UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DEFINIÇÃO DO BANCO DE DADOS DOG PARA OBTENÇÃO DE ORTOLOGIA EM MÚLTIPLOS PROTEOMAS ATRAVÉS DO PADRÃO PVOM

Vinicius Garibaldi Martelli

SÃO CARLOS

2007

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DEFINIÇÃO DO BANCO DE DADOS DOG PARA OBTENÇÃO DE ORTOLOGIA EM MÚLTIPLOS PROTEOMAS ATRAVÉS DO PADRÃO PVOM

Vinicius Garibaldi Martelli

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências Exatas e de Tecnologia da Universidade federal de São Carlos, como parte dos requisitos para obtenção do titulo de Mestre em Ciência da Computação.
(Campo de pesquisa: Banco de Dados)

SÃO CARLOS

2007

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

M376db

Martelli, Vinicius Garibaldi.
Definição do banco de dados DOG para obtenção de ortologia em múltiplos proteomas através do padrão PVOM / Vinicius Garibaldi Martelli. -- São Carlos : UFSCar, 2007.
79 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2006.

1. Banco de dados. 2. Bioinformática. I. Título.

CDD: 005.74 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Definição do Banco de Dados DOG para a obtenção de ortologia em múltiplos proteomas através do padrão PVOM"

VINICIUS GARIBALDI MARTELLI

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

Prof. Dr. Mauro Biajiz
(Orientador – DC/UFSCar)

Prof. Dr. Nalvo Franco de Almeida Junior
(DCT/UFMS)

Prof. Dr. Guilherme Pimentel Telles
(ICMC/USP)

São Carlos
Agosto/2006

"Uma longa viagem começa com um único passo".

(Lao-Tsé)

Dedico este trabalho aos meus pais e à memória de meu avô.

AGRADECIMENTOS

Agradeço aos meus pais, à minha irmã e à Valéria pela paciência e incentivo.

Ao apoio dos meus amigos João, Santiago, Juninho, Fabiano, Anderson, Fábio, Neto e a três criaturinhas especiais: Sacha, Raika e Enzo.

Agradeço ao meu orientador, Prof. Dr. Mauro Biajiz, pela colaboração, ajuda, conselhos e amizade durante os trabalhos desenvolvidos durante a graduação e o mestrado.

Agradeço também ao Prof. Dr. Nalvo Franco de Almeida Junior e à Luciana Montera pelo auxílio na definição e desenvolvimento deste projeto.

Não poderia esquecer de prestar agradecimentos à PT Inovação, por ter me dispensado às sextas-feiras, em especial ao Takao pela compreensão.

Muito Obrigado!

MARTELLI, V. G. **Definição do Banco de Dados DOG para a obtenção de ortologia em múltiplos proteomas através do padrão PVOM.** Dissertação (Mestrado em Ciência da Computação) – Departamento de Computação (DC), Universidade Federal de São Carlos. São Carlos, 2006.

RESUMO

Este trabalho apresenta uma modelagem de um banco de dados e o processo criado para sua instanciação com o objetivo de armazenar as informações necessárias para a aplicação de uma nova forma de obtenção de regiões de genes contíguos que conservaram seus conteúdos, ordem e funções entre várias espécies de organismos procariontes durante o processo evolutivo, a fim de possibilitar a predição de genes e proteínas desconhecidas baseando-se em informações já existentes. Apresenta também uma ferramenta de consultas criada para viabilizar ao usuário o acesso a estas informações.

MARTELLI, V. G. **DOG Data Base definition for obtaining orthology in multiples proteoms using the PVOM pattern.** Thesis (Master degree in Computer Science) – Department of Computer Science (DC), Federal University of São Carlos (UFSCar), São Carlos, 2006.

ABSTRACT

This work presents a modeling of a data base and the process created for its instantiation with the objective to store the information necessary for the application of a new form of attainment of regions of contiguous genes that had conserved its contents, order and functions between some species of organisms prokaryotes during the evolutionary process, in order to make possible the prediction of genes and unknown proteins being based on existing information already. It also presents a tool of consultations created to make possible to the user the access to this information.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Em meados de 1990 a comunidade científica mundial deu início à era Pré-Genômica, etapa que tinha como principal meta o seqüenciamento do código genético de várias espécies, entre eles o genoma humano, a fim de obter conhecimentos que possibilitassem tratamentos genéticos, produção de novos medicamentos e outros avanços. Entretanto, os resultados conquistados ao final dessa fase representam uma imensa quantidade de informações que demandam interpretação.

Uma das descobertas provindas desta etapa está no reduzido número de genes que constituem um ser humano, cerca de 30 mil, um terço do que se imaginava anteriormente. Os genes contêm a informação de como fazer proteínas, mas não todas as instruções de como montá-las de modo que o resultado final seja um ser vivo. Há poucos genes para tamanha diversidade, sendo estimada a existência de 300 mil a 1 milhão de proteínas diferentes no organismo. O gene é uma pequena parte de um imenso processo que envolve milhares de substâncias e reações orgânicas.

A complexidade não está na quantidade de genes, mas na capacidade do organismo de combiná-los e transformar-se numa usina bioquímica produtora de proteínas. A missão de identificá-los, determinar sua localização, função e interação é o principal alvo da denominada era Pós-Genômica (LENGAUER, 2000).

Decifrar a função das proteínas é a tarefa fundamental da biologia molecular e da bioquímica, porém a crescente produção de informação genômica faz com que a bioinformática seja um elemento indispensável na descoberta dessas funções, principalmente baseando-se em informações já existentes para auxiliar na detecção de características de proteínas cujas funcionalidades são desconhecidas.

Métodos de predição de funções têm como base o relacionamento evolutivo existente entre as proteínas, uma vez que tais características tendem a se preservar durante a evolução. Um passo além de descobrir os genes que se conservaram é detectar as regiões de genes próximos que mantiveram uma certa ordem e funcionalidade.

Tal resultado é obtido a partir da comparação dos genomas de diferentes espécies e posterior análise de seus resultados para identificar semelhanças. Em ALMEIDA (2002) foi apresentada uma ferramenta que realiza a comparação do conteúdo gênico de duas espécies, obtendo como um dos principais resultados suas regiões preservadas; tal ferramenta, conhecida como EGG (Extended Genome-Genome Comparison), é descrita com maiores detalhes na seção 3.9.

Dando continuidade a esse trabalho MONTERA (2004) desenvolveu a ferramenta BAGRE (Base de Dados de Genes, Genomas e Regiões Ortólogas), apresentado na seção 3.10, que, a partir dos resultados fornecidos pelo EGG, obtém as regiões preservadas em várias espécies. Contudo, o método utilizado para esta detecção não garante o conjunto completo de respostas nem que a resposta seja a melhor possível.

A motivação deste trabalho emerge do mencionado estudo e da possível obtenção de conjuntos completos e complexos de respostas a partir do armazenamento dos resultados do EGG e da descoberta de padrões nestas informações, as quais foram implementadas por procedimentos em bancos de dados.

Este projeto tem como objetivo gerar avanços sobre os processos de bancos de dados voltados para a resolução de problemas encontrados na bioinformática, mais especificamente no reconhecimento de genes e regiões de genes que se preservaram entre as espécies durante a evolução, além de possibilitar que o usuário tenha acesso a estas informações de forma interativa, através de um sistema de consultas que disponibiliza, via WEB, o conteúdo do banco de dados.

O presente texto se encontra distribuído da seguinte forma. A seção 2 traz conceitos básicos da biologia molecular e da bioinformática, além das definições necessárias para o entendimento deste trabalho, finalizando com uma visão geral sobre bancos de dados genômicos. A seção 3 descreve o conceito geral dos algoritmos de alinhamento de seqüências genômicas e um aprofundamento do BLAST, contendo ainda os procedimentos realizados por diversas ferramentas, as quais identificam grupos de genes semelhantes dispersos em diferentes espécies. Na seção 4 são apresentados o modelo, a base de dados denominada DOG (Database of Orthologous Groups) e todo o processo de sua instanciação. Ainda nessa seção é descrito um padrão encontrado nas consultas sobre a base para a obtenção de regiões de genes que se preservaram em diferentes espécies, o PVOM (Padrão de Verificação de Ortologia Múltipla), e uma ferramenta para consulta, chamada ISMOG (Interactive Search of Multiple Orthologous Groups). A conclusão, as evoluções obtidas e sugestões para trabalhos futuros estão contidas na seção 5. Finalizando este documento estão dispostas as referências bibliográficas.

2. CONCEITOS DA BIOLOGIA MOLECULAR

Todos os organismos vivos possuem entidades básicas chamadas **genes**, que são as unidades herdadas com funções biológicas e estrutura. Um organismo herda esses genes de seus pais e transmite seus próprios genes para seus descendentes. O estudo da hereditariedade e variações dos organismos é chamado **genética**. Seus fundamentos foram estabelecidos no século XIX, no entanto a noção de que os filhos se assemelham a seus pais é usada há muito tempo. Agricultores e criadores de animais domésticos, por exemplo, através de cruzamentos, faziam o que se chama hoje de "seleção artificial", conseguindo variedades portadoras de características economicamente interessantes, como a produção de muito leite no gado, a maior quantidade de carne em galinhas, a alta produtividade de trigo e de outros vegetais utilizados como alimento (SASSON e SILVA, 2002).

Foi em 1.866 que o monge austríaco Gregor Mendel conduziu experiências de cruzamento entre plantas que o levaram às regras das quais nasceu a genética. Ele postulava que "cada característica genética de um organismo é condicionada por dois fatores, um proveniente do pai e outro da mãe". Mendel utilizava o termo fator, o qual foi substituído pelos pesquisadores que o sucederam por gene. Assim, tanto o pai quanto a mãe lhe forneciam informações hereditárias. Além disso, quando o indivíduo fosse reproduzir-se apenas um gene do par estaria na célula sexual. A reunião de uma célula sexual masculina com outra feminina restabelecia o par de genes necessário para codificar a característica (SBFIS, 1998).

Foi na década de 1.940 que se concluiu que os genes são feitos da substância DNA (ácido desoxirribonucléico). Crick e Watson, em 1.953, propuseram uma estrutura em dupla hélice conforme ilustrado na figura ao lado, concluindo que sua forma específica é fundamental para a função do DNA como um agente que armazena e transmite a informação genética. O ramo da biologia responsável pelo estudo do código genético (genes) de cada organismo é chamada de **biologia molecular**.

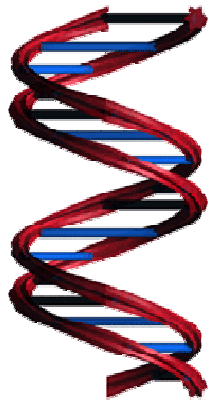


Figura 1. Molécula de DNA

A biotecnologia vem sendo aplicada há muito tempo nos processos de produção de pão e bebidas fermentadas. Além disso, os Astecas cultivavam em lagos variedades de algas utilizadas como fontes de alimentos. No final da década de 1970 esse ramo da biologia foi revolucionado e teve inicio a "nova" biotecnologia,

que consiste na aplicação dos princípios científicos da biologia molecular e da genética, através de agentes biológicos como microrganismos, células, enzimas e anticorpos, para prover bens como alimentos, bebidas, produtos químicos, energia, produtos farmacêuticos, pesticidas etc. Um exemplo de aplicação biotecnológica é a modificação direta do genoma do organismo alvo pela introdução de fragmentos de genes que possuem uma função conhecida. Sendo assim, por meio de engenharia genética o DNA que contém a informação para síntese de uma definida proteína de interesse (detalhada na próxima seção) pode ser transferido para outro organismo que então produzirá grandes quantidades da substância. Exemplos de substâncias assim desenvolvidas são a insulina humana e plantas resistentes a herbicidas. Outro uso importante da biotecnologia implica na produção de bactérias utilizadas para biodegradação de vazamentos de óleos ou lixos tóxicos (UCB, 2004).

Complementar à biotecnologia, a bioinformática utiliza técnicas computacionais para análises de caracterização molecular, integrando modelos matemáticos e estatísticos utilizados na interpretação e análise dos problemas biológicos (IAL, 2006).

2.1 SÍNTESE DE PROTEÍNAS

Existem dois tipos de ácidos nucléicos: o **DNA** (ácido desoxirribonucléico), responsável pela hereditariedade e o **RNA** (ácido ribonucléico), que, junto ao primeiro, são responsáveis pela síntese de proteínas, as quais possuem funções estruturais ou fisiológicas nos seres vivos. Todos esses elementos estão localizados nas **células**, que são as unidades básicas que determinam a estrutura e funções dos organismos (SEIBEL, 2000).

Os seres vivos, dependendo de sua organização, podem ser divididos em dois grupos: **procariontes** e **eucariontes**. Os primeiros possuem uma constituição celular mais simples, onde o material genético fica disperso pelo citoplasma, conforme pode ser visto na Figura 2A; dentro deste grupo estão as bactérias e algas azuis. Os eucariontes representam os demais organismos, possuindo um maior grau de sofisticação e o material genético de suas células é separado do citoplasma por uma membrana, ilustrada na Figura 2B (SILVA, 2000).

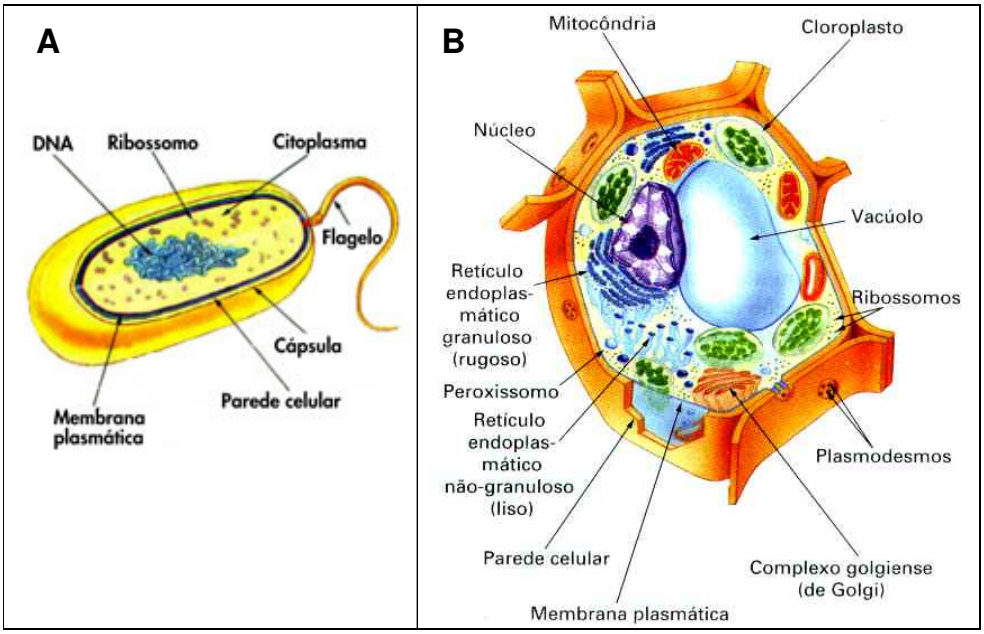


Figura 2 - A) Célula procariótica bacteriana; B) Célula eucariótica vegetal – retiradas de KESSLER (2006).

No núcleo das células eucarióticas e no citoplasma das procarióticas é onde se encontra o DNA, responsável pelo armazenamento da informação genética e que é composto de nucleotídeos dispostos em dois longos filamentos unidos, antiparalelos e em forma de hélice. Um nucleotídeo é formado por uma pentose (cinco carbonos), um grupo fosfato (P – phosphate) e uma base nitrogenada e é nomeado pela base nitrogenada que o compõe. Estas bases são classificadas como purinas, englobando a **Adenina** (A) e a **Guanina** (G) e pirimidinas, **Citosina** (C) e **Timina** (T). Uma purina só possui ligação com a pirimidina correspondente e assim, A e T são complementares, do mesmo modo que C e G, sendo ligadas por duas pontes de hidrogênio formando um par A-T, enquanto três pontes de hidrogênio formam um par G-C, conforme ilustrado na Figura 3 (GEWANDSZNAJDER e LINHARES, 2003). No RNA não existe a Timina, sendo substituída pela **Uracila** (U), a qual faz par com a Adenina.

As moléculas de DNA contêm as informações para todas as proteínas produzidas na célula. Um **códon** é formado pela seqüência de três bases ao longo da fita de DNA e consiste em uma “chave” química para codificar um dos 20 aminoácidos que compõem as proteínas (SILVA, 2000).

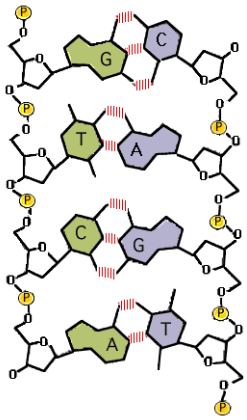


Figura 3 - Pareamento das bases do DNA, com a pentose, o fosfato e as pontes de hidrogênio – retirada de CHYNOWETH (2006).

Usando várias técnicas diferentes, os cientistas foram capazes de descobrir quais as etapas que ocorrem desde a transmissão das mensagens provenientes do código do DNA até a produção final de proteínas no citoplasma da célula (BSCS, 1995). Este procedimento está exemplificado na Figura 4.

Uma cadeia de molécula de DNA controla a síntese de um tipo específico de RNA. Os nucleotídeos adequados, ou seja, a Adenina, a Timina, a Citosina e a Guanina do DNA pareiam-se, respectivamente, com a Uracila, a Adenina, a Guanina e a Citosina do RNA. Dessa forma, a molécula de RNA formada copia a mensagem contida no DNA mediada pela enzima RNA-polimerase, processo conhecido como **transcrição** (ROBERTIS, 2001).

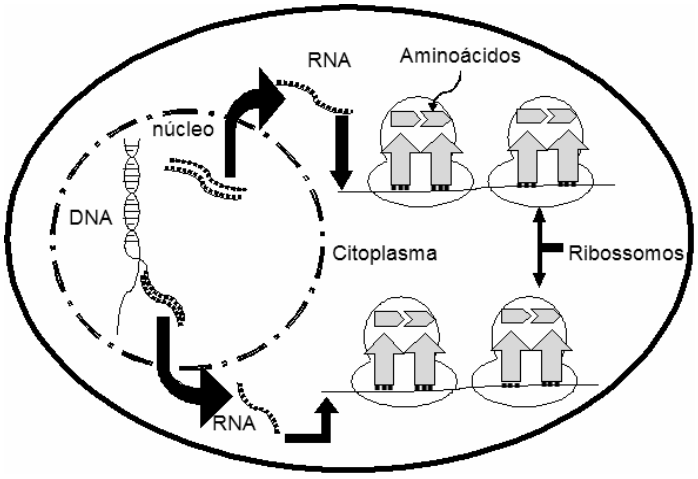


Figura 4 - Síntese de proteína.

Essa nova molécula de RNA chamada **RNA-mensageiro** (mRNA), migra para o citoplasma onde se prende a um ribossomo, formando um molde para a síntese de proteínas;

as moléculas conhecidas como **RNA-transportador** (tRNA) ligam-se a aminoácidos simples que estão no citoplasma. Cada aminoácido é apanhado por um tipo específico de tRNA que o encaixa no lugar adequado no mRNA. Portanto, o mRNA indica quais aminoácidos serão usados e qual o lugar que devem ocupar na molécula de proteína. Esta parte do processo de síntese de proteína é chamada de **tradução** (SPINGA, 1998).

Depois da síntese as novas moléculas de proteína se separam do ribossomo e do tRNA. Cada molécula deste RNA pega outra molécula de aminoácido, como foi feito anteriormente, trazendo-a novamente para o ribossomo. O tRNA deve também ser codificado de forma a reconhecer a mensagem do mRNA.

Desse modo, as mensagens do DNA servem para que a célula fabrique proteínas específicas em um processo que se repete continuamente, havendo sempre síntese dos mesmos tipos de proteínas. Em resumo: DNA "fabrica" RNA e RNA "fabrica" proteína.

Genes ou regiões codificantes são os trechos do DNA responsáveis pela síntese de proteínas. Para cada um destes trechos pode haver uma proteína equivalente, mudando, entretanto, o alfabeto de que são constituídos. Genes e DNA possuem o mesmo alfabeto de bases nitrogenadas (A, T, C e G); já a unidade básica da proteína é o **aminoácido**, codificado pela seqüência de três bases do DNA. Um exemplo fictício: considerando que uma proteína seja formada pela seqüência XYZ de aminoácidos, que os trios de bases TCG, TAC e GTC codifiquem respectivamente os aminoácidos X, Y e Z, e que exista um trecho de uma seqüência de DNA AATCGTACGTCAG, a região do DNA que corresponde ao gene seria TCGTACGTC. O conjunto de genes de um ser vivo é chamado de **genoma** enquanto que ao conjunto de proteínas preditas pelos genes dá-se o nome de **proteoma**.

2.2 BIOLOGIA COMPUTACIONAL

O seqüenciamento do genoma das mais variadas espécies exige dedicação humana, uma vez que a tecnologia existente restringe a metodologia de seqüenciamento. Atualmente os métodos utilizados sequenciam apenas uma parte (500 bases nitrogenadas) do DNA por vez (LEMOS, 2003). Dessa forma, para o DNA ser totalmente seqüenciado é quebrado em fragmentos que são montados um a um a fim de obter a cadeia inicial completa. Este processo não é trivial, pois freqüentemente ocorrem erros no seqüenciamento e problemas durante a quebra e cópia destes fragmentos, fazendo com que partes do genoma não sejam devidamente seqüenciadas, vindo a causar problemas posteriores na montagem do mesmo. Quando tais

fragmentos são agrupados formam-se os ***contigs*** e a união de todos os *contigs* formam um genoma.

Alguns projetos de seqüenciamento, entretanto, não visam obter o genoma completo da espécie abordada devido à sua complexidade, à sua extensão ou por interesse, obtendo apenas as regiões mais importantes que são partes do genoma que supostamente traduzem proteínas.

A fim de demonstrar tal grandeza de dados, o genoma humano é composto por aproximadamente 3 bilhões de bases do DNA. Cada seqüência de DNAdo genoma humano possui entre 50 a 250 milhões de pares de bases, sendo que o número de registros aumenta a cada ano, inviabilizando a busca desses dados sem o auxílio de computadores. E não são apenas esses dados que devem ser armazenados, mas também as relações existentes entre as bases, como estão dispostas, quais conjuntos de bases formam um gene, sua disposição e localização, qual a função de cada gene, no que ele poderá influir durante a vida do organismo em que está inserido, quais proteínas ele codifica, entre outros.

O DNA é formado por uma fita dupla, possuindo ambas orientações opostas sendo cada fita o complemento reverso da outra. Tais fitas recebem a denominação de positiva (+) e negativa (-). Independentemente da orientação da fita em que foi obtido, cada gene possui uma proteína predita equivalente, disposta seqüencialmente no proteoma, possuindo um serial correspondente à sua posição, como pode ser visto na Figura 5.

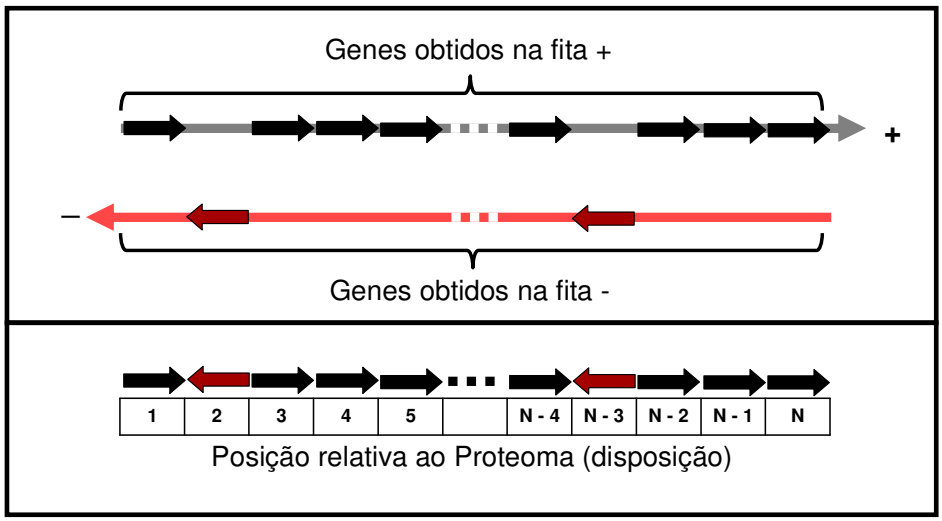


Figura 5 - Disposição e orientação de genes.

Do ponto de vista de um cientista da computação, a dupla hélice de DNA é um engenhoso sistema de armazenamento e transmissão de informação. Como no alfabeto binário

{0,1} usado em computadores, o alfabeto de 4 letras {A,T,C,G} pode codificar mensagens complexas quando dispostos em longas seqüências (AUNG, 2001).

O tamanho de um gene corresponde ao seu número de pares de bases (**bp** – base pairs) e é obtido pela posição final deste no genoma, subtraído da sua posição inicial, a cujo resultado é somado o valor 1 como fator corretivo. Ou então, multiplicando-se o número de aminoácidos (AA) por 3, representando os três pares de bases necessários para formar um aminoácido, uma vez que uma seqüência de aminoácidos tem 1/3 do tamanho do seu trecho de ácidos nucléicos. Fórmula denotada na Figura 6. O exemplo mostra uma proteína com 17 aminoácidos (AA) predita por um gene que tem como coordenadas inicial e final na fita de DNA, 35844 e 35894, respectivamente. Em ambas prova-se que seu tamanho equivale a 51 bp.

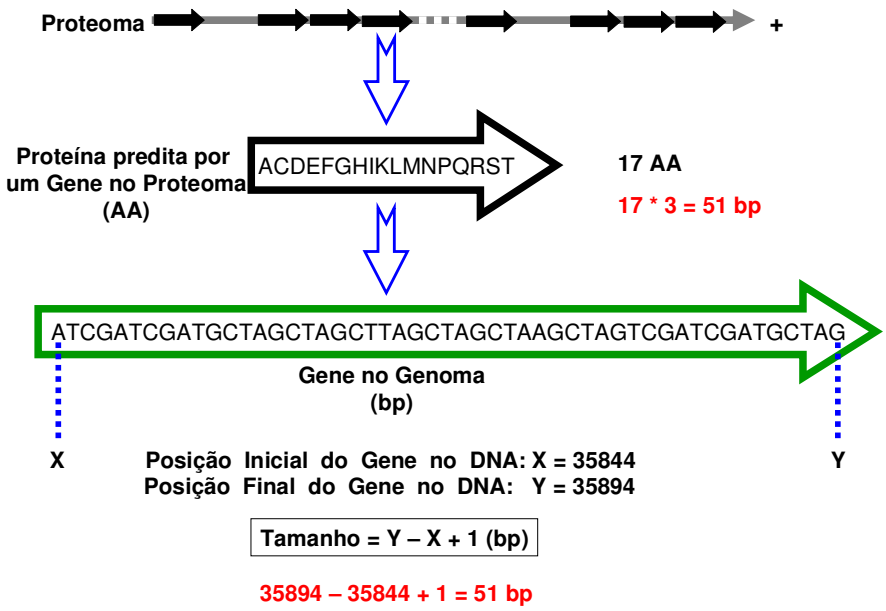


Figura 6 - Tamanho de um gene.

Os organismos possuem, cada qual, seu conjunto de genes, o que os tornam diferentes uns dos outros, porém, devido à evolução das espécies, os seres vêm sofrendo modificações e com eles, seus genes. Entretanto, não é raro encontrar genes semelhantes que codificam proteínas que desempenham uma mesma função em seres completamente diferentes na escala evolutiva, ou mesmo denotam a maneira como tal gene vem sendo transmitido no decorrer dos tempos.

Existem dois tipos de homologia: a **paralogia,** que consiste em uma duplicação gênica

ocorrida numa mesma linhagem e a **ortologia,** originada da especiação, ou seja, da geração de espécies a partir de ancestral em comum (JENSEN, 2001). Como pode ser visto na Figura 7, um gene ancestral A da espécie W sofre uma duplicação originando os genes A1 e A2, os quais possuem entre si uma relação de paralogia. Em um outro momento da evolução a espécie W deixa de existir devido a uma especiação, originando duas novas espécies X e Y. Assim, os genes A1x e A2x continuam parálogos, bem como A1y e A2y, porém os genes A1x e A2x são ortólogos em relação a A1y e A2y.

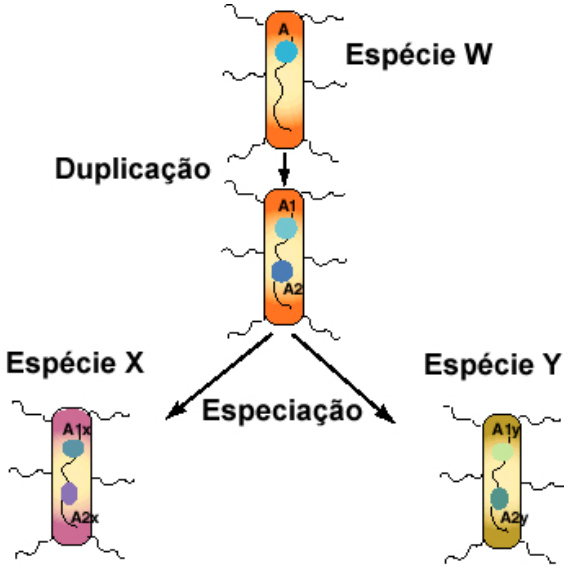


Figura 7 - Especiação X Duplicação.

Os genes que descendem de um ancestral em comum e desempenham uma mesma função são chamados de **genes homólogos**.

Quando genes homólogos são encontrados em um mesmo genoma, denominam-se de **genes parálogos**. Sua detecção é obtida comparando as proteínas preditas de uma espécie com ele mesmo e analisando as semelhanças dos genes que se localizam em coordenadas diferentes.

Já um par de genes homólogos pertencentes a proteomas diferentes recebem o nome de **genes ortólogos (GO)**, enquanto que os **genes ortólogos múltiplos (GOM)** são formados por três ou mais genes de proteomas distintos, tais que, qualquer par de genes deste conjunto forma um GO.

Um **gene específico** consiste em um gene pertencente exclusivamente a um genoma, ou seja, um gene x de um genoma X é considerado específico em relação a outro genoma Y caso não haja nenhum gene y pertencente a Y ortólogo a x (ALMEIDA, 2000).

Considerando a Figura 8, são apresentadas 3 espécies, X, Y e Z, e um conjunto de genes ortólogos ilustrados por [A1x, A1y], [A1x, A1z], [A1y, A1z], [A2x, A2y], [A3x,A2y], [A2y, A2z] e [A4x, A3z]. Todavia, há apenas uma GOM nesse exemplo: [A1x, A1y, A1z], pois é o único caso em que todos os genes pertencentes a espécies diferentes são ortólogos entre si. O gene A4z é específico a Z em relação a X e Y, uma vez que ele não se relaciona com nenhum outro gene das espécies X e Y.

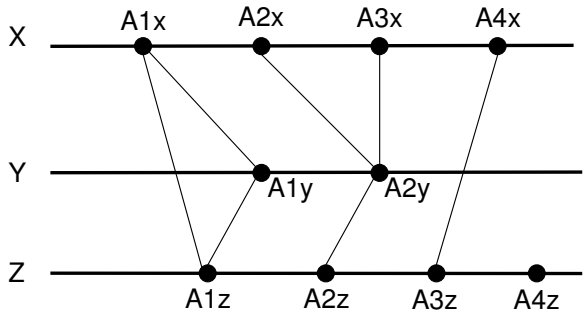


Figura 8 - GO, GOM e gene específico.

Tais relacionamentos podem ser considerados para genes, individualmente, assim como para **regiões de genes consecutivos (RGC)**, ou seja, trechos dos genomas que contêm um conjunto de genes próximos, onde a ordem e a funcionalidade são preservadas, bem como suas coordenadas de início, independentemente da orientação da fita em que foi obitda. Dependendo da proximidade considerada entre os genes, o próprio genoma pode ser visto como somente uma RGC. Conforme a Figura 9, supondo-se um valor de proximidade **α**, caso a distância entre dois genes seja superior a α, estes não farão parte da mesma região. Uma região apenas pode conter elementos cujas distâncias entre os adjacentes seja inferior ou igual ao valor α estipulado, por isso, A5 e V1 não fazem parte da mesma RGC.

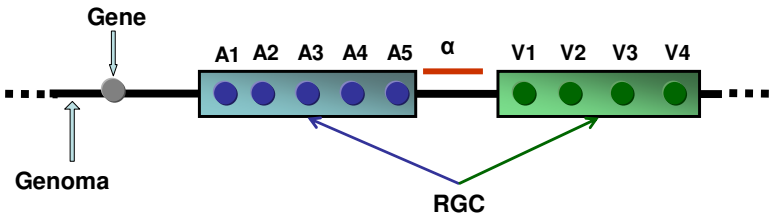


Figura 9 - RGC.

A ortologia também pode estar presente entre RGCs de organismos distintos, formando **regiões ortólogas (RO)**, as quais são compostas por um **par** de RGCs de genomas

diferentes, ortólogas entre si. Possuem aproximadamente o mesmo número de genes, além de serem descendentes de uma mesma região ancestral.

Uma **região ortóloga múltipla (ROM)** consiste de um conjunto de RGCs de três ou mais proteomas distintos, tais que, qualquer par de RGCs deste conjunto forma uma RO. A Figura 10 exibe três RGCs muito similares de três genomas diferentes, formando uma ROM, simbolizada pela figura adjacente (MONTERA, 2004).

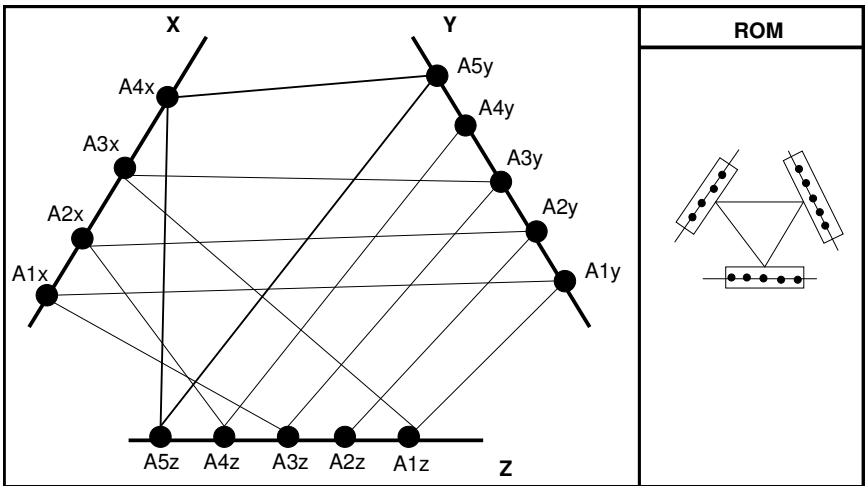


Figura 10 - ROM entre 3 proteomas e sua representação por RGCs.

Um passo além de encontrar as ortologias entre dois organismos é identificá-las entre vários organismos diferentes. É nesse contexto que esse trabalho se insere: na detecção de regiões de genes que se preservaram em diversos organismos, baseando-se principalmente nos trabalhos de ALMEIDA (2002) e de MONTERA (2004), os quais empenham-se em resolver, respectivamente, a comparação entre dois genomas e a comparação simultânea de vários genomas.

2.3 BANCOS DE DADOS GENÔMICOS

As informações sobre as seqüências de DNA e de proteínas juntamente com suas anotações, estão armazenadas em grande parte em arquivos texto, sem uma formatação padrão definida. Somente em algumas instituições estes dados são disponibilizados de uma forma mais segura e organizada, através da criação e manutenção de bases de dados, as quais têm como característica o rápido crescimento do seu volume de dados (BAXEVANIS, 2001). Como exemplo, pode-se citar o crescimento do GenBank da NCBI (National Center for

Biotechnology Information) (NCBI, 2006), cujo tamanho dobra aproximadamente a cada 15 meses (BENSON *et al*., 2004; BENSON *et al*., 2006). Tais bases de dados possuem diferentes propósitos, esquemas, estruturas de armazenamento e métodos de acesso. Entretanto, a maioria delas armazena as seqüências em formato de arquivo ("flat files"), embora utilizem o nome de bases de dados.

As seqüências armazenadas em um banco de dados irão caracterizar o seu tipo. Caso as seqüências em questão sejam de nucleotídeos, tem-se um banco de dados de nucleotídeos como o americano GenBank (GENBANK, 2006; WHEELER *et al.*, 2006; WHEELER *et al.*, 2003), o europeu EMBL Nucleotide Database da EMBL (European Molecular Biology Laboratory) (EMBL, 2006; COCHRANE *et al.*, 2006) e o japonês DDBJ (DNA Data Bank of Japan) (DDBJ 2006); recentemente a INSDC (International Nucletotide Sequence Database Collaboration) (BAIROCH *et al.*, 2005) vem buscando unificar os três bancos mencionados numa tentativa de centralizar as informações, tratar as redundâncias e garantir dados padronizados e de qualidade. Seu crescimento, destacado por participações de cada instituição, pode ser observado na Figura 11.

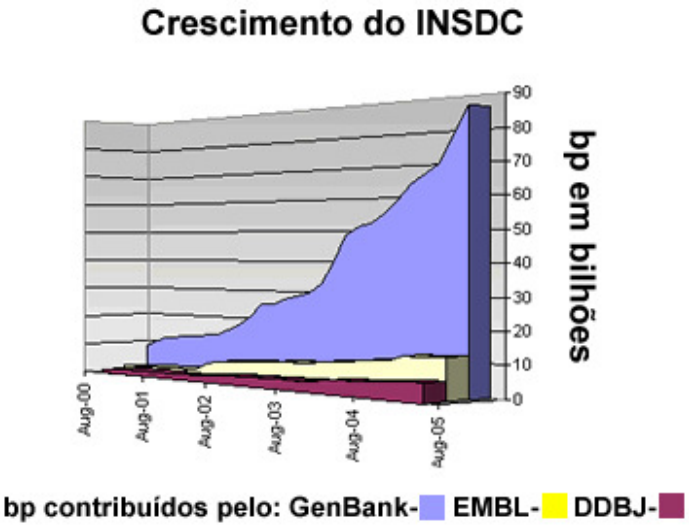


Figura 11 - Gráfico de crescimento do INSDC – retirada de BAIROCH *et al.* (2005).

No entanto, se o produto referido for proteínas, trata-se de uma banco de dados protéicos, como o TrEMBL da EMBL, o SwissProt da SIB (Swiss Institute of Bioinformatics) (SWISS-PROT, 2006) e o PIR (Protein Information Resource) da NBRF (National Biomedical Research Foundation) (PIR, 2006) e o Genbank. Com os mesmos objetivos do INSDC, a UniProt (The Universal Protein Resource) (WU *et al.*, 2006) unificou as informações contidas

nestes três bancos protéicos.

O TIGR (The Institute for Genomic Research) mantém também bases de dados contendo seqüências de DNA e de proteínas tratadas de forma a manter sempre a qualidade dos dados de suas pesquisas (TIGR, 2006).

A cada novo gene obtido pelo GenBank, é atribuído um identificador denominado **GI** (GenInfo Identifier), consistindo em um número inteiro simples que distingue cada gene, nunca sendo reaproveitado ou repetido para outro gene/organismo e mesmo que as seqüências codificadoras se diferenciem por apenas um par de bases, ambas possuirão GIs diferentes (BENSON *et al*., 2004).

Com o propósito de busca e alinhamento de seqüências, muitos desses sistemas de banco de dados utilizam como principal ferramenta o BLAST, que será detalhado na seção seguinte. Para outros objetivos, tais como análises e mineração de dados, são utilizadas diversas outras ferramentas que podem ser consultadas em CHEANG et al. (1994) e GALISSON (2001).

3 AGRUPAMENTO DE ENTIDADES ORTÓLOGAS

Esta seção apresenta inicialmente os conceitos básicos dos algoritmos de alinhamento de seqüências e seu exemplo mais representativo, o BLAST. Em seguida serão descritas ferramentas que têm por finalidade obter resultados semelhantes aos almejados pelo projeto ora desenvolvido, porém, utilizando algoritmos, procedimentos e tratamentos diferentes.

3.1 ALINHAMENTO DE SEQÜÊNCIAS

Em algumas ferramentas de comparação de seqüências biológicas como o BLAST (*Basic Local Alignment Search Tool*)(seção 3.1.2), descritos adiante neste capítulo, faz-se uso de alinhamentos de seqüências para a análise de similaridades.

Um alinhamento é uma forma de se comparar seqüências biológicas (DNA ou proteína). Ao se estabelecer um alinhamento entre essas seqüências, é possível descobrir se elas estão evolutivamente relacionadas ou não.

Uma seqüência biológica é representada através de uma cadeia de caracteres escrita com um determinado alfabeto[1]. Assim, comparar duas seqüências biológicas é equivalente a comparar duas cadeias de caracteres. Exemplo:

ACCTATGCAC
ACCATGCAC

Neste exemplo as seqüências possuem tamanhos diferentes. Por isso, é preciso igualar os tamanhos usando um caractere nulo "-" conhecido como *gap* ou espaço. Com a inserção do caractere nulo obtém-se:

A C C T A T G C A C
A C C - A T G C A C

A seguir, alinhamento é formalmente definido, de acordo com ALMEIDA (2002).

[1] Uma seqüência de DNA é escrita com um alfabeto de quatro letras, referentes às bases nitrogenadas. Uma seqüência de proteína é escrita com um alfabeto de vinte letras, sendo que cada letra corresponde a um dos vinte diferentes aminoácidos existentes na natureza.

Definição de Alinhamento: Dadas as seqüências $s = s_1 \ldots s_m$ e $t = t_1 \ldots t_n$, com símbolos pertencentes ao alfabeto Σ, e com m, n ≥ 0, um alinhamento de s e t é um mapeamento de s e t nas seqüências s' e t', respectivamente, cujos símbolos pertencem ao alfabeto $\Sigma' = \Sigma \cup \{ - \}$, onde o símbolo '-' é chamado de espaço, tal que:

1. $|s'| = |t'| = l$;
2. a remoção dos espaços de *s*' e *t*' leva a *s* e *t*, respectivamente; e
3. não é permitida a condição $s_i' = - = t_i'$, $1 \leq i \leq l$.

No alinhamento entre duas seqüências, o pareamento (de bases no caso de DNA e de aminoácidos no caso de proteínas) é estabelecido com base em um esquema de pontuação, onde se procura alinhar a seqüência com o objetivo de obter a pontuação mais alta para medidas de similaridade.

Considerando as seqüências $s = s_1 \ldots s_m$ e $t = t_1 \ldots t_n$, com símbolos pertencentes ao alfabeto Σ, um esquema de pontuação é dado por uma tupla (*p*, *g*) onde a função $p : \Sigma x \Sigma \rightarrow \mathbb{R}$ determina a pontuação de cada par de caracteres alinhados e *g* é usado para penalizar *gaps* ($g < 0$ na maioria das vezes). O esquema de pontuação fornece um valor numérico a cada alinhamento possível. Sendo (*s*', *t*') o alinhamento das seqüências *s* e *t*, se adiciona *p(a, b)* cada vez que um caractere *a* de *s*' está pareado com um caractere *b* de *t*'. Quando um caractere de *s*' ou *t*' está alinhado a um *gap*, *g* é agregado à pontuação. A pontuação total denotada como *score* (*s*', *t*') é a soma de todas as pontuações em todas as posições do alinhamento (*s*', *t*').

Sendo ψ o conjunto dos possíveis alinhamentos, a semelhança entre *s* e *t* é dada por:

$$sem(s,t) = \max_{(s',t') \in \psi} score(s',t')$$

Para uma seqüência de DNA, a pontuação de um alinhamento é ditada por uma matriz de substituição de base. A matriz apresentada na Figura 12(a) se preocupa apenas com os *matches* (equivalência entre duas bases de seqüências diferentes), enquanto que a matriz apresentada Figura 12(b) leva em consideração *matches* e *mismatches* (não equivalência entre duas bases de seqüências diferentes), além de uma penalidade para *gaps*.

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

(a)

	A	G	C	T	-
A	1	-1	-1	-1	-2
G	-1	1	-1	-1	-2
C	-1	-1	1	-1	-2
T	-1	-1	-1	1	-2

(b)

Figura 12 - Matrizes de substituição de base (a) Matriz de Substituição de Bases com *matches*; (b) Matriz de Substituição de Bases com *matches*, *mismatches* e penalidade para *gaps*.

Um exemplo de utilização da matriz da Figura 12(b) é apresentado a seguir na Figura 13.

G A A - G G A T T A G
G A T C G G A - - A G

Total de *matches*: 7
Total de *mismatches*: 1
Total de *gaps*: 3

Pontuação: 7x1 + 1x(-1) + 3x(-2) = 0

Figura 13 - Exemplo do cálculo da pontuação de um alinhamento através da utilização de uma matriz de substituição de base.

Na comparação de seqüências de DNA, as matrizes de substituição são relativamente simples, sendo que o mesmo não ocorre quando se deseja comparar seqüências de proteínas. A comparação destas seqüências leva em conta critérios evolutivos, por isso, é preciso um esquema de pontuação mais elaborado. As matrizes de substituição PAM (*Point Accepted Mutation*) e BLOSUM (*Blocks Substitution Matrix*) são muito utilizadas para a comparação de proteínas.

As matrizes PAM (DAYHOFF *et al*, 1978) estabelecem esquemas de pontuação para grupos de seqüências relacionadas. Foram construídas observando-se substituições de

aminoácidos através de alinhamentos. Para isso utilizou-se de um amplo conjunto de proteínas relacionadas, as quais tinham sofrido certa divergência evolucionária.

Cada matriz PAM é determinada por um número que indica o grau de divergência entre as seqüências usadas. Dessa forma, uma matriz PAM1 é uma unidade de divergência evolutiva na qual ocorreu 1% de substituição dos aminoácidos.

As matrizes BLOSUM (HENIKOFF e HENIKOFF, 1992) foram construídas através da extração de segmentos sem *gaps*, chamados blocos, de múltiplos alinhamentos de famílias de proteínas, envolvendo seqüências com pouca relação (diferentemente das matrizes PAM, nas quais foram usadas seqüências relacionadas), e então agrupadas com base na porcentagem de identidade. Da mesma forma como nas matrizes PAM, as matrizes BLOSUM são seguidas de números que se referem ao nível máximo de identidade entre as seqüências.

3.1.1 TIPOS DE ALINHAMENTO

Existem dois tipos de alinhamento, o global e o local. O primeiro se estende por toda seqüência, já o segundo localiza fragmentos de seqüências que são mais similares. A definição de ambos os alinhamentos é apresentada a seguir.

Alinhamento Global: Dado um alfabeto A com uma matriz de substituição M, o alinhamento global para duas seqüências $s=\{s_1s_2s_3\ldots s_m \mid s_i \in A\}$ e $t=\{t_1t_2t_3\ldots t_n \mid t_j \in A\}$ sendo $1 \leq i \leq m$ e $1 \leq j \leq n$, consiste em encontrar cadeias de caracteres α e β, as quais são obtidas de s e t inserindo espaços no início ou no final de s e t, e cuja pontuação calculada usando M é máxima (GUSFIELD, 1997) *apud* (XU *et al.*, 2003).

Alinhamento Local: Dado um alfabeto A com uma matriz de substituição M, o alinhamento local para duas seqüências $s=\{s_1s_2s_3\ldots s_m \mid s_i \in A\}$ e $t=\{t_1t_2t_3\ldots t_n \mid t_j \in A\}$ sendo $1 \leq i \leq m$ e $1 \leq j \leq n$, consiste em encontrar subcadeias de caracteres de s e t, cujo valor da similaridade (alinhamento global ótimo) é máximo (GUSFIELD, 1997) *apud* (XU *et al.*, 2003).

O grande volume de dados gerados pelos diversos projetos de seqüenciamento de genomas torna necessária a utilização de ferramentas para análise computacional destas informações. As ferramentas de bioinformática são os programas de software projetados para extrair informações significativas da grande massa de dados biológicos.

A existência de uma grande quantidade de ferramentas disponíveis para análise genômica mostra a necessidade de se atender a diferentes objetivos e a dificuldade que se tem em englobar várias funcionalidades em uma única ferramenta. É nesse sentido que esforços são gastos continuamente, pensando em atender cada vez mais e melhor a necessidade de biólogos e afins, desenvolvendo novas ferramentas e integrando as já existentes.

3.1.2 BLAST (BASIC LOCAL ALIGNMENT SEARCH TOOL)

BLAST é uma ferramenta de comparação e alinhamento local de seqüências de nucleotídeos ou de aminoácidos depositadas em bancos de dados. É de muita importância pra quem trabalha com bioinformática, pois possibilita realizar análises de similaridades de seqüências como DNA, RNA e proteínas. Como utiliza métodos heurísticos, consegue resultados significativos em um tempo satisfatório.

Essencialmente, a partir de uma seqüência de consulta (*query)* introduzida pelo usuário, BLAST tenta achar todas as seqüências em bancos (*subject*) que possuem alinhamentos locais estatisticamente significativos. O usuário pode especificar também um limiar para o alinhamento, denominado *score*.

As seqüências similares retornadas de uma pesquisa são conhecidas como *hits* e são acompanhados de alinhamentos, juntamente com o *score*, e de uma estimativa de significância, denominada de *e-value*. O *e-value* é proporcional à probabilidade de um *hit* com o seu alinhamento ser encontrado ao acaso. Assim, quanto menor o *e-value*, mais significante é o *hit*.

A estratégia usada no BLAST para a determinação dos *hits* é a busca de "sementes", que consistem em pares de seqüências muito curtas entre as seqüências em estudo, as quais são estendidas em ambos os lados. A extensão prossegue até o alcance dos escores máximos. Nem todas as extensões são investigadas porque o programa compara os escores destas extensões com um limiar cuidadosamente escolhido. Assim, extensões significativas podem não ser localizadas. Contudo, é uma margem de erro aceitável.

Como resultado, é exibida uma lista contendo todos os hits acompanhados de seus alinhamentos, consistindo em uma medida classificada como *score*, e de uma estimativa de significância, o *e-value*. O *score* é a pontuação obtida na comparação dos genes e corresponde ao grau de similaridade entre as mesmas, isto é, quão parecidas estas seqüências são. Já o *e-value* é uma estimativa de significância, proporcional à probabilidade de uma comparação

com o mesmo alinhamento (*score*) ser encontrado ao acaso, ou seja, quanto menor o *e-value*, mais significante é o alinhamento. As respostas que possuírem um *e-value* inferior a um determinado limite (mais similares) fazem parte do conjunto de respostas conhecido como ***best hit***.

Um **BBH** (*bidirecional best hit*) é um resultado que obteve *best hit* na comparação de dois conjuntos de seqüências, A e B, em ambos os sentidos, A→B e B→A, isto é, o relacionamento entre um gene x em um genoma A e um gene y em um genoma B é chamado BBH quando x for o *best hit* de y em uma consulta contra todos os genes de A e vice-versa. O BBH é freqüentemente usado como uma definição operacional de ortologia.

De acordo com o tipo de seqüência de entrada (nucleotídeo ou aminoácido) e com o tipo de resultado esperado, existe um programa BLAST específico:

- **BLASTP:** Compara uma seqüência *query* de aminoácidos contra um banco de dados de seqüências de proteínas;
- **BLASTN:** Compara uma seqüência *query* de nucleotídeos contra um banco de dados de seqüências de nucleotídeos;
- **BLASTX:** Compara uma seqüência *query* de nucleotídeos contra um banco de dados de seqüências de proteínas;
- **TBLASTN:** Traduz uma seqüência de aminoácidos para nucleotídeo e compara com o banco de dados de genes;
- **TBLASTX:** Traduz uma seqüência de nucleotídeo para aminoácidos e compara com o banco de proteínas.

3.2 KEGG

O KEGG (Kyoto Encyclopedia of Genes and Genomes) (KANEHISA *et al.*, 2002; KEGG, 2006) é composto por seis bases de dados: GENES, SSDB, PATHWAY, LIGAND, EXPRESSION e BRITE, todas baseadas em grafos, ou seja, contendo objetos como nós e seus relacionamentos como arestas formando árvores. Cada base aborda um contexto diferente do outro como: genes, proteínas, proteomas, compostos químicos, reações químicas, ortologias e interações de proteínas, com a finalidade de obter grupos ortólogos e suas características. No entanto, será dado um maior detalhamento sobre as bases KEGG/GENES e KEGG/SSDB, responsáveis pela detecção de ortologias, que é de interesse deste projeto.

KEGG/GENES é um catálogo de genes dos genomas que foram seqüenciados completamente e são representados através dos grafos, onde os nós representam os genes, enquanto que as arestas indicam suas adjacências.

O KEGG/SSDB contém informações sobre a similaridade de seqüências de aminoácidos de todos os genes que codificam proteínas em genomas completos. Os pares de genes homólogos são obtidos computacionalmente a partir da base de dados KEGG/GENES sobre o qual é executado um algoritmo de comparação de seqüências chamado Smith-Waterman (**SW**) (PEARSON,1991). Dessa forma, o KEGG/SSDB armazena apenas os *best hits* e os BBHs, representando o universo protéico, onde os nós são as proteínas e as arestas são suas similaridades.

Conforme pode ser visto na Figura 14A, cinco consultas em relação a um gene S são disponibilizadas: na primeira todos os genes que possuem algum relacionamento com S são envolvidos como suas adjacências, *best hits* e *best hits* reversos, sendo que neste último S→gene não forma um *best hit*, enquanto gene→S sim. Na segunda e terceira consultas somente os *best hits* e seus reversos respectivamente são retornados. Tanto no quarto quanto no quinto exemplos apenas as arestas das relações BBHs são consideradas; o grafo representa grupos de ortologia, formando cliques parciais, entretanto *os best-best neighbors* atendem à propriedade descrita na seção 2.2 onde é definido um gene ortólogo (GO), enquanto os *best-best* cliques compreendem a definição de genes ortólogos múltiplos (GOM), em que qualquer par do grupo forma um GO.

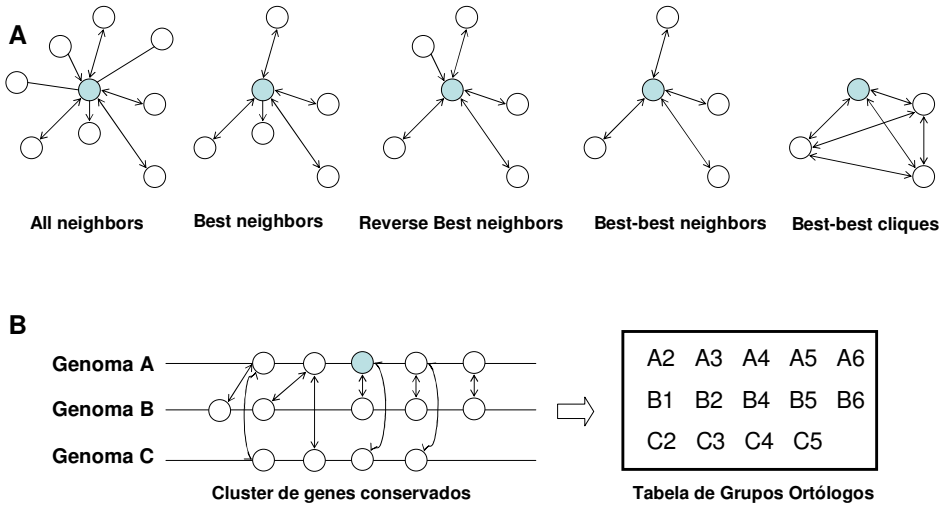


Figura 14 - Ortologia no KEGG – retirada de KANEHISA *et al.* (2002).

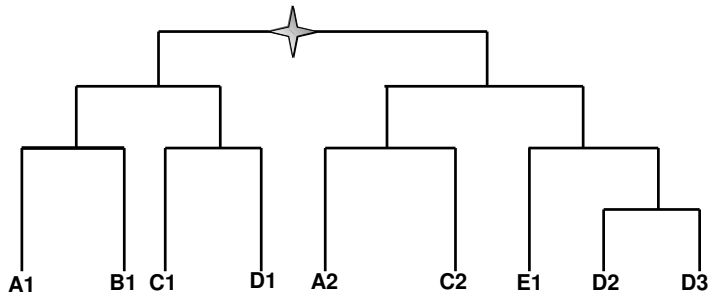
No KEGG/SSDB são detectadas regiões de genes contínuos conservados (RGC), através de conjuntos contínuos de *best-best neighbors*, formando tabelas de grupos ortólogos, ilustrado na Figura 14B. Nota-se que com este tratamento, o KEGG obtém regiões ortólogas (RO) mas não ROMs, uma vez que nem todas as RGCs formam ROs entre si, como no caso dos genomas B e C. Se ao invés da relação considerada fossem adquiridos os *best-best* cliques, a base em questão conteria tanto ROs quanto ROMs.

Recentemente, foi adicionada ao KEGG uma nova base, KEGG/OC (Ortholog Clusters) (ITOH *et al.*, 2004), que armazena os resultados obtidos a partir do KEGG/SSDB.

3.3 MBGD

A função central do MBGD (Microbial Genome Database) (UCHIYAMA, 2003; MBGD, 2006) é a criação de uma tabela de classificação de genes ortólogos para micróbios. Para isso, os relacionamentos de similaridade existentes entre os genes de diversas espécies são utilizados como fonte para um algoritmo de agrupamento de homologias.

Tais similaridades são calculadas a partir da execução do BLAST, onde todo um conjunto de genomas distintos é comparado entre si. Os resultados que atingirem um *e-value* inferior ou igual a 10^{-2} são considerados; os demais são descartados.



Grupo 1	A1	B1	C1	D1	
Grupo 2	A2		C2	D2 D3	E1

Figura 15 - Árvore dividida em grupos no MBGD – retirada de UCHIYAMA (2003).

As respostas obtidas são então submetidas a um algoritmo de agrupamento hierárquico conhecido como UPGMA (Unweighted Pair Group Method with Arithmetic mean), que tem

como tarefa inicial providenciar a construção de uma árvore contendo os genes homólogos. Após isso, um procedimento divide a árvore em domínios; como forma de classificar corretamente o resultado, finalizando em uma estrutura semelhante à disposta na Figura 15. Este processo é repetido todas as vezes que os usuários alteram o conjunto de organismos.

Nesta figura estão exemplificados nove genes (A1, A2, B1, C1, C2, D1, D2, D3 e E1) dispostos em cinco espécies (A, B, C, D e E), formando dois grupos (1 e 2). Nota-se que no Grupo 2 existem tanto genes ortólogos quanto parálogos; no caso, D2 e D3 são parálogos entre si e ortólogos em relação a A2, C2 e E1. Outro aspecto que pode ser observado é a perda de genes ocorrida durante a evolução; desta forma, os organismos B e E sofreram essas perdas conforme mostra a ausência de genes em um determinado grupo.

3.4 COG

Integrando o conjunto de ferramentas da NCBI, o COG (Cluster of Orthologous Genes) (TATUSOV *et al.*, 2001) e o KOG (euKaryotic Orthologous Groups) (TATUSOV *et al.*, 2003) têm como objetivo principal servir de plataforma para anotações funcionais. São basicamente compostas por grupos de proteínas ortólogas, as quais são organizadas pelo programa COGNITOR, responsável pela inclusão de novas proteínas dentro de cada grupo segundo a semelhança com seus objetos e seu domínio. A diferença entre o COG e o KOG está no tipo de organismo com que trabalham: o primeiro utiliza o código genético de seres procariontes, enquanto o segundo é aplicado sobre os seres eucariontes.

Existem várias versões do COGNITOR. O algoritmo original usa os *best hits* entre pares de genes de um mesmo genoma para unir as paralogias; posteriormente, identifica os grupos através do relacionamento entre, no mínimo, três seqüências de genomas diferentes. Finalizando, esses grupos são fundidos se possuírem ao menos uma aresta em comum (duas seqüências e sua similaridade), formando os grupos ortólogos constituídos também de parálogos.

Uma outra versão introduziu uma estimativa de probabilidade onde a proteína em questão é atribuída a um grupo por acaso. Assumindo que a distribuição de *hits* para um genoma nos grupos são uniformes, a probabilidade de um *best hit* estar dentro de um grupo particular é simplesmente a fração de proteínas de um genoma específico no grupo, segundo a fórmula abaixo:

$$\mathbf{f_{ij} = n_{ij}/N_i}$$

onde n_{ij} é o número de proteínas de uma espécie i no grupo j e N_i é o total de proteínas na espécie i. Estas informações estão representadas na Figura 16.

Caso tais estimativas apontem que uma proteína pertence a vários domínios, ela é individualmente separada em um novo domínio.

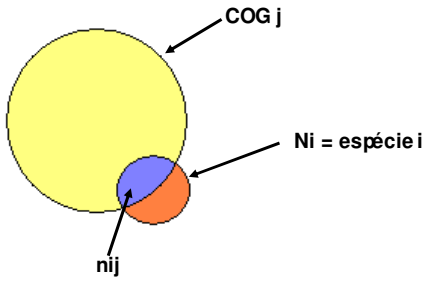


Figura 16 - Probabilidade de um gene i pertencer a um COG j – retirada de TATUSOV *et al.* (2001).

Uma forma de se visualizar os grupos disponibilizados pelo COG é ilustrada na Figura 17, onde cada grupo é exibido através das espécies e das proteínas. A ocorrência de uma proteína em uma determinada espécie no COG é sinalizada com 1, já a ausência recebe o valor 0.

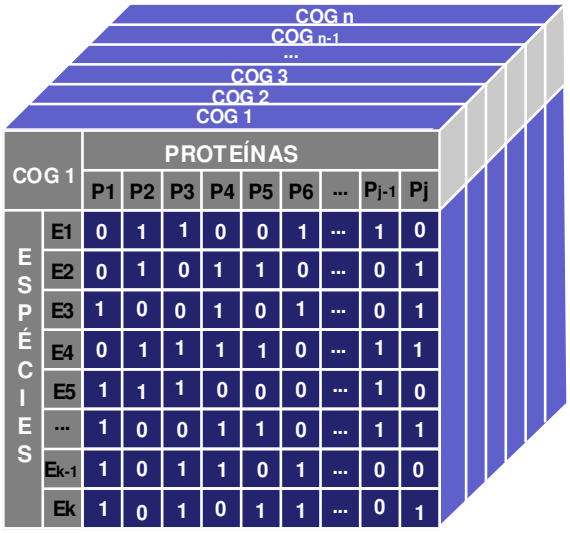


Figura 17 - Distribuição dos COGs.

Um ponto negativo do COG/KOG é que a validação de seus grupos se dá de forma manual, fazendo-se necessário que o usuário verifique se os genes atribuídos a um determinado grupo lhe são coerentes.

3.5 NCUT

Outra estratégia de agrupamento é apresentada por ABASCAL e VALENCIA (2002), onde as similaridades entre um par de seqüências são utilizadas pelo algoritmo de agrupamento denominado NCUT, a fim de reunir aquelas que possuem maior afinidade. O diferencial deste processo é obter grupos sem a necessidade de acessar o espaço completo de seqüências, focando e analisando somente as entidades que possuem relacionamento com um determinado gene, localmente, ao invés de procurar relacionamentos afastados e uma visão geral deste espaço através de seus domínios.

De um modo geral, para obter um conjunto de seqüências similares a cada gene analisado, as seqüências intermediárias são utilizadas no procedimento de busca, o qual detecta as similaridades através do BLAST. Este procedimento é baseado no princípio da transitividade, isto é, se uma proteína A é próxima a B, e B é próxima a C, então A também será próximo a C. Esta correspondência é freqüentemente complicada pela presença de proteínas que fazem parte de muitos domínios e por alinhamentos parciais, que podem criar ligações artificiais entre proteínas distantes (não-similares) através de proteínas intermediárias que pertencem a múltiplos domínios.

A cada ciclo do BLAST somente os fragmentos alinhados resultantes dos ciclos anteriores são utilizados. Esta prática, quando aplicada cuidadosamente, reduz o número de ligações artificiais criadas pelas proteínas comportadas por muitos domínios. Este procedimento está representado na Figura 18, onde cada número seqüencial corresponde a um ciclo de execução do BLAST.

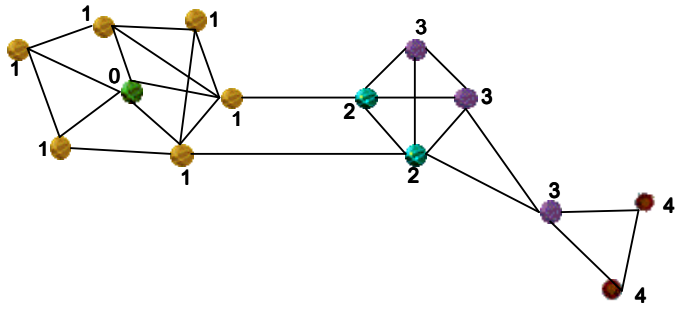


Figura 18 - Evolução passo a passo do Blast – retirada de ABASCAL e VALENCIA (2002).

Depois de encontrados os grupos de seqüências próximas, o segundo passo consiste

em uni-los a fim de obter uma representação de espécies, gerando resultados equivalentes a grupos de seqüências ortólogas.

Antes, porém, faz-se necessária a introdução do conceito básico da família de algoritmos conhecida como corte mínimo, os quais são baseados nas medidas de capacidade em teoria dos grafos, onde um grafo G(V, E) é formado por um conjunto de nós V correspondente às seqüências e um conjunto de arestas E representando as similaridades entre os nós. O peso w(A, B) é atribuído à aresta e refere-se ao *e-value* obtido pelo BLAST entre as seqüências A e B.

Um ***cut (A, B)*** no grafo G(V, E) é uma partição de V em dois conjuntos distintos dos nós A e B. A **capacidade** de um corte é a soma de todos os pesos associados às arestas que cruzam o corte, ou seja:

cut (A, B) = Soma w (i, j); i em A, j em B

sendo o corte mínimo aquele com a capacidade mínima associada. Porém esta metodologia revela uma preferência por grupos pequenos.

Uma versão normalizada do corte mínimo foi criada, chamada NCUT, de forma a melhor atender à ampliação dos grupos, baseando-se, além da capacidade, no fluxo de grafos. É calculado a partir da seguinte fórmula:

Ncut(A, B) = cut(A, B) / asso(A, V) + cut(A, B) / asso(B, V)

em que asso(A, V) é a soma dos pesos das arestas de todos os nós de A para todos os nós em V (incluindo aqueles em A). Normaliza-se a capacidade do corte pelo nível de desconexão induzido no grafo, evitando a preferência por pequenos grupos.

O algoritmo de agrupamento trabalha recursivamente; uma vez que o corte é encontrado, sua propriedade é avaliada e, caso seja necessário, um novo corte é calculado para cada subgrafo resultante.

O processo continua enquanto nenhuma das condições abaixo for satisfeita:

1 – A média aritmética da capacidade de relacionamentos dentro dos novos grupos exceder o valor da mesma medida entre eles.

2 – O número de relacionamentos existentes dentro de quaisquer dois novos grupos dividido pelo número de possíveis relacionamentos é maior que a mesma medida nos grupos já existentes.

Três parâmetros são calculados para a execução do processo de fusão dos grupos: a **capacidade média**, que é formada pela média aritmética dos relacionamentos de similaridade entre dois grupos; a **conectividade**, que consiste no número de conexões entre dois grupos, dividido pelo número de possíveis conexões que poderiam existir se todos os nós fossem

conectados; e a **entropia relativa (H)**, através da qual as informações disponíveis nos grupos vizinhos avaliam a possibilidade de fusão baseada nos potenciais de paralogia e ortologia, obtida pelo seguinte cálculo:

H(P||Q) = Sum i (P(xi) log (P(xi)/Q(xi)))

Onde i é o genoma;

Q(xi) = ni/NT; onde ni é o número de genes no genoma i e NT é o número total de genes em todos os genomas.

P(xi) é a freqüência do genoma i no conjunto de grupos avaliados.

Com esses dados a união de grupos é realizada progressivamente dos mais próximos para os mais distantes, através do seguinte algoritmo recursivo:

(0) Recupera-se o conjunto de grupos que contêm o gene em questão;
(1) Calcula-se a entropia relativa destes grupos;
(2) Procura-se o grupo com maior conectividade ao conjunto aceito. Caso dois grupos possuam a mesma conectividade, aquele com maior capacidade média é selecionado;
(3) Se dois grupos forem unidos, calcula-se a entropia relativa entre eles;
(4) Se a entropia relativa diminuir com a adição do grupo, estes são unidos e retorna-se ao passo 1; caso contrário, o processo finaliza.

Ao término desse processo, são obtidos grupos contendo informações de genes ortólogos e parálogos entre as espécies confrontadas, exibidos aos usuários através de grafos, gráficos que avaliam seus tamanhos e gráficos comparativos ao COG. Suas fontes de dados utilizadas são oriundas de bases de dados protéicas como o Swiss-Prot, porém, nenhuma informação referente ao armazenamento destes resultados é mencionada em ABASCAL e VALENCIA (2002).

3.6 INPARANOID

O Inparanoid (O'BRIEN *et al.*, 2005; O'BRIEN *et al.*, 2004) é um algoritmo que utiliza os resultados gerados a partir da comparação de genomas, aos pares, pelo BLAST, a fim de formar grupos de ortologia a partir de seus BBHs.

Contudo, esta ferramenta realiza alguns tratamentos devido ao conceito de homologia adotado, o qual afirma que é possível existir genes parálogos entre espécies diferentes desde que duplicações ocorram antes e depois de uma especiação, conforme ilustrado na Figura 19. São aplicados os termos: ***inparalogs*** para os genes que foram duplicados antes e após a

especiação e ***outparalogs*** para aqueles cuja duplicação ocorreu apenas anteriormente à especiação. Os genes *outparalogs* nunca podem ser ortólogos, enquanto somente os *inparalogs* podem formar um grupo ortólogo a outra espécie.

Um gene A em uma espécie ancestral sofre uma duplicação seguida de uma especiação, originando as linhagens B e C. C2 e C3 são *inparalogs* a que suas duplicações ocorram pós-especiação. São, portanto, co-ortólogos a B2.

C1 é *outparalog* a C2 e C3, já que sua duplicação ocorreu apenas pré-especiação. O mesmo ocorre entre B1 e B2, ao passo que B1 e C1 são ortólogos.

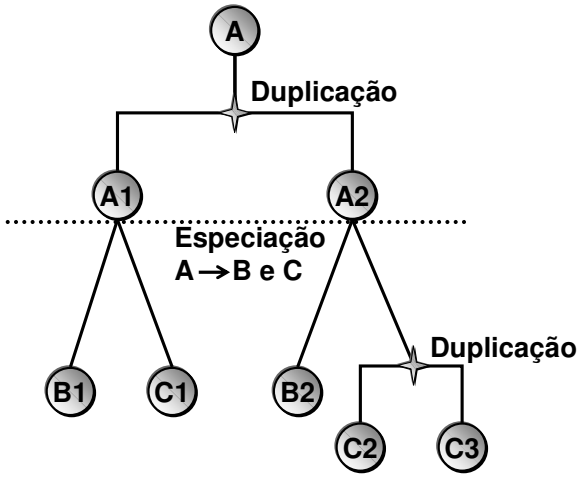


Figura 19 - Homologia no Inparanoid – retirada de O'BRIEN *et al.* (2004).

O algoritmo Inparanoid realiza o agrupamento de genes ortólogos seguindo o critério de que é possível identificar genes ortólogos e parálogos entre organismos diferentes, detectados a partir do alinhamento realizado pelo BLAST, confrontando dois genomas por vez e analisando seus resultados. Considerando dois genomas A e B quaisquer, são realizadas as seguintes comparações por esta ferramenta: A→B, B→A, A→A e B→B. Deste conjunto intermediário, apenas os BBHs são considerados, com exceção dos que possuírem entre 99 e 100% de equivalência, os quais são descartados como forma de se eliminar os *outparalogs* e os auto-relacionamentos.

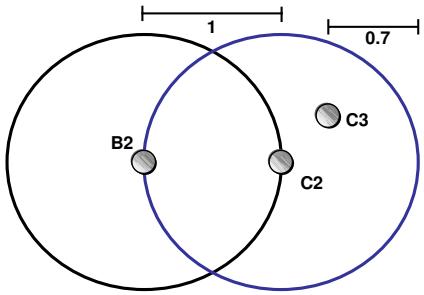


Figura 20 - Agrupamento no Inparanoid – retirada de O'BRIEN *et al.* (2004).

Do conjunto final de respostas, os pares ortólogos com maior similaridade recebem um *score* igual a 1, enquanto aos seus genes parálogos é atribuído o *score* de sua similaridade relativa, obtida por uma normalização baseada nos valores do BLAST. Dando continuidade ao exemplo da Figura 19, a distribuição desses elementos é exibida na Figura 20, onde o par [B2:C2] possui maior similaridade que [B2:C3], recebendo o *score* 1, restando atribuir a C3 seu *score* normalizado. Este valor é obtido pelo seguinte cálculo:

(BLAST[C2:C3] – BLAST[C2:B2]) / (BLAST[C2:C2] – BLAST[C2:B2]), onde BLAST[X:Y] representa a similaridade entre os genes X e Y.

No grupo do exemplo acima, C3 recebe o *score* de 0.7.

As fontes do Inparanoid são provenientes do UniProt e do Ensembl da EMBL (HUBBARD *et al.*, 2005), o qual armazena as anotações dos genomas eucariontes. Os grupos encontrados pelo algoritmo são armazenados em uma base de dados com informações como: *score* de similaridade, espécie, produto, entre outras, dando preferência ao conteúdo da UniProt.

3.7 ORTHOMCL

Esta seção aborda a forma como o OrthoMCL (LI *et al.*, 2003) obtém seus grupos ortólogos em seres eucariontes. O conceito de homologia adotado por esta ferramenta é o mesmo do Inparanoid. Utiliza o BLAST para comparações de seqüências e o algoritmo de Markov (Markov CLuster - MCL) para a formação de grupos, o qual é baseado em probabilidade e teoria dos grafos.

OrthoMCL pode ser visto como um processo de duas etapas: a primeira envolve a aplicação de regras baseada no conhecimento biológico do problema para determinar quais

seqüências podem ser incluídas, como as seqüências são conectadas e como os pesos das arestas podem quantificar o relacionamento entre duas seqüências. A segunda etapa consiste do agrupamento baseado em teoria de grafos e técnicas computacionais. O diagrama de fluxo desta metodologia é apresentado na Figura 21.

MCL simula caminhos aleatórios em um grafo utilizando matrizes de Markov para determinar as probabilidades de fluxo através dos nós. Essa metodologia gera grupos de proteínas, consistindo de paralogias e ortologias entre pelo menos duas espécies.

O procedimento do OrthoMCL inicia-se com uma comparação BLAST de todos os proteomas de interesse entre si. Relacionamentos de ortologia e paralogia são identificados entre pares de genes, cujos *e-values* de alinhamento do BLAST são reciprocamente inferiores a 10^{-5}. Posteriormente, estes resultados são convertidos em um grafo no qual os nós representam os genes e as arestas suas similaridades.

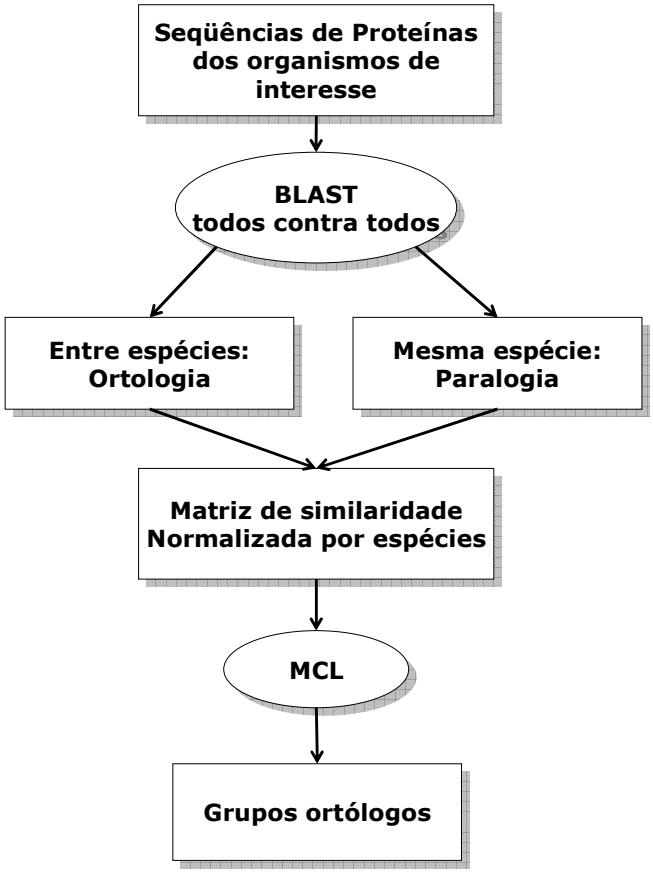


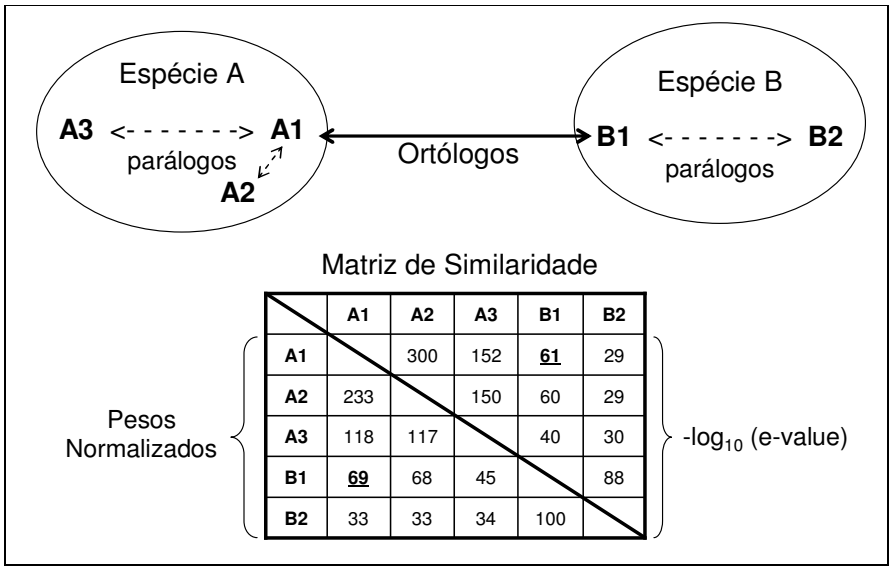
Figura 21 - Diagrama de fluxo do OrthoMCL – retirada de LI *et al.* (2003).

Devido à alta similaridade obtida nas paralogias em relação à ortologia, o processo de agrupamento pode ser comprometido. Os pesos das arestas são então normalizados por

espécies de forma a refletir um peso médio para todo par de ortologia entre duas espécies. Uma matriz simétrica de similaridade, ou matriz de Markov, é preenchida com esses valores normalizados através do cálculo de $-\log_{10}$(e-value) para cada *e-value* proveniente do alinhamento de pares de genes pelo BLAST, conforme ilustra a Figura 22.

O algoritmo de Markov é então aplicado sobre esta matriz, que utiliza simulação de fluxo para buscar as maiores médias entre e intra-espécies, de forma a obter as ortologias e paralogias respectivamente. Como resultado, este procedimento provê grupos de genes ortólogos entre pelo menos duas espécies com seus respectivos genes parálogos.

A Figura 22 exemplifica um caso em que duas espécies, A e B, possuem uma ortologia entre os genes A1 e B1, além das paralogias na espécie A entre (A1, A3) e (A1, A2) e na espécie B entre (B1, B2). Nota-se que, na ilustração, A2 está muito mais próxima de A1 do que A3, e que A1 foi escolhido como gene ortólogo a B1. Estas escolhas se justificam pelos valores encontrados na matriz de similaridade normalizada por espécies, onde os maiores valores evidenciam os pares descritos.



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

Figura 22 - Relacionamentos e matriz de similaridade – retirada de LI *et al.* (2003).

Após formarem-se os grupos, os resultados obtidos são armazenados e disponibilizadas por uma base de dados denominada OrthoMCL-DB (CHEN *et al.*, 2006).

3.8 EGO

Anteriormente chamado de TOGA (TIGR Orthologous Gene Alignment), o EGO

(Eukaryotic Gene Orthologs) (QUACKENBUSH *et al.*, 2001; LEE *et al.*, 2002) foi desenvolvido pela TIGR (TIGR, 2006) e consiste em um processo de reunião de genes similares provenientes de seres eucariontes, utilizando, para isso, bases de dados com alto grau de qualidade e alinhamentos entre estas bases através do BLAST.

A construção do EGO inicia-se pela criação de bases de dados chamadas TGI (TIGR Gene Index), nas quais os genes pertencentes a um genoma são armazenados em TGIs individuais. Para cada nova espécie adicionada nessa base, suas ESTs (Expressed Sequence Tag), possíveis regiões codificadoras e seus genes são adquiridos através de fontes públicas externas, como o GenBank, procedimento ilustrado pelo item 1 da Figura 23. Em seguida, estes dados são tratados de forma a remover seqüências com baixa qualidade, contaminadas por bactérias ou que não possuam um tamanho apropriado, e logo são comparadas a uma base de dados curada pela TIGR (item 2 da Figura 23), onde são mantidas apenas informações com excelente qualidade. Esta rotina tem como objetivo validar os dados a serem inseridos em cada TGI (item 3 da Figura 23).

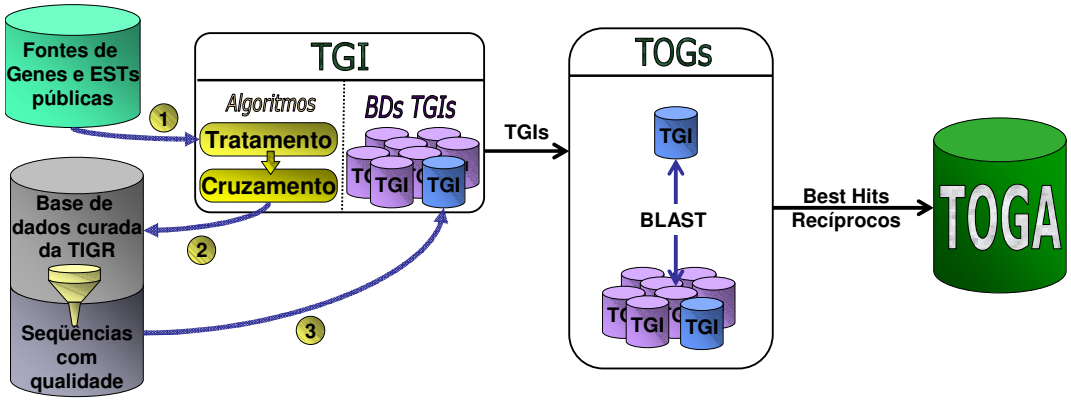


Figura 23 - Estrutura do EGO.

Com as TGIs criadas, inicia-se a fase de identificação de TOGs (Tentative Ortholog Group). Para isso, as ESTs e os genes de cada índice são comparados com as demais TGIs através do BLAST. Um TOG é formado a partir dos BBHs entre pelo menos três espécies com o *e-value* máximo de 10^{-5}.

Esses TOGs são armazenados em uma base de dados relacional, denominada TOGA, desenvolvida para capturar relacionamentos entre genes ortólogos e parálogos como objetos acessíveis, os quais são versionados através das atualizações.

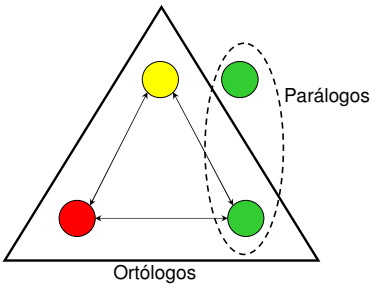


Figura 24 - Genes ortólogos e parálogos – retirada de QUACKENBUSH *et al.* (2001).

Os genes parálogos a esses grupos também fazem parte de uma instância do TOGA; são identificados como não constituintes de um BBH, porém, formam um *best hit* com algum gene contido em um TOG, conforme pode ser visto na Figura 24.

O acesso às informações armazenadas nos TOGAs dá-se por intermédio da Internet, onde podem ser realizadas consultas como: o alinhamento múltiplo de cada TOG (com ou sem seqüências parálogas) e os TOGs e BBHs dos quais uma determinada espécie participa (CHENNA *et al.*, 2003).

3.9 EGG

O EGG (Extended Genome-Genome Comparison) (ALMEIDA, 2002) é uma ferramenta que compara dois proteomas e tem como um dos seus principais objetivos encontrar as regiões de genes contíguas preservadas entre os organismos procariontes.

Sua entrada consiste de dois proteomas X e Y e suas metas são basicamente: encontrar os genes específicos de X em relação a Y e vice-versa, determinar os pares de genes ortólogos entre os dois proteomas e encontrar as regiões de genes próximos que de alguma forma preservam o conteúdo gênico nos dois proteomas.

Esse processo divide-se em três etapas. A primeira consiste em comparar todos os genes de X contra todos os genes de Y e vice-versa, através da execução do BLAST, obtendo para cada gene x do proteoma X e y do proteoma Y uma lista de *hits*, cada um acompanhado do alinhamento e do valor de *e-value* correspondente. Apenas são considerados os pares que formarem um ***match***, ou seja, um *hit* recíproco, não necessariamente um BBH, onde o *e-value* máximo considerado é de 10^{-5} e uma área de cobertura mínima de ambas as seqüências, x e y, de 60%.

A segunda fase constrói um grafo bipartido, onde os vértices são os genes e as arestas

representam suas ortologias. O grafo é disposto em uma matriz binária A_{mxn}, onde m é o número de genes de X e n o número de genes de Y; nela são armazenados os *matches* entre os genes dos proteomas X e Y de forma que $A_{i,j} = 1$ se e somente se x_i e y_j formarem um *match*; caso contrário, $A_{i,j} = 0$.

A partir da matriz binária é iniciada a terceira fase, cujo objetivo é determinar as regiões ortólogas presentes nos proteomas envolvidos na comparação. Este procedimento se dá pela obtenção das diagonais de 1's na matriz, devendo essas diagonais conter no mínimo três 1's para que possa ser caracterizada uma região ortóloga.

Dois parâmetros são definidos para determinar se duas regiões estão próximas a ponto de serem unidas em uma única região: o valor máximo e o valor mínimo de distância limite entre as regiões em número de genes, cujos valores padrão (*default*) são 5 e 2 respectivamente. Após o processo de junção, as ROs são determinadas de forma incremental, onde uma região resultante da união de outras duas podem ainda se juntar a uma terceira região próxima, localizada à sua direita, finalizando o EGG.

3.10 BAGRE

O BAGRE (Base de Dados de Genes, Genomas e Regiões Ortólogas) (MONTERA, 2004) consiste em um algoritmo para comparar regiões de genes consecutivos (RGC) de vários proteomas, utilizando para isso um conjunto de regiões ortólogas providenciadas pelo EGG, confrontando diferentes proteomas entre si.

Assim como no KEGG, a metodologia para a determinação de ROMs é baseada em algoritmos de análise de grafos fundamentada na obtenção de cliques. O procedimento utilizado pelo BAGRE pode ser dividido em quatro etapas.

Numa primeira etapa o EGG é executado para cada um dos pares de genomas possíveis entre os n proteomas envolvidos na comparação múltipla, para que as regiões ortólogas envolvendo dois organismos sejam encontradas. São necessárias n*(n-1)/2 execuções do EGG para que todas as comparações entre proteomas distintos sejam realizadas. O resultado dessa etapa aponta as ROs existentes para cada par de organismos comparados.

O segundo passo tem como finalidade a construção de um grafo onde cada vértice corresponde a uma RGC integrante das ROs obtidas na etapa anterior. Uma aresta unindo dois vértices representa o relacionamento de ortologia entre essas duas RGCs, consumando uma RO. Ao final desta etapa é gerado um grafo com todas as relações de ortologia existentes

entre os organismos envolvidos. Porém, um tratamento adicional em relação à sobreposição de RGCs se torna necessário quando duas RGCs possuírem um número considerável de genes em comum, podendo estar erroneamente representadas por vértices diferentes a ponto de interferir no resultado. O critério adotado para estes casos é o da fusão de RGCs desde que a sobreposição equivalha a uma quantidade maior ou igual à 40% do tamanho da menor RGC em número de genes.

Dando continuidade ao processo, a terceira etapa tem a finalidade de detectar as ROMs, utilizando-se, para isso, da estratégia de obtenção de cliques maximais em grafos, onde uma clique de tamanho k representa uma ROM entre k genomas distintos. Porém, o problema de encontrar a clique maximal com o maior número de vértices existente em um grafo necessita encontrar as cliques máximas, que consiste em um problema NP-Difícil, dessa forma, o problema de clique maximal também se enquadra num problema desta complexidade. Devido a esse fato, uma heurística foi adotada para sua solução, a qual não garante que todos os resultados sejam encontrados nem que o obtido seja o mais completo possível. No BAGRE, cada vértice é inicialmente considerado uma clique; o algoritmo tenta expandir cada clique inicial através da inserção de novos vértices. A heurística implementada define uma ordem crescente nos vértices para tais tentativas de expansão. O processo termina quando esta expansão não é mais possível, isto é, quando a inserção de um novo vértice faz com que o conjunto não represente mais uma clique (nem todo par de vértices do conjunto possui uma aresta que os une). Para finalizar, as cliques repetidas ou integrantes de outras são eliminadas.

A quarta etapa cuida da visualização das ROMs, ou seja, do alinhamento de cada RGC progressivamente, iniciado por uma RGC qualquer.

O BAGRE ainda é incorporado por um banco de dados em MySQL com as informações necessárias às comparações dispostas em duas tabelas, uma contendo os genomas e outra com as informações referentes às suas comparações.

A tabela contendo os genomas possui campos com o nome do organismo, seu identificador, número de genes e pares de bases, além de uma referência para informações adicionais, enquanto a tabela de comparações armazena o identificador dos dois genomas, seus BBHs, o número de genes específicos de cada um e o número de ROs. Resumidamente, tais informações possuem uma finalidade demonstrativa, ou seja, o banco é utilizado apenas para a exibição dos dados, não auxiliando no processo de obtenção de ROMs.

4 ORTOLOGIA MÚLTIPLA ATRAVÉS DE BANCO DE DADOS

Este projeto está dividido em quatro etapas. Em uma primeira etapa é necessário que o EGG seja executado entre todos os pares de proteomas de interesse.

A segunda consiste na definição de um modelo de dados, na aquisição das fontes geradas pelo EGG e seu tratamento e finalmente na instanciação de um banco de dados, chamado **DOG** – Database of Orthologous Groups, de modo a conter as informações necessárias para a obtenção de GOMs parciais e de ROMs.

A partir dos dados contidos no DOG, a terceira fase executa a busca de GOMs e ROMs através de um padrão de verificação de ortologia múltipla (**PVOM**), principal etapa deste trabalho.

A quarta e última está baseada na implementação de um sistema de consultas, nomeado **ISMOG** – Interactive Search of Multiple Orthologous Groups, que tem como premissa atender às solicitações de informações feitas pelo usuário sobre a base de dados de uma forma interativa.

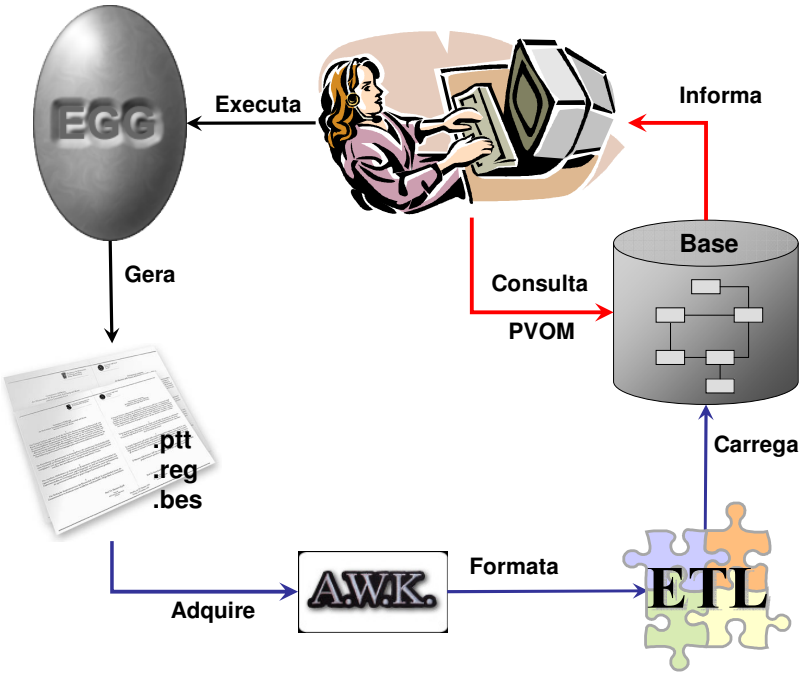


Figura 25 - Fluxo de execução.

A Figura 25 apresenta um panorama geral de como o EGG, o DOG, o PVOM e o ISMOG trabalham juntos. As setas indicadas em preto representam o processo realizado pelo EGG, executado pelo usuário entre os proteomas de seu interesse, até a geração dos arquivos

contendo os resultados. Posteriormente, o processo de carga de toda a base de dados utiliza-se de vários recursos computacionais, seguindo o fluxo representado pelas setas azuis, onde um programa **AWK** formata os registros com extensão ".ptt", ".bes" e ".reg". Um processo de transformação e padronização dos dados (**ETL**) viabiliza a carga das tabelas do DOG. As setas em vermelho ilustram o funcionamento de um sistema de consultas de ortologias de forma interativa, o ISMOG. Seus resultados são obtidos através da execução de procedimentos e funções que integram o PVOM, coletando e processando os dados da base a cada consulta em que é solicitada a ortologia múltipla.

Nos próximos itens serão detalhadas estas etapas com exceção do EGG, uma vez que este já foi descrito no capítulo anterior.

4.1 DOG

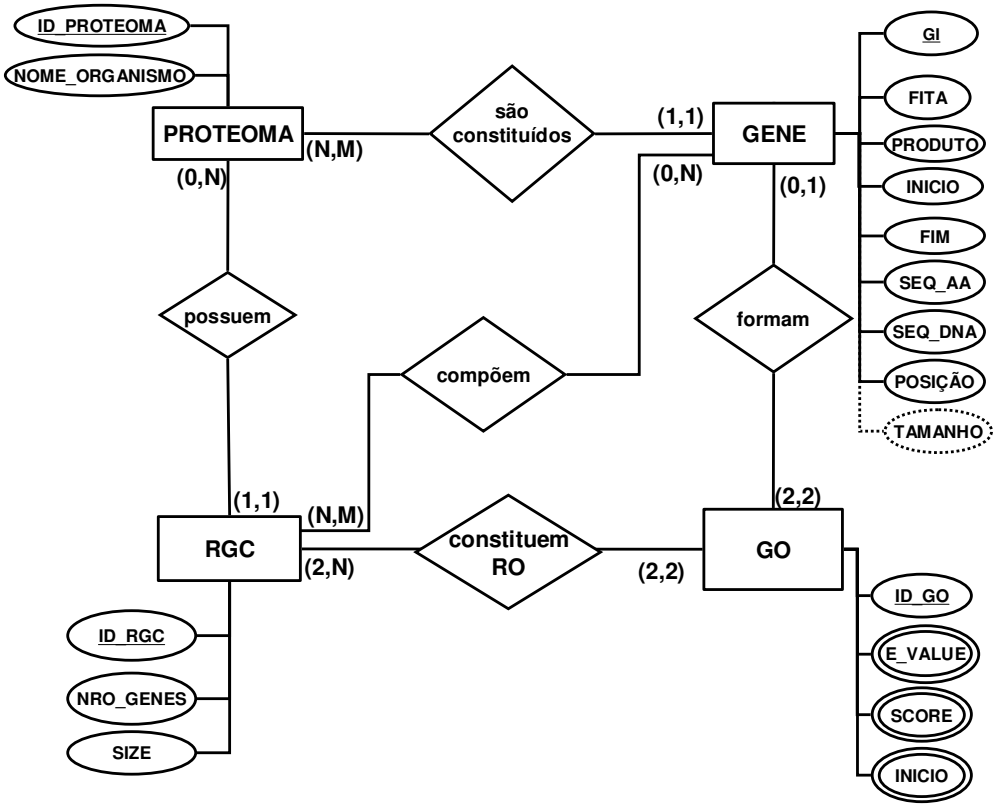


Figura 26 - DOG: Modelo de Dados conceitual.

O DOG é definido pelas entidades, características e relacionamentos envolvidos nas ortologias entre regiões e genes, armazenando as informações provindas do EGG. Seu modelo

de dados conceitual, representado pelo ME-R (modelo entidade-relacionamento), mostrado na Figura 26 (ELMASRI e NAVATHE, 2000), foi criado de forma a retratar um organismo por seu proteoma, do qual são extraídas todas as outras informações, como genes e RGCs. A entidade GENE contém informações como seu GI, a fita de que foi obtida, seu produto, tamanho, entre outras coisas, e do seu relacionamento com genes de diferentes organismos formam-se os GOs. Além disso, como já dito anteriormente, genes próximos uns dos outros formam RGCs e quando estas regiões, pertencentes a espécies distintas, apresentam entre si um considerável número de GOs, dão origem às ROs.

Partindo do modelo conceitual, obtém-se o modelo físico apresentado na Figura 27, no qual alguns relacionamentos são representados através de tabelas como no caso da relação existente entre RGC e Genes e também da composição de ROs. Neste modelo estão dispostos os tipos de dados dos atributos e a chave identificadora de cada tabela.

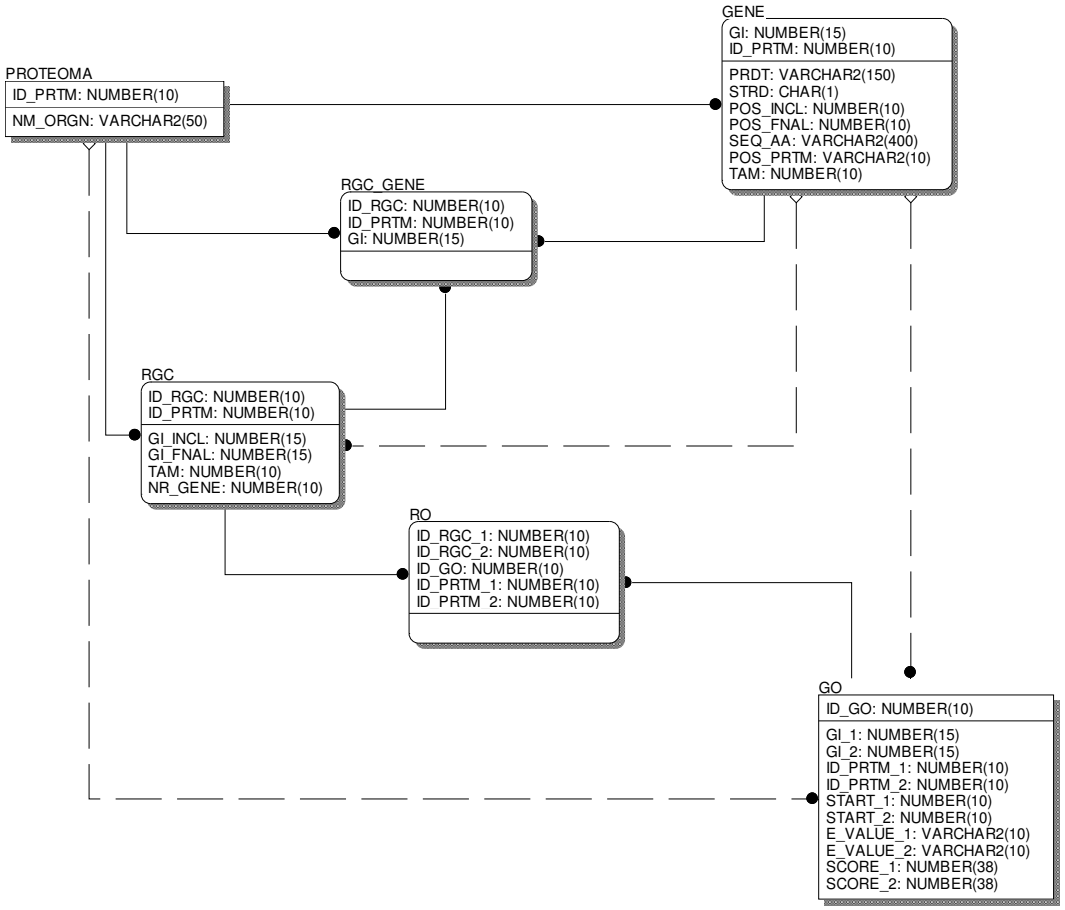


Figura 27 - DOG: Modelo de dados físico.

O principal objetivo do DOG é prover as informações necessárias para que rotinas possam obter ROM e GOM, no entanto, essas duas informações não são armazenadas, e sim

calculadas em tempo de execução da consulta. O motivo para essa decisão será justificado mais adiante na seção 4.2.3.

Com a estrutura do DOG criada, é necessário instanciá-la a partir das informações provenientes do EGG, as quais são submetidas a um processo AWK. O AWK (ROBBINS, 1996) é uma linguagem de programação criada por Alfred **A**ho, Peter **W**einberger e Brian **K**ernighan, própria para o tratamento de textos, permitindo que, a partir do reconhecimento de padrões ou expressões regulares contidas no arquivo de entrada, este seja re-formatado de acordo com as necessidades do usuário.

No caso deste trabalho, o AWK é utilizado com o intuito de gerar arquivos apropriados à população do banco de dados, baseando-se em informações contidas nos arquivos com extensão ".ptt", ".bes" e ".reg", provenientes do Genbank, Blast e EGG respectivamente.

Bifidobacterium longum NCC2705, complete genome - 0..2256646
1729 proteins

Location	Strand	Length	PID	Gene	Synonym	Product
49..288	+	79	23464629	BL0001	-	cold shock protein
526..2151	+	541	23464630	BL0002	-	chaperone
2248..2538	-	96	23464631	BL0003	-	hypothetical protein
2613..3419	+	268	23464632	BL0004	-	hypothetical protein
3412..4143	+	243	23464633	BL0005	-	response regulator of two-component system
4315..6165	+	616	23464634	BL0006	-	histidine kinase sensor of two-component system
6235..6624	+	129	23464635	BL0007	-	cold shock protein
6633..7820	+	395	23464636	BL0008	-	narrowly conserved hypothetical protein
7932..8897	-	321	23464637	BL0009	-	conserved hypothetical protein
9047..11656	+	869	23464638	BL0010	-	protease

Figura 28 - Estrutura do PTT.

Um PTT está disposto conforme ilustrado na Figura 28, o qual é lido pelo AWK gerando duas saídas. A primeira contém o cabeçalho, com os dados referentes ao organismo, responsável por alimentar a tabela de Proteomas. Na segunda saída estão contidas as informações básicas de todos os genes do proteoma, que posteriormente irão preencher os registros da tabela de Genes. O processo AWK se torna necessário sobre os arquivos PTT por não possuírem estes uma formatação padrão, podendo ter seus conteúdos com tamanho fixo, delimitados por algum caractere ou então uma mistura dos dois.

Os arquivos com extensão BES contêm todos os genes que obtiveram BBHs na comparação entre os pares de proteomas. Eles são trabalhados com o intuito de adquirir todos os GOs existentes nestes organismos. Uma amostra deste arquivo está ilustrada na Figura 29, onde estão dispostos os pares de GOs que formam BBHs, com seus respectivos valores de alinhamento (*e-value*), GIs, gene, tamanho e produto. Estas informações são a principal fonte para a carga da tabela de GOs.

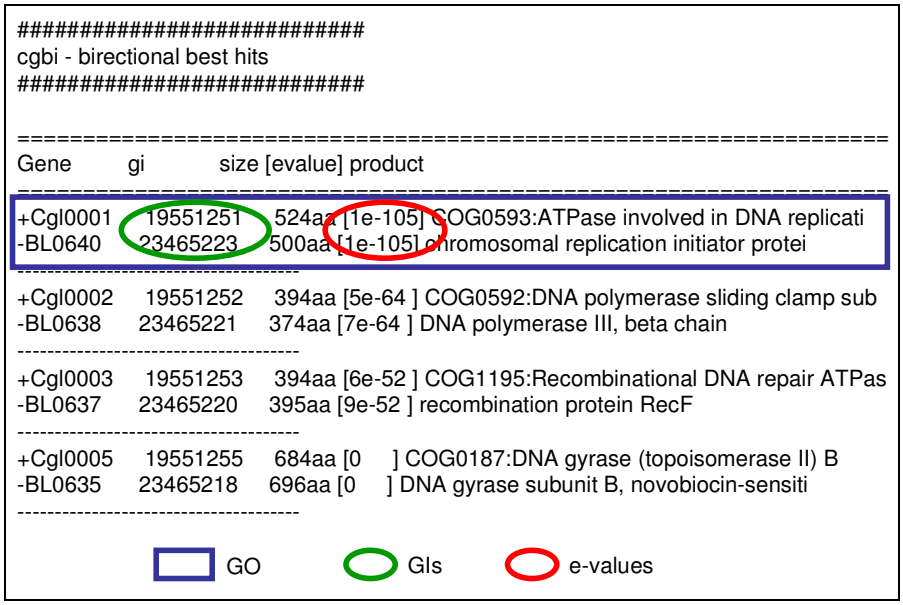


Figura 29 - Estrutura do arquivo BES.

Um programa com maiores funcionalidades foi implementado para suportar os arquivos REG, cuja estrutura é mostrada na Figura 30. Contendo as RGCs e as ROs resultantes da execução do EGG entre dois proteomas, esta família de arquivos é responsável pelos dados das tabelas de RGCs, ROs e alguns registros de GOs do DOG. O AWK separa as RGCs das ROs formadas por alguns pares de genes ortólogos. Assim, GOs e ROs são direcionados a um arquivo, enquanto as RGCs a uma outra saída. Entretanto as GOs participantes das ROs não são necessariamente BBHs, podendo constituir apenas *matches*. Tais GOs não integram o BES e são adicionados ao DOG por meio dos registros obtidos no REG.

A partir dos arquivos gerados e formatados pelo AWK, inicia-se o processo de ETL (Extract, Transform and Load), que tem como principal objetivo tratar os dados desde sua organização e padronização até a carga no banco.

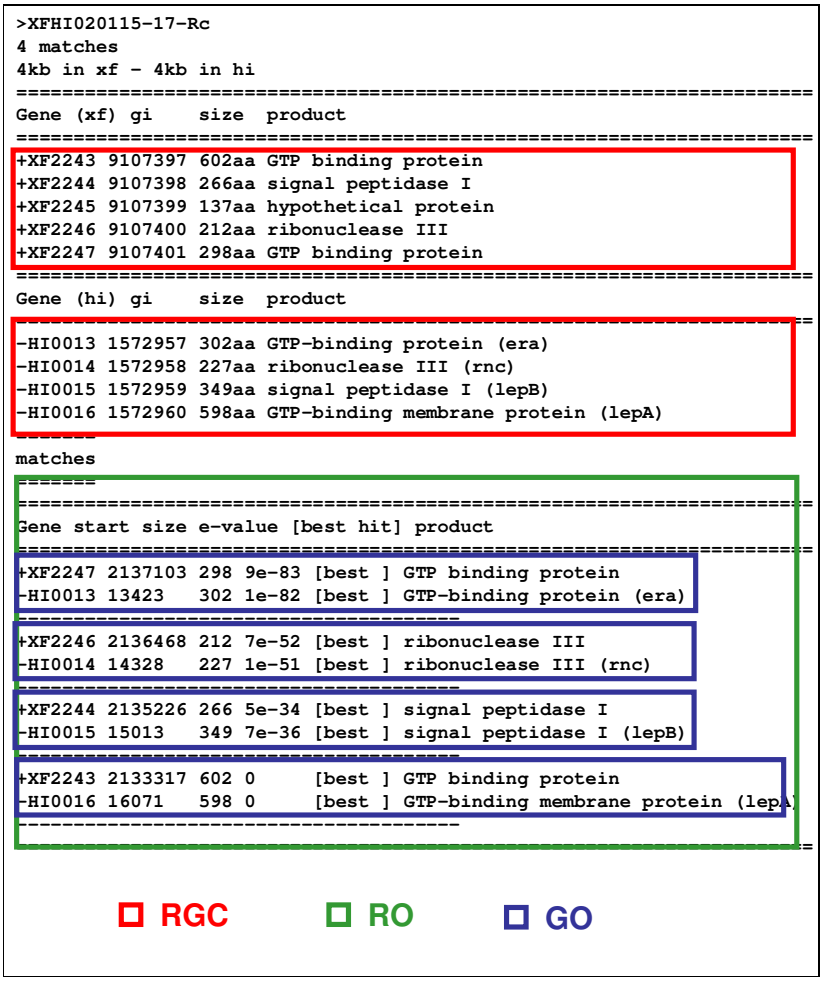


Figura 30 - Estrutura do arquivo REG.

As tabelas de Proteomas, Genes, GOs e ROs são carregadas diretamente, como ilustrado na Figura 31, passando apenas por padronizações de texto, como colocar todos os caracteres em maiúsculo, tirar os espaços em branco, converter números e outros detalhes. Para a inserção dos dados em todas as tabelas que armazenam ortologia foi estabelecida uma ordenação, de forma a evitar que registros diferentes que representam o mesmo conteúdo fossem duplicados. Por exemplo, caso um gene x de um proteoma X seja ortólogo a um gene y de um proteoma Y, esta informação poderia ser carregada tanto como x – y como y – x. Porém essas duplicidades são evitadas com o tratamento pela ordenação decrescente da chave do proteoma fornecida pelo banco, garantindo uma padronização dos dados de forma a ser inserido apenas um registro, ou seja, se o proteoma X possuir a maior chave, o registro será x – y. Caso a chave de Y seja a maior, o registro será y – x.

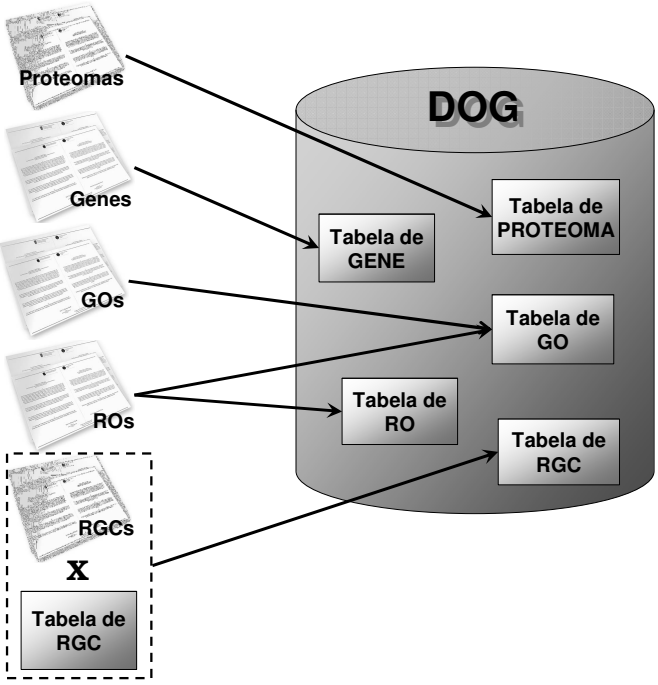


Figura 31 - Carga das tabelas do DOG a partir das fontes do EGG.

A tabela de RGC, no entanto, apesar de ter como fonte básica o arquivo gerado pelo AWK, necessita de um tratamento especial de agrupamento. Para evitar que RGCs muito parecidas fossem inseridas recebendo diferentes identificadores e conseqüentemente perdessem o vínculo na ortologia múltipla, foi adotado um tratamento que verifica as mesmas condições de unificação do BAGRE, ou seja, quando duas regiões (A e B) estão sobrepostas como ilustrado na Figura 32, e a taxa de sobreposição da região maior é igual ou superior a 40% dos genes da região menor, estas são agrupadas em uma mesma região, recebendo o nome da mais extensa (A), porém atualizada com as informações provenientes da menor (B). Tal análise é realizada tanto no arquivo gerado pelo AWK quanto na tabela de RGCs do DOG.

Este processo inicia-se pela carga do conteúdo do arquivo de RGCs em uma tabela auxiliar temporária para que haja a busca das sobreposições contidas no próprio arquivo, unindo as regiões que atendem às condições de sobreposição e atualizando suas informações. Esta etapa evita que regiões similares detectadas em uma mesma submissão recebam identificadores diferentes.

Posteriormente essa tabela auxiliar é confrontada com a tabela de RGC, aplicando-se a mesma rotina descrita anteriormente, porém agora o alvo das atualizações e inserções das informações é a própria tabela oficial. Etapa esta essencial na detecção e possível unificação

das regiões sobrepostas nas inúmeras submissões a serem realizadas, proliferando identificadores únicos entre as ortologias dos mais diversos organismos. Ao final desse procedimento, a tabela auxiliar é excluída e fica disponível para a próxima submissão.

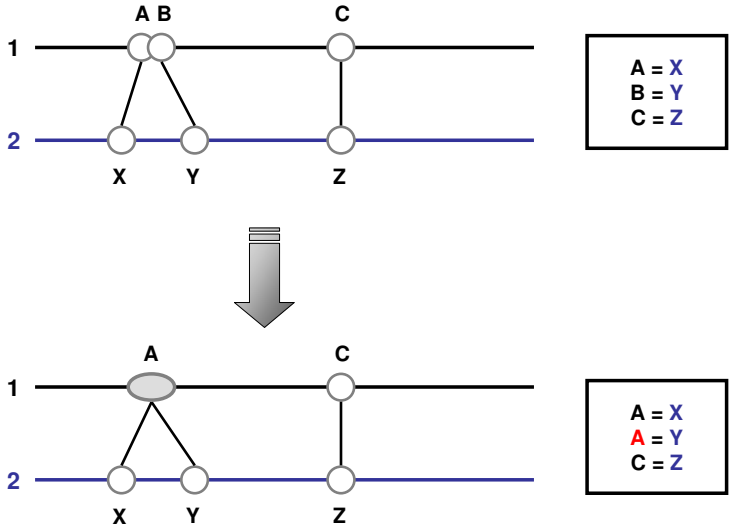


Figura 32 - Unificação de RGCs.

Existem três casos em que é necessária a atualização de informações de RGCs e eles ocorrem quando existe sobreposição de no mínimo 40% dos genes da maior região sobre a menor. Estes casos são possíveis no confronto da tabela auxiliar consigo mesma ou então na comparação entre a auxiliar e a tabela oficial desde que a RGC maior pertença à última.

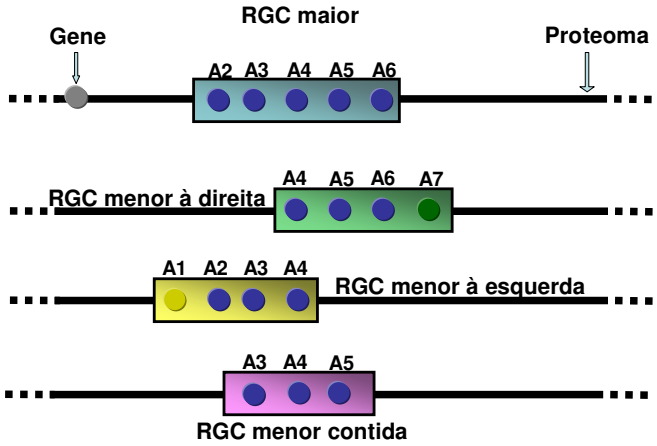


Figura 33 - Atualização da maior RGC.

Nos dois primeiros casos, a menor região situa-se nos extremos da maior, à direita e à esquerda respectivamente. A primeira possui posição inicial e final maiores que a principal, enquanto a segunda possui posições menores. Isto implica na atualização de pelo menos três informações: GI inicial ou final, número de genes e tamanho. Conforme exemplificado na Figura 33, quando as menores regiões se localizam à direita da maior, o GI final é atualizado pelo último gene que compõe a menor região, A7. Caso contrário, é renovado o GI inicial, recebendo o primeiro gene da menor, A1.

Já para corrigir as outras duas medidas é necessário realizar os seguintes cálculos. Ao número de genes da maior região é adicionado o número de genes da menor, subtraído da intersecção existente entre as duas RGCs. Ao tamanho da maior são adicionados os tamanhos dos genes que foram integrados a essa região.

O terceiro caso é o mais simples, uma vez que a região menor está contida na maior, fazendo-se necessário apenas atribuir à RGC menor a chave da maior. Em todos esses casos, o identificador da maior RGC é preservado.

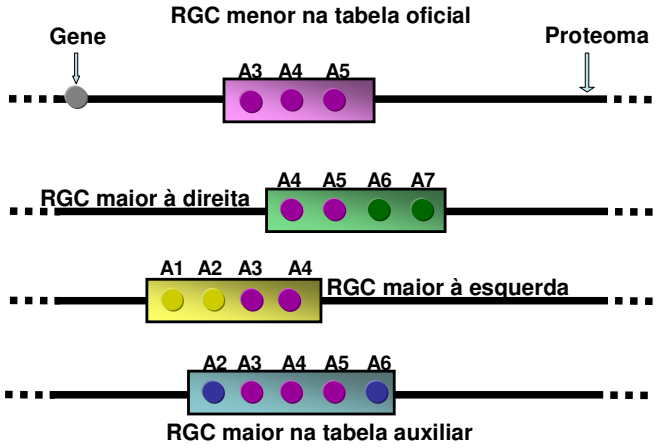


Figura 34 - Atualização da menor RGC.

Quando o confronto for entre a tabela auxiliar e a tabela oficial e a RGC maior estiver na primeira, um outro tratamento é adotado. Os mesmos três casos são possíveis, conforme ilustrado na Figura 34. Neles os registros que satisfizerem a regra de sobreposição terão que ser atualizados na própria tabela oficial de modo a preservar seu identificador. Caso a região menor esteja contida na maior, todas as informações, com exceção do identificador, serão atualizadas.

Uma última consideração deve ser feita: o processo não realiza o confronto da tabela oficial consigo mesma, para evitar que duas regiões já atualizadas, que também atendam ao

requisito da sobreposição, sejam fundidas, o que poderia resultar na representação de um proteoma inteiro por apenas uma RGC. O método utilizado efetua apenas uma primeira fusão atualizando as RGCs e garantindo que essas regiões estejam condizentes em todos os organismos submetidos.

4.2 PVOM

Com a base criada, tabelas populadas e RGCs agrupadas sem perder o vinculo nas ROs, é necessário detectar os relacionamentos onde existe a ortologia entre diferentes espécies. Este é o processo principal deste trabalho, realizado por um padrão de consultas encontrado que foi chamado de PVOM - Padrão de Verificação de Ortologia Múltipla, a fim de obter conjuntos de modo que qualquer par de objetos dentro deste tem de ser ortólogo.

Primeiramente será exposto o problema, detalhando-se alguns pontos cruciais de um caso de teste, para que posteriormente o PVOM possa ser apresentado.

4.2.1 CONSIDERAÇÕES INICIAIS

As tabelas RO e GO do DOG armazenam informações básicas para a detecção das ortologias múltiplas. Elas serão chamadas de primárias, uma vez que seus registros servirão para a criação de tabelas intermediárias até a aquisição da tabela contendo o resultado da comparação entre os organismos desejados.

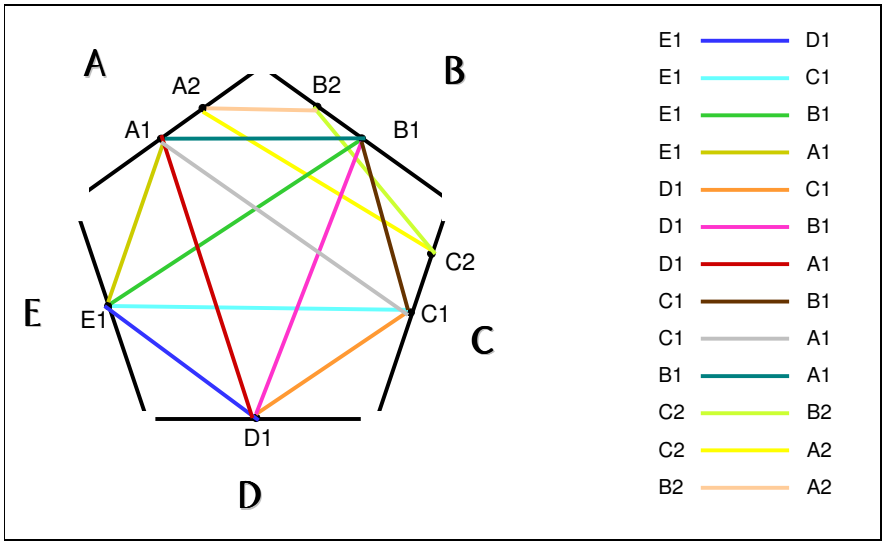


Figura 35 - Registros básicos recuperados das tabelas de ortologia para 5 organismos.

Assim, supondo o seguinte caso onde 5 diferentes proteomas (A-E) possuem algumas ortologias entre si, dispostas conforme Figura 35, estas informações são trazidas da tabela primária, lembrando que estão armazenadas seguindo uma ordenação decrescente de proteomas, propriedade a qual será mantida nas demais tabelas intermediárias e final.

As tabelas primárias armazenam pares de elementos ortólogos e com uma consulta simples sobre a base, recuperam-se todos os registros ortólogos que pertençam a um mesmo conjunto de organismos informado. Para o exemplo em questão, foram resgatadas as ocorrências existentes entre os cinco proteomas A, B, C, D e E, cada segmento unindo dois vértices representa uma ortologia, a qual poderá compor uma ortologia múltipla.

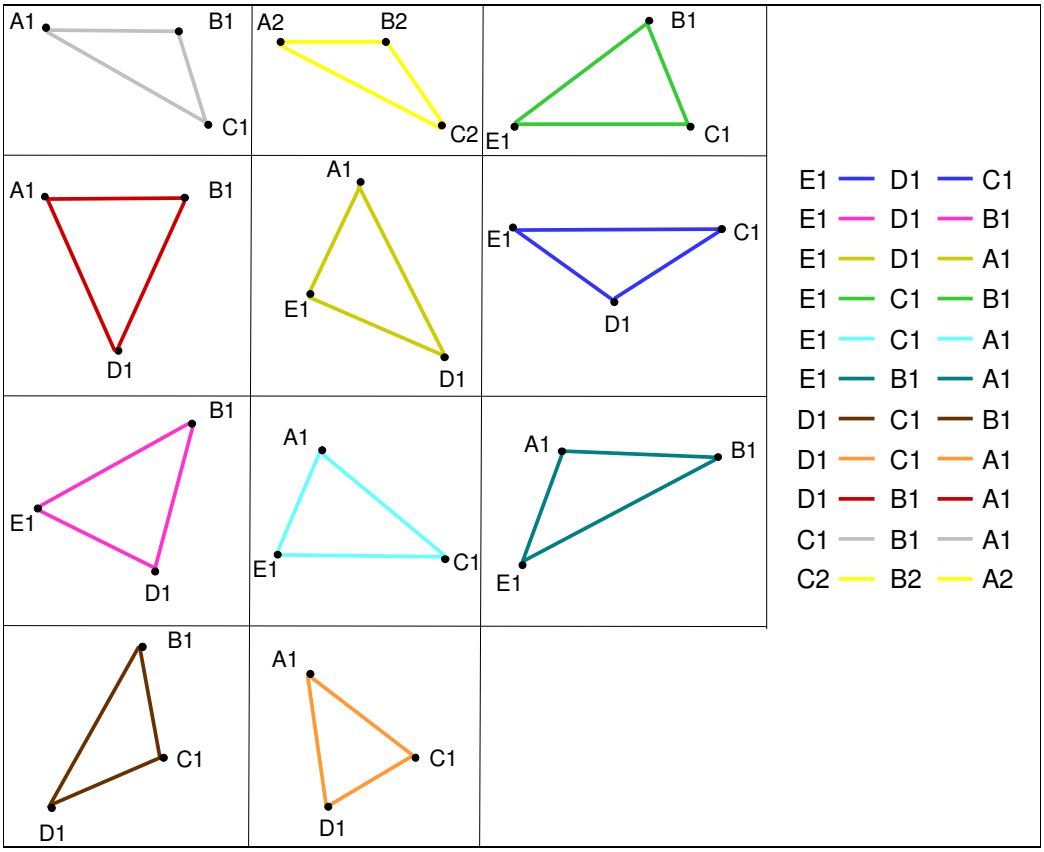


Figura 36 - Registros intermediários: Trios ortólogos.

Com as ortologias primárias recuperadas, é necessário verificar quais delas estão relacionadas de modo que cada vértice esteja ligado aos demais vértices do mesmo subconjunto. Um subconjunto refere-se à parte do conjunto que contem elementos pertencentes aos proteomas disitintos em número máximo ao permitido pelo nível, são incrementais e sempre são obtidos baseando-se nos resultados anteriores, por exemplo,

tomados as ortologias dois a dois, o próximo subconjunto a ser verificado são os que constituem ortologia múltipla entre três proteomas distintos, contendo no máximo três elementos que serão verificados, fundamentados pelas informações dos pares. Caso alguns pares não formem um triângulo, eles não serão integrados ao conjunto de resultado. Na Figura 36 são mostrados trios ortologos encontrados para o exemplo, enquanto na Figura 37 são exibidas as quadras ortólogas obtidas mediante consulta nos registros dos trios previamente calculados.

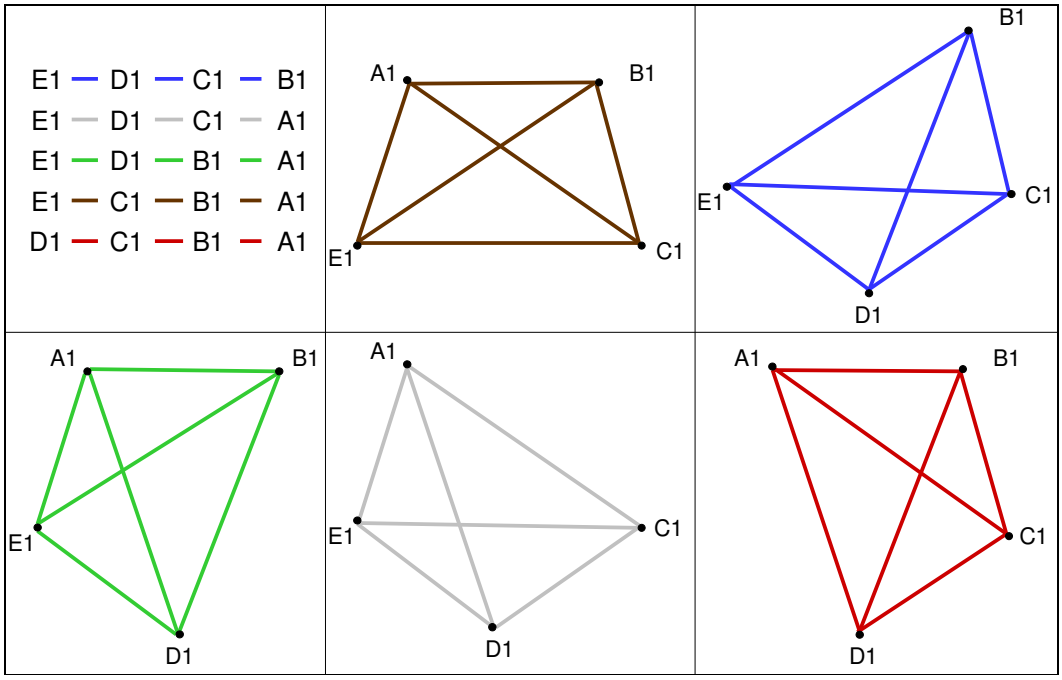


Figura 37 - Registros intermediários: Quadras ortólogas.

O processo de busca de ortologia múltipla termina quando são encontrados resultados abrangendo em um mesmo conjunto todos os proteomas selecionados, compondo a tabela final, como visto na Figura 38, ou então quando na verificação dos subconjuntos da tabelas intermediárias não são encontrados mais ocorrências que satisfaçam a condição do próximo nível.

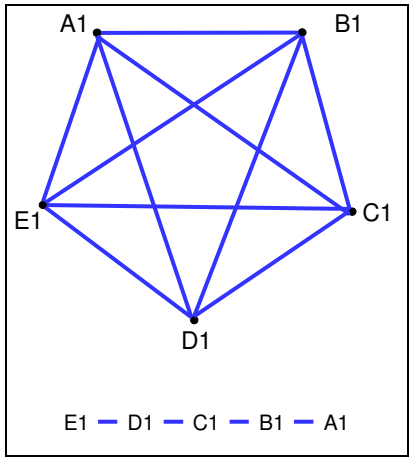


Figura 38 - Registro final.

Os conjuntos intermediários são armazenados em tabelas intermediárias e conforme aumenta a quantidade de campos que compõe a tabela, um novo nível é considerado. O nível zero é a tabela básica de RO ou GO, primeiro nível possui as informações sobre a ortologia múltipla entre três pontos, enquanto que o segundo nível, baseando nas informações do nível anterior, obtem registros sobre 4 pontos ortologos, e assim sucessivamente, até atingir a tabela final, compondo o nível N-2, onde N corresponde ao número de organismos desejados.

4.2.2 DETALHAMENTO DO PADRÃO – PVOM

Tomando como base essa simplificação, o primeiro passo na detecção do padrão é a verificação de relacionamentos entre os dados das tabelas GO e RO, a fim de obter a ortologia múltipla entre três proteomas. Os processos de obtenção de GOM e ROM são análogos, diferindo unicamente no objeto consultado: enquanto o primeiro verifica os genes através de seus GIs, o segundo certifica que as RGCs estão relacionadas. A título de entendimento do padrão, na Figura 39 está ilustrado este processo, considerado como nível 1, onde três proteomas 1, 2 e 3 encontram-se dispostos em uma matriz A. Na figura são mostradas as verificações necessárias para a passagem ao próximo nível, usando a notação de campo da matriz como [Matriz][Linha][.][Coluna]. Quando todos os elementos se relacionam entre si, é encontrado um resultado, também constante na figura, bem como a representação simbolizada por um quadrado verde adotada para demonstrar a primeira parte do padrão encontrado, denominado PVOM 1.

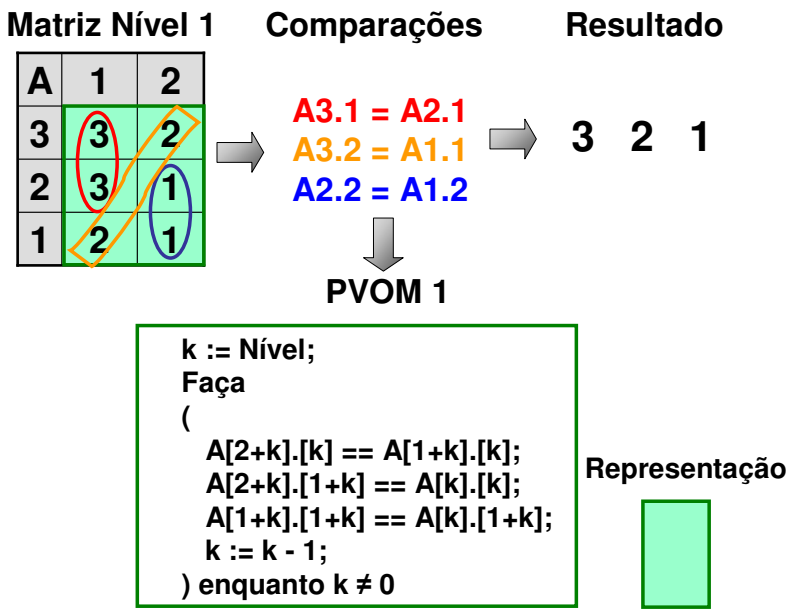


Figura 39 - Comparações de Nível 1.

No nível 2 estão as verificações de outros dois núcleos que comparam os registros extremos, o PVOM 2 e 3, os quais, juntos ao primeiro, completam a metodologia de obtenção de ortologia múltipla. Seguindo o mesmo esquema da Figura do nível 1, a Figura 40 ilustra esse processo, adicionando a cada nível um proteoma e seus respectivos conteúdos.

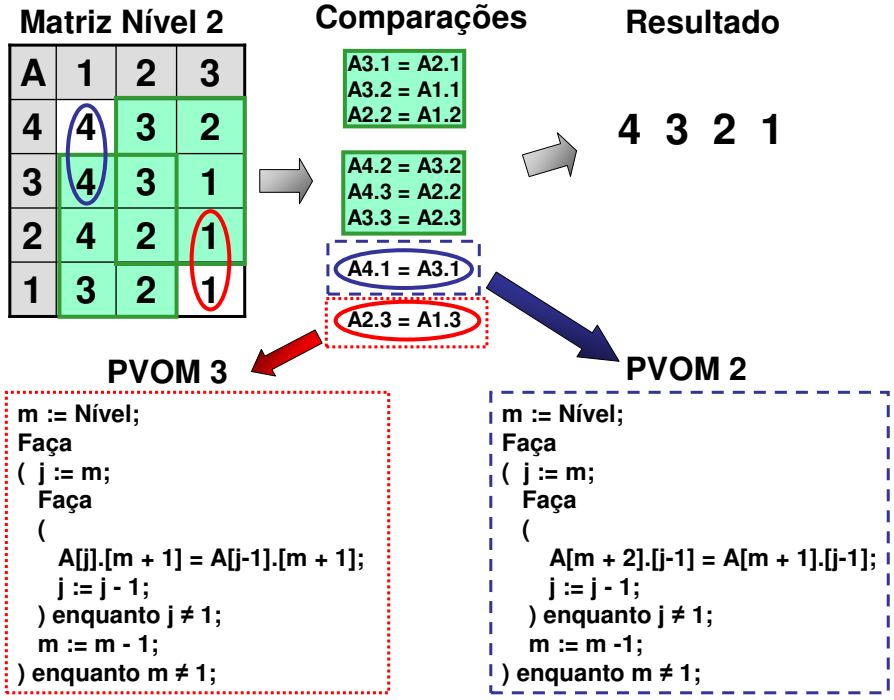


Figura 40 - Comparações de Nível 2.

Deste nível em diante, os três padrões encontrados nos passos iniciais atendem a todas as verificações e possibilidades, garantindo o intra-relacionamento dos dados.

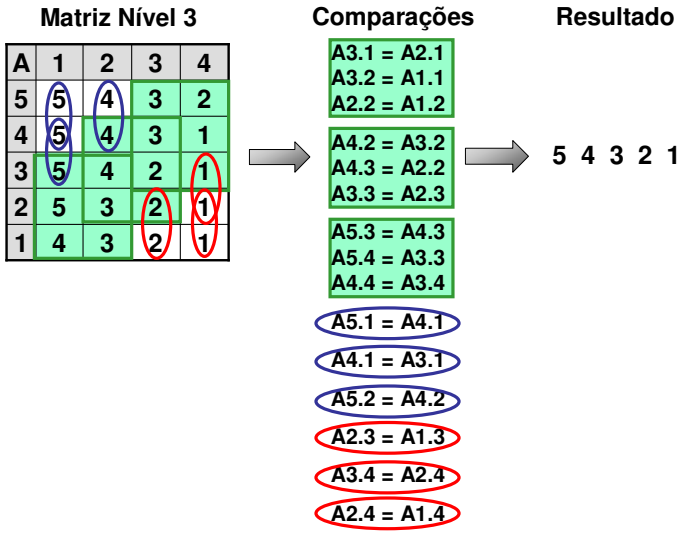


Figura 41 - Comparações de Nível 3.

A Figura 41 só vem a confirmar que o padrão encontrado se aplica a qualquer nível subseqüente.

4.2.3 CONSULTAS E TABELAS DINÂMICAS

Depois de encontrado um padrão para obtenção de ROMs e GOMs, foram implementados procedimentos e funções no banco de dados de modo a construírem essas consultas dinamicamente, além de criar, também dinamicamente, novas tabelas temporárias de resultados para cada nível, a partir da leitura dos dados da tabela do nível anterior. Isso se deve ao fato de que em cada nível, novas informações são geradas. Por exemplo, se se deseja obter os GOMs entre quatro espécies a partir dos GOs existentes entre eles, então o nível 1 conterá informações lidas da tabela de GOs, populando uma tabela de respostas que considera três espécies. No último passo, baseando-se nos dados anteriormente obtidos, construir-se-á uma nova tabela com resultados englobando os quatro elementos desejados. A cada passo intermediário desse processo, são criadas consultas e tabelas de forma a atendê-las.

5	4	3	-	-
5	4	2	-	-
5	4	1	-	-
5	3	1	-	-
5	3	2	-	-
5	2	1	-	-
4	3	2	-	-
4	3	1	-	-
4	2	1	-	-
3	2	1	-	-
5	4	3	2	-
5	4	3	1	-
5	4	2	1	-
5	3	2	1	-
4	3	2	1	-
5	4	3	2	1

Tabela 1 - Tabela esparsa.

O motivo da criação de uma tabela para cada nível está na economia de espaço e na facilidade de se localizar determinada informação. A economia é percebida na medida em que se cria uma tabela contendo todos os resultados (intermediários e finais) de uma consulta com, por exemplo, 5 proteomas que possuam uma ROM. Na Tabela 1 estão representados todos os registros das tabelas temporárias, cada nível com uma cor, necessários para a obtenção de uma ROM com esse número de proteomas. Nota-se que a tabela possui uma quantidade considerável de espaços que não são preenchidos; tabelas com essa característica recebem a denominação de esparsas.

Por essa razão não foi criada nesse trabalho uma tabela que armazenasse esses dados. As respostas são encontradas e disponibilizadas por demanda, em tempo de execução.

Para melhor entendimento do quão esparsa essa tabela seria, foi estimada a quantidade de registros necessária a cada nível para obter um único elemento ortólogo múltiplo com uma determinada quantidade de genes ou regiões. Este gráfico é apresentado na Figura 42, onde o eixo horizontal representa a quantidade de organismos distintos dos quais se deseja obter um único registro ortólogo múltiplo, enquanto o eixo vertical apresenta a quantidade de registros intermediários necessários para que se possa atingir no mínimo uma ocorrência. São estimados vários níveis em cores diferentes, como mostra o gráfico. Nota-se que para 20 organismos diferentes, são necessários mais de 800 registros no primeiro nível, para que se possa obter uma ortologia múltipla contendo essas espécies, o que equivale a 17 campos

vazios para cada um das oitocentas ocorrências, e apenas um eventual registro completamente preenchido.

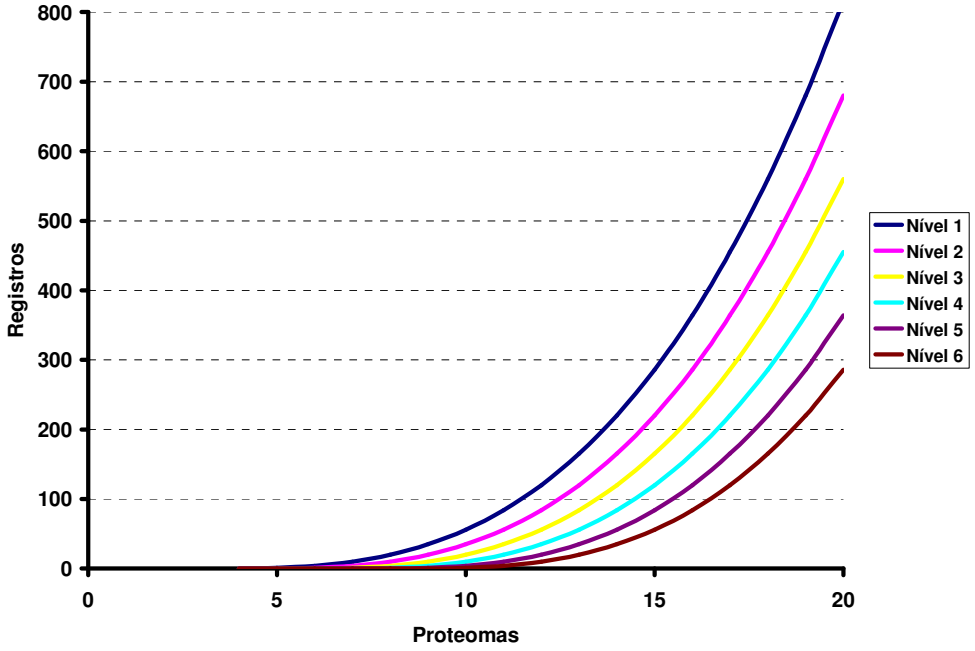


Figura 42 - Estimativa de registros.

O PVOM foi implementado por procedimentos e funções na linguagem PL/SQL (PORTFOLIO, 2001), os quais geram consultas e de tabelas dinâmicas. Seus códigos estão apresentados nos APÊNDICES A e B, dispostos por GOMs e ROMs respectivamente.

4.3 ISMOG

Finalizando a aplicação, os resultados devem ser exibidos ao usuário da forma mais transparente possível. Para isso foi implementada (ainda em processo de ajustes) uma ferramenta de consultas e visualização desses dados, através de uma aplicação WEB, que foi chamada de ISMOG (Interactive Search of Multiple Orthologous Groups). Ela é responsável por acessar os dados do DOG e disparar o PVOM para a disponibilização dos seus resultados em tabelas dinâmicas. Suas principais funcionalidades estão abaixo descritas.

Primeiramente o usuário deve escolher o tipo de informação que deseja adquirir, dentre as quais estão: GOs, ROs, GOMs e ROMs. Logo após, o conjunto de organismos contidos no DOG são exibidos para seleção, conforme procedimento ilustrado pela Figura 43.

Figura 43 - ISMOG: Seleção de organismos.

A partir dessas escolhas, para o conjunto de proteomas e para a opção selecionada ou são feitas consultas diretamente ao DOG, para os casos de RO ou GO, ou então procedimentos e funções que integram o PVOM são automaticamente disparados para os pedidos de ROM ou GOM, obtendo o conjunto de respostas de interesse do usuário, a quem deve ser exibido através do acesso às tabelas temporárias.

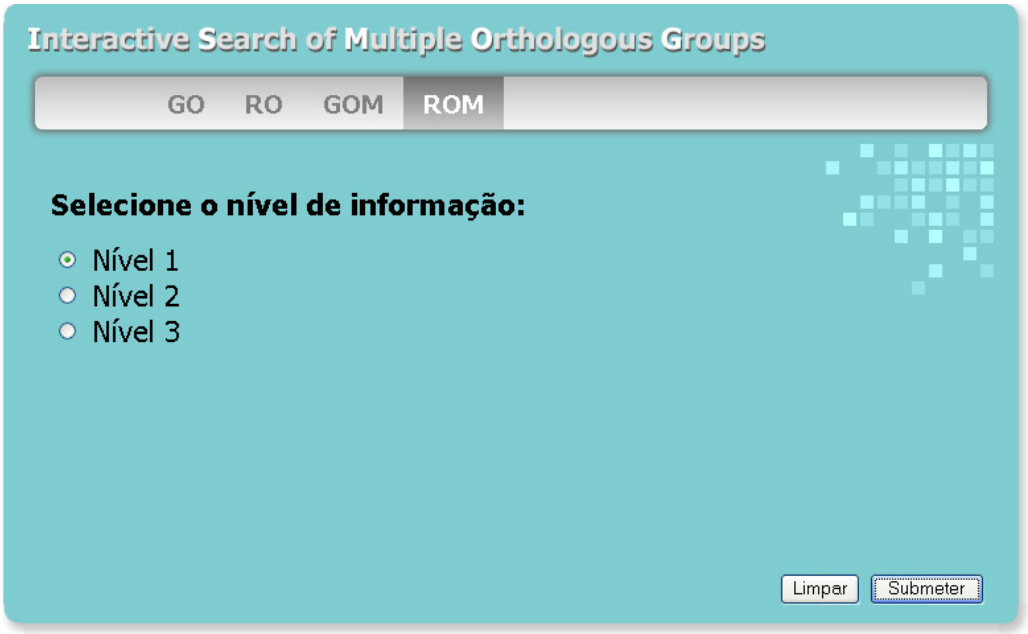


Figura 44 - ISMOG: Seleção de nível.

Os resultados são fornecidos por níveis, divisão semelhante à abordagem do PVOM, podendo o usuário navegar por diversos conjuntos de respostas de seu interesse. Tais níveis são dispostos como na Figura 44.

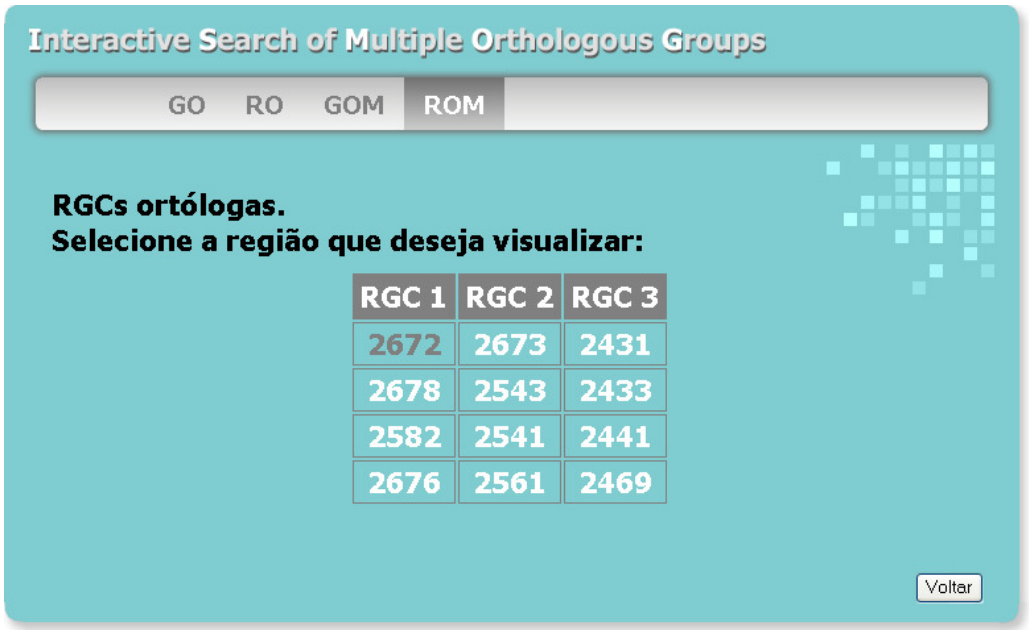


Figura 45 - ISMOG: Seleção de RGC.

Escolhida, por exemplo, uma região ortóloga de nível 1 (ortologias entre 3 proteomas), são exibidas as RGCs ortólogas, através de seus identificadores no banco, conforme pode ser visto na Figura 45.

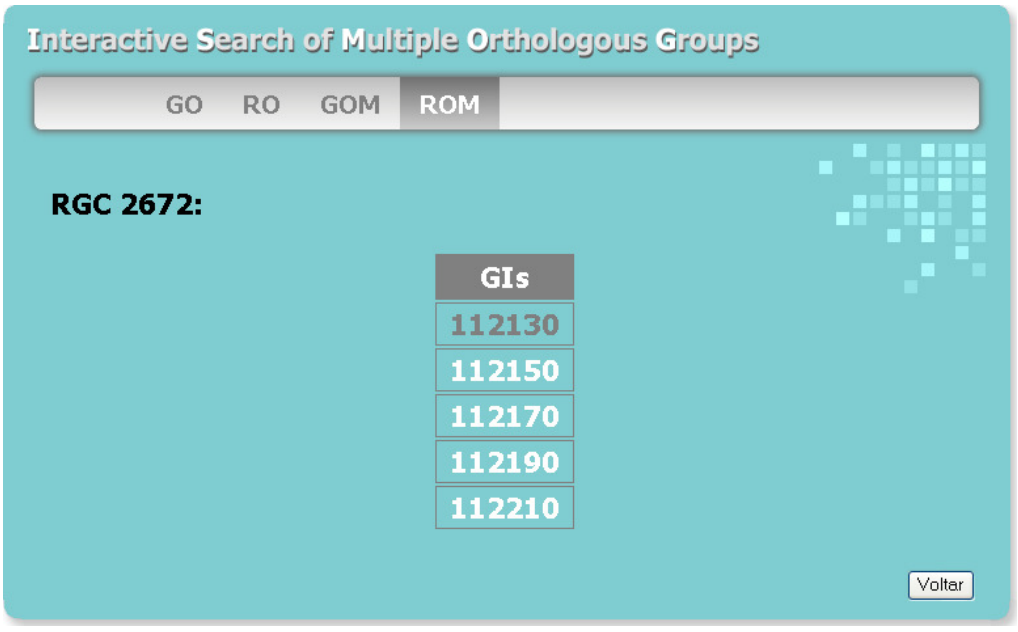


Figura 46 - ISMOG: Seleção de gene.

Selecionada a RGC, o usuário terá à sua disposição uma lista dos genes que formam esta região, representados pelos seus GIs, semelhante ao ilustrado na Figura 46.

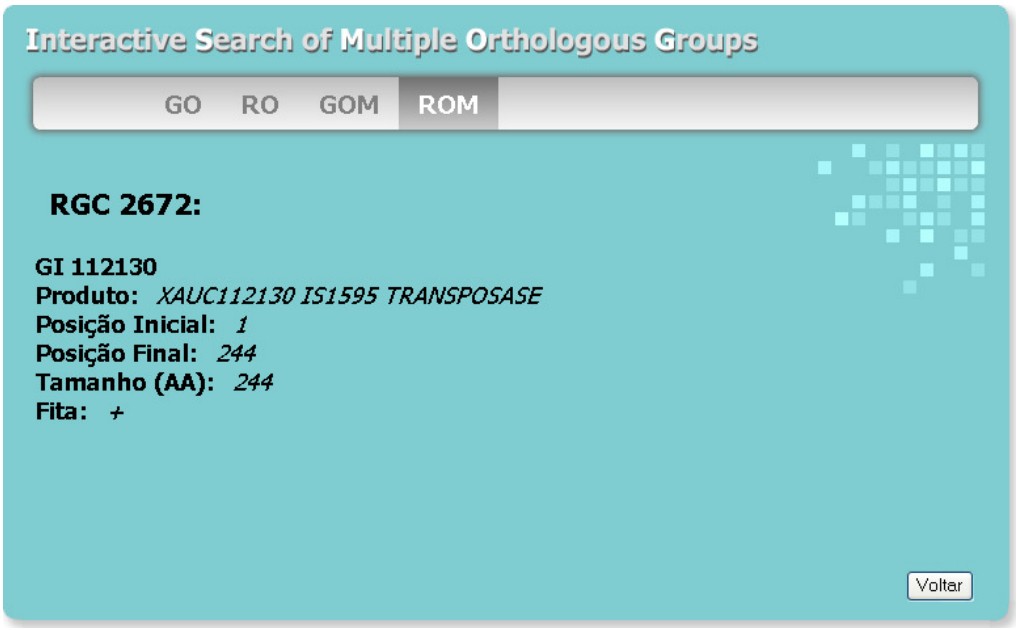


Figura 47 - ISMOG: Visualização do gene.

Finalizando, as informações de cada gene podem ser acessadas através da seleção do seu GI, sendo exibidos o seu produto, sua posição inicial e final, seu tamanho em aminoácidos e a fita em que foi obtido, como visto na Figura 47.

É importante ressaltar que outras consultas são possíveis através do acesso ao DOG e já estão sendo acrescidas ao ISMOG, como a de informações:

- de um gene pelo seu GI, espécie ou produto;
- de RGCs e ROs que apresentam um determinado gene;
- de todos os genes ortólogos a um determinado gene;
- de todas as regiões ortólogas a uma RGC.

Futuramente outros resultados serão disponibilizados, como a exibição das ROs e ROMs pelos seus genes ortólogos e a de genes específicos de cada espécie, assim como informações quantitativas, como: número de genes, de GOs, de genes específicos, de GOM, de RGCs, tamanho total do proteoma em aminoácidos e pares de bases.

4.4 DISCUSSÃO

Para comprovar que os resultados obtidos correspondem ao conjunto completo, contendo inclusive as respostas mais complexas, é necessário entender como as consultas geradas pelo PVOM funcionam no caso de GOM e ROM sobre as tabelas de GO e RO, respectivamente, as quais estão organizadas por proteomas em ordem decrescente.

Em uma primeira consulta, assim que fornecidos os proteomas desejados, estas tabelas base são acessadas e todos os dados contendo ortologia somente entre estes proteomas são recuperados. Supondo-se que a consulta executada seja sobre ROM, almejando tais regiões contidas nos proteomas A, B, C e D, conforme Figura 48. Todas as ROs existentes entre estas espécies são recuperadas: {RA1,RD1}, {RA2,RB2}, {RA2,RC1}, {RA2,RD3}, {RA3,RD2}, {RA3,RB1}, {RB1,RD2}, {RB2,,RC1}, {RB2,RD3}, {RB3,RC2} e {RC1,RD3}.

Numa segunda etapa, a partir do resultado anterior, são selecionados somente aqueles que formam triângulos, ou seja, que possuem três regiões que se relacionam entre si. O conjunto obtido é composto por 5 elementos: {RA3, RB1, RD2}, {RA2, RB2, RC1}, {RA2, RB2, RD3}, {RA2, RC1, RD3}, {RB2, RC1, RD3} Este resultado é então armazenado temporariamente em uma tabela para consultas e também para que o próximo nível procure ROMs.

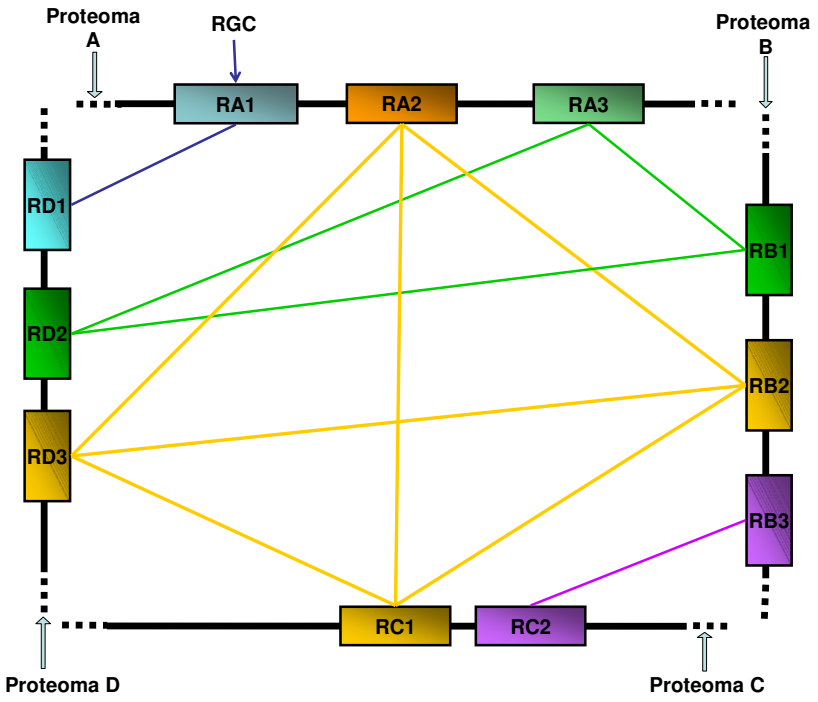


Figura 48 - Exemplo de ROM.

Este processo se repete até não serem encontradas mais ROMs ou então até completar as execuções possíveis necessárias para o conjunto de proteomas de consulta.

Nesse exemplo, ainda é realizada uma última execução de busca do PVOM, procurando por ROMs compostas por 4 RGCs onde todas se inter-relacionam. O conjunto final de respostas é formado por {RA2, RB2, RC1, RD3}.

Finalizando esse processo, o usuário tem ao seu dispor todas as respostas, incluindo as intermediárias, encontradas durante a execução do PVOM em diferentes níveis, ou seja, é fornecido o conjunto completo de respostas. Obtido esse conjunto, as que se localizam no maior nível são, obviamente, as mais complexas.

Nome do Organismo	Abreviação do Proteoma	Genes	RGCs	Total de Genes nas RGCs
Bifidobacterium Longum	**BI**	1729	66	417
Corynebacterium Glutamicum	**CG**	3040	129	1455
Leifsonia Xyli	**LE**	2456	89	877
Mycobacterium Leprae	**ML**	1605	108	1630
Mycobacterium Tuberculosis	**MT**	3918	155	2100
Streptomyces Coelicolor	**SC**	7512	180	1440

Tabela 2 - Genes e RGCs do DOG.

Como exemplo foram carregadas no DOG as saídas do EGG entre seis espécies, *Bifidobacterium Longum, Corynebacterium Glutamicum, Leifsonia Xyli, Mycobacterium Leprae, Mycobacterium Tuberculosis* e *Streptomyces Coelicolor*. Através de consultas diretas ao banco de dados as informações da Tabela 2 foram obtidas. Para facilitar as próximas análises, uma abreviação de cada organismo está disposta nessa tabela. Os números nela apresentados referem-se à quantidade de genes, RGCs e a quantidade total de genes nestas regiões.

A Tabela 3 fornece o número de GOs e ROs para cada par de proteomas confrontados pelo EGG. Tais informações estão armazenadas em tabelas no banco e são recuperadas diretamente por consultas.

Pares de Proteomas		GO	RO
CG	**BI**	863	40
CG	**MT**	1400	111
CG	**SC**	1733	92
CG	**ML**	989	100
CG	**LE**	1078	70
LE	**BI**	790	39
LE	**MT**	986	68
LE	**SC**	1467	65
LE	**ML**	771	70
ML	**BI**	625	44
ML	**SC**	1103	68
MT	**BI**	755	30
SC	**BI**	1209	43
SC	**MT**	2074	101
ML	**MT**	1663	113

Tabela 3 - ROs e GOs do DOG.

Da Tabela 4 à Tabela 7 são apresentados os resultados quantitativos de cada nível, processando o PVOM para ROMs e GOMs, assim como o tempo de execução em segundos, utilizado em cada processamento. As repostas obtidas pelo nível 1 são resultantes da execução do padrão sobre as tabelas de RO e de GO, armazenadas em uma tabela temporária criada dinamicamente, de forma a atender à quantidade de campos necessária.

Nível 1				
Proteomas			ROMs	GOMs
SC	MT	BI	18	763
ML	MT	BI	22	620
ML	SC	BI	29	672
ML	SC	MT	60	1236
LE	MT	BI	16	585
LE	SC	BI	24	989
LE	SC	MT	49	1002
LE	ML	BI	29	532
LE	ML	MT	54	779
LE	ML	SC	44	771
CG	MT	BI	20	685
CG	SC	BI	26	963
CG	SC	MT	57	1379
CG	ML	BI	33	574
CG	ML	MT	78	992
CG	ML	SC	57	848
CG	LE	BI	23	696
CG	LE	MT	50	874
CG	LE	SC	42	1165
CG	LE	ML	57	695
Tempo de execução (sec): 0.142				

Tabela 4 - Execução: ROMs e GOMs no primeiro nível.

Já é possível observar, nesta fase, quais os proteomas que apresentam maior afinidade, tanto na comparação de regiões quanto de genes isolados.

Nível 2					
Proteomas				ROMs	GOMs
ML	SC	MT	BI	14	784
LE	SC	MT	BI	15	953
LE	ML	MT	BI	23	674
LE	ML	SC	BI	33	857
LE	ML	SC	MT	45	1042
CG	SC	MT	BI	17	1122
CG	ML	MT	BI	29	666
CG	ML	SC	BI	44	797
CG	ML	SC	MT	64	1068
CG	LE	MT	BI	14	646
CG	LE	SC	BI	29	1355
CG	LE	SC	MT	55	1369
CG	LE	ML	BI	31	610
CG	LE	ML	MT	51	794
CG	LE	ML	SC	42	885
Tempo de execução (sec): 0.156					

Tabela 5 - Execução: ROMs e GOMs no segundo nível.

As informações provenientes do PVOM executado sobre as tabelas temporárias de ROM e de GOM do primeiro nível são então destinadas a uma segunda tabela dinâmica, onde a quantidade de campos necessários para o armazenamento do resultado obtido é maior. No entanto, a quantidade de registros encontrados a cada nível é menor.

Nível 3						
Proteomas					**ROMs**	**GOMs**
LE	**ML**	**SC**	**MT**	**BI**	14	1284
CG	**ML**	**SC**	**MT**	**BI**	17	1195
CG	**LE**	**SC**	**MT**	**BI**	22	2112
CG	**LE**	**ML**	**MT**	**BI**	18	866
CG	**LE**	**ML**	**SC**	**BI**	37	1328
CG	**LE**	**ML**	**SC**	**MT**	42	1593
Tempo de execução (sec): 0.295						

Tabela 6 - Execução: ROMs e GOMs no terceiro nível.

No último nível realizado para essa execução, visto na Tabela 7, são então exibidos os registros que resultam da ortologia entre quaisquer dois membros desse conjunto.

Nível 4							
Proteomas						**ROMs**	**GOMs**
CG	**LE**	**ML**	**SC**	**MT**	**BI**	9	2135
Tempo de execução (sec): 0.039							

Tabela 7 - Execução: ROMs e GOMs no quarto nível.

Ao final do processo todas essas informações estão disponíveis nas diversas tabelas temporárias correspondentes aos vários níveis, criadas para atender à execução solicitada.

5 CONCLUSÃO

O objetivo deste projeto foi a criação de um banco de dados para o suporte ao reconhecimento de genes e regiões de genes que se preservaram entre espécies durante a evolução. Em adição, através do processamento dos dados armazenados, oferecer via Web acesso a essas informações fornecendo respostas completas e complexas envolvendo o domínio de bioinformática.

Para essa finalidade foram estudados conceitos de biologia molecular, bancos de dados genômicos, ferramentas de comparação de proteomas e de genomas, ferramentas de agrupamento de ortologias, entre outros.

Os resultados obtidos foram a ferramenta ISMOG, a qual processa consultas sobre o banco de dados genômico DOG, diretamente ou através de um padrão aqui definido e que foi denominado PVOM.

Assim, as principais contribuições foram:

(1) A criação do banco de dados DOG o qual armazena informações necessárias para a obtenção de ortologias múltiplas. As principais atividades envolveram:
   - a modelagem dos dados semânticos;
   - o desenvolvimento de programas AWK para formatação dos resultados do EGG;
   - processos de carga do banco e de atualizações de RGCs.

(2) Definição do PVOM, o qual estabelece um padrão de verificação de ortologias múltiplas baseado em consultas ao banco de dados construído. As atividades envolvidas na implementação do padrão proposto foram:
   - consultas dinâmicas;
   - tabelas dinâmicas;

(3) Criação do ISMOG: ferramenta de consulta via Web sobre os dados do DOG e execução do PVOM;

(4) Resultados completos, incluindo os mais complexos, sobre ortologia múltipla.

A partir dos resultados obtidos, a seguir são listados alguns dos possíveis trabalhos futuros já identificados:

- A integração do EGG com o DOG de forma a possibilitar a automação do processo de cargas;
- O estudo da utilização de lógica nebulosa no agrupamento de RGCs;

- O estudo de bancos de dados indicados para suportar tabelas esparsas, de modo a armazenar as ortologias múltiplas obtidas, possibilitando as anotações desses grupos e evitando o re-processamento do PVOM sobre o DOG.
- Integração ao ISMOG de uma ferramenta de visualização gráfica para as ROMs e GOMs;
- Estudo da generalização do PVOM na resolução de outros problemas NP-Difíceis;
- Expansão do EGG e do DOG para suportar paralogia, de forma a aumentar a quantidade de informações disponibilizadas ao usuário e a qualidade das pesquisas, uma vez que, a maiorias das ferramentas hoje desenvolvidas possuem e utilizam esse conceito;
- Uma melhoria ao EGG é destacada nos arquivos ".reg", nos quais as RGCs são exibidas com seus GIs e produtos, enquanto que nas ROs somente são expostos os produtos. O campo que cria vínculo nesse caso é descritivo e pode ser repetido inúmeras vezes para GIs distintos. Uma sugestão é a inclusão do GI junto às ROs, de forma a não gerar informações errôneas e garantir maior integridade dos dados.

Para os desenvolvimentos aqui contidos foi utilizado um computador equipado com processador AMD Athlon XP 3000+, contendo 512Mb de Ram e um HD de 60GB, operando com dois sistemas Windows XP Home e Linux Fedora. Os softwares utilizados foram o AWK contido na instalação básica do Linux, uma ferramenta gráfica de ETL, o Informática PowerCenter 7.1 (SCHMIDT, 2004), utilizado no tratamento e carga do DOG. Esta base foi instanciada no Sistema Gerenciador de Banco de Dados Oracle 9i, disponível em <http://www.oracle.com/technology/software/products/ oracle9i/index.html> para download. O PVOM foi implementado através de procedimentos e funções em PL/SQL enquanto o ISMOG foi desenvolvido em PHP.

Os testes com as demais ferramentas principalmente com o BAGRE não foram possíveis devido às diferentes abordagens dadas à solução do problema. No BAGRE, o algoritmo funde RGCs e procura ROMs a cada execução, enquanto o PVOM agrupa as RGCs sobrepostas a cada submissão, restando para cada execução apenas a verificação de ortologia múltipla.

REFERÊNCIAS BIBLIOGRÁFICAS

ABASCAL, F. e VALENCIA, A. **Clustering of proximal sequence space for the identification of protein families**. Bioinformatics, v.18, n.7, p.908 - 921. 2002.

ALMEIDA, N. F. D. **Ferramentas para comparação genômica**. (Tese de Doutorado). Instituto de Computação, UNICAMP, Campinas, 2002.

AUNG, Z. **A Preliminary Study on the Existing Techniques for Biological Database Searching**. A Report Submitted for PhD Upgrading Qualifier Examination. 2001.

BAIROCH, A. *et al.* **The Universal Protein Resource (UniProt)**. Nucleic Acids Research, v.33, p.D154-D159. 2005.

BAXEVANIS, A. D. **The Molecular Biology Database Collection: an update compilation of biological database resources**. Nucleic Acids Research, v.29, p.1-10. 2001.

BENSON, D. A. *et al.* **GenBank: update**. Nucleic Acids Research, v.32, p.D23-D26. 2004.

_____. **GenBank**. Nucleic Acids Research, v.34, p.D16-D20. 2006.

BSCS (Biological Sciences Curriculum Study). **Biologia das Moléculas ao Homem**: Edart. 1995.

CHEANG, I. K. *et al.* **Overview of the Structures of Heterogeneous Genome Database**. Twenty-Seventh Hawaii International Conference on Biotechnology Computing. 1994.

CHEN, F. *et al*. **OrthoMCL-DB: querying a comprehensive multi-species collection of ortholog groups**. Nucleic Acids Research, v.34, p.D363-D368. 2006.

CHENNA, R. *et al*. **Multiple sequence alignment with the Clustal series of programs**. Nucleic Acids Research, v.31, n.13, p.3.497 - 3.500. 2003.

CHYNOWETH, D. P. **Course: Applied Microbial Biotechnology**. *In:* University of Florida Website. http://www.agen.ufl.edu/~chyn/age4660/4660demo.htm. Acessado em Julho de 2006

COCHRANE, G. *et al*. **EMBL Nucleotide Sequence Database: developments in 2005**. Nucleic Acids Research, v.34, p.D10-D15. 2006.

DAYHOFF, M. *et al*. **A model of evolutionary change in proteins. Atlas of protein sequence and structure**, p.345 - 352. 1978.

DDBJ (DNA DataBank of Japan). http://www.ddbj.nig.ac.jp/Welcome-j.html. Último acesso em Julho de 2006.

ELMASRI, R. e NAVATHE, S.B. **Fundamentals of Database Systems**: Addison- Wesley Publishing Company, 3a. edição, 955 pp., 2000.

EMBL (European Molecular Biology Laboratory). http://www.ebi.ac.uk/embl/ Último acesso em Julho de 2006.

GALISSON, F. **Biological Sequence Databases**. http://www.comp.nus.edu.sg /~zeyaraun/ Docs/ sequencedb-uk.pdf 2001.

GENBANK. GenBank. http://www.ncbi.nlm.nih.gov/Genbank/index.html 2006.

GEWANDSZNAJDER, F. e LINHARES, S. **Biologia Hoje: Genética, Evolução, Ecologia**: Ática. 2003.

GUSFIELD, D., Algorithms **on Strings, Trees and Sequences Computer Science and Computational Biology**. Press Syndicate of the University of Cambridge. USA, 1997.

HENIKOFF, S. e HENIKOFF, J. G. **Amino-acid substitution matrices from protein blocks**. National Academy of Sciences, v.89, p.10.915 - 10.919. 1992.

HUBBARD, T. *et al*. **Ensembl 2005**. Nucleic Acids Research, v.33, p.D447-D453. 2005.

IAL (Instituto Adolfo Lutz). **O que é Bioinformática?** http://www.ial.sp.gov.br/bioinfo/oqbioinfo.htm 2006.

ITOH, M. *et al*. **Clustering of Database Sequences for Fast Homology**. Genome Informatics, v.15, n.1, p.93-104. 2004.

JENSEN, R. A. **Orthologs and paralogs - we need to get it right**. Genome Biology, v.2, n.8, p.1002.1 - 1002.3. 2001.

KANEHISA, M. *et al*. **The KEGG database at GenomeNet**. Nucleic Acids Research, v.30, p.42 - 46. 2002.

KEGG (Kyoto Encyclopedia of Genes and Genomes). http://www.genome.jp/kegg/. Último acesso em Junho de 2006.

KESSLER, C. C. **Citologia**. *In*: Biologia na Web. http://www.cynara.com.br/. Acessado em Julho de 2006

LEE, Y. *et al*. **Cross-Referencing Eukaryotic Genomes: TIGR Orthologous Gene Alignments (TOGA)**. Genome Research, v.12, p.493 - 502. 2002.

LEMOS, M. **Um Estudo dos Algoritmos de Montagem de Fragmentos de DNA**. Departamento de Computação, PUC-RJ, 2003.

LENGAUER, T. **Computational Biology at the Beginning of the Post-genomic Era**. MPI Informatik: http://domino.mpi-sb.mpg.de/internet/. 2000.

LI, L. *et al*. **OrthoMCL: Identification of Ortholog Groups for Eukaryotic Genomes**. Genome Research, v.13, p.2.178 - 2.189. 2003.

MARTINS, W. S. **Discovery Bioinformatics Lecture Notes**. University of Delaware: http://www.capsl.udel.edu/courses/eleg667/2000/Slideindex.html 2000.

MBGD (Microbial Genome Database). http://mbgd.genome.ad.jp/ Último acesso em Julho de 2006.

MONTERA, L. **Regiões Ortólogas em Múltiplos Genomas**. (Dissertação de Mestrado). Departamento de Computação e Estatística, UFMS, 2004.

NCBI. Página da NCBI. *In*: National Center for Biotechnology Information Website. http://www.ncbi.nlm.nih.gov/ BLAST/. Último acesso em Julho de 2006.

O'BRIEN, K. P. *et al*. **OrthoDisease: A Database of Human Disease Orthologs. Human Mutation**, v.24, p.112-119. 2004.

_____. **Inparanoid: a comprehensive database of eukaryotic orthologs**. Nucleic Acids Research, v.33, p.D476-D480. 2005.

PEARSON, W. R. **Searching Protein Sequence Libraries: Comparison of the Sensitivity and Selectivity of the Smith-Waterman and FASTA algorithms**. Genomics, v.11, p.635 - 650. 1991.

PIR. Página do PIR. In: Protein Information Resource Website. http://pir.georgetown.edu/. Último acesso em Julho de 2006.

PORTFOLIO, T. **PL/SQL User's Guide and Reference**: Oracle 2001.

QUACKENBUSH, J. *et al*. **The TIGR Gene Indices: analysis of gene transcript sequences in highly sampled eukaryotic species**. Nucleic Acids Research, v.29, n.1, p.159-164. 2001.

ROBBINS, A. D. **AWK Language Programming: A User's Guide for GNU AWK**. Free Software Foundation. 1996.

ROBERTIS, E. M. F. D. **Bases da Biologia Celular e Molecular**: Guanabara Koogan. 2001. 418 p.

SASSON, S. e SILVA, C. **Biologia:** Saraiva, v.3. 2002. 480 p.

SBFIS. **Boletim**. In: Sociedade Brasileira de Fisiologia Website. http://www.sbfis.org.br/boletim/ Boletim%2023(1)%201998.htm. 23 1998.

SCHMIDT, D. C. **Informatica PowerCenter Designer Guide**: Informatica Corporation 2004.

SEIBEL, L. F. **Bio-AXS: Uma Arquitetura para Integração de Fontes de Dados e Aplicações de Biologia Molecular**. (Tese de Doutorado). Departamento de Informática, PUC-RJ, Rio de Janeiro, 2000.

SILVA, F. H. da. **Relatório Técnico: Fundamentos em Biotecnologia**: Departamento de Genética e Evolução, Centro de Biotecnologia Molecular e Estrutural, Universidade Federal de São Carlos, São Carlos, 2000.

SPINGA, R. A **Biologia nos Vestibulares**: Navegar. 1998.

SWISS-PROT. Página da SWISS-PROT. http://us.expasy.org/sprot/ 2006.

TATUSOV, R. L. *et al*. **The COG database: new developments in phylogenetic classification of proteins from complete genomes**. Nucleic Acids Research, v.29, p.22 - 28. 2001.

_____. **The COG database: an update version includes eukaryotes**. BMC Bioinformatics, v.4, n.41, p.1-14. 2003.

TIGR. Página da TIGR. *In:* The Institute for Genomic Research Website. http://www.tigr.org/. Último acesso em Julho de 2006.

UCB. **Entendendo a Biotecnologia**. *In:* Universidade Católica de Brasília Website. http://www.ucb.br/posgraduacao/biotecnologia/defini%E7%E3o.htm 2004.

UCHIYAMA, I. **MBGD: microbial genome database for comparative analysis**. Nucleic Acids Research, v.31, p.58 - 62. 2003.

XU, W. et al. **Indexing Protein Sequences in Metric Space**. Technical Report TR-04-06. The University of Texas at Austin. Department of Computer Sciences, 2003.

WHEELER, D. L. *et al*. **Database resources of the National Center for Biotechnology Information**. Nucleic Acids Research, v.31, n.1, p.28-33. 2003.

_____. **Database resources of the National Center for Biotechnology Information**. Nucleic Acids Research, v.34, p.D173-D180. 2006.

WU, C. H. *et al*. **The Universal Protein Resource (UniProt): an expanding universe of protein information.** Acids Research, v.34, p.D187-D191. 2006.

APÊNDICE A – PL/SQL DO PVOM PARA GOMS

```
/*****************************************************************
*          Procedimento de chamada do PVOM para GOMs             *
*****************************************************************/
CREATE OR REPLACE PROCEDURE find_gom AS

  --A tabela prtm_cons contem os proteomas selecionados pelo usuário
  CURSOR c_tot IS Select count(*) as total From prtm_cons;

  ---------Declaração de variáveis---------------
  rec_tot c_tot%ROWTYPE;
  i integer := 1;
  j Number;
  nr_prtms NUMBER := 0;
  nr_exec NUMBER := 0;
  c varchar(2000);

  BEGIN

    -- primeira execução através de consulta sobre a tabela de GO
    j := cria_res_tmp_gom;

    -- calcula a quantidade de níveis a serem gerados
    IF j = 0 THEN
       dbms_output.put_line('Nao ha GOMs para estes proteomas');
     ELSE
       open c_tot;
       fetch c_tot into rec_tot;
       nr_prtms := rec_tot.total;
       dbms_output.put_line(nr_prtms);
       nr_exec := nr_prtms - 3;
       dbms_output.put_line(nr_exec);
       close c_tot;

       -- gera consulta dinâmica a cada nível
       LOOP
         exit when i = nr_exec + 1;
         c := gera_query_gom(i);
         EXECUTE IMMEDIATE c;
         i := i + 1;
       END LOOP;
    END IF;
END find_gom;
/*****************************************************************/



/*****************************************************************
*     Função de criação da primeira tabela temporária - Nível 1  *
*****************************************************************/
CREATE OR REPLACE FUNCTION cria_res_tmp_gom return integer IS

  ---------Declaração de variáveis---------------
  CURSOR c_goms IS Select count(*) as goms From gom_res_1;
  rec_goms c_goms%ROWTYPE;
  r integer := 1;
  nr_goms integer;

  BEGIN

    --------Apaga a tabela de nível 1--------------
    EXECUTE IMMEDIATE 'DROP TABLE SYSTEM.GOM_RES_1';

    ----Cria a tabela de nível 1 para os organismos selecionado
    EXECUTE IMMEDIATE
    'CREATE TABLE SYSTEM.GOM_RES_1 AS
```

```
    SELECT DISTINCT
      A.GI_1 as GI_1,
      B.GI_2 as GI_2,
      C.GI_2 as GI_3,
      A.ID_GO as ID_GO_1,
      B.ID_GO as ID_GO_2,
      C.ID_GO as ID_GO_3,
      A.ID_PRTM_1 as ID_PRTM_1,
      B.ID_PRTM_2 as ID_PRTM_2,
      C.ID_PRTM_2 as ID_PRTM_3
    FROM    SYSTEM.GO A,
            SYSTEM.GO B,
            SYSTEM.GO C
    WHERE   A.GI_1 = B.GI_1  AND
            B.GI_2 = C.GI_1  AND
            A.GI_2 = C.GI_2 AND
            A.ID_PRTM_1 in (select id_prtm from prtm_cons) AND
            B.ID_PRTM_1 in (select id_prtm from prtm_cons) AND
            C.ID_PRTM_1 in (select id_prtm from prtm_cons) AND
            A.ID_PRTM_2 in (select id_prtm from prtm_cons) AND
            B.ID_PRTM_2 in (select id_prtm from prtm_cons) AND
            C.ID_PRTM_2 in (select id_prtm from prtm_cons) AND
            A.ID_PRTM_1 > B.ID_PRTM_2 AND
            B.ID_PRTM_2 > C.ID_PRTM_2
    ORDER BY
            A.ID_PRTM_1,
            B.ID_PRTM_2,
            C.ID_PRTM_2,
            A.GI_1,
            B.GI_2,
            C.GI_2';

    open c_goms;
    fetch c_goms into rec_goms;
    nr_goms := rec_goms.goms;
    close c_goms;
    IF nr_goms = 0 THEN
       RETURN (nr_goms);
     ELSE
       RETURN (r);
    END IF;
END cria_res_tmp_gom;
/*****************************************************************/



/*****************************************************************
*Função de criação dinâmica das tabelas e de criação das consultas*
*****************************************************************/
CREATE OR REPLACE FUNCTION gera_query_gom(i integer) return CLOB AS

  ---------Declaração de variáveis---------------
  c clob;
  j integer;

  BEGIN
   j := i + 1;

   --------Apaga a tabela de nível j--------------
   EXECUTE IMMEDIATE 'DROP TABLE SYSTEM.GOM_RES_'||j;

   --Geração da consulta dinâmica para criação da tabela de nível j
   c:= 'CREATE TABLE SYSTEM.GOM_RES_'||j||' AS ';

   --chamada da função de criação do select para GIs
   c:= c||sel_stmt_gi(i);

   --chamada da função de criação do select para GOs
   c:= c||sel_stmt_go(i);
```

```
   --chamada da função de criação do select para PROTEOMAS
   c:= c||sel_stmt_prtm(i);

   --chamada da função de criação da cláusula FROM para as tabelas
temporárias
   c:= c||from_stmt_gom(i);

   --chamada da função de criação da cláusula WHERE para GOMs
   c:= c||where_stmt_gom(i);

   --chamada da função de ordenação de proteomas e fechamento da consulta
   c:= c||final_stmt(i);

   --o retorno da função é um Character Large Object contendo a consulta
gerada dinamicamente para o nível i
   RETURN (c);

  END gera_query_gom;
/*****************************************************************/



/*****************************************************************
*        SEL_STMT_GI - renomeia os GIs para execução do PVOM     *
*****************************************************************/
CREATE OR REPLACE FUNCTION sel_stmt_gi(i integer) return clob is

  ---------Declaração de variáveis---------------
  c clob := ' SELECT DISTINCT ';
  k integer:= 1;
  j integer;

  BEGIN

    j := i + 3;
    LOOP
      exit when k = i + 3;
      c := c||'A'||j||'.GI_'||k||' as GI_'||k||', ';
      k := k + 1 ;
    END LOOP;

    j := k - 1;
    c:= c||'A1'||'.GI_'||j||' as GI_'||k||', ';

    RETURN (c);

  END sel_stmt_gi;

/*****************************************************************/



/*****************************************************************
*     SEL_STMT_GO - renomeia os ID_GOs para execução do PVOM     *
*****************************************************************/
CREATE OR REPLACE FUNCTION sel_stmt_go(i integer) return clob is

  ---------Declaração de variáveis---------------
  c clob;
  k integer:= 1;
  j integer;

  BEGIN

    j := i + 3;
    LOOP
      exit when k = i + 3;
      c := c||'A'||j||'.ID_GO_'||k||' as ID_GO_'||k||', ';
      k := k + 1 ;
    END LOOP;
```

```
    j := k - 1;
    c := c||'A1'||'.ID_GO_'||j||' as ID_GO_'||k||', ';

    RETURN (c);

  END sel_stmt_go;
/*****************************************************************/



/*****************************************************************
*     SEL_STMT_PRTM - renomeia os ID_PRTMs para execução do PVOM *
*****************************************************************/
CREATE OR REPLACE FUNCTION sel_stmt_prtm(i integer) return clob is

  ---------Declaração de variáveis---------------
  c clob;
  k integer:= 1;
  j integer;

  BEGIN

    j := i + 3;
    LOOP
      exit when k = i + 3;
      c := c||'A'||j||'.ID_PRTM_'||k||' as ID_PRTM_'||k||', ';
      k := k + 1 ;
    END LOOP;

    j := k - 1;
    c := c||'A1'||'.ID_PRTM_'||j||' as ID_PRTM_'||k;

    RETURN (c);
  END sel_stmt_prtm;
/*****************************************************************/

/*****************************************************************
*FROM_STMT_GOM - criação da cláusula FROM baseada no nível anterior*
*****************************************************************/
CREATE OR REPLACE FUNCTION from_stmt_gom(i integer) return clob is

  ------Declaração de variáveis---------------
  c clob := ' FROM ';
  k integer := 1;
  m integer := 1;
  j integer;

  BEGIN
    j := i;
    LOOP
      exit when m = i + 3;
      c := c||'system.GOM_RES_'||j||' A'||m||', ';
      m := m + 1;
    END LOOP;

    IF m = i + 3  THEN
      c := c||'system.GOM_RES_'||j||' A'||m;
    END IF;

    RETURN (c);

END from_stmt_gom;
/*****************************************************************/



/*****************************************************************
```

```
*WHERE_STMT_GOM - Função onde os PVOMs 1, 2 e 3 são gerados para GOM *
**********************************************************************/
CREATE OR REPLACE FUNCTION where_stmt_gom(i integer) return clob is

  ---------Declaração de variáveis---------------
  c clob := ' WHERE ';
  a1 integer;
  a2 integer;
  a3 integer;
  k integer := 0;
  m integer;
  n integer;
  n1 integer;
  n2 integer;
  j integer;

  BEGIN

  ---------PVOM 1---------------
    LOOP
      exit when k = i + 1;
      a1 := 1 + k;
      a2 := 2 + k;
      a3 := 3 + k;
      c := c||'A'||a3||'.GI_'||a1||' = A'||a2||'.GI_'||a1||' AND ';
      c := c||'A'||a2||'.GI_'||a2||' = A'||a1||'.GI_'||a2||' AND ';
      IF k = i THEN
         c := c||'A'||a3||'.GI_'||a2||' = A'||a1||'.GI_'||a1;
       ELSE
         c := c||'A'||a3||'.GI_'||a2||' = A'||a1||'.GI_'||a1||' AND ';
      END IF;
      k := k + 1;
    END LOOP;

  ---------PVOM 2---------------
    m := i;
    LOOP
      exit when m = 1;
      j := m;
      LOOP
        exit when j = 1;
        n := m + 2;
        n1 := m + 1;
        n2 := j - 1;
        c := c||' AND A'||n||'.GI_'||n2||' = A'||n1||'.GI_'||n2;
        j := j - 1;
      END LOOP;
      m := m -1;
    END LOOP;

    ---------PVOM 3---------------
    m := i;
    LOOP
      exit when m = 1;
      j := m;
      LOOP
        exit when j = 1;
        n := j ;
        n1 := m + 1 ;
        n2 := j - 1;
        c := c||' AND A'||n||'.GI_'||n1||' = A'||n2||'.GI_'||n1;
        j := j - 1;
      END LOOP;
      m := m -1;
    END LOOP;

   RETURN (c);

END where_stmt_gom;
/************************************************************/
```

```
/*****************************************************************
*     Função de ordenação de proteomas e fechamento da consulta  *
*****************************************************************/
CREATE OR REPLACE FUNCTION final_stmt(i integer) return clob is

  ---------Declaração de variáveis---------------
  c clob := '';
  k integer := 1;
  j integer := 2;
  m integer := 1;

  BEGIN

    LOOP
       exit when k = i + 2;
       c:= c||' AND A'||m||'.ID_PRTM_'||k||' > A'||m||'.ID_PRTM_'||j;
       j := j + 1;
       k := k + 1;
    END LOOP;

    IF k = i + 2 THEN
       k := k + 1;
       c := c||' AND A'||k||'.ID_PRTM_'||m||' > A'||m||'.ID_PRTM_'||m;
    END IF;

    RETURN (c);

END final_stmt;
/*****************************************************************/
```

APÊNDICE B – PL/SQL DO PVOM PARA ROMS

```
/*****************************************************************
*          Procedimento de chamada do PVOM para ROMs             *
*****************************************************************/
CREATE OR REPLACE PROCEDURE find_rom AS

  --A tabela prtm_cons contem os proteomas selecionados pelo usuário
  CURSOR c_tot IS Select count(*) as total From prtm_cons;

  ---------Declaração de variáveis---------------
  rec_tot c_tot%ROWTYPE;
  i integer := 1;
  j Number;
  nr_prtms NUMBER := 0;
  nr_exec NUMBER := 0;
  c varchar(2000);

  BEGIN

    -- primeira execução através de consulta sobre a tabela de RO
    j := cria_res_tmp_rom;

    -- calcula a quantidade de níveis a serem gerados
    IF j = 0 THEN
       dbms_output.put_line('Nao ha ROMs para estes proteomas');
    ELSE
       open c_tot;
       fetch c_tot into rec_tot;
       nr_prtms := rec_tot.total;
       dbms_output.put_line(nr_prtms);
       nr_exec := nr_prtms - 3;
       dbms_output.put_line(nr_exec);
       close c_tot;

       -- gera consulta dinâmica a cada nível
       LOOP
         exit when i = nr_exec + 1;
         c := gera_query_rom(i);
         EXECUTE IMMEDIATE c;
         i := i + 1;
       END LOOP;
    END IF;
  END find_rom;
/*****************************************************************/


/*****************************************************************
*     Função de criação da primeira tabela temporária - Nível 1  *
*****************************************************************/
CREATE OR REPLACE FUNCTION cria_res_tmp_rom return integer IS

  ---------Declaração de variáveis---------------
  CURSOR c_roms IS Select count(*) as roms From rom_res_1;
  rec_roms c_roms%ROWTYPE;
  r integer := 1;
  nr_roms integer;

  BEGIN

    --------Apaga a tabela de nível 1-------------
    EXECUTE IMMEDIATE 'DROP TABLE SYSTEM.ROM_RES_1';
    ----Cria a tabela de nível 1 para os organismos selecionado
    EXECUTE IMMEDIATE
    'CREATE TABLE SYSTEM.ROM_RES_1 AS
    SELECT DISTINCT
       A.ID_RGC_1 as ID_RGC_1,
       B.ID_RGC_2 as ID_RGC_2,
```

```
       C.ID_RGC_2 as ID_RGC_3,
       A.ID_PRTM_1 as ID_PRTM_1,
       B.ID_PRTM_2 as ID_PRTM_2,
       C.ID_PRTM_2 as ID_PRTM_3
    FROM    SYSTEM.RO_TMP A,
            SYSTEM.RO_TMP B,
            SYSTEM.RO_TMP C
    WHERE   A.ID_RGC_1 = B.ID_RGC_1 AND
            B.ID_RGC_2 = C.ID_RGC_1 AND
            A.ID_RGC_2 = C.ID_RGC_2 AND
            A.ID_PRTM_1 in (select id_prtm from prtm_cons) AND
            B.ID_PRTM_1 in (select id_prtm from prtm_cons) AND
            C.ID_PRTM_1 in (select id_prtm from prtm_cons) AND
            A.ID_PRTM_2 in (select id_prtm from prtm_cons) AND
            B.ID_PRTM_2 in (select id_prtm from prtm_cons) AND
            C.ID_PRTM_2 in (select id_prtm from prtm_cons) AND
            A.ID_PRTM_1 > B.ID_PRTM_2 AND
            B.ID_PRTM_2 > C.ID_PRTM_2
    ORDER BY
       A.ID_PRTM_1,
       B.ID_PRTM_2,
       C.ID_PRTM_2,
       A.ID_RGC_1,
       B.ID_RGC_2,
       C.ID_RGC_2';

    open c_roms;
    fetch c_roms into rec_roms;
    nr_roms := rec_roms.roms;
    close c_roms;

    IF nr_roms = 0 THEN
      RETURN (nr_roms);
    ELSE
      RETURN (r);
    END IF;

 END cria_res_tmp_rom;
/*****************************************************************/



/*****************************************************************
*Função de criação dinâmica das tabelas e de criação das consultas*
*****************************************************************/
CREATE OR REPLACE FUNCTION gera_query_rom(i integer) return CLOB AS

  ---------Declaração de variáveis---------------
  c clob;
  j integer;

  BEGIN
    j := i + 1;

    --------Apaga a tabela de nível j--------------
    EXECUTE IMMEDIATE 'DROP TABLE SYSTEM.ROM_RES_'||j;

    --Geração da consulta dinâmica para criação da tabela de nível j
    c := 'CREATE TABLE SYSTEM.ROM_RES_'||j||' AS ';

    --chamada da função de criação do select para RGCs
    c := c||sel_stmt_rgc(i);

    --chamada da função de criação do select para PROTEOMAS
    c := c||sel_stmt_prtm(i);

    --chamada da função de criação da cláusula FROM para as tabelas
temporárias
    c := c||from_stmt_rom(i);
```

```
    --chamada da função de criação da cláusula WHERE para ROMs
    c := c||where_stmt_rom(i);

    --chamada da função de ordenação de proteomas e fechamento da consulta
    c := c||final_stmt(i);

    --o retorno da função é um Character Large Object contendo a consulta
gerada dinamicamente para o nível i
    RETURN (c);

END gera_query_rom;
/*****************************************************************/



/*****************************************************************
*     SEL_STMT_RGC - renomeia os ID_RGCs para execução do PVOM   *
*****************************************************************/
CREATE OR REPLACE FUNCTION sel_stmt_rgc(i integer) return clob is

  ---------Declaração de variáveis---------------
  c clob := ' SELECT DISTINCT ';
  k integer:= 1;
  j integer;

  BEGIN
    j := i + 3;
    LOOP
      exit when k = i + 3;
      c := c||'A'||j||'.ID_RGC_'||k||' as ID_RGC_'||k||', ';
      k := k + 1 ;
    END LOOP;

    j := k - 1;
    c := c||'A1'||'.ID_RGC_'||j||' as ID_RGC_'||k||', ';

    RETURN (c);

  END sel_stmt_rgc;
/*****************************************************************/



/*****************************************************************
*     SEL_STMT_PRTM - renomeia os ID_PRTMs para execução do PVOM *
*****************************************************************/
CREATE OR REPLACE FUNCTION sel_stmt_prtm(i integer) return clob is

  ---------Declaração de variáveis---------------
  c clob;
  k integer:= 1;
  j integer;

  BEGIN

    j := i + 3;
    LOOP
      exit when k = i + 3;
      c := c||'A'||j||'.ID_PRTM_'||k||' as ID_PRTM_'||k||', ';
      k := k + 1 ;
    END LOOP;

    j := k - 1;
    c := c||'A1'||'.ID_PRTM_'||j||' as ID_PRTM_'||k;

    RETURN (c);

  END sel_stmt_prtm;
/*****************************************************************/
```

```
/*****************************************************************
*FROM_STMT_ROM - criação da cláusula FROM baseada no nível anterior*
*****************************************************************/
CREATE OR REPLACE FUNCTION from_stmt_rom(i integer) return clob is

  ---------Declaração de variáveis---------------
  c clob := ' FROM ';
  k integer := 1;
  m integer := 1;
  j integer;

  BEGIN
    j := i;
    LOOP
      exit when m = i + 3;
      c := c||'SYSTEM.ROM_RES_'||j||' A'||m||', ';
      m := m + 1;
    END LOOP;

    IF m = i + 3 THEN
       c := c||'SYSTEM.ROM_RES_'||j||' A'||m;
    END IF;

    RETURN (C);

  END from_stmt_rom;
/*****************************************************************/




/*****************************************************************
*WHERE_STMT_ROM - Função onde os PVOMs 1, 2 e 3 são gerados para ROM *
*****************************************************************/
CREATE OR REPLACE FUNCTION where_stmt_rom(i integer) return clob is

  ---------Declaração de variáveis---------------
  c clob := ' WHERE ';
  a1 integer;
  a2 integer;
  a3 integer;
  k integer := 0;
  m integer;
  n integer;
  n1 integer;
  n2 integer;
  j integer;

  BEGIN

    ---------PVOM 1---------------
    LOOP
      exit when k = i + 1;
      a1 := 1 + k;
      a2 := 2 + k;
      a3 := 3 + k;
      c := c||'A'||a3||'.ID_RGC_'||a1||' = A'||a2||'.ID_RGC_'||a1||' AND ';
      c := c||'A'||a2||'.ID_RGC_'||a2||' = A'||a1||'.ID_RGC_'||a2||' AND ';

      IF k = i THEN
        c := c||'A'||a3||'.ID_RGC_'||a2||' = A'||a1||'.ID_RGC_'||a1;
      ELSE
        c := c||'A'||a3||'.ID_RGC_'||a2||' = A'||a1||'.ID_RGC_'||a1||' AND
      ';
      END IF;
      k := k + 1;
```

```
    END LOOP;

    ---------PVOM 2---------------
    m := i;
    LOOP
      exit when m = 1;
      j := m;
      LOOP
        exit when j = 1;
        n := m + 2;
        n1 := m + 1;
        n2 := j - 1;
        c := c||' AND A'||n||'.ID_RGC_'||n2||' = A'||n1||'.ID_RGC_'||n2;
        j := j - 1;
      END LOOP;
      m := m -1;
    END LOOP;

    ---------PVOM 3---------------
    m := i;
    LOOP
      exit when m = 1;
      j := m;
      LOOP
        exit when j = 1;
        n := j ;
        n1 := m + 1 ;
        n2 := j - 1;
        c := c||' AND A'||n||'.ID_RGC_'||n1||' = A'||n2||'.ID_RGC_'||n1;
        j := j - 1;
      END LOOP;
      m := m -1;
    END LOOP;

RETURN (C);

END where_stmt_rom;
/*****************************************************************/


/*****************************************************************
*     Função de ordenação de proteomas e fechamento da consulta  *
*****************************************************************/
CREATE OR REPLACE FUNCTION final_stmt(i integer) return clob is

  ---------Declaração de variáveis---------------
  c clob := '';
  k integer := 1;
  j integer := 2;
  m integer := 1;

  BEGIN

    LOOP
       exit when k = i + 2;
       c:= c||' AND A'||m||'.ID_PRTM_'||k||' > A'||m||'.ID_PRTM_'||j;
       j := j + 1;
       k := k + 1;
    END LOOP;

    IF k = i + 2 THEN
       k := k + 1;
       c := c||' AND A'||k||'.ID_PRTM_'||m||' > A'||m||'.ID_PRTM_'||m;
    END IF;

    RETURN (c);

END final_stmt;
/*****************************************************************/
```