No alinhamento global, é feita uma tentativa de alinhar toda a sequência, usando todos os pontos possíveis, até as extremidades de cada sequência. Sequências que são muito semelhantes e contenham aproximadamente o mesmo comprimento são ideais para realizar o alinhamento global. Já no alinhamento local, partes da sequência com

grande quantidade de pontos iguais são alinhadas, gerando um ou mais pontos de correspondência ou sub-alinhamento na sequência alinhada. (MOUNT, 2004).

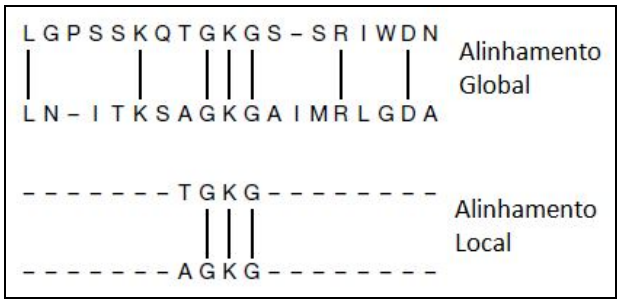


FIGURA 6 – ALINHAMENTO GLOBAL E LOCAL DE DUAS SEQUÊNCIAS.
Fonte: Adaptação de Mount (2004).

2.4.1 DISTÂNCIA DE EDIÇÃO

O termo distância de edição refere-se ao número de operações necessárias para realizar a transformação de uma sequência em outra, e envolve inserção, remoção e substituição de bases. Quanto menor é a medida da distância de edição melhor é a relação entre as duas sequências que estão sendo comparadas. A técnica é geralmente aplicada para cálculo da distância de edição entre dois textos, porém muitos outros problemas, como detecção de padrões de RNA e processamento de imagens podem ser modelados sobre esta técnica (NAVARRO, 2001).

2.4.2 IDENTIDADE

Identidade pode ser definida com a medida do número de coincidências exatas entre duas sequências, sejam elas de DNA, RNA ou aminoácidos. Descreve o grau de quanto duas ou mais sequências são idênticas em cada posição de um alinhamento, contando simplesmente o número de bases idênticas pareadas entre as sequências no alinhamento (LESK, 2008).

2.4.3 SIMILARIDADE

A similaridade é uma medida referente o quanto as sequências de aminoácidos ou nucleotídeo são parecidas, referente ao percentual de propriedades químicas semelhantes (MOUNT, 2004).

Altschul *et al.,* (1990) descrevem a similaridade entre sequências, como uma unidade de medida, em um método que atribui pontuação para inserções, exclusões e substituições de bases ou aminoácidos, ao realizar um alinhamento de duas sequências, com o objetivo de encontrar o conjunto de ação menos custoso de tais modificações. Dessa forma o alinhamento pode ser pensado como um meio de minimizar a distância evolutiva ou maximizar a similaridade entre as duas sequências comparadas. Em ambos os casos, o custo deste alinhamento é uma medida de similaridade.

2.4.4 HOMOLOGIA

A homologia correspondente à distância evolutiva entre duas sequências biológicas, estimada em função do número de eventos de mutação necessárias para explicar sua história evolutiva desde a divergência. O termo homologia relaciona-se ao estudo comparativo da evolução das espécies de forma direta, pois evolução é o processo no qual mudanças e transformações ocorrem nos seres vivos ao longo do tempo, formando novas espécies (PURVES, 2001).

Enquanto o alinhamento de sequência realiza uma medida de similaridade possuindo um caráter quantitativo, pois obtém um valor medindo a pontuação do alinhamento, a homologia é uma medida qualitativa, tratando-se de uma inferência a partir de uma análise do alinhamento (MOUNT, 2004).

2.5 BLAST

Uma das ferramentas mais usadas para comparar sequências de DNA e proteínas é a Ferramenta de Pesquisa Básica de Alinhamento Local, também conhecida com BLAST (*Basic Local Alignmnt Shearch Tool),* (ALTSCHUL *et al.*, 1990). BLAST é uma família de algoritmos de computador distribuído pelo Nacional Center for Biotechnology Information (NCBI), na forma *on-line* em seu *web site*, ou na forma de

um programa para máquina individuais. A ferramenta alinha e compara rapidamente uma sequência de DNA contra um banco de dados de sequências, tornando-se uma ferramenta fundamental para pesquisa genômica (NCBI, 2012).

Entre os algoritmos do BLAST, se destacam o BLASTP que é usado para realizar alinhamento de proteínas e o BLASTN utilizado para alinhar sequências de nucleotídeos. O BLAST funciona de maneira simples, e sua ideia central é a de que os melhores alinhamentos de sequência devam possuir segmentos que sejam pareados e este pareamento tenha um alto *score* (pontuação), esta região é conhecida como *High-scoring Segment Pairs* ou simplesmente HSP. Um segmento é uma parte de uma sequência ou uma sub-cadeia da sequência, desse modo o HSP entre duas sequências a serem alinhadas é o pareamento desses dois segmentos de mesmo tamanho (KORF *et al.*, 2003). Um exemplo de pareamento de seguimento e sua pontuação calculada a partir da matriz BLOSUM62[1] é mostrado na FIGURA 7.

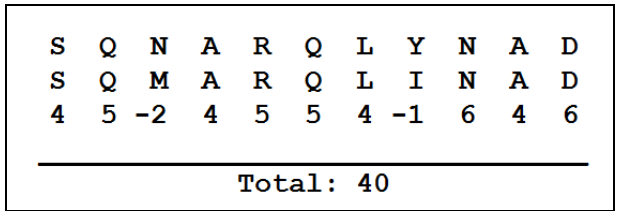


FIGURA 7 – ALINHAMENTO DE SGMENTO UTILIZANDO A MATRIZ BLOSUM62.
Fonte: O autor (2012).

Um segmento pareado máximo ou simplesmente MSP (*Maximum Segment Pair*) entre duas sequências é o segmento que apresenta a maior pontuação, e o algoritmo do BLAST, segue três passos para encontrar os MSP (SETUBAL; MEIDANIS, 1997).

1. Definido o tamanho de uma palavra como *w* e uma matriz de pontuação, cria-se uma lista (dicionário) contendo todas as palavras de tamanho *w*, aonde se alinha as palavras com a sequências de consulta, e que a pontuação seja maior que um *threshold* (*T*), dessa forma identifica-se as palavras que serão utilizadas como fonte inicial na busca de outras sequências que alinhem com a sequência de consulta no banco de dados. Estas palavras recebem a denominação de sementes (FIGURA 8).

[1] http://www.ncbi.nlm.nih.gov/Class/FieldGuide/BLOSUM62.txt

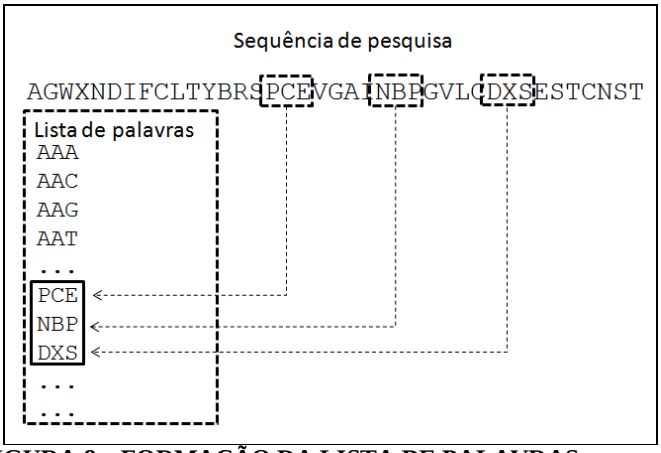


FIGURA 8 – FORMAÇÃO DA LISTA DE PALAVRAS.
Fonte: O autor (2012).

2. Com a definição das sementes na sequência de pesquisa no passo anterior, elas são agora então procuradas em cada sequência no bando de dados (FIGURA 9).

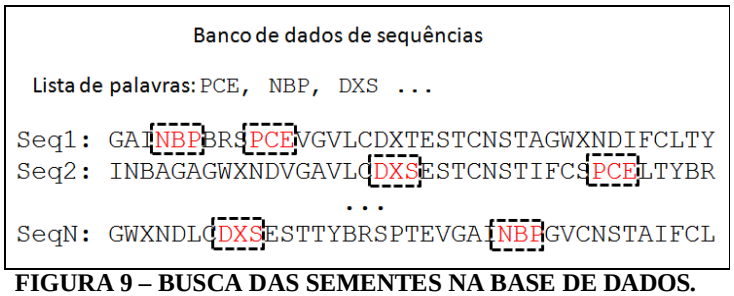


FIGURA 9 – BUSCA DAS SEMENTES NA BASE DE DADOS.
Fonte: O autor (2012).

3. Ao identificar uma semente em uma sequência na base de dados é indicado o processo de extensão do alinhamento sem a inserção de *gaps*, tentando-se alinhar a sequência encontrada com a sequência de pesquisa em ambas as direções, até que um limite inferior de pontuação seja atingido. Possibilitando assim que os melhores HSP sejam mantidos, contudo há probabilidade de que haja perda de alguma extensão importante nas sequências ao utilizar esta técnica.

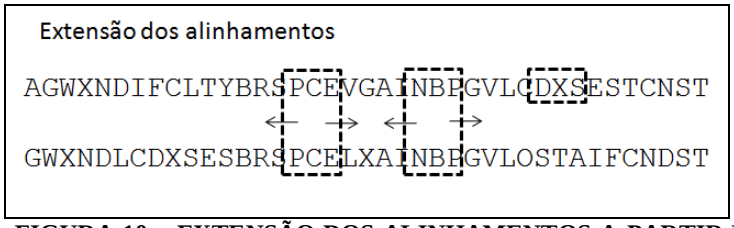


FIGURA 10 – EXTENSÃO DOS ALINHAMENTOS A PARTIR DAS SEMENTES.
Fonte: O autor (2012).

Com esta estratégia o BLAST permite identificar as sequências que possuam os maiores HSP e foca o seu esforço em estender o alinhamento das sequências minimizando assim o alinhamento em todo o banco de dados, e identificando as sequências com bons alinhamentos para sequência de pesquisa.

2.6 RECONHECIMENTO DE PADRÕES

Um padrão pode ser referenciado como uma quantidade ou descrição estrutural de um objeto ou algum outro item de interesse. Um padrão pode ser tão básico quanto um conjunto de medidas ou observações, geralmente sendo representado na forma de vetor ou matriz. O mundo pode ser visto como feito de padrões (SOUZA, 1999).

A propriedade de observar um objeto, e identificar pontos de informação e comparar estas informações com outras propriedades e comportamentos já registrados com grande rapidez, são habilidades dos serem humanos, reconhecendo assim o elemento que é observado. A simplicidade deste conceito esta presente em ambiente real, porém quando se tenta estender o reconhecimento de padrões para a linguagem computacional com a inteligência artificial, isso se torna uma tarefa complexa. Pois a utilização de máquinas e programas não é capaz de igualar-se ao cérebro humano quanto à capacidade de reconhecimento (DUDA; HART; STORK, 2001).

Para Souza (SOUZA, 1999), reconhecimento de padrões envolve uma grande quantidade de problemas de processamento de informação, na qual vai desde o reconhecimento de voz e de caracteres feitos manualmente, até a detecção de erros em equipamentos ou diagnósticos médicos. Deve-se levar em conta que as pessoas aplicam seus próprios métodos de reconhecimento de padrões em praticamente todas as áreas da atividade humana.

Segundo Kasabov (1996), as áreas aonde o reconhecimento de padrões vem sendo aplicado são:

- Análise, segmentação e pré-processamento de imagens;
- Reconhecimento de faces;
- Identificação de impressões digitais;
- Reconhecimento de caracteres;
- Análise de manuscritos;
- Visão Computacional;
- Entendimento e reconhecimento de voz;
- Diagnóstico médico.

O reconhecimento de padrões de forma artificial pode ser dividido em dois grupos, o reconhecimento de elementos abstratos e elementos concretos. Elementos abstratos são aqueles que não possuem uma forma física para ser observado, porém são observados pelos efeitos que eles produzem, como no caso da determinação de um problema. Já os elementos concretos são aqueles que podemos observar, com os objetos físicos, impressões digitais, face, voz, ou seja, tudo o que existe de forma concreta (YOUNG, 1994).

De uma forma geral três etapas de processamento são necessárias para o reconhecimento de padrões (YOUNG, 1994):

- Filtragem de entrada: Nesta etapa são eliminados os dados desnecessários, ou por algum motivo esteja distorcido, assim nesta etapa deve ficar somente os dados que sejam relevantes para o reconhecimento do elemento;
- Extração de características: A análise dos dados de entrada com a finalidade de extrair informações que possam auxiliar o reconhecimento de padrão é a função da etapa de extração de características;
- Classificação: A última etapa do reconhecimento de padrões tem a finalidade de apontar o elemento observado como sendo de uma determinada categoria ou classe, através dos dados de entrada e das análises das características.

Modelos estatísticos também podem ser utilizados para realizar a tarefa de reconhecimento de padrão, contudo esta abordagem exige um maior esforço na etapa de classificação, pois esse modelo é eficiente somente quando as suposições conseguem delimitar as classes de um objeto de forma satisfatória, e para isso uma grande quantidade de suposições avaliando diversas condições necessitam ser geradas e testadas (DEVIJVER; KITTLER, 1982).

2.6.1 EXTRAÇÃO DE CARACTERISTICAS

O método de extração de características abordado neste trabalho é a extração realizada a partir da matriz de co-ocorrência. De forma geral, o processo de extração de característica cria elementos (medidas), sejam quantitativas ou qualitativas, a partir de transformações ou combinações das características originais do objeto de estudo.

Características podem ser entendidas como qualquer medida útil extraída no processo de identificação do padrão. Intensidade de sinais são exemplos de características. As características podem ser simbólicas, numéricas ou ambas. Podem ser variáveis contínuas ou discretas (SOUZA, 1999).

Quando um algoritmo possui muitos dados para ser processado, e estes dados apresentam redundância, eles devem passar por um processo de transformação na qual seja capaz de reduzir estes dados a um conjunto de características que seja mais representativa. Este processo de transformação de dados de entrada é conhecido como extração de características.

Um método de extração de características determina um subespaço apropriado de dimensionalidade m (conjunto contendo m características) a partir de um espaço de dimensionalidade d (a cena), sendo (m <=d) (JAIN *et al.*, 2000).

2.6.2 REDUÇÃO DE DIMENSIONALIDADE

O termo dimensionalidade é utilizado para o número de características no que se refere à representação de padrões, indicando a dimensão do espaço de características. Reduzir a dimensionalidade é então à tarefa de selecionar um subconjunto do espaço de características de tal forma que não se perca a informação. Para Jain *et al.* (2000), são

três as principais razões para se utilizar a menor dimensionalidade possível. (I) custo de realizar a medição; (II) precisão do classificador; (III) com a presença somente das características mais pertinentes torna classificador mais rápido e ocupará menos memória. Ainda para Martins Junior (2004), redução de dimensionalidade é uma representação do espaço original através de um subespaço de dimensões suficientemente reduzido.

Segundo Watanabe (1985), a redução de dimensionalidade tem a sua importância para evitar um problema conhecido como mal da dimensionalidade ou fenômeno de Hughes. Este problema coloca a situação na qual é possível fazer dois padrões ficarem similar se eles forem codificados com um número suficientemente grande de características similares. Assim a ação de aumentar o número de parâmetros ocasiona a redução dos acertos no classificador como pode ser observada na FIGURA 11.

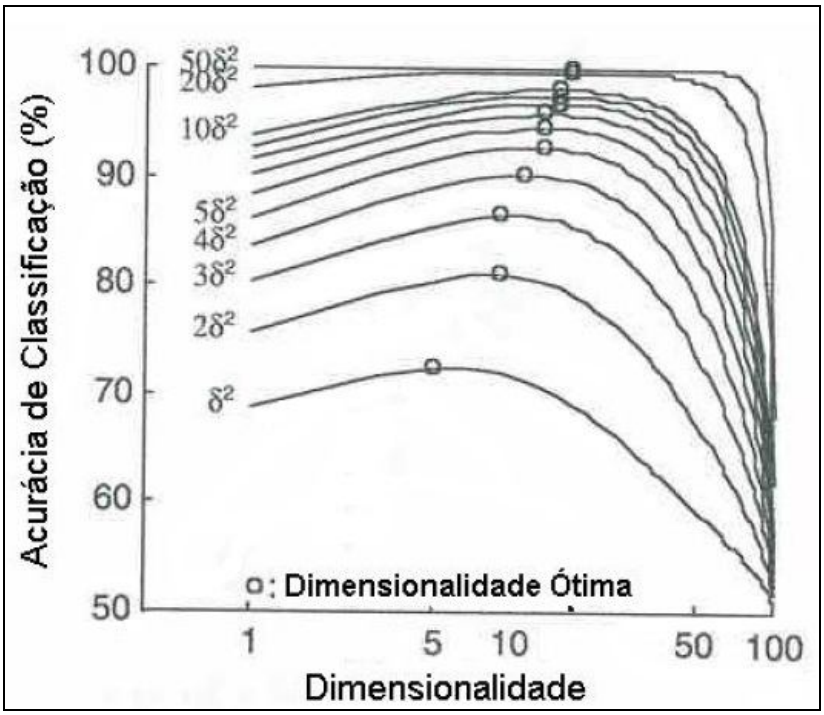


FIGURA 11 – EFEITO DE HUGHES.
FONTE: (HUGHES, 1968)

Para Souza (1999), existem três objetivos para se aplicar a redução de dimensionalidade. (I) Redução de ruídos; (II) Remoção de informações redundantes;

(III) Diminuir a quantidade de atributos para os casos onde os padrões são complexos, e as dimensões são muito grandes, como no caso de reconhecimento de padrões em imagens.

Duas abordagens para redução de dimensionalidade são propostas por Hall e Smith (1999), junção de características e seleção de características. Enquanto os algoritmos de junção de características têm a função de gerar novas características pela transformação ou combinação das características originais, os algoritmos de seleção escolhem o melhor conjunto de características segundo critérios definidos.

2.7 *OUTLIER*

Nos trabalhos detecção de *outlier* que utilizam técnicas estatísticas, não há uma definição formal aceita unanimemente, de forma simples pode-se definir *outlier* segundo Hawkins (1980) como, “um fato que desvia tanto de outros fatos a ponto de gerar suspeitas de que foi gerado por um mecanismo diferente”. *Outliers* são como pontos fora da curva, pois os mesmos não possuem a mesma distribuição de probabilidades do que a maioria das observações ou amostras. Desta forma, pode-se dizer que *outlier* são observações que estão fora do padrão normal de distribuição (TRIOLA, 1999).

Para Frota (2005) a ocorrência de *outliers* tem causas variadas, tais como:

- Fontes de ruído e erros de medidas durante a coleta dos dados;
- Falhas sistêmicas;
- Comportamento inesperado do sistema (comportamento fraudulento);
- Mudança do ponto de operação do sistema;
- Flutuações estatísticas inerentes ao conjunto de dados.

A tarefa de identificação de *outliers* tem várias aplicações, como a detecção de fraudes, detecção de intrusão, perturbações em ecossistemas, saúde púbica e medicina, (HAWKINS, 1980), pois em todos estes casos as amostras fora do padrão comum de comportamento, indicam algum problema de forma direta. Porém quando a tarefa é a análise de dados com o objetivo de reconhecimento de padrões, as amostras fora do

comportamento normal podem influenciar de forma negativa os resultados, distorcendo o modelo de classificação.

Para o processo de análise de conjunto de dados a preocupação com observações *outliers* é antiga, e inicialmente pensava-se que a melhor forma de lidar com este tipo de observações seria através da sua eliminação da análise. Porém uma análise mais criteriosa deve ser realizada sobre estes valores, é o que diz Murteira.

> "A eliminação pura e simples de um potencial *outlier* deve fazer-se com prudência e o mais aconselhável é proceder à análise com e sem a presença da respectiva observação. Se as conclusões são discordantes deve pelo menos ter-se a consistência de que o *outlier* afecta significativamente as conclusões e não há como relatar esse facto, deixando a terceiros a possibilidade de escolher o seu próprio caminho." (MURTEIRA, 1993).

Há várias técnicas para determinação de valores que são *outliers*, entre eles pode-se citar gráfico de Box, modelos de discordância, teste de dixon, teste de grubbs, z-scores, intervalo de variação, entre outros (HAWKINS, 1890). O método para detecção de *outliers* utilizado nesta pesquisa é o de intervalo de variação e sua fórmula de cálculo é apresentada na EQUAÇÃO 1.

$$\overline{X}_i - 2*dp(X_i) < X_i < \overline{X}_i + 2*dp(X_i)$$

EQUAÇÃO 1 – FÓRMULA PARA CÁLCULO DE INTERVALO DE CONFIANÇA.
FONTE: (TRIOLA, 1999).

A partir do cálculo do intervalo de variação é obtido o valor mínimo e valor máximo que as amostras devam estar em uma distribuição normal, onde $\overline{X}_i - 2*dp(X_i)$ representa o valor mínimo e $\overline{X}_i + 2*dp(X_i)$ representa o valor máximo para os seguintes dados:

- $\overline{X}_i$: É a média dos valores do atributo de toda a base;
- $2*dp(X_i)$: Duas vezes o desvio padrão dos valores do atributo de toda a base.

2.8 INREC

O modelo de redução de dimensionalidade INREC utiliza somente um número para representar um padrão, sendo uma forma simples e fácil de armazenamento e recuperação. Souza (1999) considera como objetivo da INREC a representação de infinitas informações através de apenas um número possibilitando a que padrões semelhantes possuam valores numéricos próximo.

A estratégia da INREC é ir construindo um número de forma gradual, de modo que em cada passagem da recursão ir adicionando características e ir recompondo o número. A INREC não descarta nenhuma característica, apoiada pela ideia de Pão.

> "Nenhum aspecto ou sequências de aspectos, de um padrão determina o significado daquele padrão. Ao invés disso, existe uma grande compensação entre todos os aspectos do padrão de modo que o significado de um padrão pode ser entendido se todos os aspectos estão disponíveis simultaneamente para a consideração." (PAO, 1989 apud SOUZA, 1999).

INREC é uma forma de dimensionalidade, baseada na indexação recursiva da função matemática tangente hiperbólica, para encapsular em um número apenas as informações contidas em um padrão. A função é aplicada de forma recursiva em todas as características do padrão de modo a obter um índice. Dessa forma ao indexar as variações de padrões semelhantes ou da mesma classe, obtêm índices próximos, a EQUAÇÃO 2, apresenta o algoritmo para execução da INREC.

$$r = f(x^a_1\, f(x^a_2\, f(x^a_3 \ldots f(x^a_n) \ldots)))$$

EQUAÇÃO 2 – ALGORITMO PARA GERAÇÃO DA INREC.
FONTE: SOUZA, 1999.

Onde cada item representa:

- r, índice formado pela aplicação da função de contração de forma recursiva;
- f, função de recursão tangente hiperbólica;
- x_i, as várias características que representam o padrão;

a, valor entre [1, -1], na qual sua função é promover uma padronização das características.

A INREC apresenta bons resultados, como pode ser verificado na classificação automática de dados da Íris Fisher (FISHER, (1936), apud SOUZA, 1999), em três grupos (Setosa, Virigínia e Versicolor) e a de dados cromossomos em sete grupos conhecidos como grupo de Denver (GRAHAM, (1987), apud SOUZA, 1999), mostrando que a técnica utilizando o método matemático da recursão com a função de tangente hiperbólica pode ser comparada com outros modelos para classificação de padrões, conforme pode ser observado na TABELA 1 e TABELA 2.

TABELA 1 – AVALIAÇÃO DO DESEMPENHO DO INREC PARA OS DADOS DA ÍRIS DE FISHER.

MODELO	Taxa de Acerto (%)
INREC	94,66
FAN (Free Associate Neurons)	100.00
Extração de regras de redes neurais (KT)	96,70
Árvore de Decisão	93,40

FONTE: (SOUZA, 1999).

TABELA 2 – AVALIAÇÃO DO DESEMPENHO DO INREC PARA OS DADOS DOS CROMOSSOMOS.

MODELO	Taxa de Acerto (%)
INREC	94,61
FAN (Free Associate Neurons)	95,23
Extração de regras de redes neurais (KT)	94,60
Árvore de Decisão	95,96

FONTE: (SOUZA, 1999).

Observado por Souza (1999), a INREC não oferece os melhores resultados, porém a simplicidade do método traz vantagens quando se compara em eficiência, rapidez e custo computacional.

2.9 TEXTURA

O termo textura é amplamente usado e sua noção pode ser percebida claramente, porém o mesmo não possui uma definição exata. A Textura é utilizada para explicar um padrão visual descrevendo quais são as propriedades que representam a superfície de

um objeto. De forma simples uma textura pode ser entendida com a variação de cor ou tons de cinza, seja esta variação regular ou aleatória em uma imagem (EBERT, 1994) (GOSE *et al.,* 1996),

Na literatura encontram-se diversas definições de textura. Para Sonka (1993) apud Conci *et al*. (2008), a textura pode ser definida como algo que consiste de elementos mutuamente relacionados (a primitiva de textura, que pode ser um *pixel* ou um conjunto de *pixels*). McGrogan (1997) apud Conci *et al.* (2008), define a textura como uma estrutura composta de um grande número de elementos similares mais ou menos ordenados. Haralick *et al.*, (1973) relaciona a definição de textura com o uso de coeficientes de uniformidade, densidade, aspereza, regularidade, intensidade, dentre outras características da imagem.

A textura se caracteriza pela repetição de um modelo sobre uma região. Conci *et al.* (2008) coloca que a repetição desse modelo de forma precisa, variante ou ruidosa pode ser realizada, e tamanho, formato, cor e orientação dos elementos que compõem este modelo variam em diversas regiões de forma que muitos padrões possam compor uma textura.

Haralick (1979) aponta algumas propriedades da textura, podendo elas serem classificadas com fortes quando possuem maior regularidade ou interdependência entre os *pixels* e fracas quando possuem baixa regularidade ou menor interdependência entre os *pixels* que compõem a imagem.

A análise de texturas é relacionada com o reconhecimento de padrões, de forma que possa descrever um objeto de análise a partir das características de imagem desse objeto, estabelecendo a relação de vizinhança dos elementos de textura de tal forma que se possa descrever como eles estão posicionado em relação aos demais dando um sentido de conectividade, como eles estão distribuídos de forma espacial e a sua regularidade dentro do padrão medindo a sua homogeneidade (THEODORIS, (1999) apud CONCI, 2008).

No entendimento de Conci (2008), a analise de textura é amplamente usada para:

- Segmentação: divisão de uma imagem em regiões com mesmo perfil textural;

- Descrição: extração de características baseada na quantificação de seu conteúdo de textura para discriminação entre classes de objetos;
- Classificação: rotulação de uma região com determinada textura com base em exemplos de texturas conhecidas;
- Forma: empregar a informação de textura para derivar a geometria de uma superfície tridimensional;
- Réplica: descrever uma textura visando sua reprodução.

Gonzalez e Woods (2007) descrevem que as três principais abordagens usadas para descrição de textura, a estatística, a estrutural e a espectral. Na abordagem estatística a textura é definida por um conjunto de medidas locais extraídas do padrão. A abordagem estrutural utiliza a ideia que texturas são compostas de primitivas dispostas de forma aproximadamente regular e repetitiva, de acordo com regras bem definidas. A abordagem espectral baseia-se em propriedades do espectro de Fourier, sendo principalmente utilizadas na detecção de periodicidade global, através da identificação de picos de alta energia no espectro da imagem.

Para esta pesquisa será descrito a abordagem estatística, na qual é o alvo de estudo através da matriz de co-ocorrência.

Na abordagem estatística não tenta compreender a forma hierárquica da textura, mas sim descrever a textura através de métodos estatísticos e propriedades não determinísticas, o relacionamento entre as variações de tons de cinza ou tons de cor de uma imagem.

Haralick *et al.* (1973), demonstra que o método baseado em estatísticas de segunda ordem possui alto poder de descriminação para vários tipos de imagens. A técnica utilizada por ele é a matriz de co-ocorrência na qual representa a distribuição de probabilidade da dependência espacial dos tons de cinza nos *pixels* de uma imagem.

2.10 MATRIZ DE CO-OCORRÊNCIA

Uma matriz de co-ocorrência pode ser definida com uma tabulação das diferentes combinações possíveis de valores de intensidade dos *pixels* (níveis de cinza) presentes em uma imagem. O objetivo da matriz de co-ocorrência é caracterizar a textura de uma imagem através de medidas estatística extraídas da ocorrência dos níveis de cinza em cada pixel, em diferentes direções dentro da imagem. A matriz de co-ocorrência é utilizada como ferramenta para realizar classificação de imagens, contudo o seu uso não deve ser feito de forma descriminada, pois à medida que o número de tons de cinza cresce em uma imagem o custo computacional aumenta (PEDRINI; SCHWARTZ, 2008).

A co-ocorrência na matriz pode ser definida como a frequência relativa de *P(i, j, d, q)* onde *i* e *j* são pixels vizinhos separados por uma distância *d* em uma direção *q* específica, onde a direção pode ser de 0º, 45º, 90º ou 135º, (HARALICK *et al.*, 1973). Para melhor entendimento a FIGURA 12 mostra a variação dos ângulos para construção de uma matriz de co-ocorrência.

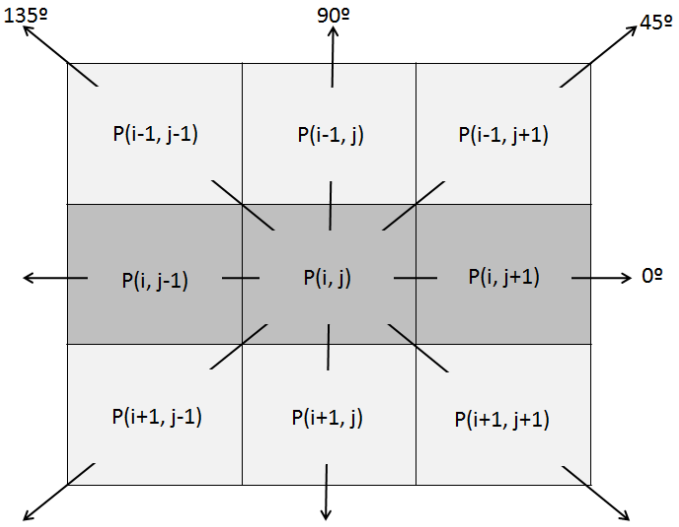


FIGURA 12 – ÂNGULOS PARA UMA MATRIZ DE CO-OCORRÊNCIA.
FONTE: Adaptação de Pedrini e Schwartz (2008).

Os ângulos de 180º, 225º, 270º e 315º não são considerados, visto que durante o processo de normalização da matriz de co-ocorrência considera-se a probabilidade de ocorrência inversa dos tons de cinza.

Toda matriz de co-ocorrência é uma matriz quadrada (CONCI *et al.*, 2008), dessa forma em uma imagem que possua três tons de cinza, representado por zero, um e dois, possui as seguintes combinações possíveis, (0,0), (0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1) e (2,2). São nove variações distribuídas em três linhas e três colunas em uma matriz (FIGURA 13).

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

FIGURA 13 –TRÊS TONS DE CINZA PARA FORMAÇÃO DA MATRIZ DE CO-OCORRÊNCIA.
FONTE: O autor (2011).

Compreendendo o conceito de *pixels* vizinhos e as possíveis combinações entre eles, bem como o conceito de distância e ângulo de leitura, a tarefa de construir uma matriz de co-ocorrência torna-se fácil. Como exemplo é possível formar uma matriz do co-ocorrência a partir de uma imagem com três tons de cinza de dimensões de 5 X 5 (FIGURA 14).

1	2	0	1	2
0	0	2	1	1
0	2	0	1	1
0	1	1	2	2
0	1	0	2	1

FIGURA 14 – IMAGEM 5X5 COM NÍVEIS DE CINZA DE 0 A 2.
FONTE: O autor (2011).

A partir da imagem é construída a matriz de co-ocorrência na qual contém o número de linhas e colunas referente à quantidade de tons de cinza. Para esta imagem,

são três os tons de cinza, 0, 1 e 2. As combinações possíveis dos tons de cinza formando a matriz é exibida na FIGURA 15.

	0	1	2
0	(0,0)	(0,1)	(0,2)
1	(1,0)	(1,1)	(1,2)
2	(2,0)	(2,1)	(2,2)

FIGURA 15 –MATRIZ DE CO-OCORÊNCIA PARA NÍVEIS DE CINZA DE 0 A 2.
FONTE: O autor (2011).

Considerando que as possibilidades de combinações de tons de cinza formam o índice das ocorrências, e uma distância $d = 1$, e um ângulo $q = 0^{o}$, lendo a imagem da esquerda para direita a co-ocorrência de zero para zero (0,0) será igual a 1. A matriz completa pode ser observada na FIGURA 16.

	0	1	2
0	1	4	3
1	1	3	3
2	2	2	1

FIGURA 16 – MATRIZ DE CO-OCORÊNCIA UMA IMAGEM 5 X 5.
FONTE: O autor (2011).

Para que se possa utilizar a matriz de co-ocorrência à mesma deve ser normalizada, sendo o processo de normalização a forma de expressar a matriz de co-ocorrência como uma probabilidade, realizando a divisão de cada elemento pela soma total dos valores, levando em consideração a probabilidade de ocorrência inversa das bases, ou seja, a leitura no ângulo aposto. A fórmula para normalização é exibida na EQUAÇÃO 3 (HARALICK *et al.*, 1973).

$$P_{i,j} = \frac{V_{i,j}}{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} V_{i,j}}$$

EQUAÇÃO 3 - NORMALIÇÃO DA MATRIZ DE CO-OCORRÊNCIA.
FONTE: (HARALICK *et al.*, 1973).

Dessa forma cada parâmetro indica:

- i representa a posição do índice da linha e j representa a posição índice da coluna;
- V representa o valor do elemento i, j na matriz de co-ocorrência;
- $P_{i,j}$ representa a probabilidade para cada elemento i, j;
- N representa o número de linhas ou colunas.

Antes de realizar a normalização devem ser atribuídos os valores da leitura no sentido inverso, sendo que estes são obtidos pela transposição da matriz de co-ocorrência, dessa forma somada as duas matrizes e aplicada fórmula de normalização, obtém-se a matriz de co-ocorrência normalizada, conforme exibido na FIGURA 17.

	0	1	2
0	1	4	3
1	1	3	3
2	2	2	1

Matriz de co-ocorrência ângulo 0º

	0	1	2
0	1	1	2
1	4	3	2
2	3	3	1

Matriz de co-ocorrência ângulo 0º (transposta)

	0	1	2
0	2	5	5
1	5	6	5
2	5	5	2

Soma das matrizes de co-ocorrência ângulo 0º e Matriz de co-ocorrência ângulo 0º (transposta)

	0	1	2
0	0,05	0,125	0,125
1	0,125	0,15	0,125
2	0,125	0,125	0,05

Matriz de co-ocorrência normalizada.

FIGURA 17 – FASES PARA NORMALIZAÇÃO DA MATRIZ DE CO-OCORRÊNCIA.
FONTE: O autor (2011).

Algumas propriedades podem ser observadas na matriz de co-ocorrência, ela é uma matriz quadrada, o número de linhas e colunas é sempre igual ao número de

variações de tons de cinza ou de cor, e a matriz de co-ocorrência é simétrica ao redor da diagonal principal.

A partir da matriz de co-ocorrência é possível calcular as características que estão presentes na seção 4.4.3 geração de características, pertencente à metodologia.

2.11 SELF-SCORE

A comparação de sequências de nucleotídeos ou proteínas em organismos iguais ou diferentes é uma ferramenta poderosa para biologia molecular. Ao encontrar semelhanças entre as sequências, através do alinhamento, os cientistas podem inferir a função de genes recém sequenciados, prever novos membros de famílias de genes, e explorar as relações evolutivas. A medida de quanto é bom um alinhamento recebe o nome de *score*, dessa forma quanto maior a pontuação, melhor é o alinhamento entre as sequências. De forma simples, o *score* é calculado a partir de uma fórmula que leva em consideração o alinhamento de partes idênticas ou semelhantes, pontuadas através de uma matriz de substituição, bem como a introdução *gaps* (NCBI, 2012b).

Já o termo *self-score*, é a pontuação obtida pelo alinhamento de uma sequência com ela mesma, indicando o máximo *score* que qualquer alinhamento com esta sequência pode alcançar (MEWES, 1997). Esta medida é útil, pois o valor de *score* do alinhamento de uma sequência contra uma sequência query que apresente até 1/3 do valor do *self-score* são consideradas sequências similares. Quando o valor é maior do que 1/3 do valor do *self-score* as sequências possuem alta similaridade (BEVAN *et al.*, 1998; BARBOSA-SILVA *et al.*, 2008; MEWES *et al.*, 1997).

Uma forma de obter a medida de *score* relativo ao *self-score é* dividir o valor do *score* do alinhamento entre sequência e a sequência *query,* com o valor do *self-score* da sequência *query* (EQUAÇÃO 4).

$$score\_relativo = \frac{score\_seq\_X\_query}{score\_query\_X\_query}$$

EQUAÇÃO 4 – CÁLCULO PARA O *SCORE* RELATIVO AO *SELF-SCORE*.
Fonte: BARBOSA-SILVA *et al.*, (2008).

2.12 SQL

O SQL (Structured Query Language) é uma linguagem estruturada de pesquisa para acessar dados em um (SGBD) Sistema Gerenciador de Banco de Dados. Essa linguagem teve seus fundamentos no modelo relacional de Codd (1970). Sua primeira versão recebeu o nome de SEQUEL (Structured English Query Language) sendo definida por CHAMBERLIN, entre outros em 1974, nos laboratórios de pesquisa da IBM (DATE 2003).

Na década de 80, o ANSI (American Nacional Standards Institute), iniciou o trabalho de padronização de uma linguagem para banco de dados relacionais. Em 1986 e 1987 respectivamente o ANSI e a ISO (International Strandards Organization), publicaram a primeira padronização do SQL (PEREIRA, 2007).

A linguagem SQL é um padrão de fato no mundo dos ambientes de banco de dados relacionais, mas pode ser adaptada a qualquer ambiente não relacional.

Vantagem do uso da Linguagem SQL:

1. Independência de fabricante é oferecida em praticamente todos os SGDBs;
2. Inglês estruturado de alto nível, formado por um conjunto bem simples de sentenças, oferecendo um rápido e fácil entendimento;
3. Consultas interativas permitem um acesso rápido aos dados, fornecendo respostas aos usuários nas questões complexas em minutos ou segundos;
4. Múltiplas visões dos Dados permitindo ao criador do banco de dados levar diferentes visões das informações a diferentes usuários (DATE, 2003).

2.12.1 SGBD - Sistema de Gestão de Banco de Dados

SGBD é formado por um grupo de programas que tem a função de abstrair do seu utilizador os detalhes de como administrar acesso, manipulação e organização dos dados contidos em um banco de dados. Dessa forma um SGBD tem a sua principal função em promover a criação e manipulação de base de dados sem a necessidade de estar atrelado a programas ou aplicações na qual o utilizam (DATE, 2003):

Um SGBD tem o seu funcionamento como uma camada de comunicação (interface), entre as aplicações ou programas com os arquivos de dados físicos,

separando as visões lógicas da estrutura real dos dados. Nesta linha são três os componentes de um SGBD:

- Linguagem de definição de dados (especifica conteúdos, estrutura a base de dados e define os elementos de dados);
- Linguagem de manipulação de dados (para poder alterar os dados na base);
- Dicionário de dados (guarda as definições de elementos de dados e respectivas características).

A arquitetura de um SGBD normalmente é baseada em três camadas principais (DATE, 2003):

1. Camada Física: Nível que armazena fisicamente os dados pertencentes à base de dados,
2. Camada Lógica: Responsável pela organização da informação em tabelas e relações,
3. Camada de Visualização: Nível referente à apresentação dos dados ao usuário através de interfaces de acesso.

2.12.2 POSTGRESQL

"O PostgreSQL é um Sistema Gerenciador de Banco de Dados (SGBD) Relacional, utilizado para armazenar informações de soluções de informática em todas as áreas de negócios existentes, bem como administrar o acesso a estas informações" (MILANI, p.25, 2008).

O PostgreSQL, surgiu na Universidade da Califórnia em Berkeley, liderado pelo professor Michael Stonebreaker. Esse SGBD, passou por um processo de desenvolvimento que durou cerca de uma década, segundo PostgreSQL (2012), e hoje dispõe suporte a quase todas as estruturas SQL e uma grande variedade de linguagens.

Quando se fala em grandes aplicações, normalmente lembra-se de tamanho em disco, ou limitação da ferramenta. No que se trata de limitação de armazenamento, os números relacionados ao PostgreSQL são muito interessantes, conforme apresenta Milani (2008), o limite dos bancos de dados do PostgreSQL dá-se pelo hardware disponível, ou seja, ilimitado.

TABELA 3 – LIMITAÇÕES DO POSTGRESQL.

Limites	Valor
Tamanho máximo da base de dados	Ilimitado
Tamanho máximo de tabela	32 TB
Tamanho máximo de linhas	1.6 TB
Tamanho máximo de campo	1 GB
Tamanho máximo de linhas por tabela	Ilimitado
Tamanho máximo de colunas por tabela	250 - 1600 dependendo do tipo da tabela
Máximo de índices por tabela	Ilimitado

Fonte: (MILANI, 2008)

> O PostgreSQL não tem limite de tamanho para seus bancos de dados, sendo a única limitação para tal critério o hardware disponível pelo computador em que o PostgreSQL está armazenando suas informações. Sua limitação dá-se em nível de tabela, com um limite máximo de 32TB por tabela. Além do mais, é possível ter registros (linhas) com até 1.6TB, campos com até um 1GB, tabelas com até 1.600 campos e índices ilimitados para aceleração da busca dos resultados. (MILANI, 2008).

Outra característica interessante a ser lembrada é sua compatibilidade com várias linguagens e sistemas operacionais. Existem já desenvolvidas bibliotecas de conexão para C/C++, Java/JSP, PHP, ASP, .NET, Perl, Python, Ruby, Tcl e Driver ODBC, entre outros, também aponta Milani (2008).

> Sobre ambientes de instalação, o PostgreSQL é uma ferramenta extremamente portável, disponibilizando instalações para diversos sistemas operacionais, como, por exemplo:
> Linux (Fedora, Core, Debian, SuSS, RedHat).
> Unix (BSD, Solaris, HP-UX, AIX).
> Max OS X Server.
> Windows (2000, 2003, XP) (MILANI, 2008)

O PostgreSQL oferece sofisticação e flexibilidade adicionais, como herança, tipos de dados, funções, restrições, gatilhos, regras e integridade de transações. Em casos mais extremos de utilização, ainda oferece suporte a cluster, que consiste em

utilizar dois ou mais computadores interligados e sincronizados para melhorar a capacidade de utilização.

2.12.2.1 Processamento de uma query consulta (query)

Quando uma *query* é enviada ao PostgreSQL, o programa terá que passar esta *query* por várias etapas até que esta seja executada, estas etapas compreendem o *parse*, re-escrita, *planner* e execução, todas as etapas (FIGURA 18) fazem acesso às tabelas para validar e executar a solicitação da query (POSTGRESQL, 2011b).

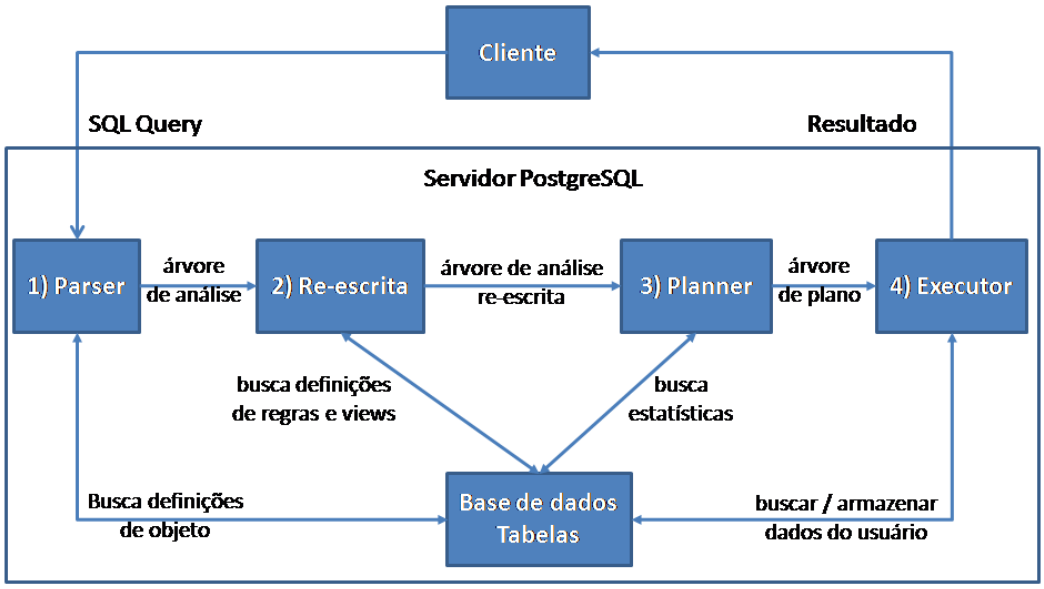


FIGURA 18 – EXECUÇÃO DE UMA QUERY.
Fonte: O autor (2011).

Parser: A fase de *parser* tem a função de validação da sintaxe da *query*, e para isso utiliza regras de gramática em linguagem C, uma vez a sintaxe estando correta é construída a *query tree* e encaminha para fase de re-escrita. Caso a sintaxe esteja incorreta será retornado erro ao cliente da query (POSTGRESQL, 2011c).

Re-escrita: A ação de *parsing* não é realizada totalmente na primeira fase, pois a correta interpretação das referências da query necessita realizar uma transação, dessa forma é interessante realizar uma primeira validação sintática para depois realizar as transações.

Na fase de re-escrita é enviada novamente a *parse tree* para fase de parse, pois a *parse tree* foi criada validando regras fixas da estrutura sintática do SQL sem olhar detalhes semânticos, dessa forma neste segundo *parser* são interpretadas as tabelas, funções e operações consultando as regras e especificações de *views* da fase anterior.

Uma nova *query tree* é construída e passada para a re-escrita, esta nova *query tree* contém uma representação de um procedimento SQL onde partes da *query* são construídas e guardadas de forma separada (POSTGRESQL, 2011d), (POSTGRESQL, 2011e).

Planner/Otimizador: Esta fase é responsável por encontrar o caminho de menor custo para acessar os dados solicitados. Este processo funciona como uma estrutura de dados contendo as ações reduzidas dos planos e o máximo de informações possíveis para o otimizador tomar a decisão do melhor caminho. Dessa forma de posse do caminho de menor custo, é construída uma árvore do plano de execução e encaminhada para fase de execução, representando assim o plano que se deseja executar.

Uma *query* pode necessitar somente de uma relação[2] ou mais, para isso o otimizador analisa cada relação e determina os planos possíveis através dos índices disponíveis em cada relação, caso seja uma junção de duas ou mais relações são consideradas as estratégias de junção de relações: Nested Loop, Merge Join e Hash Join (POSTGRESQL, 2011f).

Execução: A fase de execução é responsável por avaliar todas as operações básicas do SQL como, *select*, *insert*, *update* e *delete*. Nesta fase o executor recebe o plano de execução criado na fase anterior e o processa de forma recursiva para obter o resultado esperado pela *query*.

Cada nó da arvore do plano de execução é chamado, e devolve os resultados parciais ou indica que não há retorno, caso o nó principal seja uma junção o executor executa primeiramente os sub-planos para depois executar o nó principal e assim retornar o resultado final da *query* (POSTGRESQL, 2011g).

---

[2] Considera-se relação na fase de *Planner*: Uma tabela, *view*, *function* ou outra fonte de dados disponível para uma *query*.

2.12.2.2 Algoritmos utilizados por uma query

Algoritmos de junção

O PostgreSQL apresenta três algoritmos de junção para uma query, Nested Loop Join, Hash Join e Merge Join (DOUGLAS; DOUGLAS, 2003).

Nested Loop Join: Este algoritmo de junção realiza uma verificação dos dados à direita, para cada linha de dados à esquerda. Sendo uma estratégia de simples implementação, porém custosa para um grande volume de dados. Contudo ela pode ser melhorada quando combinada com índices em sua estratégia, assim em vez de correr a verificação na tabela de dados é percorrido os índices (FONG, 1997).

Hash Join: Junção do tipo *hash* retorna *true* para os pares de valores a esquerda e direita em que o *hash* é o mesmo em ambos os lados. Não é recomendada a utilização *hashes* para operadores que não representam igualdade (TOW, 2003).

Merge Join: Junção do tipo *merge* é baseada na ideia de primeiro ordenar, as tabelas da esquerda e da direita e depois analisá-las em paralelo. Dessa forma, ambos os tipos de dados deve ser capazes de ser totalmente ordenados, e o operador de junção deve ser aquele que só pode ter sucesso para pares de valores que se enquadram no mesmo lugar na ordem de classificação. Isso implica que este operador de junção deve se comportar como igualdade. No entanto é possível mesclar-se dois tipos distintos de dados, desde que eles sejam logicamente compatíveis (SUDARSHAN, 2006), (POSTGRESQL, 2011h).

Algoritmos de seleção

O PostgreSQL possui os seguintes algoritmos de seleção segundo Douglas e Douglas (2003):

Sequential scan: Este tipo de consulta passa por todas as tuplas[3] de uma tabela especifica, selecionando quais as tuplas que serão retornadas.

[3] Tupla, também compreendida como linha ou registro.

Index scan: Este tipo de consulta faz uso de um conjunto de índice guardado em memória, podendo ainda usar uma ou mais colunas da tabela que possuam índice para obter um acesso mais rápido as tuplas que serão retornadas.

Bitmap index scan: Este tipo de consulta utiliza índices *B-Tree* (árvore binária), criando um *bitmap* de um *bit* em memória. Dessa forma todas as condições são avaliadas usando operação de *bitwise* e o *bitmap* com valor 1 é retornado para as tuplas que são aceitas pela expressão.

2.12.2.3 Indexação

Um índice para um SGBD, permite a localização de um registro de forma mais rápida, fazendo referências associadas a uma chave, dessa forma o índice é utilizado com função de otimização.

> Índice é um recurso intimamente ligado com a otimização de um servidor de banco de dados, pois prepara algumas estruturas de organização de registros para tornar a sua busca e consulta mais rápida, precisa e eficiente (MILLANI, 2008).

Para ilustrar a importância dos índices para um banco de dados, supomos que em uma determinada base de dados não exista índices, e para encontrar um determinado registro é necessário percorrer toda a base de forma sequencial até encontrar o registro solicitado. Já com a utilização dos índices esta tarefa necessita somente de alguns passos, pois sua estrutura guarda informações da localização inicial do registro solicitado.

Os índices não beneficiam somente as operações de *select*, eles também auxiliam nas operações de *update* e *delete* bem como nas junções (*join*), podendo proporcionar mais velocidade nestas operações (PEREIRA, 2007).

Clustering: A operação de *clustering* é uma ação de ordenar fisicamente uma tabela tomando como base as informações de um determinado índice. Contudo devem ser observadas duas condições para uma boa utilização de *cluster* em uma tabela. O índice escolhido deve fazer parte da definição da tabela, e nas operações de *update* às mudanças não são refletidas para o *cluster* e necessita ser reindexado.

Índices Multi-coluna: Um índice pode ser definido através de mais de uma coluna de uma tabela.

Índices Múltiplos: É o suporte de vários índices diferentes em uma mesma tabela, atendendo os casos em que somente um índice não é suficiente para otimizar uma determinada operação.

Índices Parciais: Este tipo de índice é construído sobre parte dos dados de uma tabela através de uma expressão condicional;

Índices Únicos: A utilização deste tipo de índice esta relacionada com a ação de garantir que um determinado atributo seja único em uma tabela, quando eles forem indexados. Estes índices atende a restrição *unique*.

Índices de Expressão: Estes índices são baseados em expressões (texto), podendo ter uma ou mais colunas da tabela (DOUGLAS; DOUGLAS, 2003).

2.12.2.4 Estruturas de dados para Indexação

Várias estruturas ou algoritmos para indexação são suportados no PostgreSQL, como *B-tree, Hash, GiST* e *GIN*. Porém neste trabalho será explorado as estruturas *B-trees e Hash*.

Cada estrutura se adéqua a diferentes tipos de pesquisa e cada estrutura possui vantagens e desvantagens dependendo do caso de aplicação. Dessa forma não é possível determinar qual índice é melhor ou pior. Contudo por padrão a estrutura *B-tree* é aplicada na criação de um índice, pois este modelo melhor se adéqua a maioria dos casos (MILANI, 2008), (DOUGLAS; DOUGLAS, 2003).

B-trees: Os índices *B-tree* atuam com igualdade, e range de dados, trabalhando com os seguintes operadores <, >, =, <= ou >=. Por padrão a formação do índice é ordenada de forma ascendente e os valores *nulls* são postos no final do índice, porém é possível alterar estes valores. Em casos particulares para alguns atributos o *B-tree* é a única estrutura que permite valores únicos.

Hash: Os índices *hash* são especialistas para comparações simples como igualdade, cujo um valor obriga ser idêntico a outro. Índices de *hash* não suportam buscas *is null*. O planejador de consulta vai considerar o uso de um índice *hash* sempre que uma coluna indexada estiver envolvida e o operador = (igual) for utilizado.

2.13 JAVA

Java é uma linguagem computacional completa utilizada para programação e desenvolvimento de aplicações voltadas para internet, redes corporativas ou para máquinas individuais (CAMPIONE; WALRATH, 1996). Ela foi desenvolvida na primeira metade da década de 90 nos laboratórios da Sun Microsystes, possuindo hoje as características de suporte a orientação a objeto, código interpretado, multiplataforma e a capacidade de realizar processamento paralelo e distribuído (HORSTMANN, 2004). Java faz o uso de um interpretador para dar suporte a multiplataforma, Java Runtime Environment (JRE), na qual consiste em uma Máquina Virtual Java (JVM), diferente para cada plataforma, dessa forma os códigos escritos em Java podem ser executados em diferentes sistemas operacionais sem a necessidade de reescrever o código para cada sistema (JAVA, 2012).

3 MATERIAIS E MÉTODOS

3.1 MATERIAIS

3.1.1 BANCO DE DADOS NCBI *NUCLEOTIDE DATABASE*

Todas as sequências de nucleotídeos utilizadas nesta pesquisa foram obtidas do National Center for Biotechnology Information (NCBI). O arquivo all.ffn.tar.gz contém as sequências de DNA de genes de todos os organismos bacterianos com genoma completo depositado no NCBI, na Base de Dados *nucleotide database*. Segundo NCBI (2011b) a *nucleotide database* é uma coleção de sequências de nucleotídeos de várias fontes, incluindo GenBank, RefSeq, Anotação de Terceiros (TPA) *database* e PDB (Protein Data Bank).

O arquivo all.ffn.tar.gz esta disponível em ftp://ftp.ncbi.nlm.nih.gov/genomes/Bacteria/all.ffn.tar.gz e foi obtido em 22 de setembro de 2010.

3.1.2 SISTEMA OPERACIONAL

A plataforma de sistema operacional utilizada foi a GNU/Linux na distribuição Ubuntu 11.4 (http://www.ubuntu.com/), servindo de suporte para o desenvolvimento da metodologia, armazenagem e consulta dos resultados.

3.1.3 SERVIDOR

Foi utilizado nesta pesquisa um servidor HP Proliant ML110 G7, com as seguintes características:

- Oito Giga Byte de memória RAM 1133 MHz ECC;
- Processador Intel Xeon E31220 de 3.1GHz com quatro núcleos;
- Quatro discos Serial ATA, sendo um disco de duzentos e 250 Giga Byte para o sistema operacional e os outros três discos de um Terá Byte ligados em RAID 0 para área de armazenamento do banco de dados.

3.1.4 LINGUAGEM DE PROGRAMAÇÃO

Com o objetivo de analisar as sequências de DNA, foram desenvolvidos *scripts* para serem usados em várias etapas da metodologia bem como na integração com o BLAST durante o processo de validação. As linguagens utilizadas foram JAVA, SQL e Plpgsql.

3.1.4.1 JAVA

A linguagem Java instalada a partir do pacote JDK[4] (*Java Development Kit*) versão 7 foi utilizada para desenvolvimento dos *scripts* de manipulação das sequências de nucleotídeos.

3.1.5 POSTGRESQL

Com o objetivo de armazenar as sequências de nucleotídeo, bem como as características geradas pela metodologia foi utilizado o Sistema Gerenciador de Banco de Dados PostgreSQL na versão 8.4.

3.1.5.1 PLPGSQL

A PLPGSQL é uma extensão de linguagem estrutural da SQL, na qual tem a função de auxiliar as tarefas de programação no PostgreSQL. A PLPGSQL na versão 8.4 acrescenta à SQL características procedurais de controle de fluxo, tais como e *loops* estruturados (*for, while*) e controle de decisão (*if then else*).

[4] http://docs.oracle.com/javase/7/docs/webnotes/install/index.html

3.1.6 BLAST

Como base para validação da metodologia e comparação dos resultados obtidos pela Base de Dados, foi utilizada a ferramenta BLASTN na versão 2.2.24.

Segundo o NCBI (2012), o BLASTN é a versão do algoritmo do BLAST para encontrar sequências de nucleotídeo similares, a partir de uma tabela indexada ou de um dicionário de subsequências curtas chamadas palavras, estas palavras são então comparadas com uma base de dados. O programa pode rapidamente encontrar correspondências exatas para as palavras da consulta simplesmente olhando para uma determinada palavra no dicionário de dados. Esta comparação inicial serve como o ponto de partida para alinhamentos mais longos que são gerados em várias etapas.

Os parâmetros utilizados para execução do BLASTN foram:

- Serach Set Database: Base de Dados gerada a partir do arquivo all.ffn.tar.gz;
- Max target sequences: 20;
- Short queries: Yes
- Expect threshold: 10;
- Word size: 21;
- Match Scores: 1;
- Mismatch Scores -2;
- Gap Costs; Linear;

3.2 MÉTODOS

3.2.1 METODOLOGIA

A metodologia utilizada neste trabalho iniciou-se com a obtenção de uma Base de Dados de genes bacterianos, passando para tarefa de importar essa base para um modelo de banco de dados relacional, codificar as sequências de nucleotídeo para um formato numérico, extrair as características das sequências de genes, normalizar as características, gerar a composição na qual representará as sequências de DNA e obter respostas da Base de dados. As etapas são apresentadas no fluxograma presente na FIGURA 19 na qual são descritas na seção de resultados.

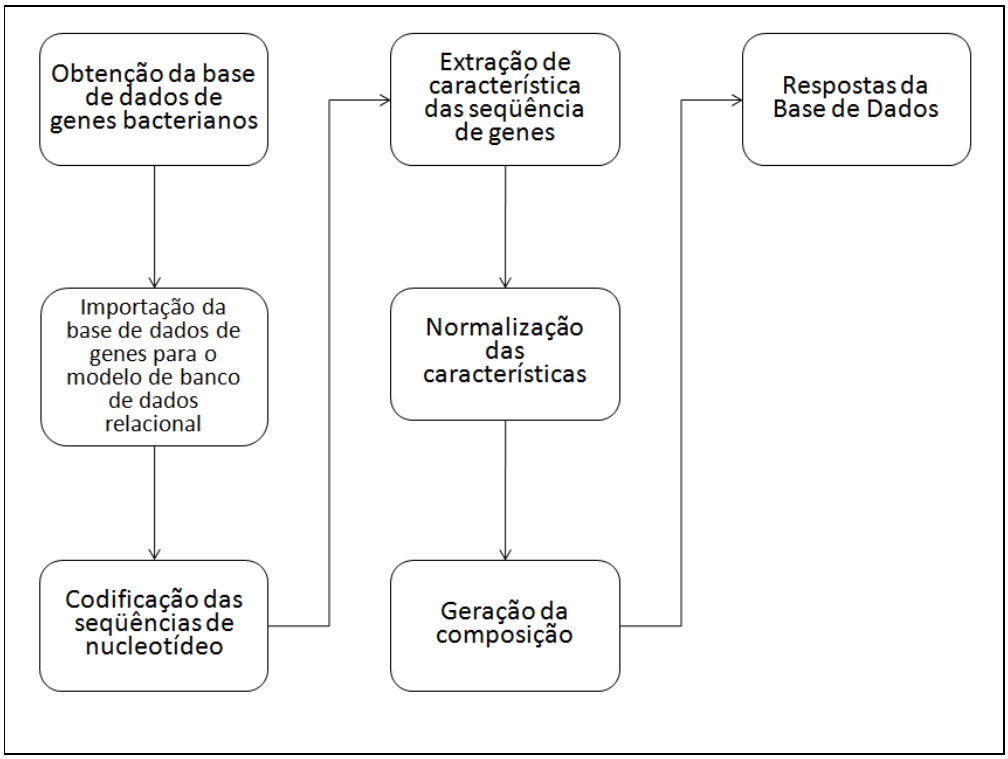


FIGURA 19 – ETAPAS DA METODOLOGIA.
FONTE: O autor (2011).

4 RESULTADOS E DISCUSSÕES

O desenvolvimento da metodologia de busca de similaridare de genes por matriz de co-ocorrência se fez seguindo as etapas presentes na metodolgia, na qual são detalahadas nesta seção.

4.1 OBTENÇÃO DA BASE DE DADOS DE GENES BACTERIANOS

A Base de Dados de genes bacterianos foi obtida através de *download* do arquivo all.ffn.tar.gz realizado na data de 22 de setembro de 2010 disponível do site ftp ftp://ftp.ncbi.nlm.nih.gov/genomes/ Bacteria/all.ffn.tar.gz.

4.2 IMPORTAÇÃO DA BASE DE DADOS DE GENES PARA O MODELO DE BANCO DE DADOS RELACIONAL

O arquivo contendo a Base de Dados dos genes bacterianos all.ffn.tar.gz contém 3.819.709 genes de todos os genomas completamente sequenciados depositados no NCBI. O arquivo contém os registros em formato fasta (FIGURA 20) onde cada conjunto de linha representa um gene. O registro contendo as informações do gene é analisado e então são separadas as seguintes informações:

1. Código de referência do organismo;
2. Posição dentro do genoma;
3. Nome do gene;
4. Nome do organismo;
5. Sequência de DNA.

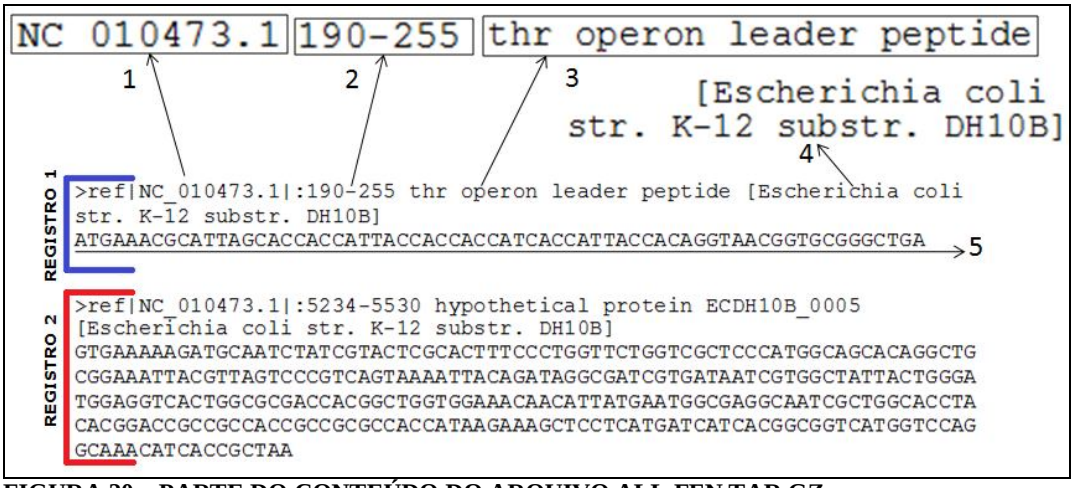


FIGURA 20 – PARTE DO CONTEÚDO DO ARQUIVO ALL.FFN.TAR.GZ
FONTE: O autor (2011).

Os dados contidos neste arquivo são então importados para um banco de dados relacional na tabela “organismo_gene” na qual pode ser observada no diagrama de entidade e relacionamento presente do APÊNDICE II e código fonte responsável pela importação no CÓDIGO FONTE 10 disponível no APÊNDICE I.

4.3 CODIFICAÇÃO DAS SEQUÊNCIAS DE NUCLEOTÍDEOS

Foram utilizadas sequências de nucleotídeo devido às características da Base de Dados obtidas. O arquivo all.ffn.tar.gz contém todas as sequências de genes dos genomas completamente sequenciados depositados no NCBI, e somente partes das sequências de genes nesta Base de Dados codificam uma proteína, assim dentro de um contexto biológico não seria adequado realizar a conversão das bases de nucleotídeo para aminoácidos, pois estaria sendo comparadas sequências de aminoácidos de uma proteína que não existe o que poderia gerar informação biologicamente incorreta.

As sequências de genes armazenadas no banco de dados estão representadas por uma cadeia de DNA contendo os nucleotídeos adenina (A), citosina (C), guanina (G) e timina (T). As sequencias de DNA foram então codificadas para um formato para aplicar a construção da matriz de co-ocorrência descrita no trabalho de Haralick (1979), onde são observadas as variações dos tons de cinza em uma imagem. Como não há tons de cinza em sequências de DNA, somente as bases de ácidos nucléicos, estas foram

convertidas para valores numéricos, para serem avaliadas segundo a sua variação entre as bases, tal como a variação de tons de cinza.

Contudo não há uma precedência entre as bases, uma forma de graduação ou uma forma de escala, para estabelecer a ordem das bases e assim atribuir um valor a elas. Dessa forma a atribuição de valores foi realizada de modo arbitrário[5], atribuindo a adenina (A) o valor 0, a citosina (C) o valor 1, a guanina (G) valor 2 e a timina (T) o valor 3. A TABELA 4 contendo os valores convertidos e CÓDIGO FONTE 1 presente no APÊNDICE I, auxiliam na compreenção.

TABELA 4 – TABELA DE CONVERSÃO DE NUCLEOTÍDEO PARA NÚMERO.

Nucleotídeo	Símbolo Nucleotídeo	Valor convertido
Adenina	A	0
Citosina	C	1
Guanina	G	2
Timina	T	3

FONTE: O autor (2011).

Durante o processo de atribuição de valores foi observado se a ordem dos valores atribuídos poderia afetar no desempenho das respostas fornecidas pelas bases. E foi observado que quando se defina a ordem de valores para as bases, esta ordem deve ser aplicada para todos os elementos armazenados bem como nas sequências na qual esta sendo pesquisada, dessa forma, a alteração não apresenta influência sobre os resultados, pois em toda a fase da metodologia foi utilizado o mesmo critério, e a variação que ocorre na matriz de co-ocorrência ocorrerá para todas as sequências, logo a classificação também se ajusta a nova atribuição (FIGURA 21).

[5] O sistema MathLab faz uma conversão semelhante pela função nt2int. Disponível em http://www.mathworks.com/help/toolbox/bioinfo/ref/nt2int.html

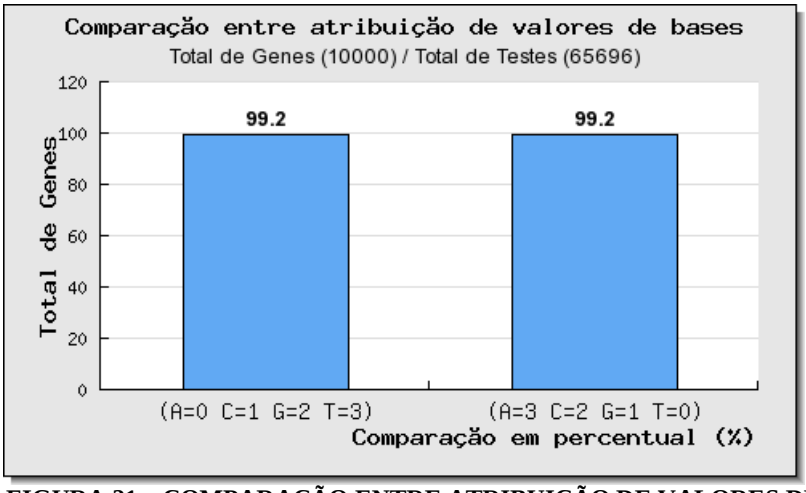


FIGURA 21 – COMPARAÇÃO ENTRE ATRIBUIÇÃO DE VALORES DE BASES.
Comparação de resultados de correspondências entre o BLAST e a Base de Dados com uma linha de corte de 70% de *score* relativo ao *self-score*, aplicado dois diferentes métodos de atribuição de valores as bases de nucleotídeo [A=0 C=1 G=2 T=3; A=3 C=2 G=1 T=0]
Fonte: O autor (2012).

4.4 EXTRAÇÃO DE CARACTERÍSTICAS DAS SEQUÊNCIAS DE GENES

A etapa de extração de característica fez uso da metodologia proposta por Haralick (1979) onde propõe uma abordagem estatística de segunda ordem, para criação de descritores capazes de medir e caracterizar textura de uma imagem. Um conjunto de 14 características de textura que poderiam ser extraídas a partir do cálculo proveniente de matrizes, mais conhecida com matriz de co-ocorrência. A construção desta matriz baseia-se em observar à alternância dos tons de cinza em uma imagem considerando uma variação de distância e direção entre os pixels vizinhos, podem ser utilizadas quatro possíveis direções, 0º, 45º, 90º e 135º.

Baraldi e Parmiggiani (1995) propõem que das quatorze medidas estatísticas baseadas na distribuição de pixel propostas originalmente por Haralick, somente seis medidas apresentam maior relevância: segundo momento angular, entropia, contraste, variância, correlação e homogeneidade, assim estas características foram utilizadas nesta pesquisa.

Para entendimento de como os descritores de textura foram utilizados, o processo de montagem da matriz de co-ocorrência para sequências de DNA, bem como a contextualização das características para o problema de pesquisa são demonstrados.

4.4.1 MATRIZES DE CO-OCORRÊNCIA METODOLOGIA

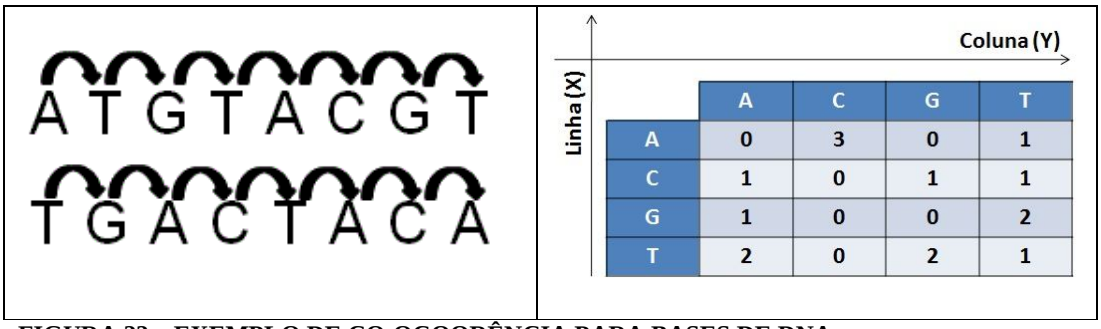
A matriz de co-ocorrência é uma tabulação das combinações possíveis entre os tons de cinza que ocorrem na imagem em uma direção específica 0º, 45º, 90º ou 135º, juntamente com a variação de distância entre pixels vizinhos (HARALICK, 1979). Porém as sequências de DNA não possuem coordenadas com múltiplas linhas e colunas, como uma imagem, possuindo somente uma linha e múltiplas colunas, referente ao tamanho da sequência, com isso limita a possibilidade de utilizar os ângulos de 45º, 90º e 135º, restando somente o ângulo de 0º referente à leitura da esquerda para direita das bases.

Como exemplo a sequência de DNA do gene 30S ribosomal protein S18 será utilizada para construção da matriz de co-ocorrência.

Sequência de DNA:

5'ATGTACGTTGACTACAAGGACCTCGAATTGCTGTCCAAGATGGTCAAC
CGTCAAGGACGCATCATGGGCCGTCGCAAAAGTGGTTGTACCGCTGCCAG
CCAACACGCTGTCACGGCTGCGATCAAACGAGCCCGCTTCATGGCCTTGT
TGCCATACGTCGGCGAATAA **3'** [Total 156 bases].

Realizando uma leitura da esquerda para direita da sequência de DNA, ou seja, ângulo de 0º, considerando uma distância entre as bases vizinhas igual a um (d=1), temos a co-ocorrência da base "A" para "T", da base "T" para "G", da base "G" para "T" sucessivamente até o fim da sequência. Esta passagem entre as bases é observada na FIGURA 22, demonstrando a passagem entre as bases e a contagem na matriz de co-ocorrência.



	A	C	G	T
A	0	3	0	1
C	1	0	1	1
G	1	0	0	2
T	2	0	2	1

FIGURA 22 – EXEMPLO DE CO-OCOORÊNCIA PARA BASES DE DNA.
FONTE: O autor (2011).

A distribuição na matriz de co-ocorrência é realizada, selecionado primeiro a posição da Linha (X) e posteriormente a Coluna (Y), dessa forma podemos observar no exemplo da FIGURA 22, que não há transição entre as bases A→A, A→G, C→C, G→C, G→G e T→C, onde cada coordenada ficou com o valor zero.

Após a geração da matriz de co-ocorrência ela é normalizada seguindo a EQUAÇÃO 3, presente na seção 2.10 MATRIZ DE CO-OCORRÊNCIA, porém antes de realizar a normalização devem ser atribuídos os valores da leitura do sentido inverso, sendo que estes são obtidos pela transposição da matriz de co-ocorrência, assim somada às duas matrizes e aplicada fórmula de normalização, obtém-se a matriz de co-ocorrência normalizada, conforme exibido na FIGURA 23 e pelos CÓDIGO FONTE 2 e CÓDIGO FONTE 3presente no APÊNDICE I.

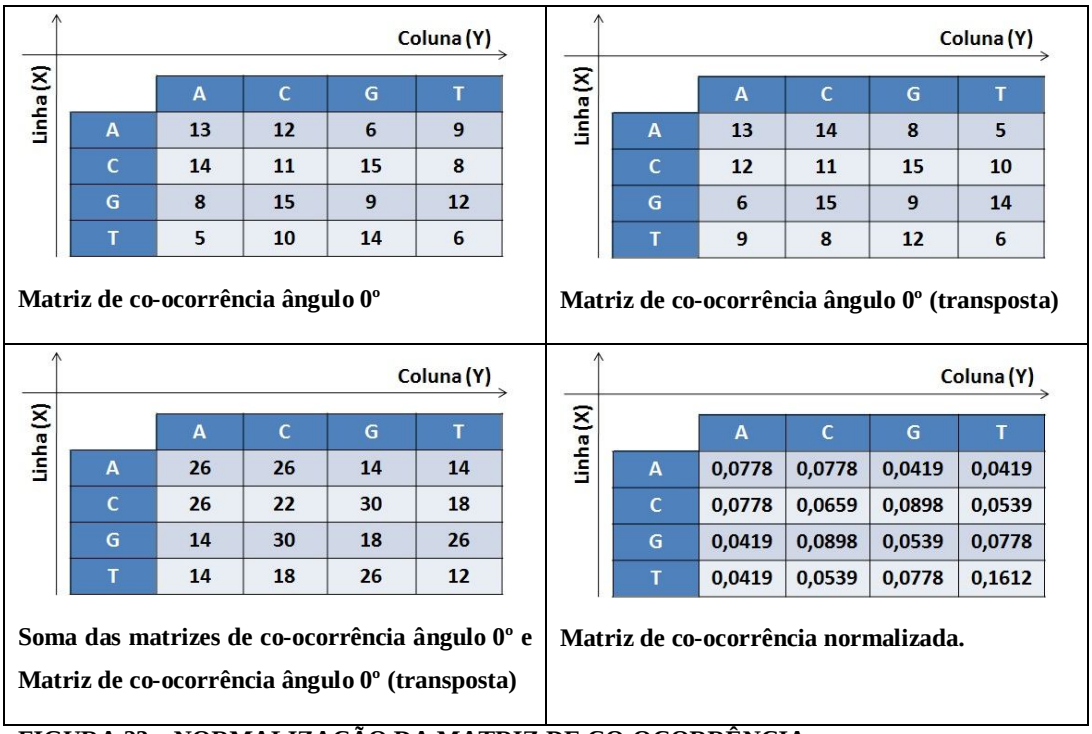


FIGURA 23 – NORMALIZAÇÃO DA MATRIZ DE CO-OCORRÊNCIA .
FONTE: O autor (2011).

4.4.2 CARACTERÍSTICAS

Segundo momento angular

Segundo momento angular (SMA) mede a uniformidade da textura, ou seja, a quantidade de repetições de pares de *pixels.* Quando houver grande repetição na variação de tons de cinza haverá valores altos para o segundo momento angular, ou seja, a matriz de co-ocorrência apresenta valores concentrados. Os valores assumidos por esta medida são positivos e menores ou iguais a 1 (PEDRINI; SCHWARTZ, 2008).

$$SMA = \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} p(i,j)^2$$

EQUAÇÃO 5 – FÓRMULA PARA CÁLCULO DO SEGUNDO MOMENTO ANGULAR. FONTE: (HARALICK *et al.*, 1973).

Contextualizando a característica quando calculada a partir de sequência de DNA, possuindo uma distribuição uniforme entre os pares de base na matriz de co-ocorrência, é então calculado um valor de SMA baixo apontando uma distribuição por igual entre as bases. A interpretação inversa é verdadeira, pois quando um valor de SMA é alto indica que há alta concentração na variação das bases, ou seja, ela foi distribuída de modo menos uniforme.

Para efeito de compreensão, serão avaliadas sequências de bases geradas para demonstrar as medidas estatísticas, através da matriz de co-ocorrência em ângulo de 0º e distância (d=1). Estas sequências não possuem função biológica e serão utilizadas somente para efeito de estudo das medidas estatísticas a partir da matriz de co-ocorrência, conforme a FIGURA 24 e FIGURA 25. O código fonte responsável pelo cálculo do SMA esta disponível no CÓDIGO FONTE 4 no APÊNDICE I.

Sequência A = [ATGGCTCAACCGAGTTA]. Valor do SMA igual a **0,0625**;

Coluna (Y) / Linha (X)

	A	C	G	T
A	0,0625	0,0625	0,0625	0,0625
C	0,0625	0,0625	0,0625	0,0625
G	0,0625	0,0625	0,0625	0,0625
T	0,0625	0,0625	0,0625	0,0625

Matriz de co-ocorrência normalizada, bem distribuída.

FIGURA 24 – EXEMPLO DE SMA BAIXO.
FONTE: O autor (2011).

Sequência B = [AAAAAAAAAAAAAAAA]. Valor do SMA igual a **1**;

Coluna (Y) / Linha (X)

	A	C	G	T
A	1	0	0	0
C	0	0	0	0
G	0	0	0	0
T	0	0	0	0

Matriz de co-ocorrência normalizada, com valor concentrado em uma posição.

FIGURA 25 – EXEMPLO DE SMA ALTO.
FONTE: O autor (2011).

Entropia

A entropia é a medida estatística na qual mede a desordem da imagem, ou seja, as muitas variações nos tons de cinza (PEDRINI; SCHWARTZ, 2008). No contexto biológico essa medida possui valores altos quando a sequência de DNA possui uma distribuição uniforme na matriz de co-ocorrência (FIGURA 26), ou seja, muita variação entre as bases. E uma entropia é baixa (FIGURA 27) quando há pouca variação de bases. A entropia está fortemente relacionada de forma inversa com o segundo momento angular, e a fórmula de calculo da entropia é vista na EQUAÇÃO 6.

$$Entropia = -\sum_{i=0}^{L-1}\sum_{j=0}^{L-1}\log(p(i,j))p(i,j)$$

EQUAÇÃO 6 – FÓRMULA PARA CÁLCULO DE ENTROPIA.
FONTE: (HARALICK *et al.*, 1973).

Sequência A = [ATGGCTCAACCGAGTTA]. Valor entropia igual a **4**;

Coluna (Y)

Linha (X)

	A	C	G	T
A	0,0625	0,0625	0,0625	0,0625
C	0,0625	0,0625	0,0625	0,0625
G	0,0625	0,0625	0,0625	0,0625
T	0,0625	0,0625	0,0625	0,0625

Matriz de co-ocorrência normalizada, bem distribuída.

FIGURA 26 – EXEMPLO DE ENTROPIA ALTA.
FONTE: O autor (2011).

Sequência B = [AAAAAAAAAAAAAAAA]. Valor entropia igual à **zero (0)**;

Coluna (Y)

Linha (X)

	A	C	G	T
A	1	0	0	0
C	0	0	0	0
G	0	0	0	0
T	0	0	0	0

Matriz de co-ocorrência normalizada, com valor concentrado em uma posição.

FIGURA 27 – EXEMPLO DE ENTROPIA BAIXA.
FONTE: O autor (2011).

O código responsável pelo cálculo da entropia é apresentado no CÓDIGO FONTE 5 no APÊNDICE I.

Contraste

Diferença espacial é a diferença entre o maior e menor valor de um conjunto de *pixels*. Assim o valor de contraste representa a distribuição espacial dos *pixels* em uma imagem, indicando um valor baixo quando há uma distribuição espacial heterogenia, mais concentrada na diagonal principal da matriz de co-ocorrência e valores altos para uma distribuição mais homogênea na matriz de co-ocorrência (PEDRINI; SCHWARTZ, 2008).

$$Contraste = \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} p(i,j)(i-j)^2$$

EQUAÇÃO 7 – FÓRMULA PARA CÁLCULO DO CONTRASTE.
FONTE: (HARALICK *et al.*, 1973).

No contexto biológico a pouca variação das bases de nucleotídeos e seguidas repetições da mesma base indicam uma distribuição heterogênea na matriz de co-ocorrência apresentando um baixo valor de contraste, bem como uma distribuição mais uniforme das bases na matriz de co-ocorrência indica uma maior variabilidade entre as bases e um alto valor de contraste.

Sequência A = [ATGGCTCAACCGAGTTA]. Valor do contraste igual a **2,5**;

Coluna (Y)

Linha (X)

	A	C	G	T
A	0,0625	0,0625	0,0625	0,0625
C	0,0625	0,0625	0,0625	0,0625
G	0,0625	0,0625	0,0625	0,0625
T	0,0625	0,0625	0,0625	0,0625

Matriz de co-ocorrência normalizada, bem distribuída.

FIGURA 28 – EXEMPLO DE CONTRASTE ALTO.
FONTE: O autor (2011).

Sequência B = [AAAACCCCGGGGTTTTT]. Valor do contraste igual a **0,1875**;

Coluna (Y)

Linha (X)

	A	C	G	T
A	0,0352	0,0010	0	0
C	0,0010	0,0352	0,0010	0
G	0	0,0010	0,0352	0,0010
T	0	0	0,0010	0,0625

Matriz de co-ocorrência normalizada, com valor concentrado na diagonal principal.

FIGURA 29 – EXEMPLO DE CONTRASTE BAIXO.
FONTE: O autor (2011).

O código utilizado para o cálculo do contraste esta disponível no CÓDIGO FONTE 6 no APÊNDICE I.

Variância

Para Pedrini e Schwartz (2008) a variância mede a heterogeneidade da textura de uma imagem. Quando os valores de tons de cinza diferem de sua média o valor de variância é alto.

A medida de variância pode ser calculada por posição da matriz de co-ocorrência como pode ser observado na EQUAÇÃO 8. Porém na metodologia busca-se uma medida única, dessa forma são somadas todas as variâncias de cada posição da matriz de co-ocorrência, conforme CÓDIGO FONTE 7 disponível no APÊNDICE I.

$$\text{variância de}_i = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} p(i,j)(i-\mu_i)^2$$

$$\text{variância de}_j = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} p(i,j)(j-\mu_j)^2$$

EQUAÇÃO 8 – FÓRMULA PARA CÁLCULO DA VARIÂNCIA.
FONTE: (HARALICK *et al.*, 1973).

No contexto biológico a medida de variância expressa o quanto a sequência de DNA é heterogenia em relação à distribuição das bases, de tal forma que quanto maior for à distribuição das bases maior será a variância (FIGURA 30).

Sequência A = [ATGGCTCAACCGAGTTA]. Valor da variância igual a **2,5**;

Coluna (Y) / Linha (X)

	A	C	G	T
A	0,0625	0,0625	0,0625	0,0625
C	0,0625	0,0625	0,0625	0,0625
G	0,0625	0,0625	0,0625	0,0625
T	0,0625	0,0625	0,0625	0,0625

Matriz de co-ocorrência normalizada, bem distribuída.

FIGURA 30 – EXEMPLO DE VARIÂNCIA ALTA.
FONTE: O autor (2011).

Sequência B = [AAAAAAAAAAAAAAA]. Valor da variância igual à **zero (0)**;

Coluna (Y)

Linha (X)

	A	C	G	T
A	1	0	0	0
C	0	0	0	0
G	0	0	0	0
T	0	0	0	0

Matriz de co-ocorrência normalizada, com valor concentrado em uma posição.

FIGURA 31 – EXEMPLO DE VARIÂNCIA BAIXA.
FONTE: O autor (2011).

Correlação

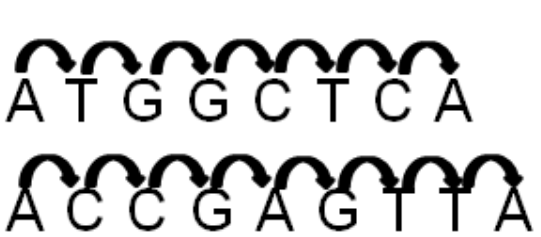
A correlação para análise de imagem mede a dependência linear do tom de cinza de um *pixel*, em relação a seu vizinho. Mostrando que valores altos de correlação indicam alta relação no nível de cinza dos pares de *pixel* (PEDRINI; SCHWARTZ, 2008).

$$Correlação = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} p(i,j)\left[\frac{(i-\mu_i)(j-\mu_j)}{(s_i)(s_j)}\right]$$

EQUAÇÃO 9 – FÓRMULA PARA CÁLCULO DA CORRELAÇÃO.
FONTE: (HARALICK *et al.*, 1973).

Em relação às sequências de DNA, a correlação mede a dependência de uma base de nucleotídeo em relação à outra, mostrando valores altos quanto maior for a dependências entre as bases (FIGURA 33), e valores baixos quando a dependência entre as bases for pequena (FIGURA 32).

Sequência A = [ATGGCTCAACCGAGTTA]. Valor da correlação igual à **zero (0)**;



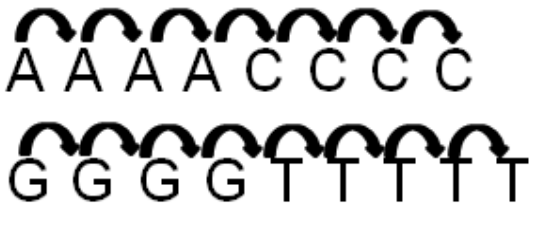
Coluna (Y) / Linha (X)

	A	C	G	T
A	0,0625	0,0625	0,0625	0,0625
C	0,0625	0,0625	0,0625	0,0625
G	0,0625	0,0625	0,0625	0,0625
T	0,0625	0,0625	0,0625	0,0625

Matriz de co-ocorrência normalizada, bem distribuída.

FIGURA 32 – EXEMPLO DE CORRELAÇÃO BAIXA.
FONTE: O autor (2011).

Sequência B = [AAAACCCCGGGGTTTTT]. Valor da correlação igual a **0,9245**;



Coluna (Y) / Linha (X)

	A	C	G	T
A	0,0352	0,0010	0	0
C	0,0010	0,0352	0,0010	0
G	0	0,0010	0,0352	0,0010
T	0	0	0,0010	0,0625

Matriz de co-ocorrência normalizada, com valor concentrado na diagonal principal.

FIGURA 33 – EXEMPLO DE CORRELAÇÃO ALTA.
FONTE: O autor (2011).

O código para cálculo da correlação é apresentado no CÓDIGO FONTE 8 disponível no APÊNDICE I.

Homogeneidade

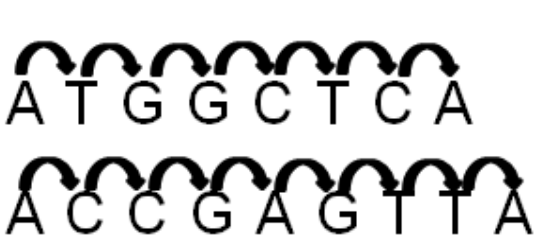
Homogeneidade ou momento de diferença inversa é à medida que expressa como a textura da imagem é homogenia, ou seja, possui pouca ou muita variação. Ela é inversamente relacionada ao contraste e ao SMA (PEDRINI; SCHWARTZ, 2008).

$$Homogeneidade = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1}\frac{p(i,j)}{1+(i-j)^2}$$

EQUAÇÃO 10 – FÓRMULA PARA CÁLCULO DA HOMOGENEIDADE
FONTE: (HARALICK *et al.*, 1973).

Adaptando para sequências de DNA, esta medida mostra o quanto às bases variam dentro da sequência, assim a repetição sequencial da mesma base eleva o valor de homogeneidade (FIGURA 34), e a dispersão das bases diminui o valor de homogeneidade (FIGURA 35).

Sequência A = [ATGGCTCAACCGAGTTA]. Valor de homogeneidade igual a **0,5**;



Coluna (Y) / Linha (X)

	A	C	G	T
A	0,0625	0,0625	0,0625	0,0625
C	0,0625	0,0625	0,0625	0,0625
G	0,0625	0,0625	0,0625	0,0625
T	0,0625	0,0625	0,0625	0,0625

Matriz de co-ocorrência normalizada, bem distribuída.

FIGURA 34 – EXEMPLO DE HOMOGENEIDADE BAIXA.
FONTE: O autor (2011).

Sequência B = [AAAAAAAAAAAAAAAA]. Valor da variância igual a **1**;

Coluna (Y) / Linha (X)

	A	C	G	T
A	1	0	0	0
C	0	0	0	0
G	0	0	0	0
T	0	0	0	0

Matriz de co-ocorrência normalizada, com valor concentrado em uma posição.

FIGURA 35 – EXEMPLO DE HOMOGENEIDADE ALTA.
FONTE: O autor (2011).

O código para cálculo da homogeneidade é apresentado no CÓDIGO FONTE 9 disponível APÊNDICE I.

Com objetivo de comparação entre as características, elas são aprsentadas na TABELA 5.

TABELA 5 – COMPARATIVO ENTRE AS CARACTERISTICAS DA MATRIZ DE CO-OCORRÊNCIA

CARACTERISTICA	VALOR ALTO	VALOR BAIXO	MEDIDA
SMA	Alta concentração das bases.	Distribuição uniforme entre as bases.	Mede a uniformidade das bases, (quantidade de repetições de pares).
Entropia	Distribuição uniforme entre as bases.	Alta concentração das bases.	Mede a desordem das bases, (variações entre as bases).
Contraste	Distribuição homogenia; Maior variabilidade.	Distribuição heterogenia; Pouca variação das bases.	Distribuição espacial (diferença entre o maior e menor valor de um conjunto de bases).
Variância	Distribuição uniforme entre as bases.	Alta concentração das bases.	Mede a heterogeneidade das bases (expressa o quanto a sequência de DNA é heterogenia em relação às bases).
Correlação	Baixa variação das bases; Alta dependência entre as bases.	Alta variação das bases; Baixa dependência entre as bases.	Mede a dependência de uma base em relação a outra.
Homogeneidade	Repetição seqüencial das bases.	Disperção entre as bases.	Medida que expressa como as bases de uma sequência são homogêneas (possui pouca ou muita variação)

FONTE: O autor (2012)

4.4.3 GERAÇÃO DAS CARACTERÍSTICAS

O processo de geração das características das sequências de genes analisa todas as sequências de DNA da Base de Dados em três pontos, sendo um ponto no início, o segundo ponto no meio e o último ponto no fim da sequência. Cada ponto deve possuir 200 bases, de forma que as sequências de DNA de um gene sejam divididas em três segmentos (FIGURA 36).

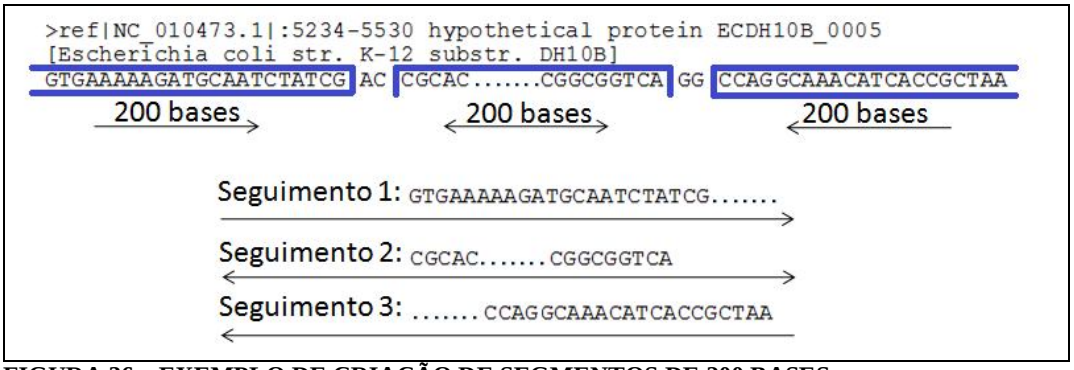


FIGURA 36 – EXEMPLO DE CRIAÇÃO DE SEGMENTOS DE 200 BASES
FONTE: O autor (2011).

Para os casos onde o gene não possua 200 bases, é considerada a quantidade de bases existentes para o segmento de inicio, meio e fim, de forma que cada parte possua a mesma sequência de DNA. Quando o gene possuir entre 201 e 599 bases, haverá pontos de sobreposição, onde parte das bases poderá participar dos segmentos do início, meio e fim. Para os demais casos cada segmento contém parte da sequência de DNA sem repetição.

Durante o processo de construção da metodologia, foram realizados vários outros experimentos, e tomando como referência os 10.000 genes testados com o BLAST e o volume de 185.039 respostas, foi observado que ao realizar o processamento de toda a sequência havia uma taxa de acerto de aproximadamente 7,9%. A partir desta observação foram analisadas as correspondências em outras três fatias das sequências, quatrocentas, duzentas e cem ultimas bases, onde se verificou que o processamento das duzentas últimas bases retorna uma quantidade maior de correspondências, correspondendo a 17.9% de respostas idênticas entre o BLAST e a Base de Dados, (FIGURA 37).

Como houve um significativo aumento na taxa de acerto analisando somente as duzentas ultimas bases, e observando que as respostas não contempladas pela Base de Dados havia forte alinhamento na parte inicial e central, foi então aplicado o processamento em três segmentos de duzentas bases para a região do início, meio e fim da sequência.

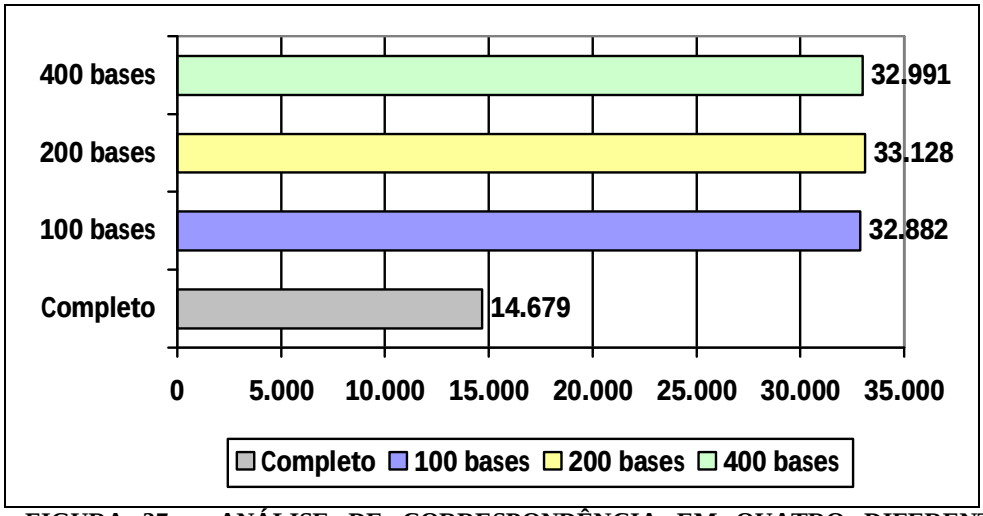


FIGURA 37 – ANÁLISE DE CORRESPONDÊNCIA EM QUATRO DIFERENTES TAMANHOS.
Comparação entre o BLAST e a Base de Dados para processamento da sequência em quatros tamanhos: completo, quatrocentas últimas bases, duzentas últimas bases, cem últimas bases.
Fonte: O autor (2012).

Com a formação dos segmentos que correspondem às três áreas de observação do gene, são processadas as sequências de modo que forme várias janelas de sobreposição do tamanho de 21 bases, variando em uma (1) em uma (1), cobrindo assim todo o segmento, totalizando 179 janelas de sobreposição quando o tamanho do segmento for de duzentas bases, (FIGURA 38, CÓDIGO FONTE 11 APÊNDICE I).

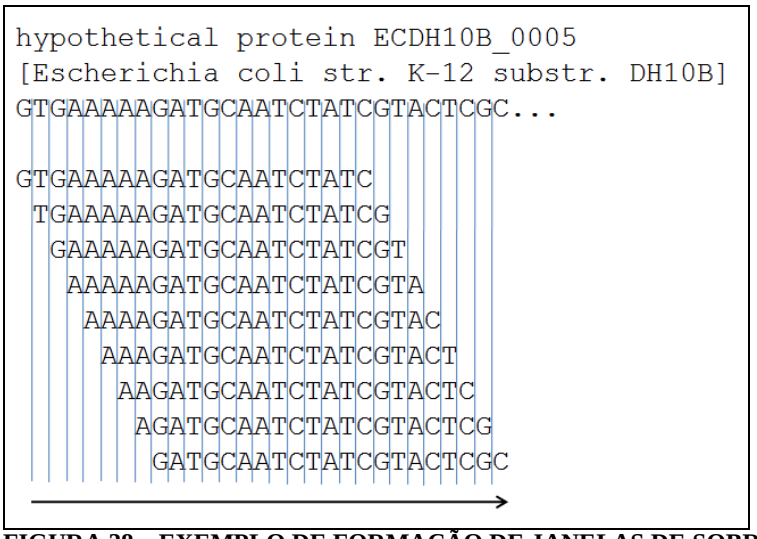


FIGURA 38 – EXEMPLO DE FORMAÇÃO DE JANELAS DE SOBREPOSIÇÃO.
FONTE: O autor (2011).

O tamanho de 21 bases nas janelas móveis foi determinado de forma experimental avaliando os resultados das respostas para janelas de tamanhos doze, dezoito, vinte e um, vinte e quatro e trinta bases. Inicialmente os testes com os vários tamanhos de janelas variavam em seis bases, ao chegar ao tamanho de vinte e quatro bases, observou-se um declínio no número de respostas correspondentes, indicando que o aumento no número de bases tornava o resultado muito específico, pois identifica somente as sequências muito semelhantes. Assim uma quantidade intermediária entre dezoito e vinte e quatro bases foi aplicada, sendo vinte e uma bases o tamanho que apresentou o melhor desempenho. A FIGURA 39 demonstra a evolução dos testes com a variação de tamanhos das janelas.

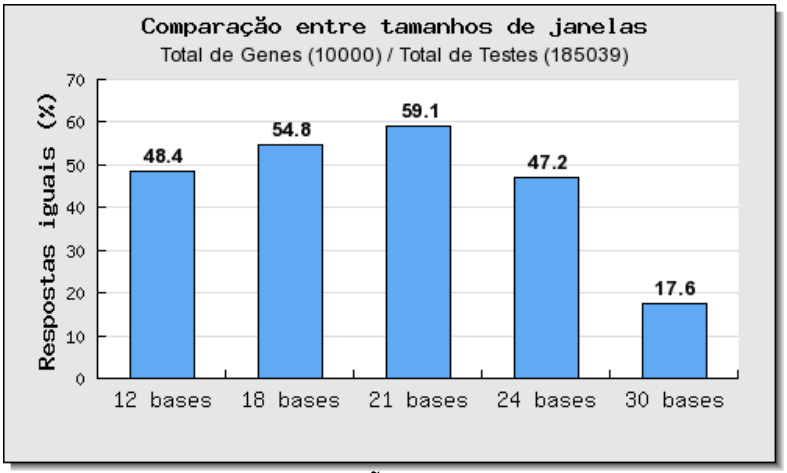


FIGURA 39 – COMPARAÇÃO ENTRE TAMANHOS DE JANELAS DE SEQUÊNCIAS.
Fonte: O autor (2012).

As janelas ou sementes serviram como fonte para geração das características e composição, de modo que a medida gerada a partir da semente de um determinado gene, seja idêntica a medida gerada pela semente de outro gene, identificando assim pontos de semelhanças entre os sequências dos genes.

As características de SMA, entropia, contraste, variância, correlação e homogeneidade são geradas para cada uma das sementes presentes nos três segmentos e armazenadas na tabela “organismo_gene_semente” (CÓDIGO FONTE 11 APÊNDICE I) para que seja formada a composição única das seis (6) características pertencente a cada semente.

4.5 NORMALIZAÇÃO DAS CARACTERÍSTICAS

O processo de normalização permite que valores de critérios diferentes, ou em escala diferentes, possam ser comparados entre si. De modo que as seis características geradas possuem escalas diferentes, elas necessitam ser normalizadas para que sejam comparadas de forma conjuntas. O método utilizado para normalização das características baseou-se no modelo de normalização severa, onde as características são re-escaladas para o intervalo de zero e um [0; 1] (PRINCIPE *et al.*, 2000).

$$x_j^{new} = \frac{x_j - \min(x_j)}{\max(x_j) - \min(x_j)}$$

EQUAÇÃO 11 – FÓRMULA PARA NORMALIZAÇÃO SEVERA.
FONTE: (PRINCIPE *et al.*, 2000).

Referente à fórmula de normalização severa entende-se:

- x_j^{new}: Nova valor normalizado;
- xj: Valor a ser normalizado;
- min(x_j): Menor valor da característica presente na Base de Dados;
- max(x_j): Maior valor da característica presente na Base de Dados.

A normalização severa foi ajustada para não sofrer influência de valores de características que sejam *outliers*, ou seja, aonde os valores dos atributos apresentem registros fora de um intervalo de variação normal, e dessa forma forçam a normalização a concentrar os resultados próximo ao valor 0 ou ao valor 1. Para este ajuste é utilizado o método de detecção de *outliers* por intervalo de variação para estabelecer qual a variação máxima e mínima dentro de uma distribuição normal em cada atributo. A partir do cálculo do intervalo de variação é obtido o novo valor mínimo e o novo valor máximo que o processo de normalização deve utilizar, alterado o comportamento da normalização severa distribuindo melhor os valores das amostras dentro do intervalo de

zero e um [0; 1], e os valores *outliers*, ficam abaixo de zero ou a cima de um, o comportamento deste ajustes pode ser observado na FIGURA 40.

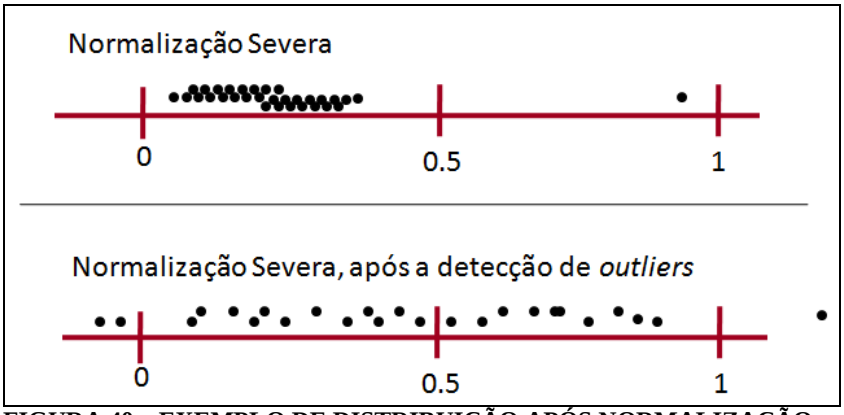


FIGURA 40 – EXEMPLO DE DISTRIBUIÇÃO APÓS NORMALIZAÇÃO.
FONTE: O autor (2011).

4.6 GERAÇÃO DA COMPOSIÇÃO

A etapa de geração da composição é a aplicação de uma técnica de redução de dimensionalidade, de tal forma que se possa expressar todas as características em um único valor.

O método de redução de dimensionalidade utilizado nesta pesquisa foi a Indexação Recursiva (INREC), proposto por Souza (1999). A INREC é uma forma simples de representar a informação, mas mantém o poder discriminatório, através de um processo que possa reduzir as *n* características para somente uma, não fazendo a eliminação ou considerando uma característica mais relevante que outra, mas sim utilizando todas as características para representar um padrão, que mantenha informação significante para que elementos similares fiquem próximos.

Para exemplo de geração de composição será utilizado uma janela de vinte e uma bases do gene hypothetical protein ECDH10B_0005

```
hypothetical protein ECDH10B_0005
[Escherichia coli str. K-12 substr. DH10B]

     GTGAAAAAGATGCAATCTATCGTACTCGC...
(01) GTGAAAAAGATGCAATCTATC
(02)  TGAAAAAGATGCAATCTATCG
(03)   GAAAAAGATGCAATCTATCGT
(04)    AAAAAGATGCAATCTATCGTA
(05)     AAAAGATGCAATCTATCGTAC
(06)      AAAGATGCAATCTATCGTACT
(07)       AAGATGCAATCTATCGTACTC
(08)        AGATGCAATCTATCGTACTCG
(09)         GATGCAATCTATCGTACTCGC
```

FIGURA 41 – JANELAS DE VINTE E UMA BASES DO PRIMEIRO SEGMENTO.
FONTE: O autor (2012).

A partir da sequência de DNA contida na janela é gerada a matriz de co-ocorrência e obtidas às características e normalizadas pelos valores mínimos de máximos ajustados de toda a base obtendo o seguinte valor:

TABELA 6 – TABELA CONTENDO OS VALORES DAS CARACTERISTICA E SUA NORMALIZAÇÃO PARA A JANELA (01) DO GENE HYPOTHETICAL PROTEIN ECDH10B_005

Característica	Valor	Valor Normalizado
Segundo momento angular	0,0916	0,3219
Entropia	3,6830	0,6404
Contraste	2,5248	0,4738
Variância	3,7884	0,7903
Correlação	0,2001	0,8769
Homogeneidade	0,5508	0,1582

FONTE: O autor (2012)

A composição é realizada com os valores das características normalizadas, seguindo o algoritmo da INREC, $r = f(x^a_1\ f(x^a_2\ f(x^a_3\ ...\ f(x^a_n)\ ...)))$, com os seguintes parâmetros:

- f: função tangente hiperbólica;
- a: 0,5;
- x_1: Segundo momento angular (0,3219);
- x_2: Entropia (0,6404);
- x_3: Contraste (0,4738);
- x_4: Variância (0,7903);
- x_5: Correlação (0,8769);
- x_6: Homogeneidade (0,1582);

Conforme as seis características a recursividade da INREC terá seis passagens como são apresentadas:

Passo 1: $f(x^{0.5}_6) = 0,37801667644892717$;

Passo 2: $f(x^{0.5}_5\ f(x^{0.5}_6)) = 0,33990571625214916$;

Passo 3: $f(x^{0.5}_4\ f(x^{0.5}_5\ f(x^{0.5}_6))) = 0,29329918265455945$;

Passo 4: $f(x^{0.5}_3\ f(x^{0.5}_4\ f(x^{0.5}_5\ f(x^{0.5}_6)))) = 0,19918814277300362$;

Passo 5: $f(x^{0.5}_2\ f(x^{0.5}_3\ f(x^{0.5}_4\ f(x^{0.5}_5\ f(x^{0.5}_6))))) = 0,15806384618885808$;

Passo 6: $f(x^{0.5}_1\ f(x^{0.5}_2\ f(x^{0.5}_3\ f(x^{0.5}_4\ f(x^{0.5}_5\ f(x^{0.5}_6)))))) = 0,0894398281275982$.

O valor 0.0894398281275982 representa as características da janela (01) do gene hypothetical protein ECDH10B_0005.

Todas as sequências contidas na Base de Dados totalizando três milhões oitocentos e dezenove mil setecentos e nove (3.819.709) genes, passaram pelas etapas da metodologia:

a) Codificação das sequências de nucleotídeo;

b) Extração das características das sequências de genes;

c) Normalização das características;

d) Geração da composição.

Com a Base de Dados contendo os valores das composições, na qual cada número real representa uma janela de 21 bases, sendo que para cada uma das sequências armazenadas possui até 537 janelas, referente aos três segmentos, início meio e fim, de até duzentas 200 bases.

4.7 RESPOSTAS DA BASE DE DADOS

Para recuperar os dados da base foi realizada uma consulta no banco de dados observando os seguintes pontos, cada registro pertence a um gene, cada janela possui a identificação de qual segmento esta contida bem como o seu valor, e cada valor contém sua posição dentro do segmento. Dessa forma é possível a partir de uma sequência a ser pesquisada, na qual já se obteve os valores das janelas para os três segmentos, realizar a comparação destes valores com a Base de Dados e assim recuperar todos os registros que possuam os mesmos valores das janelas da sequência de pesquisa.

A consulta para recuperação de dados da base possui as seguintes etapas:

a) Selecionar todos os registros da base no segmento de início, onde os valores dos registros sejam iguais aos valores das janelas do segmento de início da sequência de pesquisa;

b) Selecionar todos os registros da base no segmento do meio, onde os valores dos registros sejam iguais aos valores das janelas do segmento do meio da sequência de pesquisa;

c) Selecionar todos os registros da base no segmento do fim, onde os valores dos registros sejam iguais aos valores das janelas do segmento do fim da sequência de pesquisa;

d) Unir as três seleções em uma só resposta;

e) Ordenar as respostas por gene, segmento e posição de forma ascendente;

f) Avaliar o grau de correspondências do retorno dos registros para cada gene, obtendo uma pontuação.

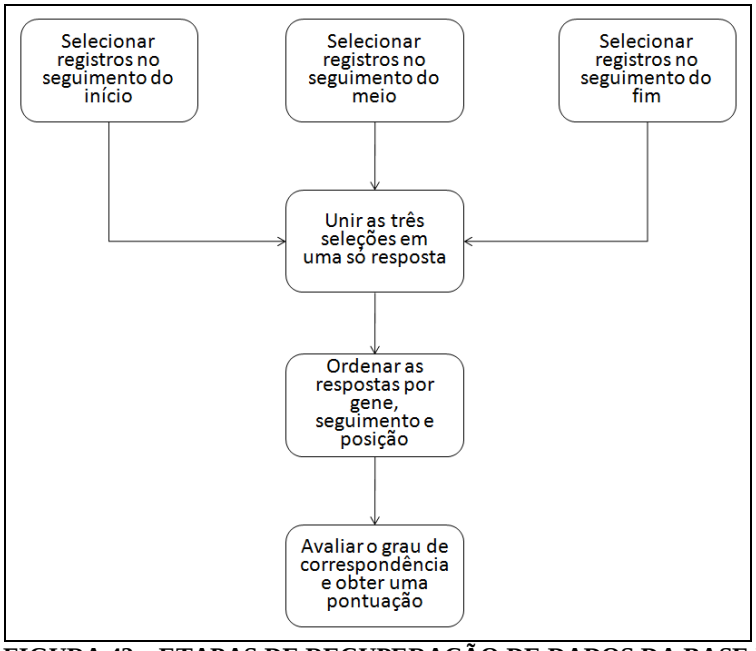


FIGURA 42 – ETAPAS DE RECUPERAÇÃO DE DADOS DA BASE DE DADOS
FONTE: O autor (2012).

O processo de avaliação do grau de correspondência foi realizado obedecendo aos seguintes critérios:

a) Para os registros correspondentes do mesmo gene e do mesmo segmento, onde a posição seja incrementada em um, o gene recebe 4 pontos por correspondência;

b) Para os registros correspondentes do mesmo gene, do mesmo segmento, onde a posição não é sequencial, ou seja, haja salto entre as posições, o gene recebe -2 pontos por salto de posição;

c) Para os registros correspondentes do mesmo gene, porém havendo mudança de segmento, o gene recebe zero ponto por correspondência;

d) Para os registros correspondentes de genes diferentes, inicia-se uma nova contagem de pontos para o novo gene;

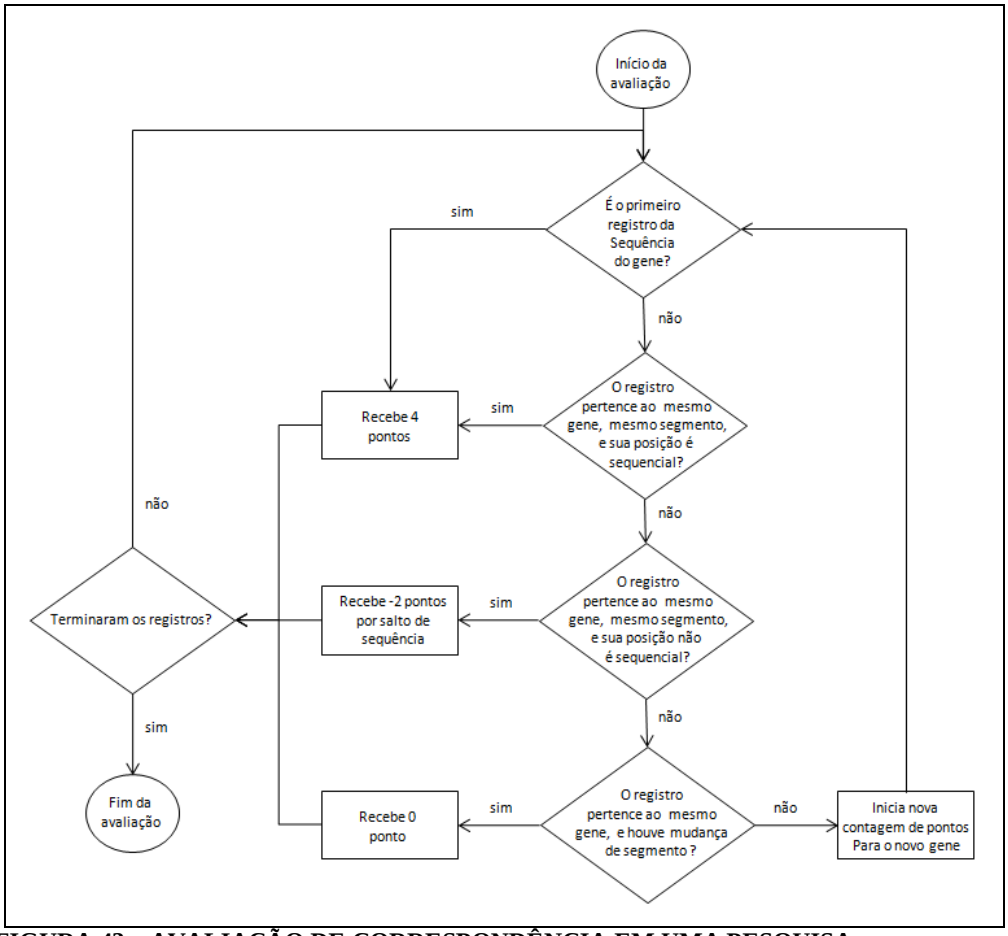


FIGURA 43 – AVALIAÇÃO DE CORRESPONDÊNCIA EM UMA PESQUISA.
FONTE: O autor (2012).

Ao termino da avaliação é retornada uma lista dos genes em ordem decrescente de valores de pontuação, indicando a ordem de semelhança dos genes (CÓDIGO FONTE 15 persente no APÊNDICE I).

Com a seleção construída e sendo possível obter as respostas da base, é necessário avaliar a pertinências das respostas, com o objetivo de certificar que os registros retornados na lista ordenada são pertinentes a sequência de pesquisa e possuam um alto grau de identidade esperado. A validação foi realizada comparando as respostas da base com as respostas do BLAST. Para realizar esta comparação foi exportado do banco de dados todas as sequências armazenadas para formar uma base no padrão de leitura do BLAST (CÓDIGO FONTE 14 APÊNDICE I) e assim ambos os algoritmos podem fornecer respostas sobre a mesma fonte de dados e sobre a mesma sequência pesquisada.

Um conjunto de teste contendo 10.000 sequências de DNA a ser pesquisada em ambos os algoritmos, foi extraído da Base de Dados de forma aleatória. E sobre este conjunto foi obtida as vinte melhores respostas do BLAST e as vinte melhores respostas da seleção da Base de Dados. Era esperado um total de 200.000 respostas, porém o BLAST para algumas sequências não retorna vinte respostas de sequências semelhantes, pois em seu algoritmo ele avalia um limite de similaridade e abaixo desse limite não considera como uma resposta satisfatória, dessa forma um total de 185.039 respostas foram obtidas do BLAST na qual serviram de base para validação das respostas da Base de Dados.

A comparação foi realizada avaliando duas situações. A primeira condição foi realizada comparando a primeira melhor resposta da base com a primeira melhor resposta do BLAST, considerando um *score* relativo ao *self-score* nas seguintes variações, 30%, 50%, 70% e 90%. A segunda condição foi avaliar todas as vinte respostas para identificar a profundidade dos acertos tomando como base as respostas do BLAST.

4.7.1 30% DE *SCORE* RELATIVO AO *SELF-SCORE*

Avaliando a primeira melhor resposta das 10.000 sequências obtidas pela Base de Dados e as comprando com o BLAST referente a 30% de *score* relativo o *self-score*, foi obtido o seguinte resultado, (FIGURA 44, TABELA 7):

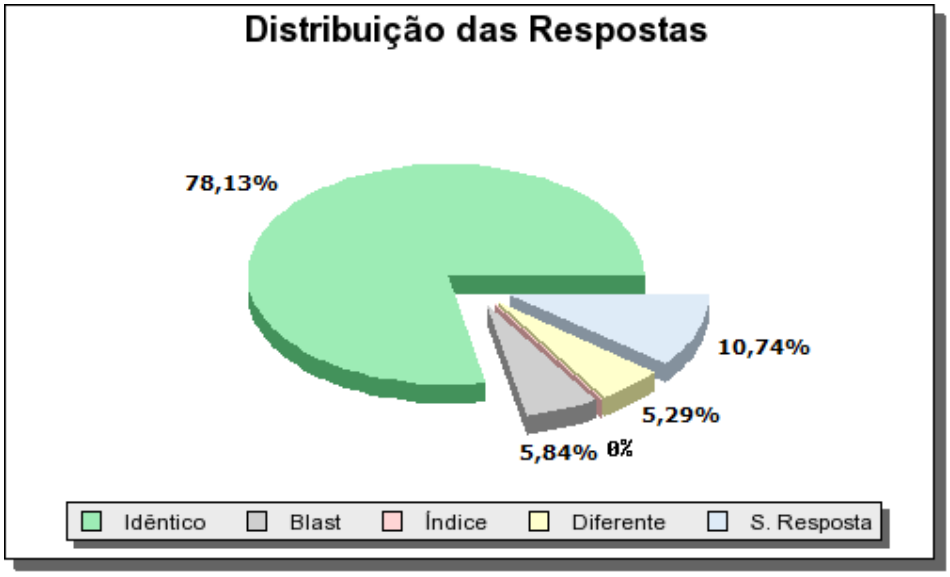


FIGURA 44 – 30% DE *SCORE* RELATIVO AO *SELF-SCORE*.
Resultado comparativo entre o BLAST e a Base de Dados referente a 30% *score* relativo ao *self-score* para 10.000 sequências.
Fonte: O autor (2012).

TABELA 7 – VALORES ABSOLUTOS E PERCENTUAIS PARA 30% DE *SCORE* RELATIVO AO *SELF-SCORE*.

Tipo de Reconhecimento	Valor Absoluto	Percentual
Idênticas	7.813	78,13%
Somente BLAST	584	5,84%
Somente Índice (Base de Dados)	0	0%
Diferentes (Entre o BLAST e a Base de Dados	529	5,29%
Sem Resposta	1.074	10,74%
Total	**10.000**	**100%**

Fonte: O autor (2012).

O resultado entre o BLAST e a Base de dados mostra que 78,13% das respostas obtidas são idênticas ao BLAST indicando para a primeira melhor resposta o grau de sensibilidade e reconhecimento na linha de corte de 30% de *score* relativo o *self-score*. Respostas divergentes onde a melhor resposta da Base de Dados são diferentes da melhor resposta do BLAST totalizam 5,29%, com 529 registros, esta divergência não representa necessariamente um erro ou respostas ruins, pois ambos os métodos encontraram boas sequências para esta linha de corte. As demais respostas são 5,84% dos registros que somente o BLAST conseguiu localizar sequências semelhantes totalizando 584 respostas, e 10,74% onde não foram identificadas sequências semelhantes em ambos os métodos.

Desconsiderando do total de genes, as sequências que não obtiveram respostas e agrupar as respostas idênticas com as respostas divergentes, aonde representam respostas válidas da Base de Dados, é obtido o resultado de 93,45% de respostas da Base dentro da linha de corte de 30% de *score* relativo ao *self-score*, ou seja, 8.342 respostas de um total de 8.926. Já o número de respostas encontradas somente pelo BLAST, representa 6,55% ou 584 registros (FIGURA 45).

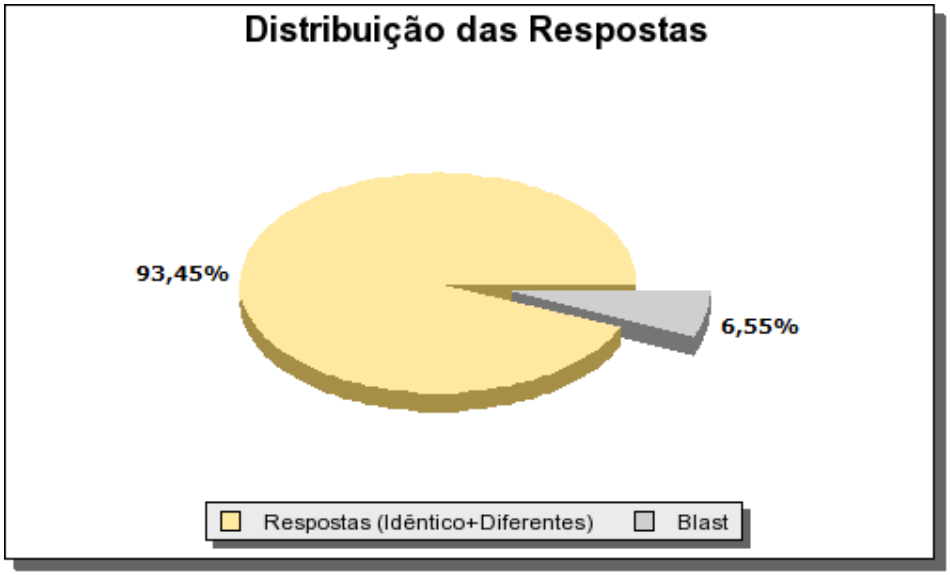


FIGURA 45 – 30% DE SCORE RELATIVO AO *SELF-SCORE* DESCONSIDERANDO AS SEQUÊNCIAS SEM RESPOSTAS.
Resultado comparativo entre o BLAST e a Base de Dados referente a 30% score relativo ao *self-score* desconsiderando as sequências sem respostas totalizando 8.926 sequências.
Fonte: O autor (2012).

4.7.2 50% DE *SCORE* RELATIVO AO *SELF-SCORE*

Avaliando a primeira melhor resposta das 10.000 sequências obtidas pela Base de Dados e as comprando com o BLAST, porém alterando a linha de corte para 50% de score relativo ao *self-score* na qual representa um refinamento das respostas, aonde somente as sequências com um grau maior de identidade são consideradas é obtido o seguinte resultado (TABELA 8).

TABELA 8 – VALORES ABSOLUTOS E PERCENTUAIS PARA 50% DE *SCORE* RELATIVO AO *SELF-SCORE*.

Tipo de Reconhecimento	Valor Absoluto (atual)	Percentual (atual)	Percentual 30% *score* relativo (anterior)
Idênticas	7.431	74,31%	78,13%
Somente BLAST	175	1,75%	5,84%
Somente Índice (Base de Dados)	0	0%	0%
Diferentes (Entre o BLAST e a Base de Dados	236	2,36%	5,29%
Sem Resposta	2.158	21,58%	10,74%
Total	**10.000**	**100%**	**100%**

Fonte: O autor (2012).

Ao comparar as respostas da Base de Dados com o BLAST entre as linhas de corte de 30% e 50% de *score* relativo ao *self-score* (FIGURA 46), é observado que todos os valores absolutos diminuíram, com exceção do valor das sequências que não obteve respostas seja pelo BLAST ou pela Base de Dados.

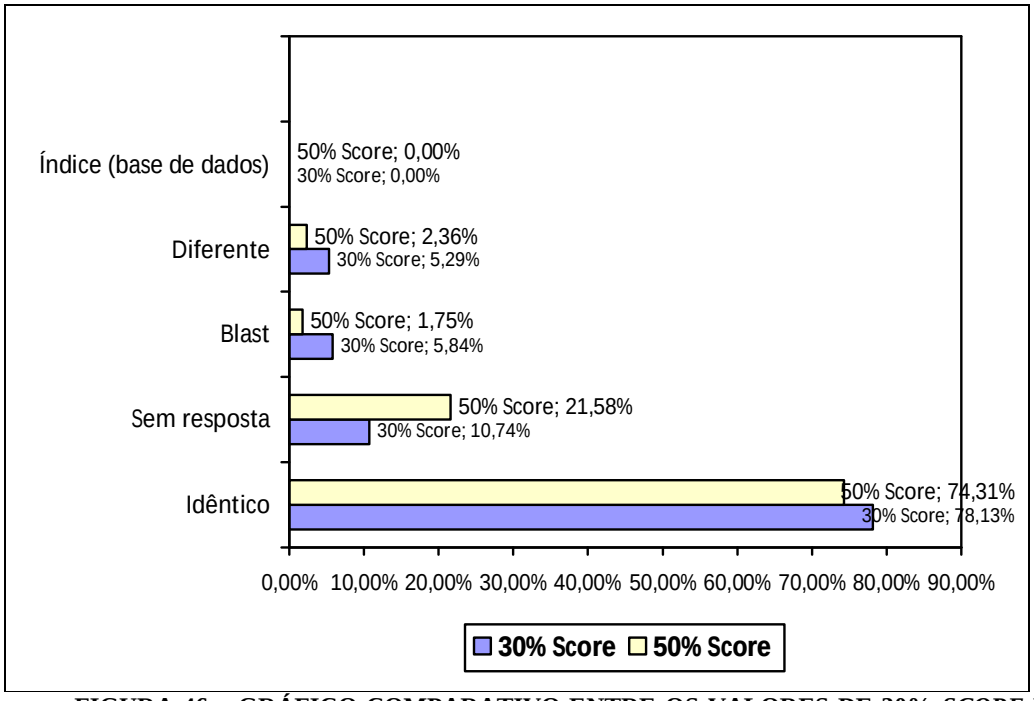


FIGURA 46 – GRÁFICO COMPARATIVO ENTRE OS VALORES DE 30% *SCORE* E 50% DE *SCORE* RELATIVO AO *SELF-SCORE*.
FONTE: O autor (2012).

Mesmo apresentando uma redução nos valores de respostas, isso não significa que os resultados são piores, pelo contrário, é um aumento de 4,32% do total de respostas válidas entre as linhas de corte de 30% e 50% de *score* relativo ao *self-score*, passando de 93,45% para 97,77% (FIGURA 47), mostrando uma precisão mais elevada das respostas quando aplicada uma linha de corte maior. Colaborando com esta análise as sequências com respostas somente pelo BLAST diminuíram de 6,55% na linha de corte anterior para 2,23% na linha de corte atual.

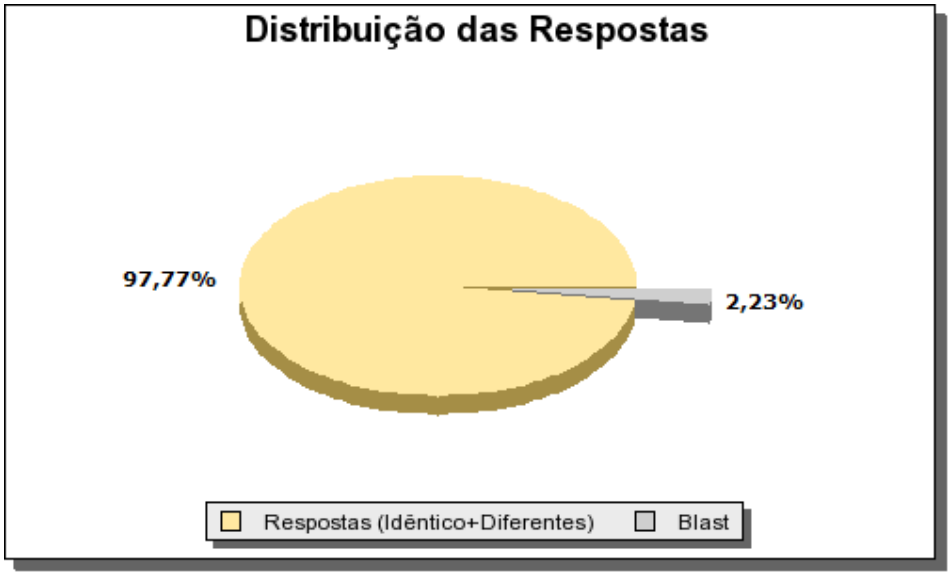


FIGURA 47 – 50% DE *SCORE* RELATIVO AO *SELF-SCORE* DESCONSIDERANDO AS SEQUÊNCIAS SEM RESPOSTAS.
Resultado comparativo entre o BLAST e a Base de Dados referente a 50% *score* relativo ao *self-score* desconsiderando as sequências sem respostas totalizando 7.842 sequências.
Fonte: O autor (2012).

4.7.3 70% DE SCORE RELATIVO AO SELF-SCORE

Com o objetivo de identificar as sequências com alto grau de identidade, foi analisada a linha de corte com 70% de *score* relativo ao *self-score*, entre o BLAST e a Base de Dados, com o seguinte resultado (FIGURA 48, TABELA 9).

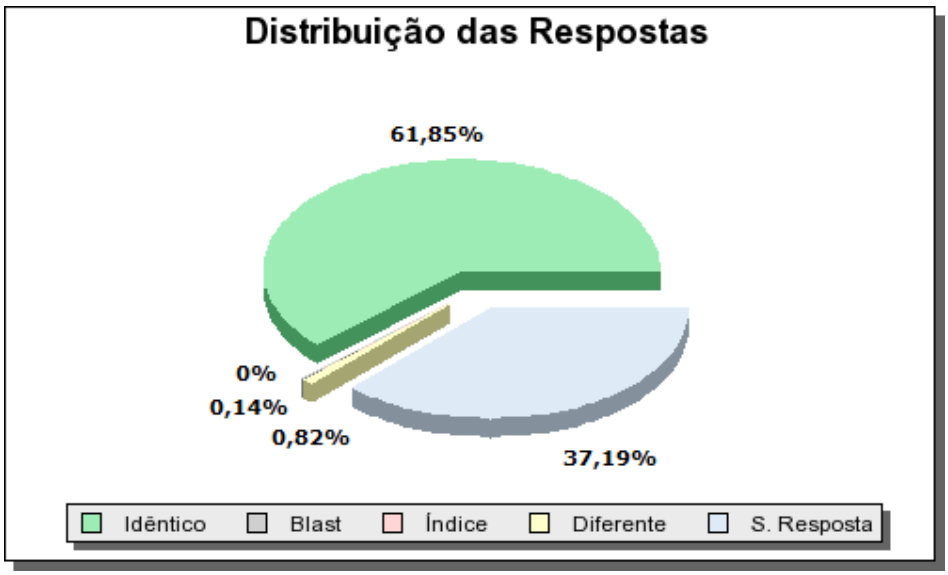


FIGURA 48 – 70% DE *SCORE* RELATIVO AO *SELF-SCORE*.
Resultado comparativo entre o BLAST e a Base de Dados referente a 70% *score* relativo ao *self-score* para 10.000 sequências.
Fonte: O autor (2012).

TABELA 9 – VALORES ABSOLUTOS E PERCENTUAIS PARA 70% DE *SCORE* RELATIVO AO *SELF-SCORE*.

Tipo de Reconhecimento	Valor Absoluto (atual)	Percentual (atual)	Percentual 50% *score* relativo (anterior)
Idênticas	6.185	61,85%	74,31%
Somente BLAST	14	0,14%	1,75%
Somente Índice (Base de Dados)	0	0%	0%
Diferentes (Entre o BLAST e a Base de Dados	82	0,82%	2,36%
Sem Resposta	3.719	37,19%	21,58%
Total	**10.000**	**100%**	**100%**

Fonte: O autor (2012).

A tendência de redução dos valores com respostas apresentadas pelas linhas de corte anteriores de 30% e 50% de *score* relativo o *self-score*, se faz seguir para avaliação de 70% de *score* relativo ao *self-score*. Respostas idênticas entre o BLAST e a Base de Dados totalizam de 62,85%, respostas obtidas somente pelo BLAST são 0,14% com somente quatorze registros, respostas diferentes entre o BLAST e a Base de Dados representa 0.82% com oitenta e duas divergências.

Removendo o percentual de sequências que não obtiveram respostas pelo BLAST ou pela Base de Dados e agrupando as respostas válidas da Base de Dados temos os seguintes valores (FIGURA 49, TABELA 10):

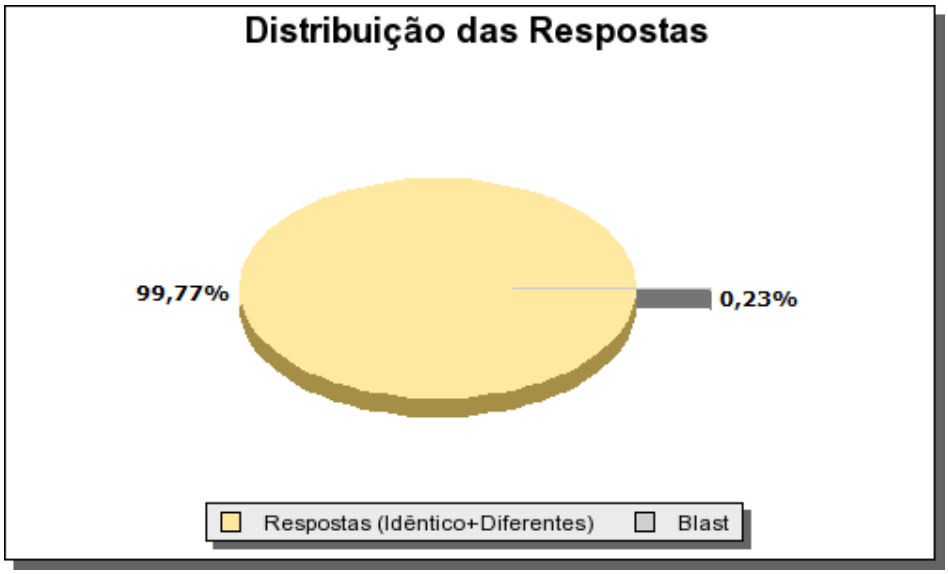


FIGURA 49 – 70% DE *SCORE* RELATIVO AO *SELF-SCORE* DESCONSIDERANDO AS SEQUÊNCIAS SEM RESPOSTAS.
Resultado comparativo entre o BLAST e a Base de Dados referente a 70% *score* relativo ao *self-score* desconsiderando as sequências sem respostas totalizando 6.281 sequências.
Fonte: O autor (2012).

TABELA 10 – VALORES ABSOLUTOS E PERCENTUAIS PARA 70% DE *SCORE* RELATIVO AO *SELF-SCORE* DESCONSIDERANDO AS SEQUÊNCIAS SEM RESPOSTAS.

Tipo de Reconhecimento	Valor Absoluto (atual)	Percentual (atual)	Percentual 50% *score* relativo (anterior)
Respostas (Idênticas+Diferentes)	6.267	99,77%	97,77%
Somente BLAST	14	0,23%	2,23%
Somente Índice (Base de Dados)	0	0%	0%
Total	**6.281**	**100%**	**100%**

Fonte: O autor (2012).

Observando a composição das respostas idênticas (6.185 registros) e divergentes (82 registros) entre o BLAST e a Base Dados, totalizando 6.267 respostas, sendo estas respostas sensíveis na linha de corte de 70% do *score* é obtido um percentual de 99,77% do total o que representa um aumento de 2% em relação à linha de corte anterior, quando se desprezando o número de sequências que não obtiveram respostas seja pelo BLAST ou pela Base Dados. Já as respostas obtidas somente pelo BLAST representam 0,23% com quatorze registros.

4.7.4 90% DE *SCORE* RELATIVO AO *SELF-SCORE*

Para avaliação de 90% de *score* relativo ao *self-score*, são esperadas respostas das sequências com alto grau de identidade, ou seja, somente as sequências de genes muito semelhantes e consequentemente a elevação do número de sequências que não possuam respostas para o BLAST como para a Base de Dados. O resultado obtido esta presente na FIGURA 50 e TABELA 11.

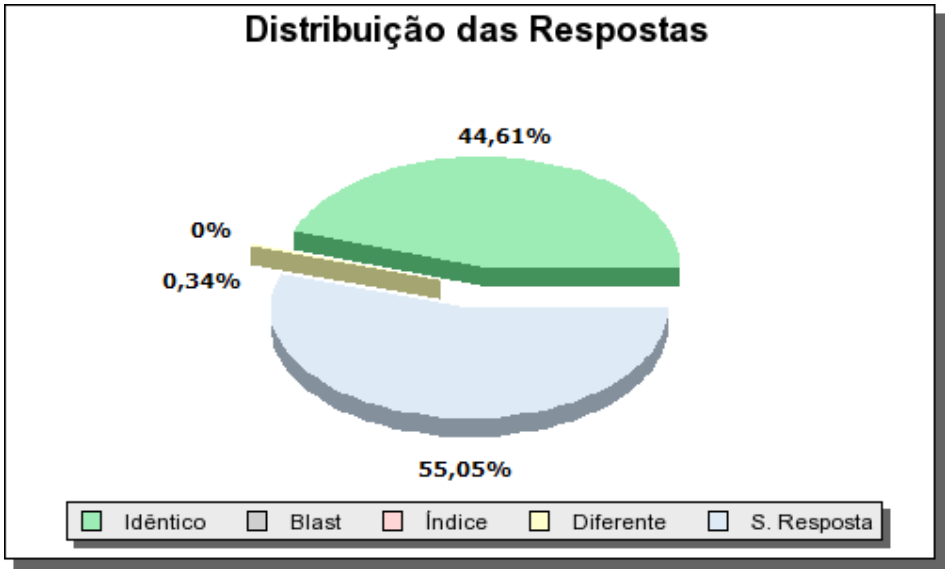


FIGURA 50 – 90% DE *SCORE* RELATIVO AO *SELF-SCORE*.
Resultado comparativo entre o BLAST e a Base de Dados referente a 90% *score* relativo ao *self-score* para 10.000 sequências.
Fonte: O autor (2012).

TABELA 11 – VALORES ABSOLUTOS E PERCENTUAIS PARA 90% DE *SCORE* RELATIVO AO *SELF-SCORE*.

Tipo de Reconhecimento	Valor Absoluto	Percentual (atual)	Percentual 70% *score* relativo (anterior)
Idênticos	4.461	44,61%	61,85%
Somente BLAST	0	0%	0,14%
Somente Índice (Base de Dados)	0	0%	0%
Diferentes (Entre o BLAST e a Base de Dados	34	0,34%	0,82%
Sem Resposta	5.505	55,05%	37,19%
Total	**10.000**	**100%**	**100%**

Fonte: O autor (2012).

O percentual de sequências sem respostas na linha de corte de 90% de *score* relativo ao *self-score* teve um aumento de 17,86%, passando de 37,19% na linha de corte anterior para 55,05% na atual, indicando que somente 44.95% do total de respostas são sensíveis a 90% de *score* relativo ao *self-score*.

Quando avaliada as resposta de 90% de *score* relativo ao *self-score*, desconsiderando as sequências sem respostas, o valor de respostas significantes na qual a Base de Dados retornou é de 100%, sendo que somente trinta e quatro 34 registros foram divergentes pelo BLAST e a BASE de Dados (TABELA 11), o que representa 0,68% do total (FIGURA 51).

FIGURA 51 – 90% DE *SCORE* RELATIVO AO *SELF-SCORE* DESCONSIDERANDO AS SEQUÊNCIAS SEM RESPOSTAS.
Resultado comparativo entre o BLAST e a Base de Dados referente a 90% *score* relativo ao *self-score* desconsiderando as sequências sem respostas totalizando 4.495 sequências.
Fonte: O autor (2012).

4.7.5 EVOLUÇÃO DOS RESULTADOS

A partir das validações realizadas entre o BLAST e a Base de Dados com *score* relativo ao *self-score* nas variações de 30%, 50%, 70% e 90% pode se observar que quanto maior o limite de sensibilidade das medidas, mais precisa são as respostas da Base de Dados, tal consideração pode ser observada pela variação dos valores apresentados pelas sequências onde somente o BLAST encontrou respostas bem como na variação dos valores das respostas divergente entre o BLAST e a Base de Dados (FIGURA 52).

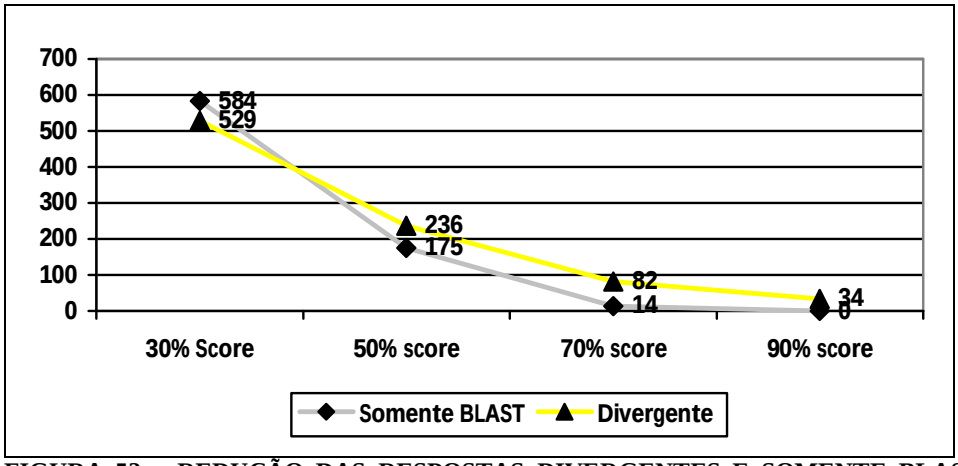


FIGURA 52 – REDUÇÃO DAS RESPOSTAS DIVERGENTES E SOMENTE BLAST PARA AS FASE DE VALIDAÇÃO DE SELF-SCORE.
Fonte: O autor (2012).

A variação dos valores encontrados somente pelo BLAST apresenta uma redução drástica em cada fase, pois a passagem da fase de 30% de *score* para 50% de *score* relativo ao *self-score* houve uma redução de aproximadamente 70,03%, a passagem da fase 50% de *score* para 70% de *score* relativo ao *self-score* obteve redução de 92%, já a ultima passagem da fase de 70% de *score* para 90% de *score* houve a redução total, 100%, não havendo mais respostas encontradas somente pelo BLAST (FIGURA 53), indicando que as respostas encontradas pelo BLAST também são encontradas pela Base de Dados.

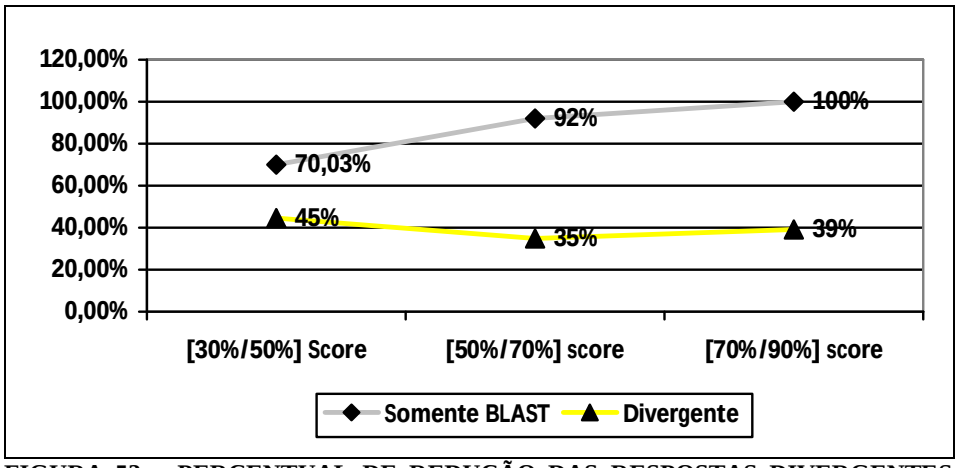


FIGURA 53 – PERCENTUAL DE REDUÇÃO DAS RESPOSTAS DIVERGENTES E SOMENTE BLAST PARA AS FASE DE VALIDAÇÃO DE SELF-SCORE.
Fonte: O autor (2012).

Já a variação da redução das respostas divergentes entre o BLAST e a Base de Dados é de aproximadamente 39% em média por passagem (FIGURA 53), apontando a precisão dos resultados da Base de Dados quando a sensibilidade da medida aumenta, pois em cada passagem a redução é acumulativa.

Sobre cada passagem de *score* relativo ao *self-score*, podemos extrair o total de respostas para cada medida, identificando qual é o nível de sensibilidade que mais possui respostas para resultados idênticos entre o BLAST e a Base de Dados. Na FIGURA 54, observa-se que na linha de corte de 90% de *score* relativo ao *self-score* apresenta o maior valor de respostas idênticas, mostrando que nesta faixa a Base de Dados tem uma alta capacidade de encontrar sequências semelhantes.

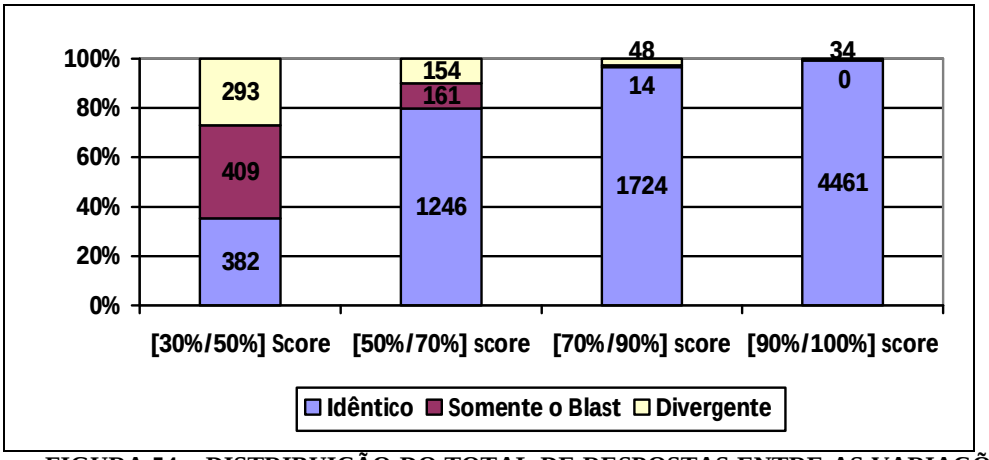


FIGURA 54 – DISTRIBUIÇÃO DO TOTAL DE RESPOSTAS ENTRE AS VARIAÇÕES DE *SCORE* RELATIVO AO *SELF-SCORE*.
Fonte: O autor (2012).

Reforçando a análise anterior de que as respostas são altamente idênticas, a FIGURA 50 apresenta o total de respostas por faixa, mostrando que a maior quantidade de respostas se obtém para 90% de score relativo ao self-score entre as 10.000 sequências avaliadas pelo BLAST e a Base de Dados.

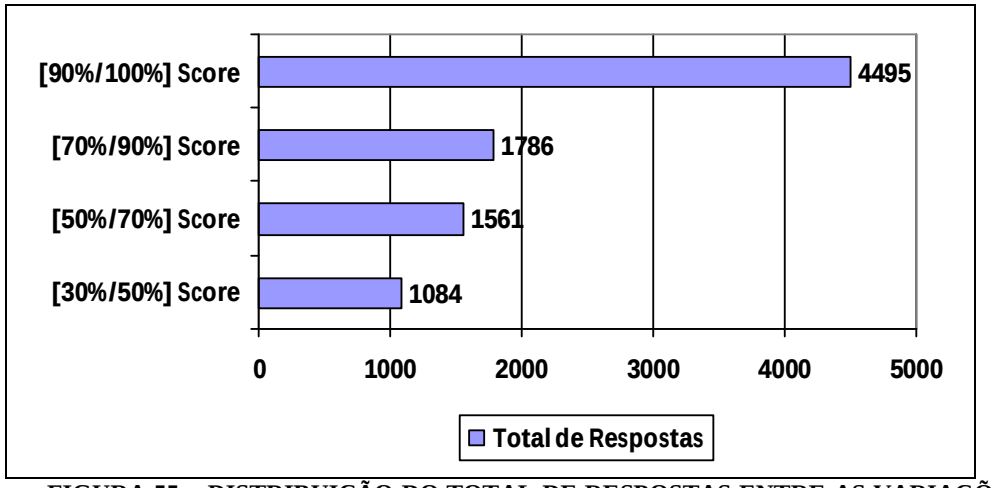


FIGURA 55 – DISTRIBUIÇÃO DO TOTAL DE RESPOSTAS ENTRE AS VARIAÇÕES DE *SCORE* RELATIVO AO *SELF-SCORE*.
Fonte: O autor (2012).

Um dado que chama a atenção na faixa de 90% de *score* relativo ao *self-score*, são os trinta e quatro valores divergentes entre o BLAST e a Base de Dados, presentes no apêndice IV, onde trinta e um registros da base apresentam valores de *score* relativo melhores do que os valores de *score* relativo do BLAST, indicando que nas respostas divergentes a Base de Dados obteve mais de 90% de respostas com melhor desempenho, os outros três registros apesar de serem sequências diferentes apresentam os mesmos valores de *score* relativo para o BLAST e a Base de Dados (FIGURA 56).

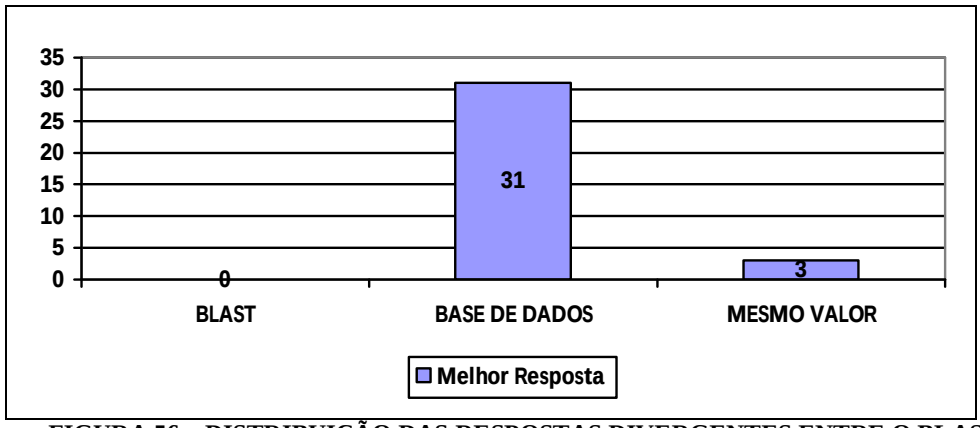


FIGURA 56 – DISTRIBUIÇÃO DAS RESPOSTAS DIVERGENTES ENTRE O BLAST E A BASE DE DADOS PARA 90% DE *SCORE* RELATIVO AO *SELF-SCORE*.
Fonte: O autor (2012).

4.7.6 AVALIAÇÃO DAS VINTE PRIMEIRAS RESPOSTAS

Para complementar a análise entre o BLAST e a Base de Dados, foi observada também o comportamento das 20 primeiras respostas de ambos os métodos, de forma que seja contabilizada como respostas idênticas (matches) as sequências que estejam presentes no BLAST e na Base de Dados, mesmo que em posições diferentes. Assim o resultado apresentado foi de pouco mais de 59.1% de respostas idênticas na qual a Base de Dados apresentou em relação ao BLAST (FIGURA 57).

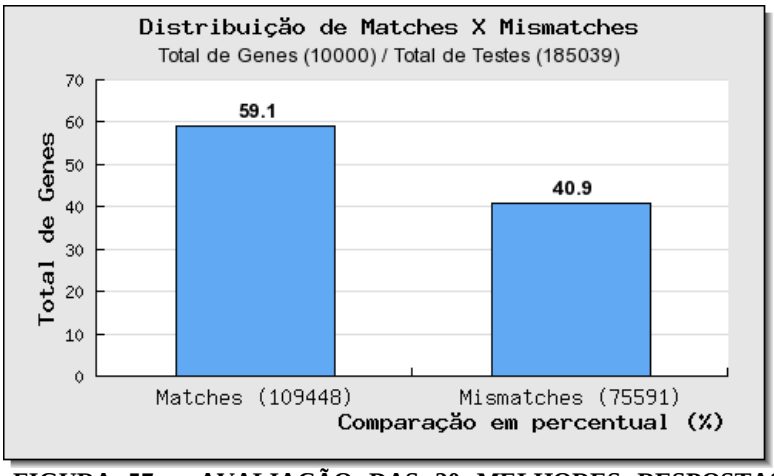


FIGURA 57 – AVALIAÇÃO DAS 20 MELHORES RESPOSTAS ENTRE O BLAST E A BASE DE DADOS.
Fonte: O autor (2012).

Ao aplicar uma linha de corte de 70% de *score* relativo ao *self-score* o que representa 70% de similaridade entre as sequências, sendo esta linha de corte adequada para pesquisa de genes similares segundo Ochmam (2002), são obtidas 65.696 respostas válidas o que representa uma redução de pouco mais de 64% do total de 185.039 respostas originalmente obtidas. Avaliando as respostas nesta linha de corte, o total de sequências que são idênticas entre o BLAST e a Base de Dados é de 99,2% (FIGURA 58), confirmando a obtenção de sequências com alto grau de identidade.

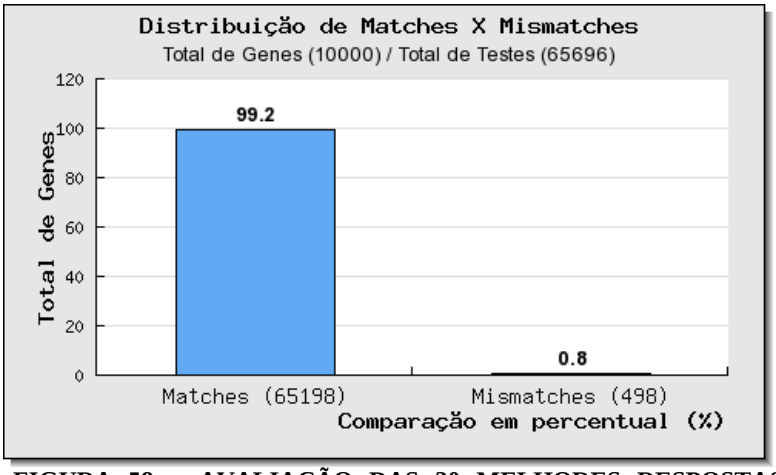


FIGURA 58 – AVALIAÇÃO DAS 20 MELHORES RESPOSTAS ENTRE O BLAST E A BASE DE DADOS COM UMA LINHA DE CORTE COM 70% DE *SCORE* RELATIVO AO *SELF-SCORE*.
Fonte: O autor (2012).

Outra análise realizada sobre as vinte melhores respostas foi quanto ao total de correspondências de respostas por sequências pesquisadas, ou seja, quantas correspondências dentro do mesmo gene foram obtidas de forma idêntica entre ambos os métodos. Verificando as duas melhores medidas dessa análise, têm em primeiro lugar, 2.613 sequências pesquisadas, na qual foram encontradas pela Base de Dados todas as vinte respostas do BLAST. Em segundo lugar mostra correspondências entre 1.495 sequências com somente uma resposta corresponde ao BLAST. As distribuições das demais posições podem ser observadas na FIGURA 59.

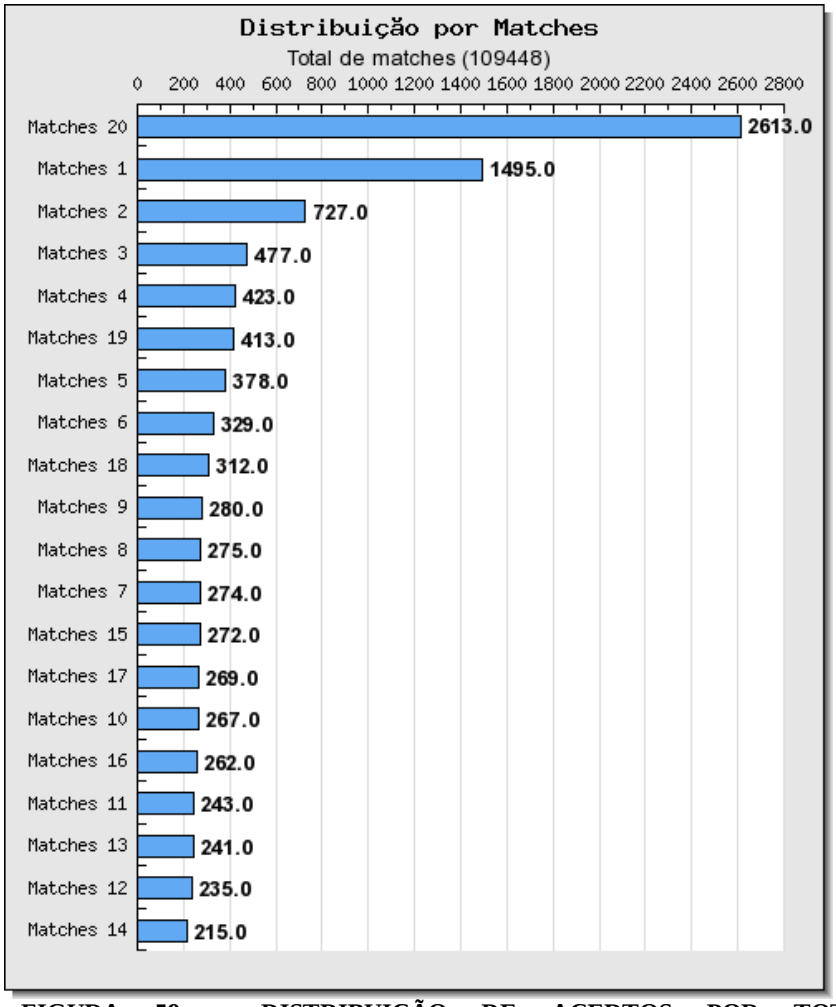


FIGURA 59 – DISTRIBUIÇÃO DE ACERTOS POR TOTAL DE CORRESPONDÊNCIAS
Fonte: O autor (2012).

Outra informação pertinente é considerar as correspondências de respostas do BLAST e da Base de Dados por posição. Nesta situação as três posições de resposta que obtiveram maior correspondência foram, a primeira posição com 9.996 respostas, a segunda posição com 8.000 respostas, e a terceira posição com 7.225 respostas idênticas, mostrando que as melhores respostas do BLAST e da Base de Dados ocorrem nas primeiras posições de ambos os métodos. As demais posições podem ser observadas na FIGURA 60.

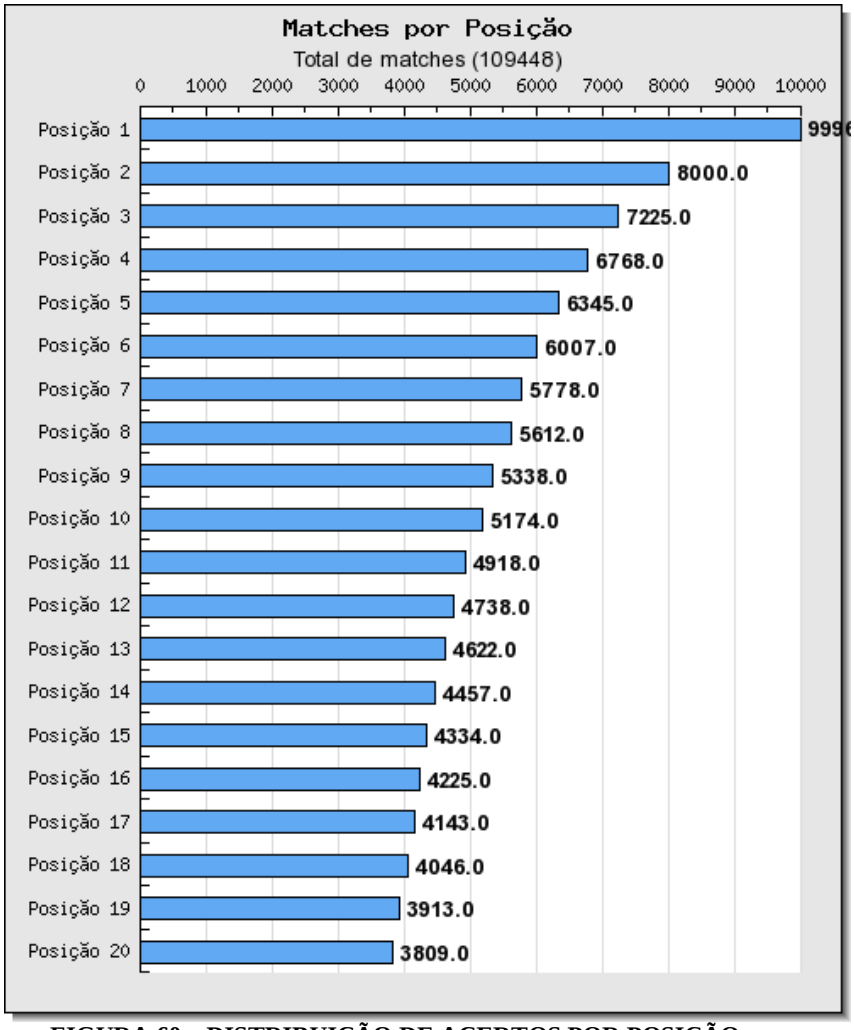


FIGURA 60 – DISTRIBUIÇÃO DE ACERTOS POR POSIÇÃO.
Fonte: O autor (2012).

4.8 DISCUSSÃO DOS RESULTADOS DA METODOLOGIA

Os resultados apresentados até o momento demonstram-se eficiente para identificação de sequências com alto grau de identidade, porém alguns pontos da metodologia aplicada devem ser discutidos, e algumas perguntas devem ser respondidas, entre elas são:

a) Aplicar a INREC diretamente nas bases de DNA não produz resultado semelhante, e assim é possível eliminar as etapas de extração de características e normalização?

O processamento da INREC diretamente nas bases de nucleotídeo é altamente sensível a variações das bases nas primeiras posições, de forma que as variações nesta posição tendem a distanciar os valores obtidos e consequentemente a não identificar sequências similares.

Para exemplificar a diferença entre a composição da INREC diretamente nas bases de nucleotídeos com a INREC das características, na TABELA 12 é mostrada seis janelas de vinte e uma bases do gene hypothetical protein AM1_0053, onde foram geradas a INREC a partir das bases e a partir das características geradas pelas bases.

TABELA 12 – DIFERENÇA ENTRE VALORES DA INREC CALCULADOS A PARTIR DAS BASES E PELAS CARACTERÍSTICAS.

Janelas	INREC Nucleotídeo	INREC Característica
(20)TTATCCTGCTCCTTGGGGTGT	0.9465060866756988	2.6690319360641066
(21)TATCCTGCTCCTTGGGGTGTT	0.8985558540598058	2.6690319360641066
(39)GTTTCTTTAAAAGCTATTTCT	0.9298943885491731	138.44522251668988
(40)TTTCTTTAAAAGCTATTTCTG	0.9570209213432536	138.44522251668988
(46)TAAAAGCTATTTCTGGTGGTA	0.7606098050832363	357.9522201432391
(47)AAAAGCTATTTCTGGTGGTAT	0.4988301681940745	357.9522201432391

Fonte: O autor (2012).

b) Por que utilizar matriz de co-ocorrência e os descritores propostos por Haralick?

O método baseado em medidas estatísticas de segunda ordem, através de matriz de co-ocorrência e descritores propostos por Haralick *et al.*, (1973), tem demostrado segundo a literatura, alto poder de discriminação em diversos tipos de imagens, uma vez que o método estatístico descreve por meio de propriedades não determinísticas as relações entre as variações de níveis de cinza ou de cor em uma imagem. (HARALICK

et. al., 1973, SHABAN; DISKIT, 1998, MUHAMAD; DERAVI, 1993; WALKER *et al.*, 1995).

Mesmo o método sendo aplicado para processamento de imagens o mesmo foi utilizado para busca de similaridade em sequências de DNA, pela semelhança do conceito de dependência e variação entre os elementos que compõem tanto as imagens (os *pixels)*, com nas sequências de DNA (as bases de nucleotídeos). Dessa forma as medidas estatísticas descrevem as variações entre as bases de nucleotídeos do mesmo modo que descreve as variações de tons entre nos *pixels* de uma imagem.

c) Quais são os pontos positivos da metodologia?

Dois pontos principais são positivos na metodologia:

1. A capacidade de identificar sequências altamente similares de forma efetiva, como demostrado nos resultados, com um custo de processamento baixo durante a consulta da sequência a ser pesquisa.

2. A facilidade de adicionar novos elementos a Base, uma vez necessário somente inserir os registros referente às novas sequências, não havendo necessidade de reconstruir toda a Base de Dados.

d) Quais são os pontos negativos da metodologia, no qual possam ser evitados ou melhorados?

O principal ponto negativo da metodologia é o pré-processamento para formação da base de dados inicial, no qual demanda tempo e capacidade de processamento.

4.9 RESULTADOS ESTRATÉGIA DE BANCO DE DADOS

Para elaboração da estratégia de recuperação das sequências, foi avaliado dois tipos de índices, *Hash* e *Btree*, bem como o plano de execução da *query* e ao final foi

realizado um *tunning* (ajuste fino) com objetivo de aproveitar melhor os recursos disponíveis na máquina.

O ponto sensível da recuperação das sequências esta em selecionar os registros da Base de Dados que contenham relação com a sequência de pesquisa. Estes registros estão distribuídos em três tabelas, composicao_inicio, composicao_meio, composicao_fim, contendo aproximadamente 680 milhões de registros em cada tabela. Para realizar a recuperação dos registros relacionados é realizada uma consulta em cada uma das tabelas do inicio, meio e fim, e a análise dessa consulta será avaliada (CÓDIGO FONTE 15 APÊNDICE I).

Os índices *Btree* e *Hash* foram testados em três diferentes planos de execução utilizando os algoritmos de junção, *Nested Loop Join*, *Hash Join* e *Merge Join*. As três tabelas de armazenamento possuem a mesma estrutura, o campo “gene” para armazenar o código do registro do gene, o campo “valor” que corresponde ao valor da janela de 21 bases, o campo “posicao” que define no registro qual a posição do valor da janela e o campo “codigo” sendo a chave primária do registro. O campo “valor” recebeu os índices para realizar a junção com a tabela “composicao_temp” que contém os valores das janelas das sequências de pesquisa e possui a mesma estrutura das tabelas anterior com acréscimo do campo “comtemp_segmento” responsável por identificar em qual segmento pertence o registro.

O SGDB PostgreSQL não sofreu nenhuma alteração de desempenho para realização dos primeiros testes. Comparando o índice *Btree* e o *Hash*, observa-se a diferença de ocupação no espaço de armazenamento, sendo 2 *Giga Bytes* são ocupados pelo índice *Btree* e quase 4 *Giga Bytes* de dados são ocupados pelo índice *Hash* em cada uma das três tabelas (**FIGURA** 61).

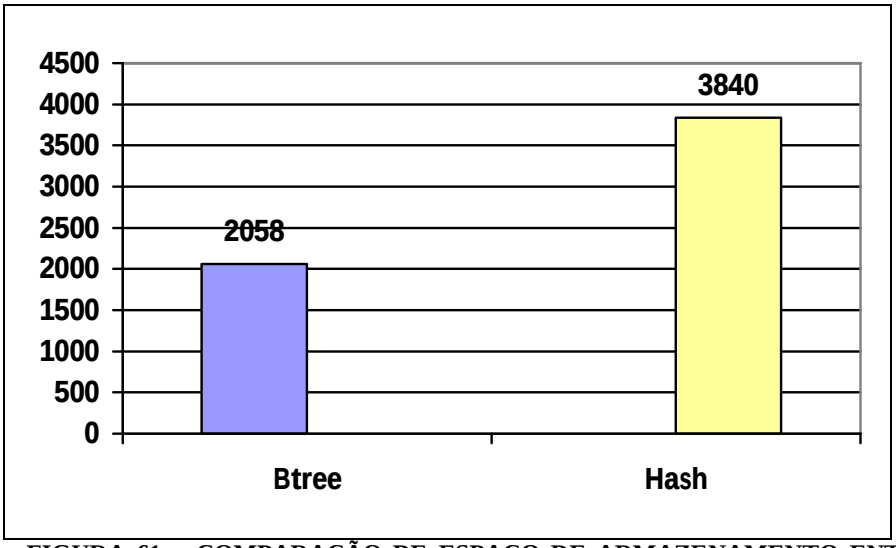


FIGURA 61 – COMPARAÇÃO DE ESPAÇO DE ARMAZENAMENTO ENTRE OS ÍNDICES *BTREE* E *HASH*.
As medidas de armazenamento estão em mega bytes.
Fonte: O autor (2012).

Os testes de tempo de resposta para recuperação dos registros que contenham relação com as sequências de pesquisa mostram que a junção *Nested Loop Join* combinada com o índice *Btree* possui o menor tempo médio de consulta (FIGURA 62), levando cerca de dezessete segundos para retornar os registros semelhantes, contra cinquenta e nove segundos do índice *Hash* para o mesmo método de junção. Esta diferença entre os índices e o método de junção *Nested Loop Join* deve-se ao fato da junção percorrer toda a tabela de segmento para cada registro da sequência de pesquisa, dessa forma quando combinado com o índice *Btree* o método de junção percorre o índice no lugar da tabela de dados, ganhando desempenho, pois se vale da estrutura de árvore binária para encontrar os valores correspondentes, além do fato da tabela estar ordenada por este índice. Já o índice *Hash*, não tem suporte à ordenação, e tem que realizar o cálculo de comparação para cada registro enquanto é percorrido.

O comparativo para o método de junção *Hash Join*, teve o índice *Hash* apresentando melhor desempenho quando comparado ao índice Btree no mesmo método, levando duas vezes menos tempo para encontrar os registros semelhantes, cerca de trintab e seis segundos contra oitenta e nove segundos para o Btree, porém ainda superior ao tempo encontrado pelo índice *Btree* no método de junção *Nested Loop* Join (FIGURA 62).

A diferença entre os índices *Btree* e *Hash* nos tempos de respostas para o método de junção *Merge Join*, apresentam pouca diferença, quando se comparado com os outros dois métodos. O tempo do índice *Hash* é de trinta e dois segundos, quanto o tempo para o índice *Btree* é de trinta e sete segundos, indicando ainda que o índice *Btree* com o método *Nested Loop Join* possui a melhor relação de tempo para recuperação de dados nesta relação de tabelas (FIGURA 62).

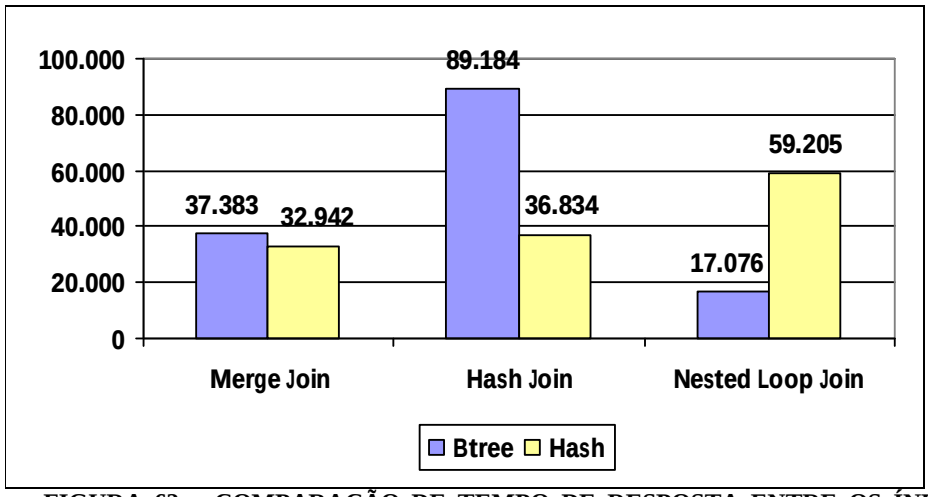


FIGURA 62 – COMPARAÇÃO DE TEMPO DE RESPOSTA ENTRE OS ÍNDICES *BTREE* E *HASH*.
Comparação de tempo de resposta entre os índices *Btree* e *Hash* para os três tipos de junções, *Merge Join*, *Hash Join*, e *Nested Loop Join*, expressa em milissegundos.
Fonte: O autor (2012).

4.9.1 TUNNING

Com o objetivo de melhorar a desempenho do tempo de resposta da Base de Dados, foi ajustado os seguintes parâmetros de configuração no SGBD PostgreSQL:

- *Shared_buffers*: Determina quanta memória dedicada o PostgreSQL pode usar para armazenar dados. Alterado de 24MB para 1GB.
- *Temp_buffers*: Define o número máximo de buffers temporários utilizados para cada sessão no banco de dados. Alterado de 8MB para 32MB.
- *Work_mem*: Especifica a quantidade de memória para ser usado por operações internas de classificação e tabelas de *hash* antes de mudar para arquivos temporários em disco. Alterado de 1MB para 1GB.

- *Effective_io_concurrency*: Define o número de I/O concorrente de disco que pode ser executado simultaneamente pelo PostgreSQL. Alterado de 1 para 3, relativo os três discos de armazenamento de dados.
- *Seq_page_cost*: Define a estimativa de custo do planejador para buscar de uma página no disco, ou uma série de páginas sequênciais. Alterado 1 para 3.
- *Random_page_cost*: Define a estimativa de custo do planejador de buscar páginas não sequenciais no disco. Alterado 4 para 3.
- *Cpu_tuple_cost*: Define a estimativa de custo do planejador para processamento em cada linha durante uma consulta. Alterado de 0.01 para 0.0051.
- *Cpu_index_tuple_cost*: Define a estimativa de custo do planejador para processar cada entrada de índice durante uma verificação de índice. Alterado de 0.005 para 0.0045.
- *Effective_cache_size*: Define o tamanho efetivo do cache de disco que está disponível para uma única consulta Alterado de 128MB para 256MB.
- Criação de uma *tablespace*, em memória para armazenar os índices das tabelas que contém os segmentos das bases de nucleotídeo.

Após os ajustes foi comparado o índice Btree como índice Btree tunnig no SGBD pelo o método de junção *Nested Loop Join*, e o ganho de desempenho apresentado foi significativo, reduzindo o tempo médio da consulta de dezessete segundos para duzentos e oitenta e um milissegundos (281ms). Esta drástica redução (

FIGURA 63) deve-se ao fato da junção *Nested Loop Join* percorrer os índices e não a tabela de dados, e os índices estarem agora em uma *tablespace* em memória e não mais no disco, o que torna a sua leitura muito mais veloz, pois a leitura em memória não apresenta a latência de IO presentes nos discos.

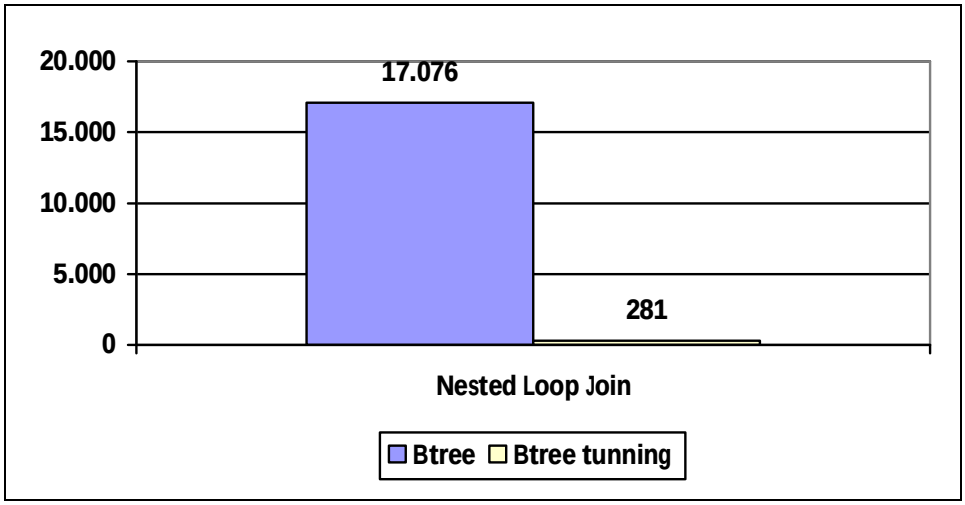


FIGURA 63 – COMPARAÇÃO DE TEMPO DE RESPOSTA ENTRE OS ÍNDICES *BTREE* E *BTREE TUNNING*.
Comparação de tempo de resposta entre os índices *Btree* e *Btree* após aplicação de *tunning* no SGBD PostgreSQL para o tipo de junção, *Nested Loop Join*, expressa em milissegundos.
Fonte: O autor (2012).

Sobre a utilização da *tablespace* em memória surge as seguintes dúvidas:

- **Como fica é o comportamento dessa área de armazenamento, podem ocorrer erros nos índices que estão em memória?**

A utilização de espaço de memória RAM como dispositivo de armazenamento não é recomendado para sistemas que não possuam memória com correção de erro (memórias ECC), pois podem ocorrem falhas no espaço de memória corrompendo os registros e tornando-os inutilizáveis ou ainda fornecer respostas incorretas as consultas. Contudo este não é o caso do sistema em uso, pois o mesmo contém 8GB de memória RAM com suporte a ECC.

- **Como é a recuperação das informações no caso de desligamento do servidor?**

Uma vez formada a Base de Dados, ela não é alterada constantemente, e este fato permite realizar uma copia dos índices em memória para uma área do disco, e

realizar o processo inverso de copiar do disco para memória durante o processo de inicialização do sistema operacional, o que leva em torno de um minuto para a base atual, assim ao iniciar o SGBD a *tablespace* em memória já contém os índices para serem utilizados.

5 CONCLUSÕES

Ao propor a Metodologia de Busca de Similaridade de Genes por Matriz de Co-ocorrência em Nucletotídeos, com objetivo de identificar sequências com alto grau de identidade em uma base de dados, de forma alternativa ao alinhamento de sequência, chama-se a atenção os seguintes fatos:

- Através da metodologia, foi possível identificar genes semelhantes na base de dados, de tal forma que não foi necessário realizar o alinhamento de sequências;

- Quando comparada as respostas dos testes de 10.000 genes com a metodologia proposta, e o BLAST que realiza alinhamento de sequências, em uma linha de corte de 50% de *score* relativo ao *self-score*, foi observado mais de 97% de correspondências entre as primeiras melhores respostas de ambos os métodos;

- Considerando todas as respostas do BLAST e da metodologia para o teste com 10.000 genes, em uma linha de corte de 70% de *score* relativo ao *self-score*, foram encontradas 65.198 respostas idênticas, totalizando quase 100% das respostas;

- A análise das medidas estatísticas obtidas através da co-ocorrência das bases de nucleotídeos mostrou-se adequade e precisa na busca de sequências gênicas com alto grau de identidade;

- O armazenamento em banco de dados relacional, das medidas estatística geradas, alinhando com uma estratégia de busca e indexação na base de dados além de tornar possível a analise das características, possibilitou a identificação rápidas das sequências semelhantes sem realizar a comparação em toda a base de dados;

- A otimização dos recuros disponíveis, bem com a construção de uma *tablespace* em memória para manipulação dos índices no SGBD, possibilitou a melhoria no tempo das respostas para cada consulta no banco de dados, passando de 17 segundo para menos de 1 segundo.

Aplicando a metodologia e observando todos os pontos positivos apresentados, bem como os resultados satisfatórios, é possível considerar que este trabalho contribui com uma forma alterativa de identificar sequências similares em uma base de dados de forma rápida e eficiente.

6 PERSPECTIVAS

A proposta da Metodologia de Busca de Similaridade de Genes por Matriz de Co-ocorrência em Nucleotídeos, não termina com a apresentação da metodologia ou pelo resutados satisfatórios encontrados. Melhorias podem e devem ser realizadas para que sistemas de Bioinformática sejam eficientes, mesmo com as mudanças de necessidades e volume de dados. E as seguintes melhorias são emergente na metologia proposta:

- Construção de uma interface de acesso para envio e consulta de sequências similares, possibilitando a interação com a metodologia e a base de dados de forma simples e interativa;

- Avaliação do tamanho das janelas móveis com variação maior de que uma base, possibilitando assim a redução do espaço de armazenamento e consequentemente uma área menor na *tablespace* em memória para manipulação dos índices de busca.

- Integração dos resultados com outras ferramentas de Bioinformática, como o BLAST e um Dot Plot para análise mais precisa dos resultados apresentados pela metodologia.

- Disponibilização de uma forma de acesso on-line da metodologia, para que outros sistemas possam utilizar os recuros porpostos sem a necessidade de implementar o sistema.

REFERÊNCIAS

ALMEIDA J. S. Insights into the immune transcriptome of the shrimp Litopenaeus vannamei: tissuespecific expression profiles and transcriptomic responses to immune challenge. American Physiological Society. 2006.

ALTCHUL, S. F., *et al*. Basic Local Alignment Search Tool. Journal of Molecular Biology 215, p. 403–410, 1990.

AUDE J. C.; LOUIS A.; OLLIVIER E.; RISLEY J. L. Massive sequence comparisons as a helpin annotating genomic Sequences. Genome Research, 2001

BARALDI, A. e PARMIGGIANI, F. An Investigation of the Textural Characteristics Associated with Gray Level Co-occurrence Matrix Statistical Parameters, 1995 – IEEE Transactions on Geoscience and Remote Sensing, 33(2) p.293–304.

BARBOSA-SILVA, *et al*. Clustering of cognate proteins among distinct proteomes derived from multiple links to a single seed sequence. BMC Bioinformatics 9:141, 2008.

BEVAN, M., *et al.* Analysis of 1.9Mb of contiguous sequence fromchromosome 4 of Arabidopsis thaliana. Nature, v.391, 1998.

CAMPIONE, M; WALRATH, K. The Java Tutorial: Object-Oriented Programming for the Internet. SunSoft Press, 1996.

CONCI, A.; AZEVEDO, E.; LETA, F. R. L. Computação Gráfica. Teoria e Pratica. Vol. 2. Elsevier, 2008.

CRICK, F. Central Dogma of Molecular Biology. Natute, v. 277, p. 561-663, 1970.

DATE, C. J. Introdução a Sistemas de Banco de dados. 7. ed. Campus, 2003.

DALE, J. W.; PARK, S. Molecular genetics of bacteria. 4. ed. John Wiley & Sons Inc, 2004.

DEVIJVER, P. A. e KITTLER, J. Pattern Recognition: A Statistical Approach. Prentice Hall Int, (1982).

DOUGLAS, K. DOUGLAS, S. PosgreSQL A comprehensive guide to building, programming and administering PostgreSQL databases. 1a. ed. Sams, 2003.

DUDA, R. O.; HART, P. E.; STORK, D. G. Pattern Classification. 2. ed. Jhon Wiley & Sons, 2001.

EBERT, D. S, Texturing and Modeling: A Procedural Approach, Cambridge: Academic Press, 1994.

FONG, Z. The Design and Implementation of the Postgres Query Optimizer. 1997.

FROTA, R. A. Avaliação de Algoritmos de Redes Neurais Artificiais em tarefa de Detecção de Novidades: Uma Abordagem Unificadora. Tese de Mestrado em Engenharia de Teleinformática, Universidade Federal do Ceará, Fortaleza, 2005.

GONZALEZ, R. C; WOODS, R. Digital Image Processing. 3a. ed. Prentice Hall. 2007.

GOSE, E., JOHNSONBAUGH, R. e Jost, S. Pattern Recognition and Image Analysis. Prentice Hall, 1996.

GREGORY T. R. The evolution of the genome. Elsevier, 2005.

GRIFFITHS, J. F. *et al.* Introduction to Genetic Analysis. 8. ed. WH Freeman, 2004.

HALL, M. A.; SMITH, L. A. Feature selection for machine learning: Comparing a correlation – based filter approach to the wrapper. In Flairs, 1999.

HARALICK, R. M., Statistical and Structural Approaches to Texture. Proceendings of the IEEE, vol. 67, n. 5, 1979.

HARALICK, R. M.; SHANMUGAM, K.; DINSTEIN, I. Textural Features for Image Classification, 1973 – IEEE Transactions on Systems, Man and Cybernetics, 3 p.610-620.

HAWKINS, D. Identification of Outliers. Chapman and Hall, London, 1980.

HORSTMANN, C. Big Java. Bookman, 2004.

HUGHES, G. F. On the mean accuracy of statistical pattern recognizers, IEEE - Transactions an Information Theory, vol. 11 - 14, No 1, pp. 55- 65, 1968.

HUGHEY, R.; KARPLUS, K Bioinformatics: a new field in engineering education. Journal of Engineering Education., p. 101-104, 2003.

JAIN, A. K.; DUIN, R. P. W. & MAO, J., 2000, "Statistical pattern recognition: a review". IEEE Transactions on Pattern Analysis and Machine Intelligence, v.22, n.1, pp. 4-37.

JAVA. What is Java? Disponível em: <http://www.java.com/en/download/faq/whatis_java.xml>. Último acesso em 09/01/2012.

JUNQUEIRA, L. C.; CARNEIRO, J. Biologia Celular e Molecular. 8. ed. Guanabara Koogan, 2005.

KASABOV, K. N. Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering. 2. ed. USA, Massachusetts Institute of Technology Press, 1996.

KORF, I. *et al.* BLAST. 1. ed. O'Relly & Associates, 2003.

LESK, A. M. Introdução A Bioinformática. 2. ed. Artmed, 2008.

LODISH, H *et al.* Biologia Celular e Molecular. 5. ed. Artmed, 2005.

LUSCOMBE, N. M.; GREENBAUM, D.; GERSTEIN, M. What is bioinformatics? A Proposed Definition and Overview of the Field, Methods of Information in Medicine 40, p. 346-358, 2001.

LYNCH M. The origins of genome architecture. Sinauer, 2007.

MATLAB. GLCM_Features4.m: Vectorized version of GLCM_Features1.m [With code changes]. Disponível em: <http://www.mathworks.com/matlabcentral/fileexchange/22354-glcmfeatures4-m-vectorized-version-of-glcmfeatures1-m-with-code-changes>. Último acesso em: 25/09/2010.

MARKOWITZ V. M.; MAVROMATIS K.; IVANOVA N. N.; CHEN I. A.; CHU K; KYRPIDES N. C. IMG ER: a system for microbial genome annotation expert review and curation, Oxford, v.25, n.17, p.2271-2278, 2009.

MARTINS JR., D. C. Redução de Dimensionalidade Utilizando Entropia Condicional Média Aplicada a Problemas de Bioinformática e de Processamento de Imagens. Dissertação de Mestrado, Ciência da Computação - Instituto de Matemática e Estatística da Universidade de São Paulo. São Paulo, 2004.

MEWES, H. W. *et al*. Overviewof the yeast genome. Nature, v.387, 1997.

MILANI, A. PostgreSQL Guia do Programador. Novatec, 2008.

MOUNT, D. W. Bioinformatics: Sequence and Genome Analysis. 2. ed. Cold Spring Harbor Laboratory Press Edition, 2004.

MUHAMAD, A. K.; DERAVI, F. Neural network texture classifiers using direct input co-occurrence matrices. IEEE Transactions on Pattern Analysis and Machine Intelligence. p. 117–120, 1993.

MURTEIRA, B. J. F. Análise exploratória de dados. Estatística Descritiva. Mcgraw-Hill, 1993.

NAVARRO. G. A guided tour to approximate string matching. ACM Computing Surveys, 33, p. 31-88, 2001.

NCBI. A Science Primer - Just the Facts: A Basic Introduction to the Science Underlying NCBI Resources. Disponível em: <http://www.ncbi.nlm.nih.gov/ About/primer/bioinformatics.html>. Último acesso em: 15/11/2011a.

NCBI. All Resources: Databases. Disponível em: <http://www.ncbi.nlm.nih. gov/guide/all/#databases_>. Último acesso em 18/11/2011b.

NCBI. Blast Program Selection Guide: Table of Content. Disponível em: <http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_ TYPE=ProgSelectionGuide>. Último acesso em 04/01/2012.

NCBI. The BLAST Sequence Analysis Tool. Disponível em: <http://www.ncbi.nlm.nih.gov/books/NBK21097>. Último acesso em 06/01/2012b.

NELSON, D. L; COX, M. M. Lehninger PRINCÍPIOS DE BIOQUIMICA. 4. ed. Sarvier, 2006.

OCHMAN H. Distinguishing the ORFs from the ELFs: short bacterial genes and the annotation of genomes. TRENDS in Genetics Vol.18 No.7. 2002.

OLAZAR, M. R. R. Algoritmos Evoluionarios Multiobjetivo para Alinhamento Múltiplo de Sequências Biológicas. Rio de Janeiro 2007. Dissertação de mestrado em Ciências em Engenharia Elétrica. Universidade Federal do Rio de Janeiro.

PEDRINI, H.; SCHWARTZ, R. W. Análise de imagens digitais: Princípios, algoritmos e aplicações. Thomson Learning, 2008.

PEREIRA, N, A. PostgreSQL: Técnicas Avançadas: Versão Open Source 7.X e 8.X: Solução para Desenvolvedores e Administradores de Banco de dados. 4. ed. Érica, 2007.

POSTGRESQL. The PostgreSQL Global Development Group. PostgreSQL 9.1 Documentation. Disponível em: <http://www.postgresql.org/docs/9.1/static>. Último acesso: 11/10/2011.

POSTGRESQL. The Path of a Query. Disponível em: <http://www.postgresql.org/docs/8.4/static/query-path.html>. Último acesso: 12/10/2011b.

POSTGRESQL. The Parser Stage. Disponível em: <http://www.postgresql.org/docs/8.4/static/parser-stage.html>. Último acesso: 12/10/2011c.

POSTGRESQL. The Query Tree. Disponível em: <http://www.postgresql.org/docs/8.4/static/querytree.html>. Último acesso: 12/10/2011d.

POSTGRESQL. The PostgreSQL Rule System. Disponível em: <http://www.postgresql.org/docs/8.4/static/rule-system.html>. Último acesso: 12/10/2011e.

POSTGRESQL. Planner/Optimizer. Disponível em: <http://www.postgresql.org/docs/8.4/static/planner-optimizer.html>. Último acesso: 12/10/2011f.

POSTGRESQL. Executor. Disponível em: <http://www.postgresql.org/docs/8.4/static/executor.html>. Último acesso: 12/10/2011g.

POSTGRESQL. Operator Optimization Information. Disponível em: <http://www.postgresql.org/docs/8.4/static/xoper-optimization.html>. Último acesso: 13/10/2011h.

POSTGRESQL. HISTORY. Disponível em: <http://www.postgresql.org/about/history/>. Último acesso: 06/01/2012.

PRODOCIMI, F. *et al.* Bioinformática: manual do usuário. Biotecnologia, Ciência & Desenvolvimento, Brasília, v. 5, n. 29, p. 12–25, Nov. 2002.

PRINCIPE, J. C.; EULIANO, N. R.; LEFEBVRE, W. C. Neural and Adaptive Systems: Fundamentals through Simulations. John Wiley & Sons, 2000.

PURVES, et al. Vida: A ciência da Biologia. 6. ed, Porto Alegre: Artmed Editora, 2001.

SHABAN, M. A.; DIKSHIT, O. Textural classification of high resolution digital satellite imagery. IEEE Transactions on Computers p. 2590–2592, 1998.

SETUBAL, J. C.; MEIDANIS, J. Introduction to Computacional Molecular Biology, Books/Cole Publishing Company, 1997.

SOUZA, J. A. Reconhecimento de padrões usando indexação recursiva. Tese de Doutorado em Engenharia de Produção, UFSC, Florianópolis, 1999.

SUDARSHAN S, K. Database System Concepts. Cap. 2, 13, 14, 26, McGraw-Hill, 5 ed. 2006.

TOW, D. SQL Tuning. 1a. ed. OREILLY & ASSOC, 2003.

TRIOLA, M. F. Introdução à Estatística. 7a. ed. Rio de Janeiro. Editora LTC, 1999.

WALKER, R.; JACKWAY, P; LONGSTAFF, I. Improving co-occurrence matrix feature discrimination. In Proc. of DICTA95 - 3rd International Conference on Digital Image Computing: Techniques and Applications, p. 643–648, 1995.

WATANABE. S. Pattern Recognition: Human and Mecanical. 1a. ed. Wiley, 1985.

WEISS, V. A. Estratégias de Finalização da Montagem do Genoma da Bactéria Diazotrófica Endofítica Herbaspirillum seropedicae SmR1. Curitiba, 2010. Dissertação (mestrado em Ciências - Bioquímica). Departamento de Bioquímica. Universidade Federal do Paraná.

WU, C. *et al.* BioGPS: an extensible and customizable portal for querying and organizing gene annotation resources. Licensee BioMed Central Ltd. 2009.

YOUNG, T. Y. Handbook of Pattern Recognition and Image Processing. 1. ed. Academic Press, 1994).

APÊNDICE I

METODOS JAVA, CÓDIGO FONTE

CÓDIGO FONTE 1 – CÓDIGO PARA CONVERSÃO DE NÚCLEÓTIDEO PARA NÚMERO.

```
public static int[][] nt2int(String sequence) {
 int matrizNt2Int[][] = new int[1][sequence.length()];
 String seq = sequence.toLowerCase();
 for (int i = 0; i < seq.length(); i++) {
   switch (seq.codePointAt(i)) {
     //a=97 c=99 g=103 t=116
     case 97:  matrizNt2Int[0][i] = 0; break;
     case 99:  matrizNt2Int[0][i] = 1; break;
     case 103: matrizNt2Int[0][i] = 2; break;
     case 116: matrizNt2Int[0][i] = 3; break;
    }
  }
 return matrizNt2Int;
}
```

FONTE: O autor (2011).

CÓDIGO FONTE 2 – CÓDIGO PARA CALCULO DA MATRIZ DE CO-OCORRÊNCIA.

```
private static double[][] CalcMatriz(double SGLD[][],int seq[][],int
hop){
    // double SGLD[][] = Recebe uma matriz com valores zeros
    // int seq[][]     = Recebe as sequências de DNA no forma de
    //                   matriz bidimensional(1) linha X (N) colunas
    // int hop         = Recebe o tamanho da distância entre os bases

    int size_x;  // Quantidade de linhas
    int size_y;  // Quantidade de colunas
    int B1;      // Variação da base B1
    int B2;      // Variação da base B2
    size_x = seq.length;
    size_y = seq[0].length;

    for (int i = 0; i < size_x; i++) {
    //Calcula matriz a co-ocorrência iniciada com Zeros
            for (int j = 0; j < size_y - hop; j++) {
                B1 = seq[i][j];       // Obtém a variação da base B1
                B2 = seq[i][j + hop]; // Obtém a variação da base B2
                SGLD[B1][B2] += 1;
            }
        }
        Matrix myMatrix = new Matrix(SGLD);
        SGLD = myMatrix.sum(myMatrix.transpose()).getData();
        SGLD = NormalizeMatriz(SGLD);
        return SGLD;
    }
```

FONTE: O autor (2011).

CÓDIGO FONTE 3 – CÓDIGO PARA NORMALIZAR A MATRIZ DE CO-OCORRÊNCIA.

```
Private static double[][] NormalizeMatriz(double SGLD[][]) {
        double fator;
        Matrix myMatrix = new Matrix(SGLD);
        fator = myMatrix.fullSum(myMatrix);

        for (int i = 0; i < SGLD.length; i++) {
            for (int j = 0; j < SGLD[0].length; j++) {
                SGLD[i][j] = SGLD[i][j] / fator;
            }
        }
   return SGLD;
}
```

FONTE: O autor (2011).

CÓDIGO FONTE 4 – CÓDIGO PARA CÁLCULO DE SMA.

```
private static double calcSMA(double SGLD[][]) {
    double result = 0;
    Matrix myMatrix = new Matrix(SGLD);
    myMatrix.setData(myMatrix.multiplyScalar(myMatrix).getData());
    //Eleva cada elemento da matriz ao quadrado;
    result = myMatrix.fullSum(myMatrix);
    //Soma todos os elementos da matriz;
    return result;
}
```

FONTE: O autor (2011), baseado em (MATLAB, 2010).

CÓDIGO FONTE 5 – CÓDIGO PARA CÁLCULO DA ENTROPIA.

```
private static double calcEntropy(double SGLD[][]) {
  double result = 0;
  Matrix myMatrix = new Matrix(SGLD);

  myMatrix.setData(myMatrix.multiplyScalar(myMatrix.log2Matrix()).
getData());
  myMatrix.setData(myMatrix.multiplyFactor(-1.0).getData());
  result = myMatrix.fullSum(myMatrix);

  return result;
}
```

FONTE: O autor (2011), baseado em (MATLAB, 2010).

CÓDIGO FONTE 6 – CÓDIGO PARA CÁLCULO DO CONTRASTE.

```
private static double calcContraste(double SGLD[][]) {
  double result = 0;
  double valor;
  Matrix myMatrix     = new Matrix(SGLD);
  Matrix myMatrixFind = new Matrix(myMatrix.findMatrix().getData());
  for(int i = 0; i < myMatrixFind.getData().length; i++) {
    valor = 0;
    valor = myMatrixFind.getData()[i][0]-myMatrixFind.getData()[i][1];
    valor = Math.pow(valor, 2);
    valor = valor * myMatrixFind.getData()[i][2];
    result += valor;
   }
   return result;
}

public Matrix findMatrix() {
        double[][] findData = new double[row * col][3];
        int w = 0;
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                if (this.getData()[i][j] != 0){
                    findData[w][0] = i;
                    findData[w][1] = j;
                    findData[w][2] = this.getData()[i][j];
                    w++;
                }
            }
        }
        Matrix matrixFind = new Matrix(findData);
        return matrixFind;
    }
```

FONTE: O autor (2011), baseado em (MATLAB, 2010).

CÓDIGO FONTE 7 – CÓDIGO PARA CÁLCULO DA VARIÂNCIA.

```
private static double calcVariance (double SGLD[][]){
  double[] res = new double[2];
  Matrix myMatrix = new Matrix(SGLD);
  double[][] valor_MatrixFind = myMatrix.findMatrix().getData();
  double[][] valor_k = new double[valor_MatrixFind.length][1];

  for (int i=0; i < valor_MatrixFind.length; i++){
    valor_k[i][0] = valor_MatrixFind[i][0]+valor_MatrixFind[i][1];
  }

  Matrix myMatrix_val_k = new Matrix(valor_k);
  double[][] valor_sum_pk=new double[myMatrix_val_k.maxMatrix()+1][1];

  for(int i=myMatrix_val_k.minMatrix();i<=myMatrix_val_k.maxMatrix();
i++){
    for(int j=0; j < valor_k.length; j++) {
      if (valor_k[j][0]==i){
      valor_sum_pk[i][0]+= valor_MatrixFind[j][2];
      }
    }
  }

  Matrix myMatrixFind_sum_pk = new Matrix(valor_sum_pk);
  double[][] valor_MatrixFind_sum_pk =myMatrixFind_sum_pk.findMatrix()
.getData();

 // Calc sum average----------------------------------------------
 for (int i=0; i<valor_MatrixFind_sum_pk.length; i++){
 res[0]+=valor_MatrixFind_sum_pk[i][0]*valor_MatrixFind_sum_pk[i][2];
}

 //Calc sum variance----------------------------------------------
 for (int i=0; i<valor_MatrixFind_sum_pk.length; i++){
   res[1]+= Math.pow((valor_MatrixFind_sum_pk[i][0]-result[0]),
2)*valor_MatrixFind_sum_pk[i][2];
 }
 return result;
}
```

FONTE: O autor (2011), baseado em (MATLAB, 2010).

CÓDIGO FONTE 8 – CÓDIGO PARA CÁLCULO DA CORRELAÇÃO.

```
private static double calcCorrelation(double SGLD[][]) {
  double result = 0;
  double valor_mu_x = 0;
  double valor_sigma_x = 0;
  double valor_mu_y = 0;
  double valor_sigma_y = 0;
  double valor = 0;
  double[][] Data_x = new double[SGLD.length][1];
  double[][] Data_y = new double[1][SGLD[0].length];

//------- Calc de valor_h_x
  for (int i = 0; i < SGLD.length; i++) {
    for (int j = 0; j < SGLD[0].length; j++) {
      Data_x[i][0] += SGLD[i][j];}
   }

   Matrix myMatrix_x = new Matrix(Data_x);
   Matrix myMatrixFind_x = new Matrix(myMatrix_x.findMatrix().
getData());

   for (int i = 0; i < myMatrixFind_x.getData().length; i++) {
     valor_mu_x += myMatrixFind_x.getData()[i][0] * myMatrixFind_x.
getData()[i][2];}

   for (int i = 0; i < myMatrixFind_x.getData().length; i++) {
    valor_sigma_x += (Math.pow((myMatrixFind_x.getData()[i][0] -
valor_mu_x), 2) * myMatrixFind_x.getData()[i][2]);}
//------- Calc de valor_mu_y e valor_sigma_y
  for (int i = 0; i < SGLD.length; i++) {
    for (int j = 0; j < SGLD[0].length; j++) {
      Data_y[0][j] += SGLD[i][j];}
    }
  Matrix myMatrix_y = new Matrix(Data_y);
  Matrix myMatrixFind_y = new Matrix(myMatrix_y.findMatrix().
getData());

  for (int i = 0; i < myMatrixFind_y.getData().length; i++) {
    valor_mu_y += myMatrixFind_y.getData()[i][1] * myMatrixFind_y.
getData()[i][2];
  }
  for (int i = 0; i < myMatrixFind_y.getData().length; i++) {
    valor_sigma_y += (Math.pow((myMatrixFind_y.getData()[i][1] -
valor_mu_y), 2) * myMatrixFind_y.getData()[i][2]);
  }
  Matrix myMatrix = new Matrix(SGLD);
  Matrix myMatrixFind = new Matrix(myMatrix.findMatrix().getData());
  int soma = 0;
  for (int i = 0; i < myMatrixFind.getData().length; i++) {
    valor += ((myMatrixFind.getData()[i][0] - valor_mu_x) *
(myMatrixFind.getData()[i][1] - valor_mu_y)) *
myMatrixFind.getData()[i][2];
  }

  result = valor / Math.sqrt(valor_sigma_x*valor_sigma_y);
  return result;
}
```

FONTE: O autor (2011), baseado em (MATLAB, 2010).

CÓDIGO FONTE 9 – CÓDIGO PARA CÁLCULO DA HOMOGENEIDADE.

```
private static double calcInverseDifferneceMoment(double SGLD[][]) {
  double result = 0;
  double valor;
  Matrix myMatrix = new Matrix(SGLD);
  Matrix myMatrixFind = new Matrix(myMatrix.findMatrix().getData());

  for (int i = 0; i < myMatrixFind.getData().length; i++) {
    valor = 0;
    valor = myMatrixFind.getData()[i][0]-myMatrixFind.getData()[i][1];
    valor = Math.pow(valor, 2);
    valor = valor + 1;
    valor = 1 / valor;
    valor = valor * myMatrixFind.getData()[i][2];
    result += valor;
  }
return result;
}
```

FONTE: O autor (2011), baseado em (MATLAB, 2010).

CÓDIGO FONTE 10 – CÓDIGO DE IMPORTAÇÃO DA BASE DE DADOS DE GENES PARA O MODELO DE BANCO DE DADOS RELACIONAL.

```
public static void UpGeneDB() throws Exception {
  ArrayList<SeqI> arquivo;
  SeqI itemArquivo;        String dir;
  File diretorio;          File diretorio_genoma;
  String[] lista_dir;      String[] lista_fasta;
  StringBuffer path        = new StringBuffer();
  StringBuffer path_genoma = new StringBuffer();
  StringBuffer sql         = new StringBuffer();
  String[] description;    String[] id;
  Boolean tipo;            db dbGenoma = new db();
  dir = "/home/administrador/sistema/genes/";
  diretorio = new File(dir);
  lista_dir = diretorio.list();
  for (int i = 0; i < lista_dir.length; i++) {
    path_genoma.append(dir); path_genoma.append("/");
    path_genoma.append(lista_dir[i]);
    diretorio_genoma = new File(path_genoma.toString());
    path_genoma      = new StringBuffer();
    lista_fasta      = diretorio_genoma.list();
    for (int j = 0; j < lista_fasta.length; j++) {
      path.append(dir);     path.append(lista_dir[i]);
      path.append("/");     path.append(lista_fasta[j]);
      arquivo = Fasta.readMany(path.toString(), 10240);
      path = new StringBuffer();
      for (int k = 0; k < arquivo.size(); k++) {
        itemArquivo = arquivo.get(k);
        description = itemArquivo.description().replace("[", ",").
Split(",");
        id = itemArquivo.id().replace("|", ",").split(",");
        if ((Integer.parseInt(id[2].split(":")[1].replace("c",
"").replace("-", ",").split(",")[0]) -
        Integer.parseInt(id[2]. split(":")[1].replace("c",
"").replace("-", ",").split(",")[1])) > 0)
        {tipo = true;
        } else {
          tipo = false;
        }
        sql.append("INSERT INTO organismo_gene(");
        sql.append("orggen_descricao, orggen_organismo, orggen_gene,
orggen_start, ");
        sql.append("orggen_end, orggen_complementar) ");
        sql.append("VALUES (?, ?, ?, ?, ?, ?)");
        String[][] valor = new String[6][2];
        valor[0][0] = "0"; valor[0][1] = description[0];
        valor[1][0] = "0"; valor[1][1] = id[1];
        valor[2][0] = "0"; valor[2][1] =
itemArquivo.subseq(itemArquivo.bounds().plus()).toString();
        valor[3][0] = "1"; valor[3][1] =
id[2].split(":")[1].replace("c", "").replace("-", ",").split(",")[0];
        valor[4][0] = "1"; valor[4][1] =
id[2].split(":")[1].replace("c", "").replace("-", ",").split(",")[1];
        valor[5][0] = "2"; valor[5][1] = tipo.toString();
        dbGenoma.query(sql.toString(), valor, 0);
        sql = new StringBuffer();
 }}}}
```

FONTE: O autor (2011).

CÓDIGO FONTE 11 - CÓDIGO DE CHAMADA PARA GERAÇÃO DOS SEGMENTOS E GERAÇÃO DAS CARACTERISTICAS

```
public static void UpGeneSementeHaralick(int tipo) throws Exception {
  ResultSet resultCodigo, resultGenoma;
  db dbCodigo      = new db();
  db dbGenoma      = new db();
  db dbSemente     = new db();
  String sql;                int codigo;
  String sequence = "";      String seq21    = "";
  double[][] Caracterisrtica; double[] DADOS = new double[6];

  sql = "SELECT orggen_codigo AS codigo FROM organismo_gene WHERE
ORDER BY orggen_codigo";

  resultCodigo = dbCodigo.query(sql, 1);
  while (resultCodigo.next()) {
    codigo = resultCodigo.getInt("codigo");
    sql = "SELECT orggen_codigo AS codigo, orggen_gene AS sequência
FROM organismo_gene WHERE orggen_codigo =" + codigo;
    resultGenoma = dbGenoma.query(sql, 1);

   if (resultGenoma.next()) {
     sequence = DNA.getSequence(tipo, 200, resultGenoma.getString
("sequência"));

     for (int i=0; i<=sequence.length()-21; i++){
     seq21 = sequence.substring(i, 21+i);
     Caracterisrtica = Haralick.HaralickDNA(seq21, 2, 0, 1, 0, 1, 4);
     for (int j = 0; j < Caracterisrtica[0].length; j++) {
     sql = "INSERT INTO organismo_gene_semente(orggensem_gene,
orggensem_valor, orggensem_caracteristica, orggensem_posicao,
orggensem_segmento) ";
       sql+= "VALUES("+codigo+", "+Caracterisrtica[0][j]+", "+(j+1)+",
"+(i+1)+", "+ tipo+")";
     dbSemente.query(sql, 0);
     }
    }
   }
  }
}
```

FONTE: O autor (2011).

CÓDIGO FONTE 12 – CÓDIGO DE GERAÇÃO DOS SEGMENTOS.

```
public static String getSequence(int tipo, int size, String sequence){
        int numero;
        int inicio;
        int meio;
        int fim;
        String retorno = sequence;

        switch (tipo){
            case 1:
                if(sequence.length()>size){
                    inicio = 0;
                    fim = size;
                    retorno = sequence.substring(inicio, fim);
                }
                break;

            case 2:
                if(sequence.length()>size){
                    meio = sequence.length()/2;
                    inicio = meio-(size/2);
                    fim    = meio+(size/2);
                    retorno = sequence.substring(inicio, fim);
                }
                break;

            case 3:
                if(sequence.length()>size){
                    inicio = sequence.length()-size;
                    fim = sequence.length();
                    retorno = sequence.substring(inicio, fim);
                }

                break;

        }
        return retorno;
}
```

FONTE: O autor (2011).

CÓDIGO FONTE 13 – CÓDIGO DE GERAÇÃO DAS CARACTERÍSTICAS DE CO-OCORRÊNCIA.

```
public static double[][] HaralickDNA(String sequence, int input_bits,
int output_bits, int hop, int direction, int origem, int size) {

// As entradas do método são:
// a)    sequence: Sequêcia para análise
// b)  input_bits: Quantos bits que representa a variação da
                   sequêcia
// c) output_bits: A resolucãoo de saída da matriz de co-ocorrência
// d)         hop: A distância entre as bases para calcular a matriz
                   de co-ocorrência, ela pode ser valores entre 0 e
                   size(image)-1
// f:   direction: O angulo no qual será calculada a matriz de co-
                   ocorrência valor 0 (zero);
// g:      origem: Identificação para Nucleotídeo, Aminoácido ou Códon
// h:        size: Tamanho da Matriz

        double[][] caracteristica = new double[1][6];
        int CO_size;    // Tamanho da matriz
        double SGLD[][];// Matriz de co-ocorrência
        int seq[][];    // Sequência em números

        switch (origem){
            case 1: seq = Genoma.nt2int(sequence); break;
            case 2: seq = Genoma.codon2int(sequence); break;
            case 3: seq = Genoma.aa2int(sequence); break;
            default: seq = Genoma.nt2int(sequence);}

        CO_size = size;
        SGLD = StartMatriz(CO_size);
        // Inicializa a matriz com o seu tamanho com valor zero;

        SGLD = CalcMatriz(SGLD, seq, hop);
        // Calcula Matriz

        //01 SMA
        caracteristica[0][0] = calcSMA(SGLD);

        //02 Entropy;
        caracteristica[0][1] = calcEntropy(SGLD);

        //03 Contraste
        caracteristica[0][2] = calcContraste (SGLD);

        double[] Sum_average_variance_antropy =
calcSumAverageVarianceEntropy(SGLD);

        //04 Variance;
        caracteristica[0][3] = Sum_average_variance_antropy[1];

        //05 Correlation;
        caracteristica[0][4] = calcCorrelation(SGLD);

        //06 Inverse_differnece_momen;
        caracteristica[0][5] = calcInverseDifferneceMoment(SGLD);

        return caracteristica;
    }
```

FONTE: O autor (2011).

CÓDIGO FONTE 14 – CÓDIGO DE EXPORTAÇÃO DE REGISTROS PARA O BLAST.

```
public static void UpBlastNTGeneFastaBig() throws Exception {
        ResultSet resultCodigo, resultCaracteristica;
        db dbCodigo = new db();
        db dbCaracteristica = new db();
        String sql;
        String Dados = "";
        String Seq;
        int codigo;
        File arquivo = new File("/home/administrador/sistema/blast/
ncbi-blast-2.2.24+/db/genes/nucleotideo_gene_db_group.fasta");
        FileOutputStream myFile = new FileOutputStream(arquivo);
        sql = "SELECT orggen_codigo AS codigo FROM organismo_gene
WHERE orggen_status = '1' ORDER BY orggen_codigo ASC";
        resultCodigo = dbCodigo.query(sql, 1);
        while (resultCodigo.next()) {
            codigo = resultCodigo.getInt("codigo");
            sql = "SELECT org_codigo AS organismo_codigo ,
orggen_codigo AS gene_codigo , org_descricao AS organismo ,
orggen_descricao AS gene , org_ncbi AS ncbi , orggen_gene AS valor,
orggen_start AS start , orggen_end AS end FROM organismo_gene LEFT
JOIN organismo ON orggen_organismo = org_ncbi WHERE orggen_codigo=" +
codigo;
            resultCaracteristica = dbCaracteristica.query(sql, 1);
            resultCaracteristica.next();
            Dados = ">ref|" + "(" +
resultCaracteristica.getString("organismo_codigo") + "-" +
resultCaracteristica.getString("gene_codigo") + ")";
            Dados += resultCaracteristica.getString("ncbi");
            Dados += "|:" + resultCaracteristica.getString("start") +
"-" + resultCaracteristica.getString("end");
            Dados += " " + resultCaracteristica.getString("gene");
            Dados += " [" +
resultCaracteristica.getString("organismo") + "]\n";
            myFile.write(Dados.getBytes());
            Seq = resultCaracteristica.getString("valor");
            for (int i = 0; i <= Math.abs(Seq.length() / 70); i++) {
                try {
                  Dados = Seq.substring(i * 70, i * 70 + 70) + "\n";
                } catch (Exception error) {
                  Dados = Seq.substring(i * 70, Seq.length()) + "\n";
                }
                myFile.write(Dados.getBytes());
            }
        }
        myFile.close();
    }
```

FONTE: O autor (2011).

CÓDIGO FONTE BANDO DE DADOS

CÓDIGO FONTE 15 – CÓDIGO SQL DE PESQUISA DE SEQUÊNCIAS.

```
CREATE OR REPLACE FUNCTION stp_select_gene_index(integer DEFAULT 1,
integer DEFAULT 20, integer DEFAULT 20)
  RETURNS SETOF select_semente_index AS
$BODY$

DECLARE val_gene INT;                -- Gene de Comparação Input
DECLARE val_gene_correspondecia INT; -- Mínimo de Correspondência por
                                       Genes
DECLARE val_gene_total INT;          -- Total de Genes a ser retornado
DECLARE val_string_exec VARCHAR(9000); -- String para execução
DECLARE val_record_exec RECORD;      -- Record para execução
DECLARE val_index_gene INT;          -- Gene do Record
DECLARE val_index_posicao INT;       -- Gene Posição do Record
DECLARE val_index_slice INT;         -- Gene Slice do Record
DECLARE val_score_gene INT;          -- Gene Score Calc
DECLARE val_index_gene_ant INT;      -- Gene Anterior Record
DECLARE val_index_posicao_ant INT;   -- Gene Posição Anterior Record
DECLARE val_index_slice_ant INT;     -- Gene Slice Anterior Record
DECLARE val_index_record RECORD;     -- Gene Record

BEGIN

  val_gene               := $1;
  val_gene_correspondecia := ($2 * 3); -- Minimo * 3;
  val_gene_total         := $3;
  val_score_gene         := 0;
  val_index_gene_ant     := 0;
  val_index_slice_ant    := 0;

  CREATE TEMPORARY TABLE index_gene_tmp(
  indgen_gene integer,
  indgen_posicao integer,
  indgen_slice integer); --1 START; 2 MIDDLE; 3 END

 -- Recupera os segmentos do início
 val_string_exec := 'INSERT INTO index_gene_tmp
 (SELECT comini_gene AS ‘gene’
 , comini_posicao    AS ‘posiscao’
 , 1                 AS ‘segmento’ --Segmento início
 FROM composicao_inicio inicio
 INNER JOIN
 (SELECT comtemp_valor AS valor
 FROM   composicao_temp                --Sequências pesquisadas
 WHERE  comtemp_gene = ' ||val_gene||' --Gene de pesquisa
 AND    comtemp_segmento = 1           --Segmento início
 ORDER BY comtemp_valor) genes
 ON inicio.comini_valor = genes.valor)';
 EXECUTE val_string_exec;

 -- Recupera os segmentos do meio
 val_string_exec := 'INSERT INTO index_gene_tmp
 (SELECT commeio_gene AS ‘gene’
 , commeio_posicao    AS ‘posiscao’
 , 2                  AS ‘segmento’ --Segmento meio
 FROM composicao_meio meio
```

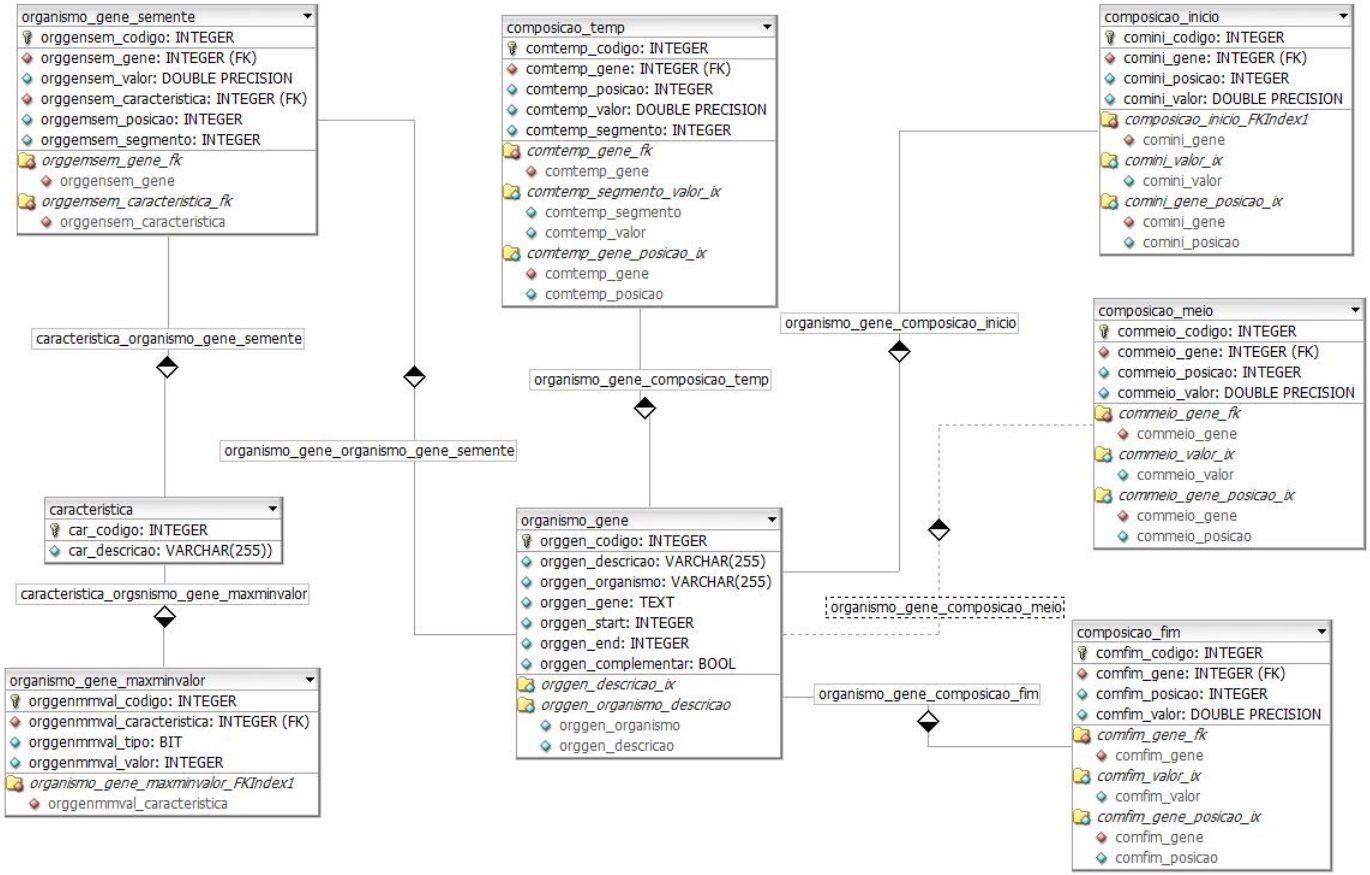
```
 INNER JOIN
 (SELECT comtemp_valor AS valor
 FROM   composicao_temp                --Sequências pesquisadas
 WHERE  comtemp_gene = ' ||val_gene||' --Gene de pesquisa
 AND    comtemp_segmento = 2          --Segmento meio
 ORDER BY comtemp_valor) genes
 ON meio.commeio_valor = genes.valor)';
 EXECUTE val_string_exec;

-- Recupera os segmentos do fim
 val_string_exec := 'INSERT INTO index_gene_tmp
 (SELECT comfim_gene AS ‘gene’
 , comfim_posicao    AS ‘posiscao’
 , 3                 AS ‘segmento’ --Segmento fim
 FROM composicao_fim fim
 INNER JOIN
 (SELECT comtemp_valor AS valor
 FROM   composicao_temp                --Sequências pesquisadas
 WHERE  comtemp_gene = ' ||val_gene||' --Gene de pesquisa
 AND    comtemp_segmento = 3           --Segmento meio
 ORDER BY comtemp_valor) genes
 ON fim.comfim_valor = genes.valor)';
 EXECUTE val_string_exec;

  -- Processamento do score
  CREATE TEMPORARY TABLE index_gene_top_tmp(
  indgen_gene integer,
  indgen_score integer);


FOR val_index_record IN SELECT indgen_gene, indgen_posicao,
indgen_slice FROM index_gene_tmp
WHERE indgen_gene IN (
SELECT indgen_gene
FROM index_gene_tmp
GROUP BY indgen_gene
HAVING count(indgen_gene) >= val_gene_correspondecia
ORDER BY count(indgen_gene) DESC
)
ORDER BY indgen_gene, indgen_posicao, indgen_slice

LOOP
 val_index_gene    := val_index_record.indgen_gene;
 val_index_posicao := val_index_record.indgen_posicao;
 val_index_slice   := val_index_record.indgen_slice;

 IF val_index_gene_ant = val_index_gene OR val_index_gene_ant = 0 THEN
   IF (val_index_posicao_ant + 1) = val_index_posicao OR
       val_index_slice_ant != val_index_slice THEN
        -- A Cada correspondencia sequencial 4 pontos
        val_score_gene := val_score_gene + 4;
    ELSE
        -- A Cada quebra de correspondencia -2 ponto
        val_score_gene := val_score_gene - ((val_index_posicao -
val_index_posicao_ant)*2);
    END IF;

 ELSE
      INSERT INTO index_gene_top_tmp(indgen_gene, indgen_score)
VALUES(val_index_gene_ant, val_score_gene);
```

```
      val_score_gene        :=0;
      val_index_posicao_ant :=0;
      val_score_gene        := val_score_gene - ((val_index_posicao -
val_index_posicao_ant)*2);

  END IF;

    val_index_gene_ant    :=val_index_gene;
    val_index_posicao_ant := val_index_posicao;
    val_index_slice_ant   := val_index_slice;

  END LOOP;

  INSERT INTO index_gene_top_tmp(indgen_gene, indgen_score) VALUES
(val_index_gene_ant, val_score_gene);
  -- Fim  Processamento do score

RETURN QUERY
SELECT indgen_gene, indgen_score FROM index_gene_top_tmp ORDER BY
indgen_score DESC, indgen_gene LIMIT val_gene_total OFFSET 0;

  DROP TABLE index_gene_tmp;
  DROP TABLE index_gene_top_tmp;

END

$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100
  ROWS 1000;
ALTER FUNCTION stp_select_semente_index(integer, integer, integer)
OWNER TO postgres;
```

FONTE: O autor (2011).

APÊNDICE II

DIAGRAMA DE ENTIDADE E RELACIONAMENTO



APÊNDICE III

DICIONÁRIO DE DADOS

TABELA 13 – Legenda

Boolean	Boolean lógico (true / false)
Bit	Sequência de bits de comprimento fixo
Souble precision	dupla precisão número de ponto flutuante
Integer	Valores inteiros de -2.147.483.648 até 2.147.483.647
Text	Sequência de caracteres de comprimento variável. Armazena textos.
Varchar	Sequência de caracteres de comprimento variável

TABELA 14 – Organismo_gene – Tabela para armazenamento das sequências de nucleotídeos dos genes.

ORGANISMO_GENE						
CAMPO	**TIPO**	**TAM.**	**NULL**	**DESCRIÇÃO**	**PK**	**FK**
orggen_codigo	Integer	4 bytes		Código do gene	*	
orggen_descricao	Varchar	255		Descrição do gene		
orggen_organismo	Varchar	255		Nome do organismo do gene		
orggen_gene	Text			Sequência de DNA		
orggen_start	Integer	4 bytes		Posição inicial do gene no genoma		
orggen_end	Integer	4 bytes		Posição final do gene no genoma		
orggen_complementar	Boolean			Indicativo de qual fita de DNA encontrase o gene		

TABELA 15 – Composicao_inicio – Tabela para armazenamento das composições das caractersticas do segmento de início.

COMPOSICAO_INICIO						
CAMPO	**TIPO**	**TAM.**	**NULL**	**DESCRIÇÃO**	**PK**	**FK**
comini_codigo	Integer	4 bytes		Código da composição	*	
comini_gene	Integer	4 bytes		Código do gene		*
comini_posicao	Integer	4 bytes		Posição dentro da composição		
comini_valor	Double Precision	8 bytes		Valor da Compposição		

TABELA 16 – Composicao_meio – Tabela para armazenamento das composições das caractersticas do segmento do meio.

COMPOSICAO_MEIO						
CAMPO	**TIPO**	**TAM.**	**NULL**	**DESCRIÇÃO**	**PK**	**FK**
commeio_codigo	Integer	4 bytes		Código da composição	*	
commeio_gene	Integer	4 bytes		Código do gene		*
commeio_posicao	Integer	4 bytes		Posição dentro da composição		
commeio_valor	Double Precision	8 bytes		Valor da Compposição		

TABELA 17 – Composicao_fim – Tabela para armazenamento das composições das caractersticas do segmento do fim.

COMPOSICAO_FIM						
CAMPO	**TIPO**	**TAM.**	**NULL**	**DESCRIÇÃO**	**PK**	**FK**
comfim_codigo	Integer	4 bytes		Código da composição	*	
comfim_gene	Integer	4 bytes		Código do gene		*
comfim_posicao	Integer	4 bytes		Posição dentro da composição		
comfim_valor	Double Precision	8 bytes		Valor da Compposição		

TABELA 18 – Composicao_temp – Tabela para armazenamento das composições das caractersticas do gene de pesquisa.

COMPOSICAO_TEMP						
CAMPO	**TIPO**	**TAM.**	**NULL**	**DESCRIÇÃO**	**PK**	**FK**
comtemp_codigo	Integer	4 bytes		Código da composição	*	
comtemp_gene	Integer	4 bytes		Código do gene		*
comtemp_posicao	Integer	4 bytes		Posição dentro da composição		
comtemp_valor	Double Precision	8 bytes		Valor da Composição		
comtemp_segmento	Integer	4 bytes		Indicação de qual ssegmento pertence a composicao		

TABELA 19 – Organismo_gene_semente – Tabela para armazenamento das características dos genes geradas.

COMPOSICAO_TEMP						
CAMPO	**TIPO**	**TAM.**	**NULL**	**DESCRIÇÃO**	**PK**	**FK**
orggensem_codigo	Integer	4 bytes		Código da semente	*	
orggensem_gene	Integer	4 bytes		Código do gene		*
orggensem_valor	Double Precision	8 bytes		Valor da Semente		
orggensem_caracteristica	Integer	4 bytes		Código da Característica		*
orggemsem_posicao	Integer	4 bytes		Posição da Semento		
orggemsem_segmento	Integer	4 bytes		Indicação de qual semgmento a semente pertence		

TABELA 20 – Caracteristica – Tabela para armazenamento das decrições das caracteristicas.

CARACTERISTICA						
CAMPO	**TIPO**	**TAM.**	**NULL**	**DESCRIÇÃO**	**PK**	**FK**
car_codigo	Integer	4 bytes		Código da característica	*	
car_descricao	Varchar	255		Descrição da caracteristica		

TABELA 21 – Organismo_gene_maxminvalor – Tabela para armazenamento dos valores máximo e mínimos das caracteriticas dos genes.

ORGANISMO_GENE_MAXMINVALOR						
CAMPO	**TIPO**	**TAM.**	**NULL**	**DESCRIÇÃO**	**PK**	**FK**
orggenmmval_codigo	Integer	4 bytes		Código do valor máximo/mínimo	*	
orggenmmval_caracteristica	Integer	4 bytes		Código da característica		*
orggenmmval_tipo	Bit			Indicação para valor máximo ou mínimo		
orggenmmval_valor	Double Precision	8 bytes		Valor máximo ou mínimo		

APÊNDICE IV

RESULTADOS DIVERGENTES ENTRE O BLAST E A BASE DE DADOS PARA 90% DE *SCORE* RELATIVO AO *SELF-SCORE*

TABELA 22 – VALORES DIVERGENTES PARA 90% DE *SCORE* RELATIVO AO *SELF-SCORE*.

Gene Query	Blast Self-score	Gene Blast	Base Self-score	Gene Base
hypothetical protein NCgl0467	0.982456140350877	hypothetical protein cgR_0587	1	hypothetical protein cg0571
ABC-type transporter	0.995944849959449	hypothetical protein cgR_1013	1	ABC transporter ATPase
NAD synthetase	0.973821989528796	NAD synthetase	1	NAD synthetase
hypothetical cytosolic protein	0.992924528301887	hypothetical protein COXBURSA331_A0674	0.996462264150943	hypothetical cytosolic protein
dihydroorotate dehydrogenase 2	0.927142857142857	dihydroorotate dehydrogenase 2	0.928571428571429	dihydroorotate dehydrogenase 2
Ribulose bisphosphate carboxylase (RuBisCO)	0.965753424657534	ribulose-1	0.969178082191781	ribulose bisphosphate carboxylase small chain
tyrosine recombinase XerD	0.954258675078864	integrase/recombinase	1	tyrosine recombinase XerD subunit
NADH-ubiquinone oxidoreductase chain 4L	0.934673366834171	NADH-ubiquinone oxidoreductase chain 4L	0.979899497487437	NADH-ubiquinone oxidoreductase chain 4L
integrase family protein	0.927440633245383	integrase family protein	1	integrase family protein
integrase catalytic subunit	0.994987468671679	integrase catalytic subunit	0.996240601503759	integrase catalytic subunit

F0F1 ATP synthase subunit gamma	0.957482993197279	ATP synthase F1	0.97108843537415	F0F1 ATP synthase subuni gamma
major cold shock protein	0.964788732394366	major cold shock protein	0.97887323943662	major cold shock protein
AsnC-family transcriptional regulator	0.918181818181818	AsnC-family regulatory protein	1	Leucine-responsive regulatory protein
hypothetical protein EAM_2344	0.929133858267717	hypothetical protein ETA_11520	1	UPF0115 protein yfcN
O2-independent coproporphyrinogen III oxidase	0.954214360041623	O2-independent coproporphyrinogen III oxidase	1	oxygen-independent coproporphyrinogen III oxidase
Uncharacterized protein yccV	0.921658986175115	heat shock protein HspQ	1	hypothetical protein EAM_1386
fused predicted transporter subunits of ABC superfamily: ATP-binding components	0.99581589958159	putative transporter fused subunits of ABC superfamily: ATP-binding components	0.997489539748954	ABC transporter ATP-binding protein
hypothetical protein E2348C_2466	0.994805194805195	hypothetical protein ECED1_2790	1	hypothetical protein ECP_2365
glycerol-3-phosphate transporter periplasmic binding protein	0.996710526315789	glycerol-3-phosphate transporter periplasmic binding protein	0.99890350877193	glycerol-3-phosphate transporter periplasmic binding protein
hypothetical protein ECP_0333	0.924295774647887	hypothetical protein ECS88_2097	0.934859154929577	hypothetical protein ECO26_3416
allantoin permease	0.993871297242084	putative allantoin transporter	0.997957099080695	allantoin permease
neutral amino-acid efflux protein	0.99746835443038	neutral amino-acid efflux protein	1	neutral amino-acid efflux protein
N-acetylmuramoyl-L-alanine amidase	0.998833138856476	putative amidase	1	N-acetylmuramoyl-L-alanine amidase

dTDP-glucose 4	0.995989304812834	dTDP-glucose 4	1	dTDP-glucose 4
hypothetical protein EcolC_3221	0.989100817438692	hypothetical protein S0357	1	predicted lipoprotein
TMAO reductase III (TorYZ)	0.978007761966365	TMAO reductase III (TorYZ)	0.981888745148771	cytochrome c-type protein torY
intracellular growth attenuator protein	0.996635262449529	putative membrane protein igaA-like protein	0.997981157469717	predicted inner membrane protein
hypothetical protein EcHS_A2131	0.967272727272727	hypothetical protein ECED1_2348	1	hypothetical protein ECO26_2890
outer membrane lipoprotein	1	outer membrane protein induced after carbon starvation	1	Slp family outer membrane lipoprotein
Não Informado	1	putative adhesin	1	Ig domain-containing protein
predicted fused mannitol-specific enzyme IIBC component of PTS	0.997847147470398	PTS system	1	putative fused mannitol-specific PTS enzymes: IIB component/IIC component
putative transcription regulator	0.996784565916399	putative AraC-like transcriptional regulator	1	putative AraC-like transcriptional regulator
putative outer membrane usher protein	0.998663101604278	fimbrial usher protein	1	truncated outer membrane protein
ISXoo11 transposase	1	transposase	1	ISXoo11 transposase