Ancoragem de genomas incompletos em genomas completos

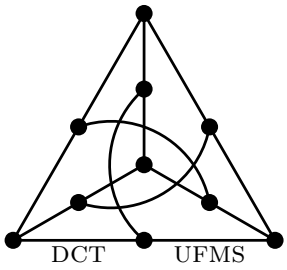
André Chastel Lima

Dissertação de Mestrado

Orientação: Prof. Dr. Nalvo Franco de Almeida Junior

Área de Concentração: Biologia Computacional

Dissertação apresentada como requisito para a obtenção do título de mestre em Ciência da Computação.



Departamento de Computação e Estatística
Centro de Ciências Exatas e Tecnologia
Universidade Federal de Mato Grosso do Sul
3 de agosto de 2007

Resumo

Um projeto genoma usual possui três etapas. A primeira, denominada seqüenciamento e montagem, consiste na determinação da seqüência exata de todos os seus cromossomos. A segunda, conhecida como anotação, consiste na descoberta da posição exata dos genes de cada cromossomo, quais as proteínas produzidas por eles e qual a função de cada uma delas. Ao final, a análise do genoma consiste na descoberta de funcionalidades específicas ou comuns a outros organismos. Algumas vezes não se deseja terminar completamente a primeira fase. Nesses casos tem-se, ao invés de cromossomos, apenas trechos contíguos e disjuntos dos cromossomos, chamados de *contigs*. Mesmo assim, ainda é possível chegar a conclusões importantes acerca das funcionalidades do organismo estudado. Este trabalho consiste no estudo de algumas ferramentas de comparação de seqüências, visando usá-las como instrumento para mapear os *contigs* de um genoma incompleto no genoma completo de um organismo evolutivamente próximo. Dessa forma, é possível auxiliar projetos cujo objetivo seja apenas entender mecanismos biológicos importantes do organismo, sem que haja a necessidade de completar seu genoma.

Abstract

An usual genome project has three steps. The first one, called sequencing and assembling, consists in determining the exact sequence of each chromosome. The second one, called annotation, consists in finding out the exact position of all chromosome genes, the proteins they produce and what the function of these proteins. Finally, the third step aims the analysis of the genome by determining its specific and common functionalities. Sometimes the first step is not completed. In these cases, instead of a set of chromosomes, one has only a set of contiguous and disjoint pieces, called *contigs*. Nevertheless, it is still possible draw conclusions about the organism functionalities. This work consists of a study of some sequence comparison tools, aiming using them as a mapping tool of *contigs* of an incomplete genome onto a close related complete genome. In this way it is possible to help projects in which the goal is just understanding important biological features without finishing the whole genome.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Nos últimos anos vivenciamos grandes avanços na biotecnologia e muitos desses resultam em temas que constantemente são destaques, e fontes de muitas discussões, em toda a sociedade. A biologia molecular é uma das áreas que muito tem contribuído para esses avanços, particularmente o desenvolvimento de técnicas de seqüenciamento de moléculas de DNA que são empregadas nos chamados projeto genoma. A evolução dessas técnicas propiciaram a geração de uma enorme quantidade de dados de genomas, grande parte desses estão disponibilizados em bancos de dados públicos. A necessidade de transformar todos esses dados em informações biológicas relevantes deu origem a uma nova área de conhecimento chamada de Bioinformática ou Biologia Molecular Computacional, que envolve a Matemática, a Ciência da Computação, a Biologia Molecular e a Estatística.

A Bioinformática tem como principal objetivo aplicar, ou criar, métodos computacionais e matemáticos para solucionar problemas da biologia molecular; e assim desenvolver ferramentas para extrair informações biológicas dos organismos, armazenar e disponibilizar os dados gerados pelos projetos.

Muitos projetos genomas estão em andamento. Atualmente (julho de 2007) cerca de 564 já estão completos e 1018 genomas ainda estão incompletos [3]. Muitos dos incompletos estão parados com centenas ou milhares de contigs já seqüenciados e por motivos diversos não terminarão o seqüenciamento do genoma todo. Entende-se por genoma completo a determinação exata das bases de todos os replicons do organismo estudado - fase de seqüenciamento e montagem; a determinação exata da posição de cada gene do genoma, assim como a identificação da função de cada um deles (quais proteínas ele expressa) - fase de anotação do genoma. Após o término dessas duas fases, o objetivo

passa a ser o de fazer a análise do genoma, em particular no que diz respeito às suas funcionalidades específicas e também comuns a outros organismos, além do estudo de suas interações com o ambiente em que vive.

Muitos dos projetos em andamento visam a determinação das funcionalidades e características biológicas de organismos filogeneticamente próximos a organismos que já tiveram seus genomas completados. Espera-se desses novos genomas muitas similaridades com os anteriores, tanto no nível do DNA que eles contêm, quanto no nível das proteínas por eles produzidas. Assim, é de extrema utilidade a comparação de genomas, mesmo que essa comparação envolva genomas completos e incompletos. Nosso objetivo é identificar ou desenvolver ferramentas computacionais úteis na comparação de genomas completos e incompletos. Em particular, estamos interessados em investigar aspectos estruturais comuns, capazes de identificar fatores funcionais importantes dos organismos, mesmo sem finalizar os projetos dos genomas ainda incompletos.

Nosso trabalho consiste em utilizar métodos e ferramentas capazes de fazer a ancoragem, processo que determina uma provável ordenação de pedaços de um genoma incompleto, os chamados *contigs*, de um organismo em relação ao genoma completo de um outro organismo, desde que sejam organismos evolutivamente próximos, e identificar os métodos que trouxeram os melhores resultados. Para isso, criamos conjuntos de contigs, a partir de genomas completos e fizemos a ancoragem desses contigs nos genomas de origem, partindo da premissa de que uma ferramenta de ancoragem deve saber ancorar corretamente os contigs de um genoma no próprio genoma. Para tanto, usamos as ferramentas: PROmer [11], NUCmer [11], Mega BLAST [26] e Egg [4], que já foram empregadas em diversos projetos genomas com muito sucesso. Para comparar a qualidade das ferramentas, definimos uma medida de qualidade para as ancoragens. Nossos resultados mostram que PROmer teve um desempenho superior a 95% de acertos, e mostraram também que Egg teve um excelente desempenho quando temos poucos e longos contigs. Este estudo, que inclui ainda o desenvolvimento de uma ferrarmenta de visualização gráfica das ancoragens e sua disponibilização na web, resultou nas publicações [5] e [6].

Nosso trabalho está organizado da seguinte maneira. No capítulo 2 apresentamos tópicos sobre a comparação de genomas, começando pelos conceitos básicos de biologia molecular, importantes técnicas computacionais para a comparação de seqüências e terminamos com a descrição dos métodos computacionais e estrutura de dados empregados pelas ferramentas aqui estudadas. No capítulo 3 apresentamos uma definição formal de ancoragem, como utilizamos as ferramentas nesse processo e explicamos como foi feita a adaptação necessária à ferramenta Egg para que pudesse fazer a ancoragem. A definição

da medida de comparação, a construção dos casos de testes e um estudo comparativo dos resultados obtidos estão no capítulo 4. No capítulo 5 descrevemos um visualizador gráfico para as ancoragens usando o GBrowse [23]. Por fim nossas conclusões são apresentadas no capítulo 6.

Capítulo 2

Comparação de Genomas

Neste capítulo descrevemos os conceitos básicos de biologia molecular e tópicos sobre a principal abordagem de comparação de genomas que é a comparação de seqüências. Apresentamos também as definições de alinhamento e similaridade, que são noções muito importantes e mostramos uma técnica computacional para calcular o alinhamento e a similaridade entre as seqüências. Descrevemos ainda uma ferramenta de busca de uma seqüência em um banco de seqüências e uma estrutura de dados muito útil quando as seqüências comparadas são muito grandes. Os conceitos e definições desse capítulo foram retirados de [4, 18, 19]. Descrevemos ainda as ferramentas utilizadas no trabalho.

2.1 Conceitos e Definições

Os conceitos de biologia molecular que precisamos ter para melhor entendimento deste trabalho são: *gene*, *cromossomo*, *plasmídeos*, *genoma* e *contig*. Os genes são responsáveis por carregar características genéticas de um ancestral para seus descendentes. Um **gene** é um pequeno trecho contínuo de uma molécula de DNA capaz de codificar proteínas ou RNA. As moléculas de DNA formam os cromossomos e os plasmídeos, que são as estruturas responsáveis pela replicação das células. Os **cromossomos** são grandes moléculas de DNA cujo tamanho varia entre centenas de milhares e milhões de nucleotídeos, e são essenciais à replicação dos organismos. Já os **plasmídeos** são moléculas bem menores que os cromossoms e não são essenciais à replicação dos organismos, tanto que estão presente somente nas bactérias e em alguns eucariotos. O número de cromossomos e plasmídeos em uma célula é característico de cada espécie. Plasmídeos e cromossomos são chamados de **replicons**. O **genoma**

de um organismo é o conjunto de todos os cromossomos e plasmídeos dentro de uma célula.

Um outro conceito que tem importância relevante nesse trabalho é o de *contig*, por isso o apresentamos com mais detalhes. Os contigs surgem ainda na fase de montagem de fragmentos dos projetos genomas. Máquinas seqüenciadoras são utilizadas para determinar a seqüência de nucleotídeos que formam uma molécula de DNA, porém essas máquinas não conseguem ler a molécula inteira de uma só vez, conseguem ler em torno de 700 a 800 nucleotídeos por vez. Para determinar a seqüência completa da molécula, várias cópias da mesma são fragmentadas aleatoriamente e seus fragmentos são submetidos, todos juntos, aos seqüenciadores. As seqüências desses fragmentos são chamados de **reads**. O objetivo da fase de montagem é determinar a ordem dos reads com base na sobreposição que existe entre eles, para assim obter a seqüência inteira da molécula original. Quando uma amostragem de fragmentos é produzida, pode acontecer que determinadas regiões da molécula não estejam representadas por um desses fragmentos, essas regiões são ditas não possuirem cobertura. Nos casos em que existam regiões sem cobertura, a reconstrução da seqüência completa fica comprometida, originando então várias regiões contíguas disjuntas, chamadas **contigs**.

Alguns erros na fase de montagem podem comprometer as seqüências dos contigs e do genoma como um todo. Os principais erros são o *base-call*, *contaminação das seqüências*, *seqüências quiméricas* e *repetições*. O erro chamado de ***base-call*** ocorre no processo de leitura dos nucleotídeos feita pelos seqüenciadores, ou seja, ao tentar ler as bases de uma seqüência de DNA o aparelho determina erroneamente uma base. A principal conseqüência desses erros é que as seqüências passam a ter divergências entre suas bases, e com com isso, sobreposições aproximadas entre as seqüências devem ser consideradas. Já o caso de **seqüências contaminadas** ocorre quando o fragmento, além do DNA alvo, trás parte do DNA do vetor usado para copiar a molécula. Por isso, antes de inserir essas seqüências no conjunto de entrada, elas são submetidas ao processo de *screening*, que tenta retirar a contaminação. Seqüências contaminadas podem influenciar de forma negativa a determinação da sobreposição das seqüências.

As seqüências **quimeras** ou **quiméricas** são dois fragmentos de partes distintas do DNA alvo que se unem pelas extremidades formando uma única seqüência que não existe no DNA original. Esses erros devem ser detectados antes e retirados do conjunto de entrada para não formarem seqüências erradas. **Repetições** são trechos que aparecem duas ou mais vezes em uma molécula de DNA, e independentemente do tamanho desses trechos, atrapal-

ham o processo. Pois quando os trechos são grandes, de modo que não haja um fragmento que o comporte, seqüências com ambigüidade de arranjos podem ser geradas. E quando são muito pequenos e ocorrem seguidamente, repetições podem ser montadas em um único trecho, gerando uma seqüência diferente da real.

A comparação de genomas tem como principal abordagem a comparação de seqüências, que de maneira simplória, podemos dizer que tem por objetivo encontrar regiões distintas e regiões comuns nas seqüênciass. Dentro dessa aparente simplicidade estão inseridos diversos problemas cujas soluções eficientes requerem algoritmos e estrutura de dados distintas. Entre esses problemas podemos citar a comparação global e a comparação local entre duas seqüências, a comparação entre múltiplas seqüências, a procura por uma seqüência em um banco com milhares de seqüências e a comparação entre seqüências extremamente grandes. Esses exemplos usam noções de similaridade e alinhamento para determinar suas soluções. Similaridade de duas seqüências determina uma medida de o quanto similares as seqüências são. Alinhamento é uma maneira de colocar uma seqüência sobre a outra, ou outras no caso de comparação entre múltiplas seqüências, tornando claro a correspondência entre caracteres ou subseqüências similares nas seqüências. A seguir esses conceitos são definidos formalmente.

Alinhamento entre duas seqüências é definido como a inserção de espaços em locais arbitrários ao longo das seqüências tal que elas terminem com o mesmo tamanho, criando uma correspondência entre caracteres ou espaços na primeira seqüência e caracteres ou espaços na segunda seqüência, evitando a correspondência entre espaço e espaço. Dado um alinhamento, nós podemos determinar um *score* para ele da seguinte maneira. Para cada coluna do alinhamento atribuímos uma pontuação dependendo do seu conteúdo. Se existirem dois caracteres idênticos a pontuação atribuída é 1, se forem dois caracteres distintos a pontuação é -1 e no caso de existir um caracter e um espaço a pontuação é -2. O *score* do alinhamento é a soma das pontuações das colunas. O melhor alinhamento é aquele com o *score* máximo. Este *score* máximo é chamado de **similaridade** entre duas seqüências, denotado por $sim(s, t)$. Para as seqüências s e t, podem existir muitos alinhamentos com o *score* máximo. Outros esquemas de pontuação podem ser usados para cada par de símbolos e para os espaços.

A principal técnica computacional empregada em algoritmos que determinam alinhamentos locais ou globais e as respectivas similaridades entre seqüências é a **programação dinâmica**. Esta técnica consiste, basicamente, em solucionar uma instância de um problema aproveitando as soluções já existentes

de instâncias menores do mesmo problema. No caso de comparação de duas seqüências, digamos s e t, ao invés de calcular a similaridade entre elas como seqüências inteiras, construímos a solução calculando a similaridade entre prefixos das duas seqüências. Começando com os prefixos menores e usando os resultados computados previamente para solucão dos prefixos maiores. Vamos exemplificar a aplicação da técnica descrevendo um algoritmo que computa a similaridade e o alinhamento global entre duas seqüências.

Dadas duas seqüências s e t, de tamanhos m e n respectivamente, definimos uma matriz A, $m \times n$, onde cada entrada A_{ij} recebe o valor da similaridade entre os prefixos $s_1...s_i$ e $t_1...t_j$. Assim, a similaridade entre s e t é dado por A_{mn} ($sim(s,t) = A_{mn}$), e para alcançarmos esse valor, através da programação dinâmica, temos que calcular todos os valores A_{ij} para $0 \leq i \leq m$ e $0 \leq j \leq n$. A inicialização da matriz para solucionar nosso exemplo é dada abaixo, onde a função σ determina o esquema de pontuação das correspondências entre os caracteres ou espaços ($-$) nas colunas.

$$\begin{aligned} A_{0,0} &= 0, \\ A_{i,0} &= \textstyle\sum_{k=1}^{i} \sigma(s_k, -)e \\ A_{0,j} &= \textstyle\sum_{k=1}^{j} \sigma(-, t_k). \end{aligned}$$

Para computar as demais entradas da matriz a fórmula de recorrência é dada por:

$$A_{i,j} = max \begin{cases} A_{i-1,j} + \sigma(s_i, -) \\ A_{i,j} + \sigma(s_i, t_j) \\ A_{i,j-1} + \sigma(-, t_j) \end{cases}$$

Com isso determinamos a similaridade entre s e t com uma complexidade $O(mn)$. Para determinar um alinhamento, basta armazenar, para cada (i, j), qual dentre as entradas $A_{i-1,j}$, $A_{i,j}$ e $A_{i,j-1}$ foi usada. Isso é feito com uma complexidade $O(m + n)$.

A programação dinâmica é uma excelente ferramenta para fazer comparações aproximadas entre seqüências, ou seja, a computação de alinhamentos globais, locais ou semi-globais entre duas seqüências, porém se as seqüências forem muito grandes, problemas de espaço e desempenho impedem seu uso.

Outro problema para o qual a programação dinâmica não tem bom desempenho é a pesquisa, em um banco contendo milhares de seqüências, por seqüências com um certo grau de similaridade a uma dada seqüência de consulta. Para solucionar esse problema uma heurística muito eficiente é usada na ferramenta BLAST, de *Basic Local Alignment Search Tool*, desenvolvida inicialmente por Altschul [7]. Descrevemos agora resumidamente essa ferramenta.

A seqüência consultada é chamada de seqüência **query** enquanto as seqüências retornadas são chamadas de **hits**. BLAST faz a procura e retorna uma lista de *hits* acompanhados por uma estimativa de significância estatística, chamada de **e-value**, o alinhamento entre as seqüências *query* e os *hit*, e um valor para esse alinhamento, chamado de *score*. O e-value basicamente indica o número de *hits* ao acaso esperado, ou seja, quanto menor o valor de e-value menor é a probabilidade de um *hit* ter sido encontrado ao acaso. O procedimento do BLAST começa pela procura de pequenas subcadeias da seqüência *query* cujo alinhamento com subcadeias do mesmo tamanho das seqüências do banco sejam no mínimo igual a um primeiro valor de corte. As subcadeias da seqüência *query* e das seqüências do banco que atenderem a esse requisito têm seus alinhamentos estendidos em ambas extremidades, na tentativa de se obter segmentos maiores cuja pontuação seja no mínimo igual a um segundo valor de corte. Esses segmentos são utilizados na construção do alinhamento.

Para o problema de comparação exata de seqüências, uma estrutura de dados bastante usada é a *árvore de sufixos* que armazena os sufixos de uma seqüência.

> **Definição** Seja $s = s_1 s_2 ... s_m$ uma seqüência com m símbolos de um alfabeto Σ, onde $s_m = \$$ é um símbolo especial que não ocorre em nenhuma outra posição de s[1]. A **árvore de sufixos** T_s para a seqüência s é uma árvore enraizada, com m folhas e no máximo $m - 1$ nós internos, tal que:
>
> 1. as arestas de T_s são orientadas no sentido raiz $\rightarrow$ folha; e são rotuladas com subcadeias de s;
> 2. cada nó interno tem pelo menos dois filhos;
> 3. quaisquer duas arestas que saem de um mesmo nó estão rotuladas com subcadeias de prefixos distintos;
> 4. cada folha está rotulada com um inteiro i, $1 \leq i \leq m$; e

[1] O símbolo `$` é necessário para garantir que nenhum sufixo seja prefixo de outro sufixo. Isso, por sua vez, garante a existência e unicidade da árvore.

5. a concatenação dos rótulos das arestas de um caminho da raiz até uma folha i corresponde ao sufixo de s que começa na posição i.

A aplicação típica de árvores de sufixo é a busca de todas as ocorrências de uma seqüência p de tamnho m em um texto s de tamanho n, dada a árvore de sufixos T_s. A idéia parte do princípio de que qualquer sufixo $s[i \ldots m]$ de s deve estar representado na árvore, através de um caminho único da raiz de T_s até a folha rotulada com i. Por outro lado, qualquer subcadeia $s[i \ldots j]$ de s, $i \leq j \leq m$, é um prefixo do sufixo $s[i \ldots m]$ e, portanto, deve rotular a parte inicial desse caminho único. Assim, para descobrir se há ocorrências de p em s, deve-se percorrer esse caminho, comparando os símbolos de p com os símbolos que rotulam as arestas do caminho, até que p seja integralmente encontrado ou nenhum casamento seja mais possível. No primeiro caso, todas as folhas na subárvore abaixo do nó onde a busca terminou rotulam as posições onde p ocorre em s. No segundo caso, p não ocorre em s.

O caminho a ser percorrido na busca é único porque quaisquer duas arestas que saem de um vértice interno de T_s são rotuladas com subcadeias de s que obrigatoriamente começam com símbolos diferentes, pela própria definição da árvore de sufixos.

Na figura 2.1 podemos ver que a busca por $p = xa$ terminará em um vértice interno, cuja subárvore tem as folhas rotuladas com 1 e 4, significando, portanto, que p ocorre nessas posições de s. Se fizermos $p = bxa\$$ e efeturamos outra busca na árvore da Figura 2.1 terminaremos na folha rotulada com 3, indicando a posição de s onde ocorre o sufixo. O último caso na busca ocorre se fizermos $p = xab$, nenhum vértice é alcançado o que indica que p não é um sufixo de s.

2.2 Ferramentas de Comparação

As ferramentas que escolhemos para trabalhar são NUCmer e PROmer [2, 10, 11, 14], Mega BLAST [26] e Egg [4]. São ferramentas para comparação de seqüências e que já têm seu mérito reconhecido pela comunidade científica. Essas ferramentas usam técnicas distintas e nos permitem uma aplicação direta ao problema de ancoragem, exceto Egg, para a qual tivemos que desenvolver uma pequena adaptação. Mega BLAST e Egg utilizam BLAST em seu núcleo, porém Egg usa uma metodologia de procura por estruturas organizacionais e Mega BLAST usa um algoritmo guloso para alinhamento

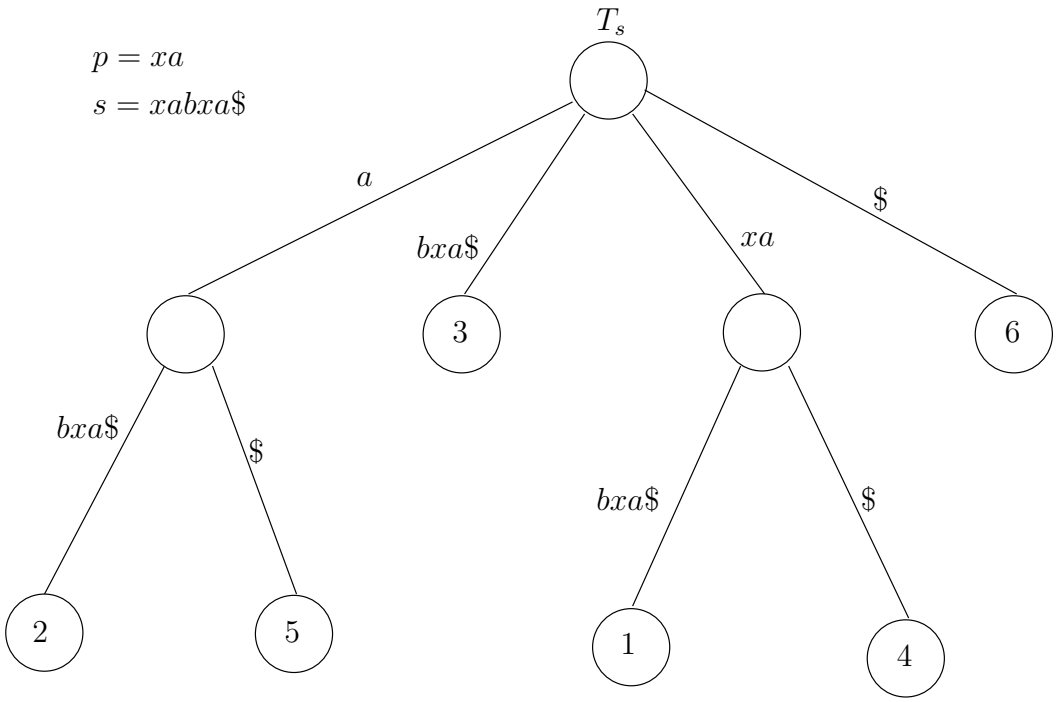


Figura 2.1: Árvore de Sufixos T_s para a seqüência $xabxa\$$

de seqüências. NUCmer e Promer usam em seu núcleo a ferramenta MUMmer, que implementa uma árvore de sufixos muito eficiente na determinação de trechos idênticos das seqüências, porém NUCmer utiliza a seqüência de nucleotídeos enquanto PROmer traduz a seqüência de nucleotideos nas seis seqüências de aminoácidos possíveis. Passamos agora a descrever cada uma dessas ferramentas.

2.2.1 MUMmer

MUMmer é um sistema capaz de fazer o alinhamento entre duas seqüências, uma chamada de seqüência referência e outra chamada seqüência *query*. As seqüências podem ser de genomas inteiros e/ou diferentes montagens de um genoma. O nome MUMmer teve origem na primeira versão, quando o sistema construía o alinhamento a partir de *matches únicos maximais* (MUM). Os **MUM's** são subseqüências idênticas nas duas seqüências que não estão contidos em subseqüências idênticas de comprimento maior e que acontecem uma única vez em cada seqüência. As versões seguintes da ferramenta constroem o alinhamento a partir de outras estruturas não tão rígidas como o MUM.

A segunda versão usa candidatos a matches únicos maximais (MUM's candidatos). Essa estrutura não exige que a subseqüência seja única na seqüência *query* mas, que seja única na seqüência referência. Na terceira versão a restrição de unicidade também foi retirada da seqüência referência e então o alinhamento é construído sobre todos os *matches* maximais. A ferramenta permite fazer o alinhamento sobre qualquer uma dessas três estruturas, sendo os MUM's candidatos a estrutura padrão.

MUMmer tem como principais métodos computacionais na construção do alinhamento uma árvore de sufixos, um algoritmo modificado para solução do problema computacional conhecido como subseqüência crescente mais longa ou LIS (*Longest Increasing Subsequence*) e um algoritmo de programação dinâmica baseado no algoritmo de alinhamento de Smith-Waterman [22]. Cada um desses métodos é aplicado em uma das três fases que compõem o sistema.

A primeira fase consiste em encontrar os *matches*, que são as sementes do alinhamento. Para executar tal tarefa é empregada a árvore de sufixo. Primeiramente, todos os sufixos da seqüência referência são inseridos na árvore, depois os sufixos da seqüência *query* são procurados na árvore e assim são identificados os MUM's candidatos ou os *matches* maximais. Para identificar os MUM's, da primeira versão, os sufixos da seqüência *query* também são inseridos na árvore.

A segunda fase de MUMmer é responsável por determinar a seqüência de *matches* que formam a base do melhor alinhamento entre os genomas. Para isto, a ferramenta primeiro ordena os *matches* por sua posição no genoma de referência, depois faz a ordenação pelas posições no genoma *query*. Cada *match* passa a ter dois inteiros como atributos, um representando sua ordem na referência e outro representando a ordenação na *query*. Então executa o algoritmo de LIS na ordenação dada pela *query* para determinar a mais longa subseqüência de *matches* em ordem crescente e assim determinar a base do alinhamento global. MUMmer percorre os *matches* da esquerda para a direita considerando, além da ordenação, o tamanho e o fato de que pode haver sobreposições entre os *matches* para determinar a melhor subseqüência.

Na terceira fase, onde a base do alinhamento global já está definida, muitos programas são usados para fechar os gaps locais. Um **gap** é definido como uma interrupção no alinhamento global e é classificado em **SNP**(*Single Nucleotide Polymorphisms*), diferenças de um único nucleotídeo, inserção, região altamente polimórfica e repetições. Para cada uma dessas classificações de inserção são aplicados um ou mais programas para a execução do alinhamento local. No processamento de SNP's são duas as formas de identificação no MUMmer, a mais simples acontece quando a base é rodeada por *matches* e nesse caso o sistema identifica um SNP. A outra forma é quando o SNP é ad-

jacente a seqüências que não são únicas nos genomas. Nesta situação o SNP é identificado pelo processo que trata de repetições.

Uma inserção, no contexto de MUMmer, é uma região de tamanho moderado que aparece em um genoma e não aparece no outro, onde o outro nesse momento é a base do alinhamento global. MUMmer ainda divide as inserções em dois grupos, transposições e inserções simples. Transposições são subseqüências que foram retiradas de uma posição e colocadas em outra (relativamente ao outro genoma), e são detectadas num passo de pós-processamento pois aparecem como *matches* mas fora de ordem. Inserções simples são subseqüências que aparecem em apenas um dos genomas.

Regiões Polimórficas são regiões com um grande número de diferenças, por terem muitas diferenças é menos significante definí-las em forma de SNP's. Para essas regiões é executado um algoritmo de programação dinâmica equivalente a Smith-Waterman, ou se a região for grande existe a possibilidade de rodar o MUMmer recursivamente diminuindo o tamanho mínimo do MUM.

Para efetuar o processamento de repetições foram criados dois programas *repeat-match* e *exact-tandems*. *repeat-match* é um algoritmo que usa árvore de sufixos e está projetado para encontrar repetições exatas maximais em uma única seqüência. Usa um algoritmo similar a MUMmer. *exact-tandems* retorna uma lista de repetições *tandem* exatas [2] em um única seqüência. Na experiência dos autores, muitas das repetições encontradas eram repetições *tandem* e apareciam adjacentes a seqüências únicas e os *matches* apareciam nas extremidades dessas repetições *tandem* em repetições dele mesmo.

Para fazer o alinhamento entre duas seqüências é preciso invocar o *script* `run-mummer1` ou `run-mummer3`, ambos são responsáveis por chamar todos os demais *scripts* e programas para a confecção do alinhamento. `run-mummer1` faz o alinhamento usando as definições da primeira versão e `run-mummer3` usa as definições da terceira versão. Esses *scripts* recebem como parâmetro um arquivo fasta contendo o genoma de referência, um arquivo fasta contendo o genoma *query* ou um arquivo multi-fasta contendo os *contigs queries*, além das opções da ferramenta, e retornam 4 arquivos. O arquivo `.out` contém a lista dos *matches* exatos encontrados, o arquivo `.gaps` listando os *gaps* encontrados, `.align` que contém o alinhamento dos *matches* e `errorgaps`.

[2] Repetições concatenadas de uma mesma sub-seqüência.

NUCmer

NUCmer (NUCleotideos MUMmer) é um sistema capaz de fazer o alinhamento entre múltiplos-*contigs* que tem o MUMmer em seu núcleo. Foi disponibilizado juntamente com a segunda versão de MUMmer. Na época seus criadores observaram que os projetos de seqüenciamento de genomas inteiros do tipo *shotgun*(WGSS) estavam adiando sua fase de finalização por tempo indeterminado e seus resultados estavam se resumindo a centenas de *contigs* que representam um nível de montagem *shotgun* $5x$, $6x$[3] [4]. Observaram também que geralmente para os organismos desses projetos existiam um outro organismo bastante próximo evolutivamente já seqüenciado, então desenvolveram um método computacional para alinhar múltiplos *contigs* com um genoma completo. Outra motivação foi o desejo de comparar os *contigs* resultantes de um nível de montagem *shotgun* com os resultantes de outro nível de montagem, por exemplo $5x$ com $6x$. NUCmer também executa o alinhamento em três fases como MUMmer.

NUCmer recebe como entrada as seqüências referência e *query* através de arquivos multi-fasta, primeiramente concatena as seqüências dos *contigs* de cada um dos arquivos de entrada resultando em duas seqüências. Essas seqüências são submetidas a MUMmer que retorna todos os *matches* exatos entre os dois genomas. Esses *matches* são mapeados de volta aos *contigs* em cada um dos arquivos de entrada.

Na segunda fase, NUCmer roda um programa de agrupamento dos *matches* ao longo de cada *contig*. Os *matches* são agrupados se estiverem dentro de uma distância especificada pelo usuário. A seguir o sistema roda uma variação do algoritmo de programação dinâmica de alinhamento de Smith-Waterman para alinhar as seqüências entre os agrupamentos de *matches*. Para evitar a computação excessiva, o número de *mismatches* é limitado nos *gaps* entre os *matches*. A quantidade exata de *mismatches* é definida pelo usuário.

NUCmer constrói dois arquivos como resposta, um arquivo **.cluster** e um **.delta**. O arquivo .cluster contém a lista dos *matches* agrupados para cada duas seqüências comparadas. O arquivo .delta contém uma representação codificada dos alinhamentos de todas seqüências *queries* contra todas as seqüências de referência.

NUCmer permite que alguns parâmetros essenciais para a construção dos al-

[3]Cobertura de uma posição p é o número de fragmentos que possuem uma de suas bases passando em p.

[4]Montagem ou cobertura $5x$ indica que existem pelo menos 5 fragmentos passando pela posição p

inhamentos sejam controlados pelo usuário.

PROmer

PROmer (PROtein MUMmer) é um sistema cujo objetivo é propiciar uma anotação comparativa entre genomas através da comparação das proteínas desses genomas. Acreditando que as seqüências de proteínas conservam-se por um período muito maior, em uma escala de tempo evolucionária, que as seqüência de DNA, PROmer tenta identificar relacionamentos que não são visíveis nos alinhamentos de seqüências de DNA.

Assim como NUCmer, PROmer também usa MUMmer para construir os alinhamentos entre as seqüências e por isso usa os mesmo métodos computacionais. A principal diferença entre as ferramentas está no fato de que PROmer traduz as seqüências de DNA, dadas em dois arquivos multi-fasta da entrada, em seqüências de aminoácidos.

Inicialmente todas as seqüências de DNA são traduzidas em todos os seis *frames* de leitura, nesse momento um índice é criado para mapear as seqüências de aminoácidos e seus tamanhos nas seqüências de DNA originais. Feita a tradução, PROmer filtra as seqüências de aminoácidos que contém um número excessivo de *stop codons* que podem não fazer parte de uma proteína. As seqüências restantes são concatenadas, separadamente de acordo com os dois arquivos de entrada, formando duas seqüências de aminoácidos que representam as potenciais proteínas de cada genoma. Essas seqüências são submetidas a MUMmer que identifica os *matches* entre elas. Para finalizar essa fase PROmer mapeia os *matches* nas seqüências de DNA originais usando o índice criado na tradução.

Com os *matches* identificados PROmer faz o agrupamento desses usando as coordenadas das seqüências de DNA. Ao agrupar os *matches* pode acontecer desses estarem inconsistentes segundo o *frame* de leitura usado na tradução. Para resolver esse problema, a ferramenta assume que o *frame* a ser usado é aquele que tem o maior grau de homologia entre as seqüências de proteínas. Assim cada agrupamento termina com uma série de *hits* de aminoácidos em um único *frame* de leitura. Com os agrupamentos de *matches* construídos a ferramenta passa para a terceira fase que é extender os alinhamentos entre os agrupamentos, aumentando a área de cobertura total da região, usando um algoritmo eficiente para alinhamento de seqüências e a matriz de pontuação BLOSUM62.

PROmer apresenta seus resultados no mesmo formato que NUCmer, um ar-

quivo **.cluster** e um arquivo **.delta** contendo, respectivamente, a listagem dos agrupamentos de *matches* para cada duas seqüências comparadas e os alinhamentos entre as seqüências codificados. PROmer permite a manipulação de parâmetros essências para a construção dos alinhamentos.

2.2.2 Mega BLAST

Mega BLAST é uma ferramenta para alinhamento de seqüências de DNA onde essas seqüências se difereciam apenas por erros de seqüenciamento ou erros equivalentes. Usando um algoritmo guloso Mega BLAST produz um alinhamento tão bom quanto os gerados por algoritmos que usam programação dinâmica, porém chega a ser até 10× mais rápido trabalhando com dados apropriados. Ao fazer a comparação entre duas seqüências, digamos s e t cujos respectivos tamanhos são m e n, o algoritmo usa uma matriz $m \times n$, assim como a programação dinâmica, porém não são computadas todas as entradas dessa matriz. Mega BLAST percorre algumas diagonais calculando a distância entre as seqüências. A **distância** D_{ij} entre os prefixos $s_1 \dots s_i$ e $t_1 \dots t_j$ é definida pelo menor número de operações de edição (substituição, inserção ou remoção) necessárias para transformar um sufixo no outro.

Mega BLAST trabalha diretamente sobre a distância entre as seqüências. Para determinar quais diagonais devem ser percorridas, ele aplica uma restrição baseada na similaridade entre os sufixos e assim percorre apenas as diagonais cujas similaridades são maiores do que $-\infty$. Sendo T a maior similaridade computada até a diagonal anterior e X um valor definido pelo usuário ($X > 0$), o algoritmo determina que para toda entrada (i,j) da matriz cuja $sim(i, j) < T - X$, seja atribuído $-\infty$ à $sim(i, j)$. Por isso o algoritmo precisa transformar distância ($dist(s, t)$) em similaridade ($sim(s, t)$) de uma maneira eficiente. Segundo [19], isso é possível fazendo:

$$sim(s,t) = \frac{M}{2}(m+n) - dist(s,t)$$

onde :

M é uma constante,

$m = |s|$,

$n = |t|$.

Mega BLAST considera somente os alinhamentos que contiverem um *match* idêntico com pelo menos W nucleotídeos, e é mais eficiente quando $W > 16$.

2.2.3 Egg

A ferramenta Egg (*Extended Genome-Genome comparison*) [4] foi desenvolvida para fazer a comparação entre os proteomas[5] de dois organismos e determinar como a reordenação e o reagrupamento dos genes influenciam nas diferenças entre as funcionalidades dos genomas. Baseada em uma metodologia de busca por estruturas organizacionais, a sua implementação usa a teoria dos grafos, mais especificamente dois grafos bipartidos e a procura por cliques maximais. As estruturas organizacionais referidas são *genes ortólogos*, *genes específicos*, *regiões de genes específicos*, *regiões de genes ortólogos*, *espinha dorsal* e *famílias de genes parálogos*.

As estruturas organizacionais são definidas a partir da menor organização possível num proteoma, o gene. Um **gene** é um trecho do genoma que carrega informações funcionais. Um gene pode ser traduzido em uma proteína ou ser codificado em RNA. **Proteína** é uma seqüência traduzida para aminoácidos de uma ORF(*open read frame*) predita como pertencendo a um gene. Dentre os genes a ferramenta procura por casos especiais que são os genes parálogos, ortólogos e específicos. **Genes ortólogos** e **genes parálogos** são genes descendentes de um mesmo gene ancestral, através do eventos de especiação e duplicação respectivamente. A diferença é que o primeiro ocorre em genomas diferentes, e o segundo ocorre num mesmo genoma. **Genes específicos** de um genoma são aqueles genes que não têm um gene ortólogo entre os genes do outro genoma.

Além dos genes especiais, a metodologia define as regiões de ocorrência desses genes. **Região de genes consecutivos (RGC)** é definida como um conjunto de genes consecutivos em um proteoma ordenados pelas coordenadas de início de cada gene. **Região específica (RE)** de um genoma G em relação a um genoma H é uma região "rica" em genes específicos, ou seja uma região onde a grande maioria dos genes ocorrem apenas no genoma G. **Região ortóloga (RO)** é definida como sendo um par (α, β) onde α é uma RGC no genoma G e β é uma RGC em H, além disso α e β devem ser descendentes de uma mesma região ancestral e terem aproximadamente o mesmo número de genes.

Para completar as estruturas organizacionais é preciso definir cruzamento e espinha dorsal de RGCs. O **cruzamento** ocorre quando dois pares ortólogos

[5]Conjunto de todos os genes de um genoma que codificam proteína.

(g, h) e (g', h') têm a ordem de g e g' em G invertida em relação a h e h' em H. A **espinha dorsal** de duas RGCs, α e β, é uma seqüência de pares ortólogos (g, h) onde cada gene g em α tenha no máximo um gene ortólogo h a ele em β e vice-versa, e que não ocorra nenhum cruzamento entre os pares.

A implementação de EGG é feita em três fases. Na primeira é feita a comparação dos genes, na segunda são definidas as arestas dos grafos e na última fase são determinadas as estruturas organizacionais.

Para fazer a comparação dos genes é utilizada a ferramenta BLAST que fornece, para cada gene g_i, uma lista dos genes considerados similares dentre os genes de H. Cada gene h_j retornado por BLAST, que é chamado de *hit*, vem acompanhado de algumas informações, usadas por EGG são o *e-value*, o *score* e o alinhamento entre g_i e h_j. O *e-value* é uma significância estatística do *hit*, ou seja, indica a probabilidade esperada do *hit* ter sido encontrado ao acaso. Quanto menor o *e-value*, menor a probabilidade do *hit* ser encontrado ao acaso,ou seja, mais significativo é o *hit*.

O EGG considera um gene g no genoma G órtologo a um gene h no genoma H se e somente se :

- a medida de significância estatística $s(g, h)$ da similaridade entre g e h seja menor ou igual a um limite fixo S; e
- dado o alinhamento de g e h, pelo menos $P\%$ de $|g|$ e $P\%$ de $|h|$ sejam cobertos pelo alinhamento.

Para EGG, um gene g é específico em um genoma G em relação a outro genoma H, quando a medida de significância estatística $s(g, h)$ da similaridade entre eles é maior que um limite S', para todo gene h no genoma H, tal que $S' > S$.

EGG implementa como medida de significância o *e-value* quando determina os genes específicos. Assim construindo, para cada gene g_i, uma lista ligada contendo os *hits* retornados por BLAST em ordem não-decrescente de *e-value*, EGG determina os genes específicos como sendo aqueles que tenham *e-value* maior do que S'.

No inicio da fase de determinação das arestas do grafo, EGG encontra os pares ortólogos usando o valor da similaridade do alinhamento do *hit* (*score*), como a medida de significância $s(g, h)$, e o alinhamento, ambos passados por BLAST, e tendo como valores limites, descritos anteriormente, $S = 10^{-5}$, $S' = 10^{-3}$, e $P = 60\%$. Observe que o intervalo entre 10^{-5} e 10^{-3} é considerado uma zona de dúvida para a similaridade, já os *scores* abaixo de 10^{-5} têm alta similaridade, e os genes com *score* acima de 10^{-3} são considerados genes específicos.

Cada par ortólogo encontrado, obedecendo aos limites é chamado de *match* e representa uma aresta no grafo. Sendo m e n o número de genes de G e H, respectivamente, Egg armazena os *matches* em uma matriz binária $A_{m \times n}$, tal que $A_{i,j} = 1$ se, e somente se, g_i e h_j formam um *match*.

Para implementar o conceito de par de genes fortemente ortólogos, Egg usa o que chamamos de **BBH**, de *Bidirectional Best Hit*. Um par (g_i,h_j) de genes ortólogos forma um BBH se h_j é o melhor *hit* encontrado por g_i, ou seja, aquele com menor *e-value*, e vice-versa.

Na terceira fase é preciso determinar os **runs**, que pelas definições dadas acima podemos dizer que são seqüencias consecutivas com pelo menos dois *matches*. Como os *matches* estão armazenados na matriz A, os *runs* são determinados pelas diagonais de A onde as posições estão preenchidas com 1. O programa gera um código para cada *run* encontrado, o código traz as informações sobre os genomas comparados, ano, mês e dia da comparação, um número seqüencial do *run* naquela comparação, se ele é paralelo ou anti-paralelo e se é consistente ou inconsistente.

Para encontrar as regiões ortólogas, o programa procura *runs* com pelo menos $M = 3$ *matches*, ou a união de *runs* que distam no máximo K genes. Na implementação o valor de K é determinado por uma estratégia, essa determina l_{min} como a menor distância entre os genes finais de um *run* e os genes iniciais do outro em cada um de seus respectivos *matches* e l_{max} é a distância máxima respectivamente. Com o uso de dois parâmetros **max.small.gap** e **max.large.gap** passados pelo usuário, que determinam o tamanho máximo do menor e do maior intervalo entre os runs, em número de genes, os *runs* são juntados para formar uma RO se

$$l_{min} \leq \textbf{max.small.gap} \text{ e } l_{max} \leq \textbf{max.large.gap}$$

Os valores padrões são 5 para **max.large.gap** e 3 para **max.small.gap**. Para encontrar a espinha dorsal dos proteomas, objetivando encontrar o alinhamento entre os proteomas Egg implementa um algoritmo de programação dinâmica para encontrar a seqüência comum mais longa sobre os BBH's encontrados pelo BLAST na fase de comparação dos genes. E para determinar as familias de genes parálogos foi implementado um algoritmo que encontra os cliques maximais do grafo onde os vértices são os genes do genoma. O algoritmo primeiro cria as arestas (matches) do grafo G, onde cara vértice é um gene do genoma. Depois cada aresta é considerada uma clique de G e vértices são adicionadas um-a-um nas cliques até que não seja mais possível

adicionar[6]. As famílias então começam em cliques maximais. No caso de genes que pertençam a mais de uma familia, o valor adotado como critério de desempate é o *score* retornado pelo BLAST.

[6] Clique é um conjunto de vértices adjacentes dois-a-dois. Uma clique C é máximal se não existe clique C' que contenha C

Capítulo 3

Ancoragem

Nesse capítulo apresentamos os objetivos e as idéias que nos motivaram a desenvolver este trabalho, bem como a metodologia usada para a execução dos testes. Tentamos transmitir a principal função do processo chamado de ancoragem, bem como o que fizemos com cada uma das ferramentas para que essas pudessem fazer a ancoragem dos contigs no genoma completo. Apresentamos ainda as ferramentas computacionais que desenvolvemos para operacionalizar todo o processo de testes.

3.1 Definições e objetivos

Nossa principal abordagem consiste na observação de alinhamentos capazes de estimar a posição exata de partes não-contíguas de um genoma incompleto em um genoma completo, a partir dos resultados obtidos de ferramentas de comparação de seqüências. Usaremos as ferramentas descritas no capítulo 2, capazes de gerar alinhamentos entre seqüências. Algumas delas já determinam diretamente uma posição de ancoragem, e para as outras adicionamos tal funcionalidade. As ferramentas foram aplicadas sobre um conjunto de testes composto por 10 genomas completos comparados com dez conjuntos respectivos, contendo 100, 200, 300, 400, 500, 600, 700, 800, 900 e 1000 *contigs* dos próprios genomas.

Ancoragem é um processo que determina uma provável ordenação de pedaços de um genoma incompleto, os contigs, de um organismo em relação ao genoma completo de um outro organismo, desde que sejam organismos evolutivamente próximos. Ou seja, dado um genoma G com coordenadas $[1 \ldots n]$ e $C =$

$\{C_1 \dots C_k\}$ o conjunto de k *contigs* de um genoma H, chamamos de **ancoragem de C_i em** G, e denotamos por X_i, a posição onde uma ferramenta M ancorou a primeira base de C_i em G, $1 \leq i \leq k$. A Figura 3.1 exemplifica a ancoragem dos contigs $C_1 \dots C_k$.

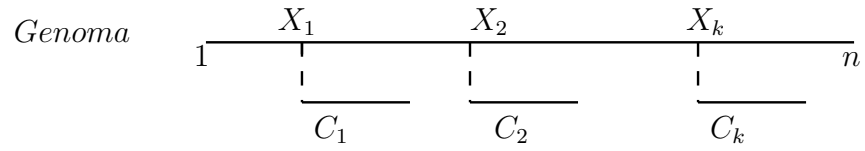


Figura 3.1: Ancoragens $X_1 \dots X_k$ dos contigs $C_1 \dots C_k$ no genoma completo.

3.2 Ferramentas para Ancoragem

A idéia nessa seção é mostrarmos como cada uma das ferramentas, descritas na seção 2.2, efetivamente determina a ancoragem dos *contigs* nos genomas completos, e principalmente os *scripts* em Perl que desenvolvemos para agilizar o processo de testes.

Desenvolvemos várias ferramentas computacionais (*scripts* em Perl) para agilizar o processo de ancoragem usando as ferramentas Promer, NUCmer, Mega BLAST e Egg. O primeiro *script*, nomeado `breakSeq.pl`, é responsável por fazer a quebra de um genoma inteiro em k contigs. Descrevemos os detalhes de implementação e funcionamento dessa ferramenta na seção 4.2. O importante, por enquanto, é sabermos que essa ferramenta cria três arquivos, `contigs.brk`, `contigs.res`, e `contigs.fasta`. No primeiro estão os dados que permitem a geração das seqüências dos contigs, no segundo estão os resultados esperados pelas ancoragens, ou seja, as posições e fitas de onde cada um dos contigs foram retirados da seqüência, e no terceiro arquivo temos as k seqüências de DNA dos contigs. O segundo script, `genAnchors.pl`, tem a finalidade de controlar todo processo de ancoragem, e isso envolve em um primeiro momento a aplicação das ferramentas nos contigs e genomas, e no segundo momento computar as medições das ancoragens resultantes de cada uma das ferramentas.

Os processos de ancoragens são feitos diretamente por `genAnchors.pl` para as ferramentas NUCmer, PROmer e Mega BLAST, ou seja, o script faz uma chamada direta às ferramentas, passando os parâmetros necessários. No caso de Egg que, como dito na seção 2.2.3, usa BLAST para comparar proteínas preditas, criamos uma ferramenta que gera k arquivos fasta contendo a seqüência de nucleotídeos de cada um dos contigs e k arquivos contendo a anotação de

cada um dos contigs. Assim executamos Egg comparando os contigs, um a um, com o genoma inteiro, o que implicou no desenvolvimento de outro script que faz a preparação do banco de dados de BLAST com as seqüências envolvidas.

Para cada uma das ancoragens, `genAnchors` gera três arquivos: o primeiro, `.res`, contém as posições e fitas de ancoragem para cada um dos contigs ancorados, os outros são os arquivos `.conf` e `.gff` que são usados pela ferramenta de visualização gráfica GBrowse, descrita no capítulo 5, para exibição das ancoragens. Scripts específicos para capturar cada um dos arquivos `.res` de cada ferramenta e computar as medições estão descritos nas subseções a seguir, onde descrevemos também as técnicas usadas em cada ferramenta na ancoragem.

3.2.1 NUCmer e PROmer

A segunda versão do pacote MUMmer, apresentada na seção 2.2.1, disponibilizou as ferramentas NUCmer e PROmer, capazes de fazer comparações entre um genoma completo e multi-*contigs* resultando em alinhamentos do tipo todos contra todos. Esse pacote também disponibilizou uma ferramenta chamada `show-tiling`, com o objetivo de determinar o melhor mapeamento dos *contigs* em uma única na seqüência de referência, baseando-se nos alinhamentos encontrados. `Show-tiling` tenta construir um trajeto com os *contigs*, mapeando-os na seqüência referência. A partir dos alinhamentos encontrados entre uma seqüência de referência longa e muitos pequenos *contigs*, `show-tiling` determina a melhor posição de mapeamento para cada *contig*. É importante dizer que cada *contig* é usado na cobertura apenas uma vez o que traz dificuldades ao sistema na ocorrência de repetições.

`Show-tiling` recebe como entrada um arquivo `.delta` resultante de NUCmer ou PROmer. A saída padrão de `show-tiling` traz, para cada seqüência de referência, uma lista dos *contigs* mapeados com 8 colunas informando as posições de início e fim na seqüência referência, o tamanho do *gap* para o próximo *contig*, o tamanho, a cobertura do alinhamento, o percentual de identidade, a orientação e o ID do *contig*. `Show-tiling` trata o caso do genoma ser circular e também permite que o usuário determine o percentual mínimo de cobertura e identidade que cada alinhamento deve ter para ser considerado na determinação do trajeto.

Fazemos as medições das ancoragens resultantes de PROmer e NUCmer com o script `medida.pl`, que toma como entrada os arquivos `contigs.res`, que contêm a ancoragem esperada, e o arquivo com a ancoragem resultante da

ferramenta. Então verificamos se o contig foi ancorado ou não pela ferramenta, se foi então observamos a fita em que foi ancorado e por fim calculamos a distância entre a posição de ancoragem e a posição esperada. A saída contendo as distâncias de ancoragem para cada um dos contigs é dada através de um arquivo `.dist`.

3.2.2 Mega BLAST

Mega Blast permite a escolha de vários formatos de saída, nós optamos pela opção 3, que traz uma listagem dos hits encontrados em 12 colunas com as seguintes informações: `Query id`, `Subject id`, `% identidade`, `tamanho do alinhamento`, `número de erros`, `abertura de gaps`, `início na query`, `fim na query`, `início na referência`, `fim na referência`, `e-value` e `bit score`. Os hits são ordenados pelo `% de identidade`, `tamanho do alinhamento`, `query id` e assim por diante. Executamos Mega BLAST passando como parâmetro os arquivos fasta com as seqüências do genoma inteiro e com as seqüências dos k contigs, além da opção de formato para saída.

Implementamos o cálculo das medições em `medmblast.pl`, e nesse, para cada contig listado, encontramos seu primeiro hit, que pode ser considerado o melhor, e então, supondo que q_i e q_f denotam a posição de início e fim, respectivamente, do alinhamento do hit no contig, e que s_i e s_f a posição de ínicio e fim, respectivamente, do alinhamento do hit no genoma, computamos p_i, a posição de ancoragem do contig, de duas maneiras : a primeira quando o alinhamento é dado na mesma fita seguimos o demonstrado na Figura 3.2; a segunda quando o alinhamento é dado em fitas contrárias seguimos o demonstrado na Figura 3.3.

$$p_i = |s_i - q_i|$$

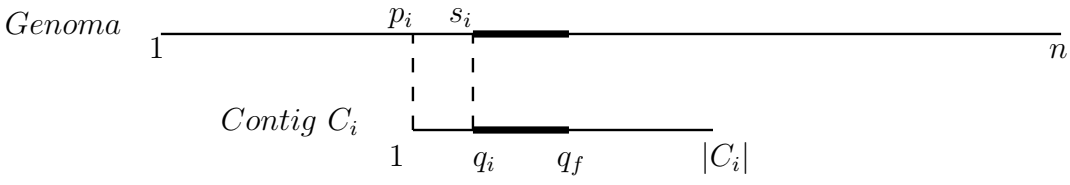


Figura 3.2: Cálculo da ancoragem de Mega BLAST na mesma fita.

Com as posições de ancoragem dos contigs definidas, computamos os acertos observando o arquivo `contigs.res`, e assim criamos o arquivo `mblast.dist`.

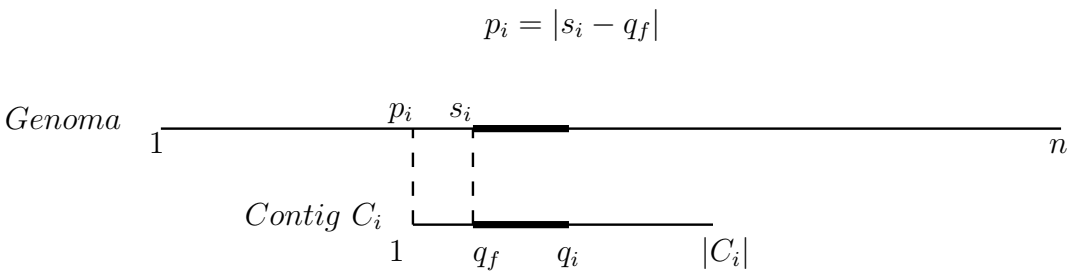


Figura 3.3: Cálculo da ancoragem de Mega BLAST em fitas contrárias.

3.2.3 Egg

A primeira versão de Egg foi projetada para comparar pares de genomas inteiros. Recentemente, uma atualização permitiu comparações entre multi-*contigs* anotados. Contudo, Egg ainda não estava apto a fazer a ancoragem dos *contigs* em um genoma completo. Desenvolvemos uma técnica para ancorar os *contigs*, usando os BBH's encontrados entre os genes do genoma completo e os genes dos *contigs*.

Egg mapeia as proteínas de cada *contig* no genoma de referência, particularmente os BBH's, ao invés de mapear o *contig* inteiro. Assim nós precisamos traduzir o conjunto de BBH's formados pelos genes do *contig* C_i e genes do genoma inteiro G em uma posição X_i. X_i, chamada de *mapeamento* de C_i, é a posição onde um método qualquer mapeia a primeira base do *contig* C_i, pode ser qualquer valor entre 1 e o tamanho do genoma inteiro G. Formalmente, seja (g, h) um BBH, onde g é um gene de G e h um gene de C_i. Nós simulamos um sistema de coordenadas Cartesiano onde G e C_i são dois eixos horizontais e as coordenadas (x, y) de g e h são dadas por $(x_g, 1)$ e $(x_h, 0)$. X_i é calculada encontrando a posição δ em G que minimiza a somatória das distâncias Euclidianas entre todos os pares de genes dos BBHs entre G e C_i , considerando C_i deslocado de $\delta - 1$ posições em G. Assim, X_i é dada pela fórmula abaixo e a Figura 3.4 ilustra dois BBHs e o deslocamento δ.

Implementamos a tradução de BBH's em ancoragem no script `bbh2res.pl`, que a partir de cada um dos k arquivos `.bbh` gera um único arquivo `.res` que depois submetemos ao `medida.pl` que gera o arquivo `.dist` contendo as medições.

$$X_i = \arg\min_{\delta=1\ldots n} \sum_{\forall \text{ BBHs em } C_i \times G} \sqrt{1 + (x_g - x_h - \delta)^2}$$

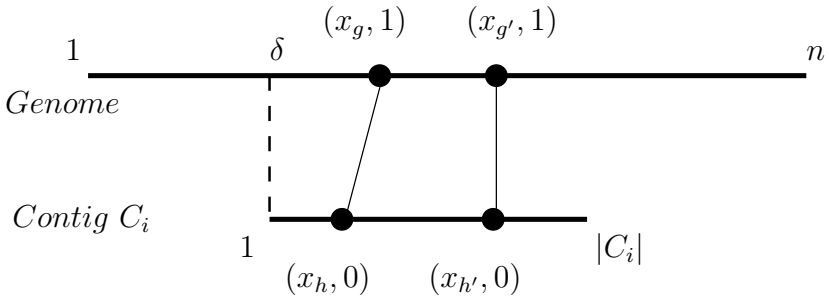


Figura 3.4: Uma ilustração de dois BBHs, (g, h) e (g', h'), mapeados em G.

Capítulo 4

Estudo comparativo

Nesse capítulo apresentamos um estudo comparativo entre as ancoragens resultantes da aplicação das ferramentas NUCmer, PROmer, Mega BLAST e Egg em nossos conjuntos de testes. Antes, definimos a medida usada para determinar a qualidade e apresentamos nossos conjuntos de testes.

4.1 Medida de comparação

Para avaliarmos o quanto uma ancoragem feita por uma ferramenta M é melhor do que outra ancoragem, definimos uma medida de comparação de ancoragens, denotada por S_M. Essa medida de comparação é baseada na distância entre a ancoragem dada para um contig i e a posição de onde ele foi retirado do genoma, uma vez que contruímos nossos contigs a partir de genomas completos. A construção dos conjuntos de testes está descrita na seção **??**. Definimos a distância a seguir.

> Definição: Dado um genoma G com coordenadas $[1 \dots n]$ e $C = \{C_1 \dots C_k\}$ o conjunto de k contigs de G, sejam P_i a primeira posição da primeira base de C_i em G, e X_i a ancoragem de C_i em G. Chamamos de **distância de ancoragem de C_i em G**, e denotamos por $d_{M,i}$, a distância entre P_i e X_i, de acordo com as coordenadas de G e a ancoragem construída por M.

Essa distância precisa ser calculada considerando que os genomas podem ser circulares ou lineares. Quando o genoma é linear o cálculo da distância de ancoragem é dado por $d_{M,i} = |X_i - P_i|$, como mostra a Figura 4.1.

$$d_{M,i} = |X_i - P_i|$$

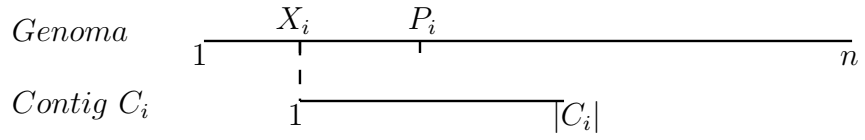
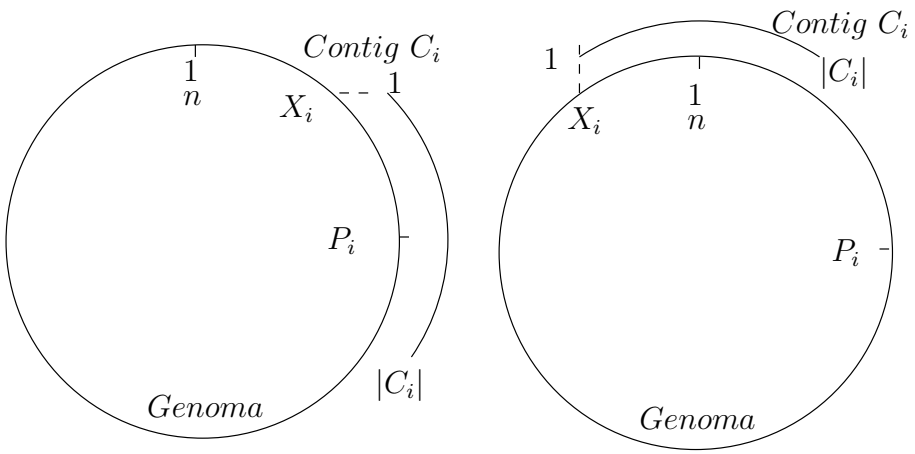


Figura 4.1: No genoma linear a medida $d_{M,i}$, que determina a distância entre a posição esperada P_i e a ancoragem X_i, é dada por $|X_i - P_i|$.

No caso de G ser um genoma circular a distância de ancoragem é dada por $d_{M,i} = \min\{|P_i - X_i|,\ n - |P_i - X_i|\}$, para garantir que essa distância seja a menor. Observe a Figura 4.2.

$d_{M,i} = |X_i - P_i|$ $\qquad$ $d_{M,i} = n - |X_i - P_i|$



$$d_{M,i} = min\{n - |X_i - P_i|, |X_i - P_i|\}$$

Figura 4.2: Medida de distância $d_{M,i}$ em genoma circular. No caso (a) a distância é determinado por $|X_i - P_i|$. No caso (b) o contig está localizado sobre a posição inicial do genoma, o que faz a distância ser dada por $n - |X_i - P_i|$.

Podemos dizer que a ancoragem correta é aquela onde $d_{M<i} = 0$. Nossa medida de comparação de ancoragens faz a contagem dos acertos que uma ferramenta obteve ao ancorar k contigs, porém ela considera como acerto uma ancoragem não tão distante de P_i.

A **medida de comparação de ancoragem** S_M é definida como o

número de distâncias de ancoragem $d_{M,i}$ tal que $d_{M,i} \leq \alpha|C_i|$, onde α é um fator que permite soluções aproximadas (não tão distantes de P_i).

A Figura 4.3 mostra a ancoragem dos contigs C_1, C_2 e C_3 nas posições X_1, X_2, e X_3 respectivamente, e traz as posições P_1, P_2, e P_3 de onde esses contigs foram retirados. Ao adotarmos $\alpha = 0,5$, ou seja, estamos considerando como intervalo de acerto até 50% do tamanho do contig, teremos que $S_M = 2$.

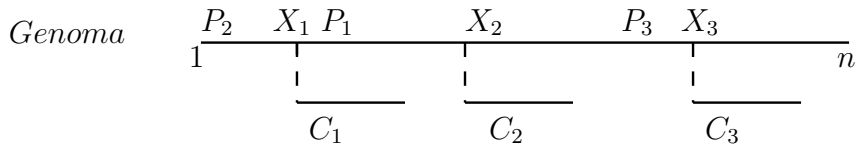


Figura 4.3: Exemplo da ancoragem de três contigs num genoma completo e como a medida de comparação S_M atuaria, fazendo $\alpha = 0,5$.

Nós não consideramos, no cálculo de S_M, os contigs ancorados em fita errada ou não ancorados.

4.2 Conjunto de Testes

Nossos conjuntos de testes foram montados a partir de 10 genomas completos retirados do *GenBank*. Esses genomas foram arbitrariamente quebrados em conjuntos de 100, 200, 300, 400, 500, 600, 700, 800, 900 e 1000 contigs. Descrevemos o método de geração dos contigs e relacionamos os genomas escolhidos para os testes nessa seção.

Genomas

Foram utilizados genomas completos oriundos de projetos genomas completos e depositados no *GenBank*. *GenBank* é a base de dados de seqüências genéticas do *National Institute Health of United States of America*, uma coleção de seqüências de DNA anotadas e disponíveis ao público. Os organismos escolhidos estão relacionados na Tabela 4.1. A tabela traz ainda informações como tamanho do genoma em milhões de *basepair*(Mbp), o número de cromossomos, identificador no *Genbank*, o identificador no *RefSeq*. *Reference Sequence (RefSeq)* é uma coleção, disponibilizada pelo NCBI, que fornece um

conjunto detalhado, integrado, não redundante das seqüências, incluindo o DNA, o RNA transcrito e os produtos genômicos da proteína, dos principais organismos pesquisados.

Tabela 4.1: Genomas selecionados. A tabela informa o tamanho dos genomas em MBp, o número de cromossomos, o código no Genbank e o código do RefSeq.

Organismos	Tam	Cr	Genbank	RefSeq
Baccillus anthracis str. Sterne	5.23	1	*AE*017225.1	*NC*_005945.1
Buchnera aphidicola str. Sg (Schizaphis graminum)	0.64	1	*AE*013218.1	*NC*_004061.1
Candidatus Blochmannia pennsylvanicus str. BPEN	0.79	1	CP000016.1	*NC*_007292.1
Escherichia coli W3110	4.65	1	*AP*009048.1	*NC*_000091.1
Haemophilus influenzae 86-028NP	1.91	1	*CP*000057.1	*NC*_007146.1
Helicobacter pylori HPAG1	1.61	1	*CP*000241.1	*NC*_008086.1
Streptococcus pneumoniae R6	2.04	1	*AE*007317.1	*NC*_003098.1
Synechococcus sp. CC9605	2.51	1	*CP*000110.1	*NC*_007516.1
Xanthomonas campestris pv. campestris str. ATCC3913	5.08	1	*AE*008922.1	*NC*_003902.1
Xanthomonas campestris pv. campestris str. 8004	5.15	1	*CP*000050.1	*NC*_007086.1

Todos os genomas escolhidos são circulares.

Processo de geração dos *contigs*

Para efetuarmos a quebra de um determinado genoma inteiro em uma determinada quantidade de *contigs*, nós criamos um conjunto de *scripts* em Perl, que são os seguintes :

- `genContigs.pl` - responsável por coordenar a execução dos demais *scripts*.
- `breakSeq.pl` - promove a quebra do genoma em k *contigs* e determina a ancoragem esperada.
- `genFASTA.pl` - constrói o arquivo multi-fasta contendo as seqüências de nucleotídeos dos contigs e o arquivo GFF retratando a ancoragem esperada para ser mostrado na ferramenta *GBrowse*. Mais detalhes sobre o formato GFF serão vistos na seção 5.1

Geramos contigs de tamanho e orientação diferentes e garantimos que não existe sobreposição entre eles. Apresentamos aqui o processo de quebra de forma geral e posteriormente, nas subseções, apresentamos uma breve descrição dos *scripts* dentro do processo. O principal resultado da quebra será um arquivo no formato multi-fasta contendo as seqüências de nucleotídeos dos *contigs*.

Para iniciar o processo devemos informar o arquivo FASTA contendo a seqüência do genoma a ser quebrado, o número de *contigs* a serem criados (`nContigs`) e se ocorrerá ou não a inversão de orientação dos *contigs*. Primeiramente determinamos os tamanhos máximo e mínimo para os *contigs*. O tamanho máximo nós definimos como sendo o tamanho do genoma, em nucleotídeos, dividido pelo número de *contigs* desejado. O tamanho mínimo como sendo 10% do tamanho máximo. Dentro desses limites, para cada um dos `nContigs`, fazemos uma escolha aleatória determinando seu tamanho.

Com o tamanho do *contig* determinado, preocupamo-nos agora com a posição inicial do *contig* dentro do genoma. Para evitarmos a sobreposição, cada um dos *contigs* é retirado da i-ésima parte do genoma, onde i está entre 1 e `nContigs`. Assim, fazemos uma escolha aleatória para posição inicial do *contig* entre a posição inicial da i-ésima parte do genoma e a última posição possível de caber o *contig*. Esta posição foi definida subtraindo da posição final da i-ésima parte do genoma, o tamanho do *contig*. A orientação do *contig* também foi determinada de forma aleatória. Na verdade, observamos o tamanho do *contig*. Quando esse é ímpar o *contig* tem a orientação revertida, caso contrário a orientação é mantida.

Assim, tendo o tamanho, a posição inicial e a orientação de cada um dos *contigs*, passamos a construir os arquivos de saída, `contigs.res` e `contigs.fasta`. O primeiro arquivo contém, para cada contig, informações como sua posição inicial, seu tamanho e sua direção em relação ao genoma. Temos então a ancoragem esperada.

4.3 Resultados e Análise Comparativa

Apresentamos nesta seção os resultados do nosso trabalho através de 3 gráficos que trazem o percentual de acertos das ancoragens dadas pelas ferramentas. As figuras 4.4, 4.5 e 4.6 mostram os resultados quando usamos $\alpha = 0.1$, $\alpha = 0.25$ e $\alpha = 0.5$, respectivamente. Lembramos que α é um parâmetro que usamos para determinar um intervalo onde consideramos a ancoragem correta, ou seja, quando fazemos $\alpha = 0.5$ estamos considerando um intervalo de 50% do tamanho do contig em torno da posição esperada de ancoragem como sendo uma ancoragem correta. Os resultados mostam que apenas Mega BLAST, dentre as ferramentas, tem seus resultados afetados pela variação de α, embora os comportamentos de todas as ferramentas tenham permanecido inalterados relativamente ao α aplicado.

Os gráficos trazem no eixo x o número de *contigs* em cada conjunto de testes.

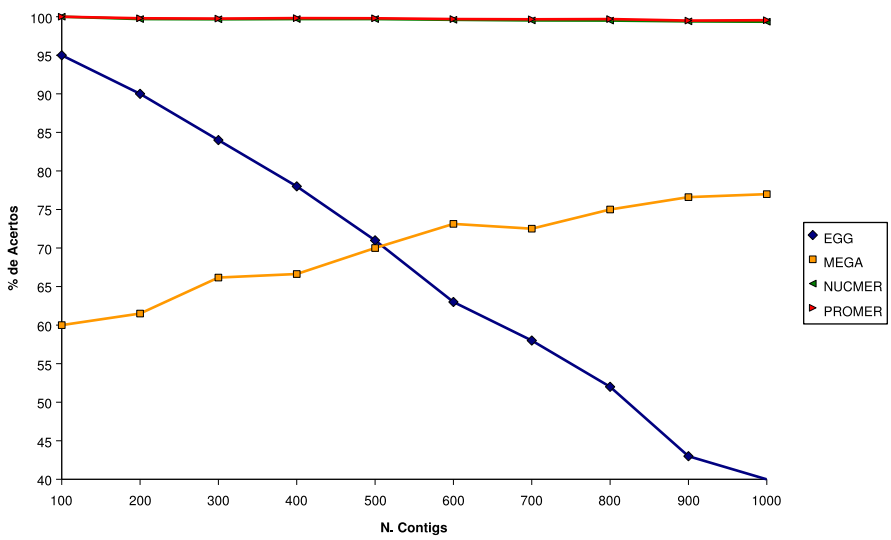


Figura 4.4: Resultados para $\alpha = 0.1$. O eixo y representa o percentual de acerto na ancoragem feita com todos os 10 genomas. O eixo x indica o número de *contigs nos casos de testes*

No eixo y é dado o percentual de acertos considerados por S_M para todos os 10 conjuntos de x *contigs*. Por exemplo, observando a figura 4.4, Egg obteve 95% de acertos com os conjuntos de 100 *contigs*. Isso significa que dos 10×100 *contigs* a serem ancorados, Egg acertou 950. Tanto PROmer como NUCmer apresentam ancoragens muito próximas da ideal com ligeira vantagem para PROmer. Egg mostra-se com um bom desempenho para contigs maiores enquanto que Mega BLAST melhora quando os contigs são menores.

As ferramentas PROmer e NUCmer mantiveram um resultados acima dos 95% de acerto. Entendemos que a procura por matches idênticos maximais foi determinante para esse desempenho, e a vantagem de PROmer em relação a NUCmer está no fato de usar as seqüências de aminoácidos, que tendem a ter um menor índice de diferenças entre as cadeias uma vez que basta um símbolo diferente na cadeia para que a árvore de sufixos desconsidere a identidade entre os trechos das seqüências.

Observando os gráficos, identificamos o comportamento de Egg de obter melhores resultados quando os contigs são maiores. Isso é explicado pelo fato de que ela usa os BBH's entre os genes das seqüências. Assim, quanto maior o contig, maior é número de genes presentes e conseqüentemente mais BBH's são detectados e avaliados no processo de ancoragem tornando seus resultados mais seguros. Outra razão para isso é que a presença de genes parálogos

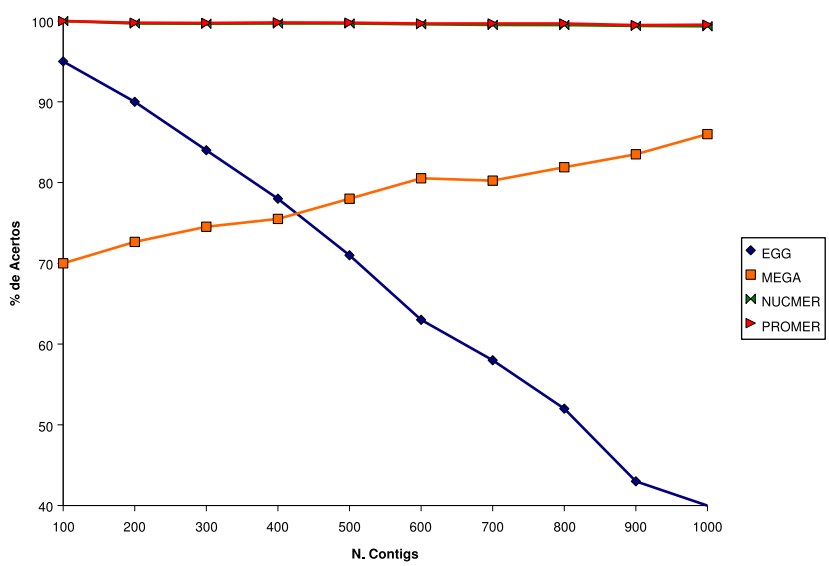


Figura 4.5: Resultados para $\alpha = 0.25$

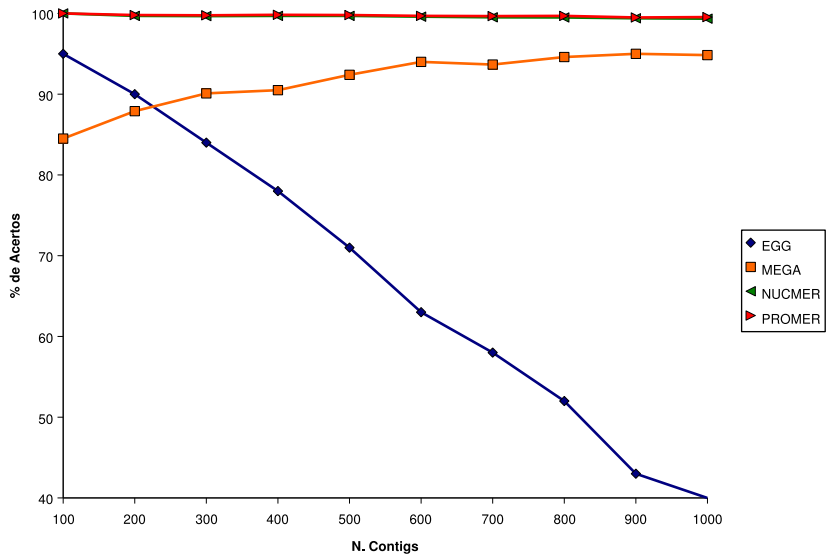


Figura 4.6: Resultados para $\alpha = 0.5$

podem atrapalhar a determinação correta dos BBH's.

Mega BLAST é a única das ferramentas que apresentou uma alteração mais acentuada em relação a variação de α. Observando os gráficos, verificamos que os acertos estão entre 83% e 97% quando $\alpha = 0.5$, entre 73% e 82% para $\alpha = 0.25$ e entre 60% e 79% para $\alpha = 0.1$. Porém em todos, o número de

acertos aumenta quando o tamanho do *contig* diminui. Acreditamos que pelo fato de Mega BLAST usar alinhamentos locais e a existência de repetições nas seqüências, falsas ancoragens foram encontradas pela ferramenta. O mesmo fato repercute no número de acertos maior quando o *contig* é pequeno, pois assim a chance de uma falsa ancoragem diminui pois com *contigs* menores repetições são mais difíceis de ocorrerem.

Capítulo 5

Ferramenta de Visualização das ancoragens

A visualização gráfica proporciona uma maneira rápida e fácil de apresentar os *contigs* ancorados no genoma completo. Optamos por usar a ferramenta GBrowse, desenvolvida pelo Projeto GMOD [1].

5.1 GMod

O projeto *GMOD-Generic Model Organism Database*, Base de Dados Genérico de Organismo Modelo, é financiado pelo *National Institute of Health*-NIH, e tem o objetivo de gerar componentes reutilizáveis para criação e manutenção de modelo de banco de dados de organismo (MOD) [23]. GMOD é uma coleção de aplicações e banco de dados interconectados que os biólogos usam como repositórios e como ferramentas.

Os MODs são ferramentas importantes para as pesquisas científicas. Elas compartilham um conjunto de tarefas em comum tais como: coletar e apurar dados a partir da literatura científica, integrar essas informações com os resultados de experimentos em grande-escala e mais recentemente, providenciar uma nomenclatura padrão de genes, termos anatômicos e outros elementos do vocabulário científico. Essas informações são disponibilizadas para a comunidade via uma página Web, que serve ainda como conexão para discussão, anúncios de interesse para a comunidade e submissão de dados [23]. Com relação às pesquisas em biologia molecular, é importante destacar que existe uma regra chave nos MODs que tenta relacionar as características genômicas

às características biológicas clássicas dos organismos. Os MODs contém regras interpretativas e apurativas que os distingue dos banco de dados que apenas armazenam as informações.

O NIH, observando o crescimento do número de importante projetos de seqüenciamento de genomas e visando gerenciar os dados resultantes desses, financiou o projeto GMOD, com o objetivo de construir ferramentas para construção e manutenção de MODs. Essas ferramentas estão disponibilizadas para a comunidade através do endereço `http://www.gmod.org`.

5.2 GBrowse

O GBrowse (*Generic Genome Browser*) é uma das ferramentas disponibilizadas pelo projeto GMOD e é uma combinação de base de dados e página Web interativa para manipular e indicar anotações e características em genomas. A ferramenta contém muitos recursos que outros *browsers* (navegadores *web*) disponibilizam, tem a vantagem de ter sido projetada para ser portável e extensível.

Os dados dos genomas a serem exibidos são fornecidos de duas maneiras para a ferramenta. A maneira mais fácil é através de um arquivo no formato GFF(*General Features Format*), descrito ainda nessa seção. O site do NCBI disponibiliza o arquivo GFF dos genomas lá depositados. A outra maneira de fornecer os dados para o GBrowse é através de um banco de dados para o qual a ferramenta tenha suporte, que hoje são PostGrees, MySQL, BioSQL e CHADO.

As anotações dos genomas são exibidas em forma de trilhas (*tracks*), que são linhas horizontais compostas por desenhos e textos. Existem vários formatos de desenhos para as trilhas que o usuário da ferramenta pode associar às características dos genomas. Algumas associações entre desenhos e características já são sugeridas na ferramenta. Para fazer a definição das trilhas a serem mostradas pelo GBrowse, precisamos criar um arquivo texto, com extensão `.conf`, com as configuraçãoes das trilhas. Na Figura 5.1 temos um exemplo de configuração de trilhas retirado do arquivo `.conf`. Nesse exemplo, temos a definição de duas trilhas, *oContigs_Anchors* e *Contigs_Anchors*. Cada uma das trilhas está sendo definida pelas palavras chaves `feature`, `glyph`, `height`, `bgcolor` e `Key`, onde:

- `feature`: determina a característica a ser mostrada (contig), e a fonte de tal característica (Ref). Essa palavra-chave está diretamente associada

aos dados do genoma armazenados no banco de dados ou no arquivo GFF

- `glyph`: determina o desenho associado à característica a ser mostrada na trilha. O tipo `generic` é um retângulo. A palavra-chave `stranded`, que aparece na definição da segunda trilha, com o valor 1 determina que uma seta deve aparecer em uma das extremidades do retângulo definido uma orientação para a característica.
- `height`: determina a altura do desenho.
- `bgcolor`: determina a cor de preenchimento do desenho.
- `Key`: determina o texto que aparece a esquerda da trilha na janela de exibição, como se fosse o identificador da trilha.

```
################## TRACK CONFIGURATION ####################
# the remainder of the sections configure individual tracks
###########################################################

[oContigs_Anchors:overview]
feature      = contig:Ref
glyph        = generic
height       = 5
bgcolor      = blue
Key          = Contigs

[Contigs_Anchors]
feature      = contig:Ref
glyph        = generic
stranded     = 1
height       = 10
bgcolor      = blue
Key          = Contigs
```

Figura 5.1: Trecho de um arquivo `.conf`, onde são configuradas as trilhas.

Outras palavras-chaves existem para complementar a definição das trilhas. Deve existir um arquivo `.conf` para cada genoma, ou para cada visualização a ser exibido na ferramenta.

O formato GFF(*General Feature Format*) foi criado com o objeivo de descrever, de maneira simples, as características encontradas nos genomas dos organismos. O formato GFF não tem a intenção de ser usado para um completo gerenciamento de dados de análise e anotação de seqüencias genômicas.

Formado linhas contendo 9 campos que são: o identificador da seqüência, a fonte, a característica, início, fim, pontuação, fita, frame e atributos. O identificador da seqüência indica a seqüência a qual se refere a característica. A fonte geralmente indica o nome do programa que fez a predição da característica. A característica indica o tipo da característica, existe um conjunto padrão de característica porém não é imposto o uso dessas. Início e fim indicam as respectivas posições dentro da seqüência onde inicia-se e finda a característica. Score é um número real que indica a pontuação com que o programa da fonte encontrou a característica. A fita indica a fita em que foi encontrada a característica, assim como o frame indica o frame de leitura quando pertinente à característica. A seguir temos o trecho inicial de um arquivo GFF.

```
##gff-version 3
##sequence-region   reference 1 5175554
reference       RefSeq  genome  1       5175554 .       +       .       ID=genome;Name=genome
reference       REF     contig  57332   97661   .       +       .       ID=c2;Name=c2
reference       REF     contig  108269  141983  .       -       .       ID=c3;Name=c3
reference       REF     contig  152527  193478  .       +       .       ID=c4;Name=c4
reference       REF     contig  204248  253174  .       -       .       ID=c5;Name=c5
reference       REF     contig  253816  296334  .       -       .       ID=c6;Name=c6
reference       REF     contig  333824  350385  .       +       .       ID=c7;Name=c7
reference       REF     contig  372654  398248  .       -       .       ID=c8;Name=c8
```

A extendibilidade do GBrowse é simples. Usando BioPerl, podemos criar novos adaptadores para banco de dados, ou novos agregadores responsáveis por gerar gráficos, e até mesmo plugins para novas formas de visualização, anotação e consultas das características dos genomas.

Existem vários tipos de gráficos e agregadores que possibilitam a visualização das características dos genomas de muitas formas. A configuração dessas visões é feita de modo simples permitindo uma exploração muito grande dos dados. Para detalhes basta acessar a tutorial do GBrowse em `http://www.gmod.org`.

5.2.1 Apresentação Visual da Ancoragem através do GBrowse

Nossa apresentação visual da ancoragem traz duas trilhas, compostas por *contigs* com as orientações e identificações visíveis. A primeira trilha, rotulada **Contigs**, apresenta a ancoragem esperada dos *contigs* na cor azul. A segunda, rotulada **Mapping**, apresenta a ancoragem dos *contigs* dada pelas ferramentas na cor preta. Na figura 5.2 apresentamos a ancoragem do conjunto de testes com 100 contigs de *Xhantomonas campestri* pv *campestris* (XCC) no genoma completo da *Xhantomonas anoxonopodis citri* (XAC) feita pela ferramenta

PROmer. Nessa figura destacamos a visão geral da ancoragem que está colocada na porção de ***Overview***, onde a representação da largura completa do genoma é mostrada. Podemos ver que na porção ***Details*** aparece apenas uma mensagem dizendo que a visualização dos detalhes é possível quando a visão for limitada a 1 Mbp.

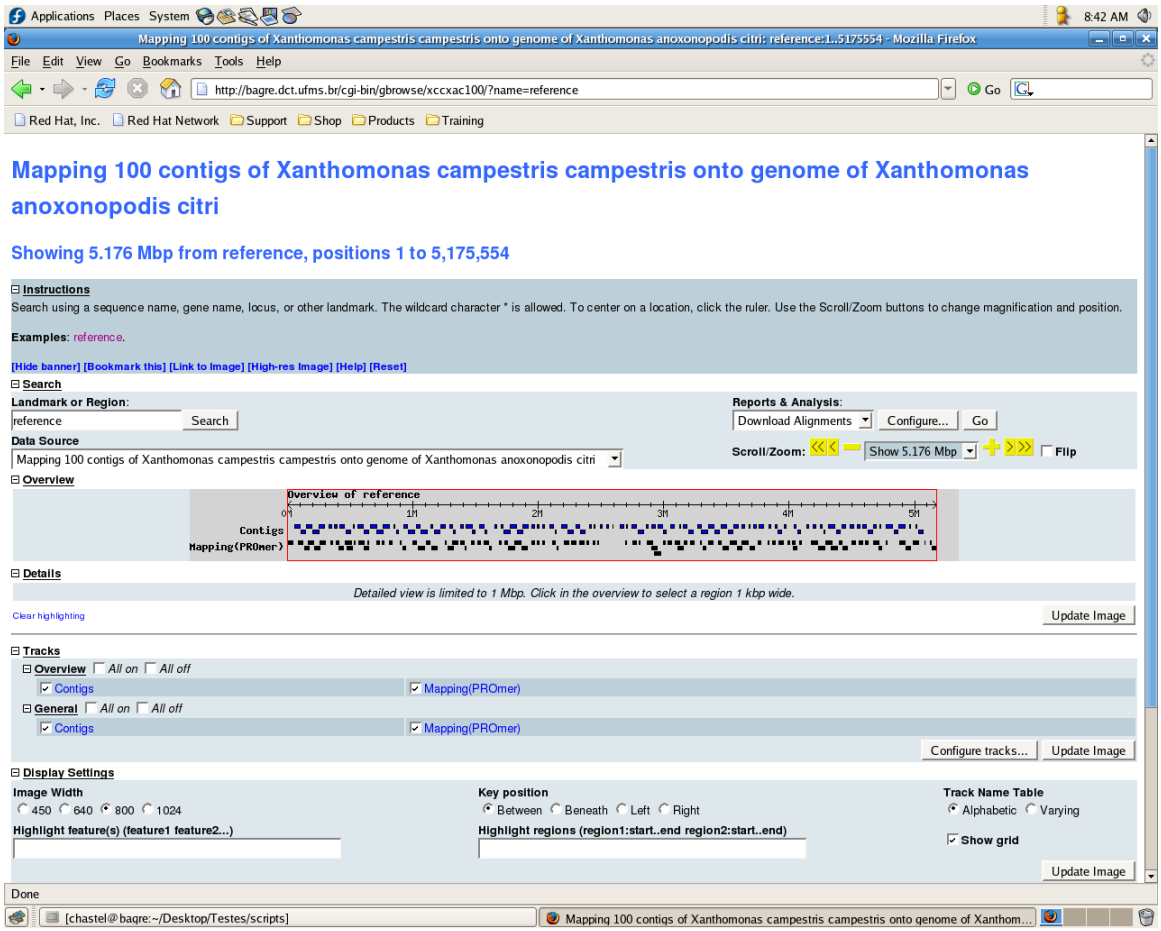


Figura 5.2: Visão Geral, usando GBrowse, da ancoragem de contigs de *Xhantomonas campestri campestris* no genoma completo da *Xhantomonas anoxonopodis citri* feita por PROmer. Destaque da porção *Overview*.

Na figura 5.3 apresentamos os detalhes da ancoragem. Na porção *Overview*

aparece um retângulo vermelho, que mostra a porção do genoma que está sendo detalhada. Observando a porção *Details*, identificamos as duas trilhas com as representaçãoes dos contigs, onde dos contigs são representados tamanho, orientação e posição de ancoragem. Os contigs em azul, e indentificados por c_i, estão colocados nas posições e orientações esperadas, enquanto que os contigs na cor preta e identificados por m_i estão nas posições e orientações definidas por PROmer. Nessa ancoragem podemos ver que os contigs m_i estão quase sempre colocados abaixo dos respectivos c_i, salvo um pequeno deslocamento para a direita, porém abaixo dos contigs c_{51} e c_{52} não há nenhum contig ancorado mas os respectivos foram ancorados próximos, m_{51} à esquerda de c_{48} e m_{52} à direita de c_{53} indicando que estão presentas no genoma da *Xhatomonas anoxonopodis citri* e que algum evento ocorreu nesse trecho do DNA.

Nas figuras 5.4 e 5.5 podemos observar outros eventos. Na primeira figura os contigs m_{81}, m_{82}, m_{83} e m_{85} de XCC estão ancorados na porção inicial de XAC. Na segunda figura os contigs m_{14}, m_{15}, e m_{16} de XCC estão ancorados na porção final de XAC. Podemos assim identificar inversões nas seqüências do DNA dos organismos. Essas inversões podem ser confirmadas pela Figura 5.6, retirada de [4], que mostra os BBH's dos genes desses dois organismos.

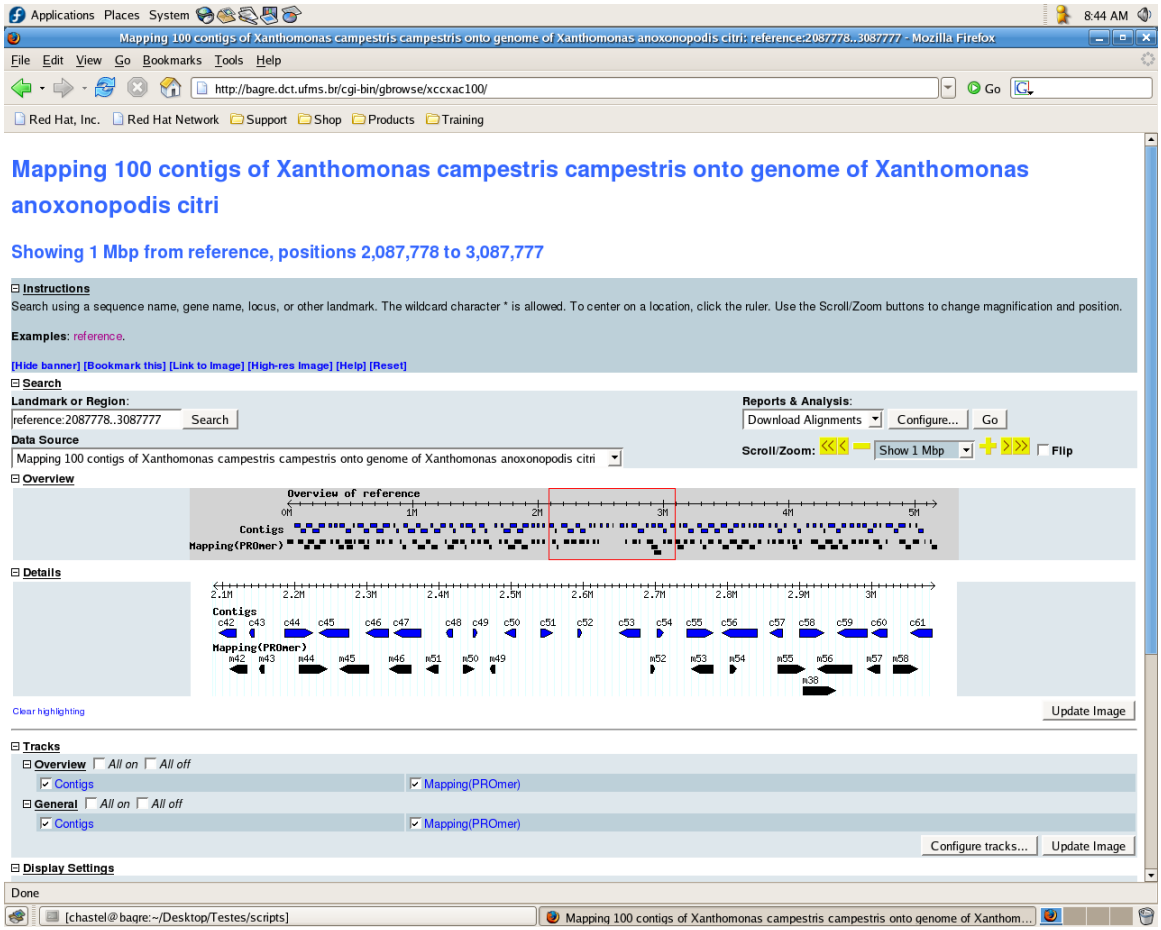


Figura 5.3: Visão dos detalhes da ancoragem.

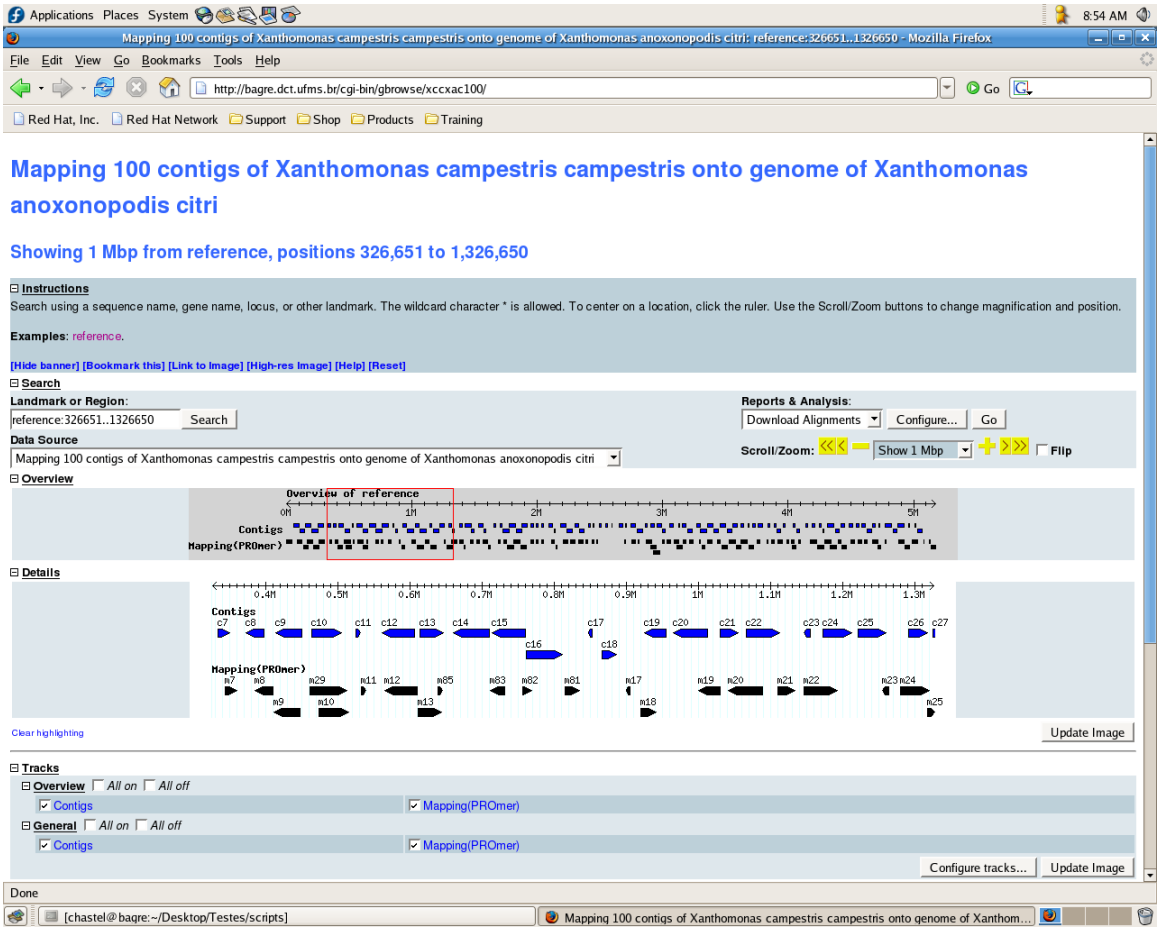


Figura 5.4: Visão dos detalhes da ancoragem. Os contigs m_{81}, m_{82}, m_{83} e m_{85} de XCC foram ancorados na porção inicial de XAC.

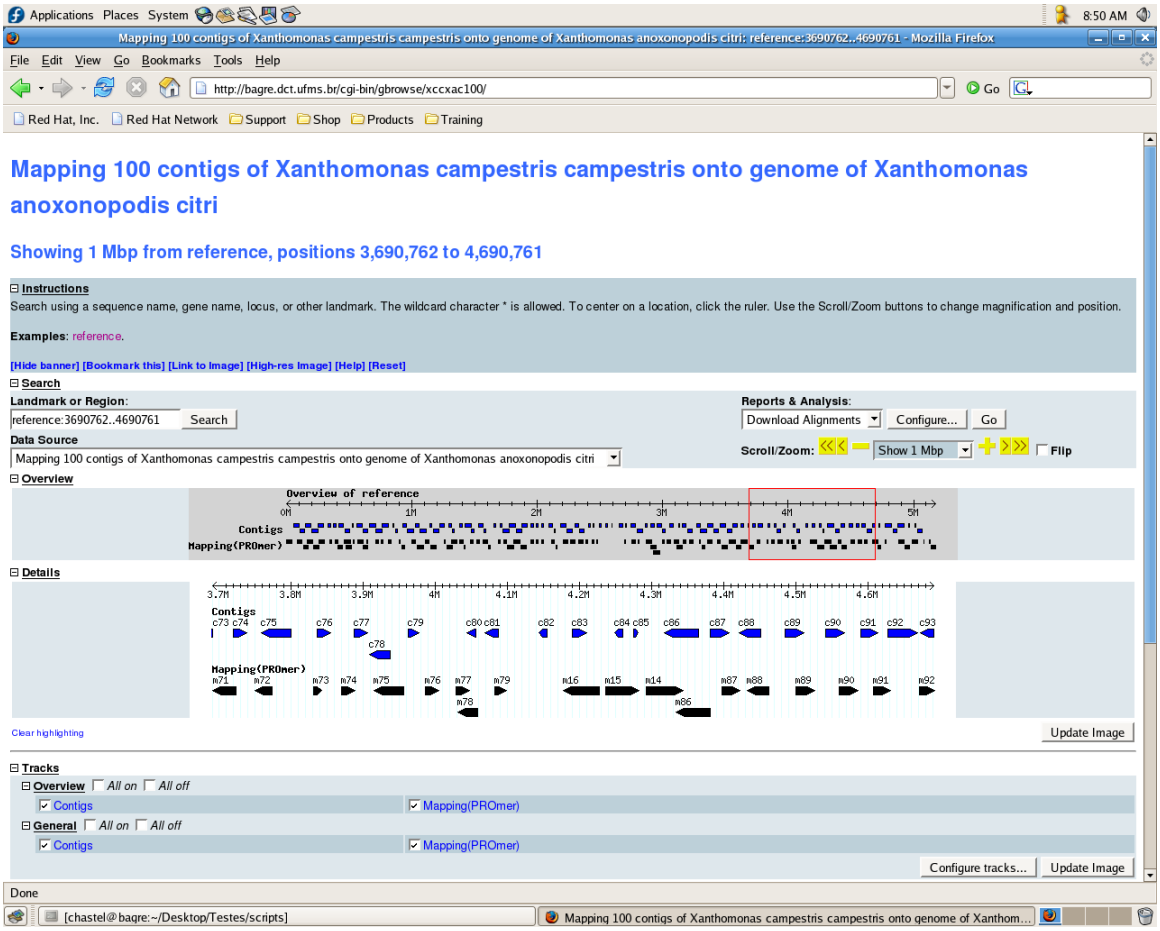


Figura 5.5: Visão dos detalhes da ancoragem. Os contigs m_{14}, m_{15}, e m_{16} de XCC foram ancorados na porção final de XAC.

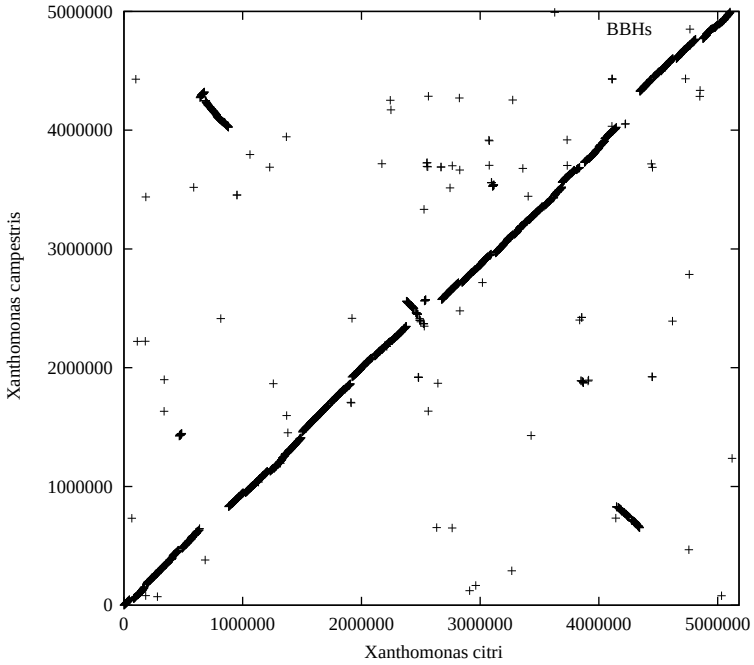


Figura 5.6: BBH's dos genes de XAC e XCC (Retirada de [4]).

Capítulo 6

Conclusão

Nosso trabalho apresenta um estudo de ferramentas de comparação genômica para o uso em ancoragem de contigs em genomas completos e tem como uma das contribiuções a determinação de uma medida que assegura a qualidade dessas ancoragens. A medida S_M, baseada na distância entre a posição de ancoragem esperada e a posição efetiva dada pela ferramenta, permite importantes conclusões a respeito das ferramentas e técnicas usadas.

As ferramentas PROmer e NUCmer alcançaram excelentes resultados no processo de ancoragem, com uma pequena vantagem para PROmer, o que nos permite afirmar que o uso da árvore de sufixos de MUMmer na identificação de *matches* idênticos maximais é de extrema relevância no processo e o uso da cadeia de aminoácidos foi determinante na vantagem detectada em PROmer. Na ancoragem de todos os casos de testes tanto PROmer quanto NUCmer ficaram acima dos 99% de acerto, a vantagem de PROmer sempre foi pequena, por exemplo, como os conjuntos de 1000 contigs NUCmer obteve $99,35\%$ e PROmer $99,55\%$.

A ferramenta Egg e a técnica de ancoragem baseada em BBH's também alcançou bons resultados quando aplicada sobre *contigs* longos, porém com *contigs* curtos seus resultados foram os piores. Isso devido ao fato de quanto maior o contig é provável um maior número de genes o que pode gerar um maior número de BBH's com o genoma completo. Com um número maior de BBH's a ancoragem é mais precisa. Dessa forma Egg mostra-se uma excelente ferramenta para projetos que estejam quase finalizados.

Mega BLAST obteve resultados sempre acima de 60% de acertos, sendo mais eficiente quando os *contigs* são mais curtos, porém acreditamos que seus resultados podem melhorar ao aumentarmos o tamanho dos *matches* idênticos que

são as sementes dos alinhamentos locais que determinam sua ancoragem.

A visualização gráfica das ancoragens através do uso da ferramenta GBrowse, outra contribuição desse trabalho, também mostra-se muito útil para uma melhor análise de como os contigs se alinharam ao genoma completo.

Ao fim desse trabalho confirmamos as análises iniciais apresentadas em [5] e [6], que já indicavam um melhor desempenho de PROmer com relação as outras ferramentas. Porém, se o conjunto de *contigs* ao qual se deseja ancorar for composto por *contigs* longos e anotados a melhor opção é usar a ancoragem de Egg. Isso acontece, em geral, em projeto em fase final e com poucos trechos não montados.

Resultados detalhados, ferramentas e as ancoragens definidas pelas ferramentas estão disponíveis no endereço http://www.egg.dct.ufms.br.

Trabalhos Futuros

Entendemos que muita coisa ainda pode ser feita para melhorar as metodologias e ferramentas aqui propostas e o detalhamento dos seus resultados. Listamos a seguir algumas dessas possibilidades.

Agilizar a ancoragem baseada nos BBH's

A técnica de ancoragem baseada nos BBH's implementada precisa deslocar cada um dos contigs através de todo o genoma âncora, nucleotídeo a nucleotídeo, isso deixa o processo muito lento. Por exemplo, para fazer a ancoragem dos 100 contigs de *Xanthomonas campestris* pv *campestris* no genoma da *Xanthomonas anoxonopodis citri* foram necessárias pouco mais que 24 horas. Acreditamos que seja possível determinar a posição correta de ancoragem sem a necessidade desse deslocamento por todo o genoma âncora, para isso é preciso fazer uma formulação matemática que possa comprovar tal suposição.

Duas possíveis solução são:

- Usando uma aboradagem de divisão e conquista podemos dividir o genoma em três partes iguais, colocamos o contig no centro de cada porção do genoma, calculamos distância entre o contig e o genoma e descartamos a porção com a maior distância. Aplicamos, recursivamente, esse mesmo processo com as duas partes do genoma que não foram descartadas. Não

garatimos a otimalidade das ancoragens com essa aboradagem porém nossa complexidade, que atualmente é $O(n)$, será dada por

$$T(n) = 3k + T(\tfrac{2n}{3}),$$

o que resulta em $O(log_3 2n)$.

- Ao invés de percorrer toda a extensão do genoma, percorrer apenas a porção do genoma entre os BBH's extremos do contig. Com essa heurística garantimos a otimalidade da ancoragem mas não garantimos um ganho de tempo.

Revisar as ancoragens de Mega BLAST

Mega BLAST, assim como PROmer e NUCmer, também procura por um match idêntico, porém não se preocupa em usar o maximal. Em nossos testes usamos o tamanho padrão para o match, 28. Acreditamos que aumentando esse tamanho os resultados possam ser melhorados, porém não devem alcançar os mesmos de PROmer e NUCmer.

Ampliar ancoragem de Egg

Estudar a possibilidade de usar, além dos BBH's, outras estruturas genômicas para fazer a ancoragem e assim aumentar o percentual de acertos quando os *contigs* forem menores.

Adaptar EggView para apresentar ancoragens

EggView [6] é uma ferramenta desenvolvida sobre o GBrowse para visualização das estruturas detectadas por Egg na comparação entre dois genomas completos, entre as estruturas apresentadas estão os BBH's e os genes exclusivos. A apresentação dessas estruturas numa ancoragem serão de grande importância para a extração de informações relevantes ao projeto.

Referências Bibliográficas

[1] http://gmod.org, Junho 2007.

[2] http://mummer.sourceforge.net/manual, Janeiro 2007.

[3] http://www.ncbi.nlm.nih.gov/genomes/static/gpstat.html, Maio 2007.

[4] N. F. Almeida. *Ferrmanetas para comparação genômica*. Tese de Doutoramento, IC-UNICAMP, 2002.

[5] N. F. Almeida, A. L. Chastel, S. S. Adi, C.J. Viana, M. Y. Nakazaki, A. Tamura, L. Hiratsuka, e L. Brazil. Mapping contigs onto reference genomes. *BSB07'LNBI 4643*, páginas 155–158, August 2007. Marie-France Sogot and Maria Emília T-Walter, Springer-Verlage Berlim Heidelberg 2007.

[6] N. F. Almeida, M. Y. Nakazaki, A. Tamura, L. Hiratsuka, A. L. Chastel, S. S. Adi, C.J. Viana, , e L. Brazil. EGGview: Visualization of EGG compartive date using gbrowse. *Proccedings of BSB'07*, August 2007. ISBN 978-85-7669-123-5.

[7] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, e D. J. Lipman. A basic local alignmentsearch tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[8] G. S. Araújo. Filogenia de proteomas. *Dissertação de Mestrado, DCT-UFMS*, 2003.

[9] A. C. R. da Silva, J. C. Setubal, e N. F. Almeida et al. Comparison of the genomes of two xanthomonas pathogens with differing host specifities. *Nature*, 417:459–463, 2002.

[10] A. L. Delcher, S. Kasif, R. D. Fleischmann, J. Peterson, O. White, e S. L. Salzberg. Alignment of whole genomes. *Nucleic Acids Research*, 27:2369–2376, 1999.

[11] A. L. Delcher, A. Phillippy, J. Carlton, e S. Salzberg. Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Research*, 30:2478–2483, 2002.

[12] M. S. Felipe, M. E. Walter, M. M. Brígido, e N. F. Almeida et al. Transcriptome characterization of the dimorphic and pathogenic fungus paracoccidioides brasiliensis by est analysis. *Yeast*, 20:263–271, 2003.

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

[14] S. Kurtz, A. Phillippy, A. L. Delcher, M. Smoot, M. Shumway, C. Antonescu, e S. L. Salzberg. Versatile and open software for comparing large genomes. *Genome Biology*, 5:R12, 2004.

[15] C. Monteiro-Vitorello, L. E. A. Camargo, e N. F. Almeida et.al. The genome sequence of the gran-positive sugarcane pathogen leifsonia xyli subsp. xyli. —, 2003.

[16] L. Montera. Regiões ortólogas em múltiplos genomas. *Dissertação de Mestrado, DCT-UFMS*, 2004.

[17] L. M. Moreira, R. F. de Souza, N. F. Almeida, J. C. Setubal, J. C. F. Oliveira, L. R. Furlan, J. A. Ferro, e A. C. R. da Silva. Comparative genomics analyses of citrus-associated bacteria. *Annu. Rev. Phytopathology*, 42:163–184, 2004.

[18] V. K. Okura. Bioinformática de projetos genoma de bactérias. *Dissertação de Mestrado, IC-UNICAMP*, 2002.

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

[20] J. C. Setubal e R. F. Werneck. A program for building contig scaffolds in double-barrelled shotgun genome sequencing. 2004.

[21] M. A. Van Sluys, J. C. Setubal, e N. F. Almeida et al. Comparative analyses of the complete genome sequences of pierce's disease and citrus variegated chlorosis strains of xylella fastidiosa. *J. Bacteriology*, 185:1018–1026, 2003.

[22] T. Smith e M. Waterman. Identification of common molecular subsequences. *Journal of Moleular Biology*, 147:195–197, 1981.

[23] L. D. Stein, C. Mungall, S. Q. Shu, M. Caudy, M. Mangone, A. Day, E. Nickerson, J. E. Stajich, T. W. Harris, A. Arva, e S. Lewis. The generic genome browser: A building block for a model organism system database. *Genome Research*, 12:1599–1610, 2002.

[24] G. P. Telles, M. M. Brigido, N. F. Almeida, C. J. M. Viana, D. A. S. Anjos, e M. E. M. T. Walter. A method for comparing three genomes. *BSB05*, LCBI 3594:160–169, 2005.

[25] D. W. Wood, J. C. Setubal, e N. F. Almeida et al. The genome of the natural genetic engineer agrobacterium tumefaciens c58. *Science*, 294 (5550):2317–2323, 2001.

[26] Z. Zhang, S. Schwartz, L. Wagner, e W. Miller1. A greedy algorithm for aligning dna sequences. *Journal of Computational Biology*, Volume 7(1/2):203–214, 2000.