Ferramentas de Auxílio ao Seqüenciamento de DNA por Montagem de Fragmentos: um estudo comparativo

Said Sadique Adi

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO GRAU DE MESTRE
EM
CIÊNCIA DA COMPUTAÇÃO

Área de Concentração: Ciência da Computação
Orientador: Prof. Dr. Carlos Eduardo Ferreira

– Durante o desenvolvimento deste trabalho, o autor recebeu apoio financeiro da CAPES –

— São Paulo, março de 2000 —

Ferramentas de Auxílio ao Seqüenciamento de DNA por Montagem de Fragmentos: um estudo comparativo

Este exemplar corresponde à redação
final da dissertação devidamente corrigida
e defendida por Said Sadique Adi
e aprovada pela comissão julgadora.

São Paulo, 2 de maio de 2000.

Banca examinadora:

- Prof. Dr. Carlos Eduardo Ferreira (orientador) (IME-USP)
- Prof. Dr. Sérgio Russo Matioli (IB-USP)
- Prof. Dr. João Meidanis (IC-UNICAMP)

Aos meus irmãos Ashjan,
Kaher e Sandro.

Agradecimentos

Em primeiro lugar, gostaria de agradecer ao meu orientador, professor Carlos Eduardo Ferreira, pelo apoio direto durante o desenvolvimento desta dissertação, assim como durante a fase inicial de meu mestrado. Sem a sua atenção e palavras de ânimo o término deste trabalho seria impensável.

Agradecimentos do fundo da alma ofereço aos meus pais que, mesmo estando a quilômetros de distância, nunca me deixaram desamparado. Muito obrigado pai pelas palavras de conforto a cada telefonema...muito obrigado mãe por toda a preocupação que sempre demonstrastes por mim.

Aos amigos...bom, a eles apenas agradecimentos não bastam. De qualquer forma, meu muito obrigado aos amigos Ricardo e Uirá pelo clima de descontração no instituto e pelos futebóis que pudemos jogar e assistir juntos. Muito obrigado Fábio, Marco, Eliany e Luiz por não me deixarem sentir tão distante dos amigos de Campo Grande e por terem me ajudado em muito durante a fase de adaptação a esta cidade. Um agradecimento em especial ao amigo Raphael que, além de ótimo companheiro de "república", é também um excelente revisor de textos :).

Com certeza, nunca conseguiria expressar em uma única página, ou várias delas, os meus agradecimentos a todos aqueles que, de uma forma ou de outra, me acompanharam durante o desenvolvimento deste trabalho. Por isso, peço desculpas aos que, por falta de espaço (ou descuido mesmo), não foram citados.

Said Sadique Adi

Resumo

Atualmente, existe um grande número de ferramentas para montagem de fragmentos de DNA disponíveis. Neste trabalho apresentamos um estudo comparativo das ferramentas **CAP2**, **FAKtory**, **TIGR** e **PHRAP**. Para realizarmos este estudo, primeiramente executamos esses sistemas de montagem sobre doze casos de testes distintos. Após isso, tomamos os resultados obtidos por cada um deles e os comparamos com as seqüências de onde os fragmentos foram originalmente obtidos. Os testes utilizados avaliam a eficiência dos programas com relação a três problemas associados ao processo de montagem (erros no seqüenciamento, fragmentos quimeras e regiões repetidas) e pudemos ver que nenhum dos sistemas é claramente superior aos demais no tratamento de todos eles. Cada ferramenta de montagem parece tratar de melhor forma um problema em especial. Além de avaliarmos os resultados, realizamos também um estudo comparativo do tempo de CPU gasto por cada uma das ferramentas durante suas execuções.

Abstract

Nowadays, several packages for DNA fragment assembly are available. In this work we present a comparative study of the performances of the programs **CAP2**, **FAKtory**, **TIGR** e **PHRAP**. To get to our objectives, we first ran each of these programs on twelve instances. After this, we compared the outputs with the sequences from which the fragments were originally obtained. In this comparison, we took into consideration three problems related to fragments assembly (sequencing errors, chimeric fragments and repeats regions). We conclude that no one of the packages we tested is more efficient than the others when considering all the problems cited above. If we consider a particular problem, then we observed different performances among the programs. Even more, we compare the packages with respect to theirs CPU times.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Talvez a Biologia seja hoje uma das ciências que desperta maior interesse e, ao mesmo tempo, maior número de incertezas nos pesquisadores. Parte destas incertezas é decorrente da falta de um conjunto adequado de técnicas e ferramentas capaz de manipular uma grande quantidade de dados e gerar resultados que auxiliem no desvendamento de alguns dos segredos da vida. Na tentativa de suprir esta falha, a Ciência da Computação se juntou à Biologia determinando o surgimento de uma nova área de pesquisa, conhecida como **Biologia Computacional**.

Um dos problemas tratados pela área de Biologia Computacional e do qual tem-se falado muito nos últimos anos corresponde ao problema do seqüenciamento de DNA, que tem por objetivo determinar todas as bases componentes dessa molécula. Após o descobrimento do DNA em 1869, por Johann Miescher, vários cientistas concentraram seus esforços na determinação da composição, estrutura e funções desse ácido dentro das células. Estes estudos lançaram uma luz sobre muitas questões a respeito das características físicas dos indivíduos e como a hereditariedade se processa nos organismos vivos. Além disso, descobriu-se também uma estreita relação entre as bases componentes do DNA e os aminoácidos que compõem as proteínas produzidas pelas células, descoberta esta que levou à percepção dos grande benefícios que o conhecimento da seqüência daquela molécula poderia trazer.

Vários métodos passaram então a ser criados no intuito de determinar, com a maior exatidão possível e de forma eficiente, as bases componentes das moléculas de DNA. De início, estes métodos de seqüenciamento utilizavam-se basicamente de processos químicos para realização de suas tarefas e permitiam apenas o seqüenciameno de pequenas cadeias de DNA ou de fragmentos obtidos de extensas moléculas. A tarefa de seqüenciamento total de longas cadeias (como por exemplo aquelas componentes do genoma humano contendo, aproximadamente, 3 bilhões de bases) começou a ser posta em prática somente na última década, com o desenvolvimento de métodos mais poderosos

e que necessitam de grande processamento computacional para atingir seus objetivos. Um destes métodos é conhecido como **Método Shotgun de Seqüenciamento**, que corresponde ao mais utilizado atualmente e um dos principais objetos de estudo neste trabalho.

O método *Shotgun* de seqüenciamento possui como passo inicial a clonagem da molécula a ser determinada. No segundo passo, estes clones são quebrados em vários fragmentos, que terão então suas bases determinadas através de algum método tradicional de seqüenciamento. O último passo do método *Shotgun* realiza a montagem dos fragmentos, já seqüenciados, obtidos na fase anterior de forma a reconstituir a seqüência da molécula original. Esta última fase necessita de uma quantidade considerável de processamento computacional, utilizando-se de ferramentas de montagem que recebem como entrada as seqüências de bases relacionadas aos fragmentos e devolvem como saída as bases componentes de toda a seqüência original ou de partes dela.

Atualmente existe uma série de ferramentas de montagem disponíveis, que acabam sendo utilizadas como "caixas-pretas" pelos biólogos no sentido de que a seqüência final gerada por elas é sempre tomada como correta e posteriormente conferida frente a seqüências similares. Esta prática acaba sendo condenada ao considerarmos que existem disparidades entre os resultados gerados por cada sistema de montagem ao serem aplicados sobre o mesmo conjunto de dados, o que nos leva a discutir qual deles obtém uma resposta mais próxima à seqüência do DNA original. A diferença entre os resultados produzidos por cada uma das ferramentas está relacionada com a forma que os programas lidam com o problema de montagem e com certas falhas que ocorrem durante a obtenção e seqüenciamento dos fragmentos. Dentre elas, podemos citar erros no seqüenciamento dos fragmentos, falta de cobertura total da molécula de DNA e inexistência de sobreposições adequadas entre os fragmentos. Cada uma destas falhas dificulta a recomposição exata do DNA original, além de interferir também no desempenho de processamento de cada ferramenta, outra questão merecedora de discussão.

Nosso objetivo neste trabalho é avaliar, de forma empírica, quatro ferramentas de montagem (**CAP2**, **PHRAP**, **TIGR**, **FAKtory**) disponíveis atualmente. Através da comparação dos resultados gerados por cada uma delas, entre si e com a seqüência original sendo montada, tentaremos determinar qual(is) da(s) ferramenta(s) melhor se adapta(m) às exigências dos biólogos e consegue(m) tratar da melhor forma alguns dos problemas que podem surgir durante a tarefa de determinação das bases componentes de longas cadeias de DNA. Como os projetos de seqüenciamento manipulam um número cada vez maior de fragmentos, além da exatidão dos resultados gerados por cada sistema de montagem, faremos também um estudo comparativo do tempo de CPU ocupado por eles durante a tarefa de leitura e montagem de fragmentos pertencentes a instâncias de tamanhos variados.

O restante desta dissertação é dividida em sete capítulos. No capítulo 2 daremos

uma breve descrição do DNA e falaremos também a respeito do problema de seqüenciamento desta molécula, enumerando suas motivações e descrevendo alguns métodos utilizados para resolvê-lo, como o método *Shotgun*. Por último, esse capítulo inclui uma definição formal do problema da montagem de fragmentos sobrepostos e enumera algumas dificuldades que podem surgir durante seu tratamento. No capítulo 3 daremos uma visão geral dos algoritmos utilizados por cada uma das ferramentas citadas anteriormente e descreveremos também a forma com que elas tratam alguns dos problemas relacionados ao processo de montagem. No capítulo 4 comentaremos a respeito de três ferramentas auxiliares utilizadas por nós durante o desenvolvimento do projeto. Os capítulos 5 e 6 incluem uma descrição detalhada dos resultados obtidos por cada um dos sistemas de montagem e, finalmente, o capítulo 7 encerra a dissertação com uma conclusão do trabalho.

Capítulo 2

Noções Preliminares

Neste capítulo falaremos a respeito de alguns conceitos básicos relacionados ao seqüenciamento de DNA, descrevendo a estrutura dessa molécula, definindo formalmente o problema, enumerando suas motivações e comentando a respeito de alguns métodos utilizados para resolvê-lo, como o método *Shotgun* de seqüenciamento. Antes disso, porém, daremos uma visão bem geral desta nova área da Ciência da Computação conhecida como Biologia Computacional, que inclui uma gama considerável de problemas.

2.1 Um Pouco de Biologia Computacional

Com o avanço dos estudos dentro da Biologia, em especial da Biologia Molecular, uma série de problemas envolvendo conjuntos consideráveis de dados vieram à tona. Além de se mostrarem numerosos, estes dados nem sempre se encontravam bem comportados e exigiam uma elevada quantidade de processamento, fatores estes que impediam qualquer tentativa manual de resolução dos problemas a que estavam relacionados. Várias técnicas e ferramentas computacionais passaram então a ser desenvolvidas e aplicadas sobre estes problemas no intuito de resolvê-los de forma eficiente, dando origem a uma nova área da Ciência da Computação denominada de **Biologia Computacional**[30], área dentro da qual este trabalho se insere.

Apesar de se tratar de um campo de pesquisa relativamente novo, já são inúmeros os trabalhos desenvolvidos dentro da área de Biologia Computacional, com cada um deles tratando dos mais variados problemas, como por exemplo:

- Comparação de seqüências: tarefa de determinar partes semelhantes entre duas ou mais seqüências.

- Seqüenciamento de DNA (montagem de fragmentos): tarefa de determinar as bases componentes de uma molécula de DNA a partir da montagem de fragmentos sobrepostos provenientes de clones desta molécula.
- Predição da estrutura secundária do RNA: tarefa de determinar a estrutura secundária de uma molécula de RNA com base em sua seqüência de nucleotídeos.
- Predição da estrutura terciária de proteínas: tarefa de determinar a estrutura terciária de uma proteína com base em sua seqüência de aminoácidos.
- Reconstrução de árvores filogenéticas: tarefa de construir árvores de filogenia que permitam deduzir a evolução genética de um dado organismo.

Vários algoritmos têm sido formulados na tentativa de resolver os problemas acima citados e, apesar da NP-completude de alguns deles, os resultados vêm sendo bastante gratificantes. A grande maioria dos algoritmos desenvolvidos até o momento consegue realizar sua tarefa em tempo satisfatório e, mais importante que isso, devolve como saída respostas próximas àquelas esperadas pelos biólogos.

Neste projeto focalizaremos nossa atenção sobre o problema de seqüenciamento de DNA por montagem de fragmentos, que será abordado de forma minuciosa nas próximas seções. Além disso, falaremos também um pouco mais a respeito do problema de comparação de seqüências que, de certa forma, está relacionado com uma das fases do processo de montagem a ser vista.

2.2 O Problema do Seqüenciamento de DNA

Dentre os problemas acima mencionados, o problema do seqüenciamento de DNA talvez seja aquele que vem despertando o maior interesse nos cientistas nos últimos anos, o que se reflete no grande número de projetos de seqüenciamento já concluídos e que ainda se encontram em andamento.

Antes de definirmos formalmente o problema do seqüenciamento e descrevermos suas motivações, falaremos um pouco a respeito do DNA, enumerando os elementos que constituem sua estrutura e as funções desta molécula dentro da célula.

2.2.1 A molécula de DNA

O **DNA** (do inglês *desoxyrribonucleic acid*) foi descoberto em 1869 pelo bioquímico alemão Johann Miescher[5], porém somente em 1912 é que os componentes desta molécula foram totalmente desvendados.

Utilizando-se de termos químicos, uma molécula de DNA é composta de unidades básicas denominadas nucleotídeos, que por sua vez constituem-se de uma molécula de acúcar com cinco carbonos (pentose), um fosfato, presente na forma de ácido fosfórico e uma base nitrogenada, que pode ser a **Adenina**, a **Citosina**, a **Guanina** ou a **Timina**.

Quarenta e um anos após a determinação dos componentes da molécula de DNA, James Watson e Francis Crick apresentaram um modelo para a estrutura desta molécula. Segundo estes dois cientistas, a molécula de DNA seria constituída de duas cadeias de nucleotídeos dispostas em hélice ao redor de um eixo imaginário. Estas duas cadeias se manteriam unidas por meio de pontes de hidrogênio que se estabelecem entre bases específicas da molécula (Adenina com Timina e Citosina com Guanina). Estas bases são denominadas **bases complementares** e correspondem à unidade de medida mais comumente utilizada para determinação do tamanho de uma molécula de DNA, possuindo como abreviação as letras **bp**. Além da estrutura, Watson e Crick estabeleceram também uma orientação específica para cada uma das fitas componentes do DNA. De acordo com o modelo proposto por eles, as duas cadeias seriam antiparalelas, ou seja, as ligações entre os seus nucleotídeos estariam orientadas no sentido $5' \rightarrow 3'$ (do carbono $5'$ de um nucleotídeo ao carbono $3'$ do nucleotídeo adjacente) em uma delas, enquanto que na outra a orientação seria inversa, de $3' \rightarrow 5'$. Na figura abaixo representamos a composição, a estrutura e as orientações relacionadas a uma molécula de DNA.

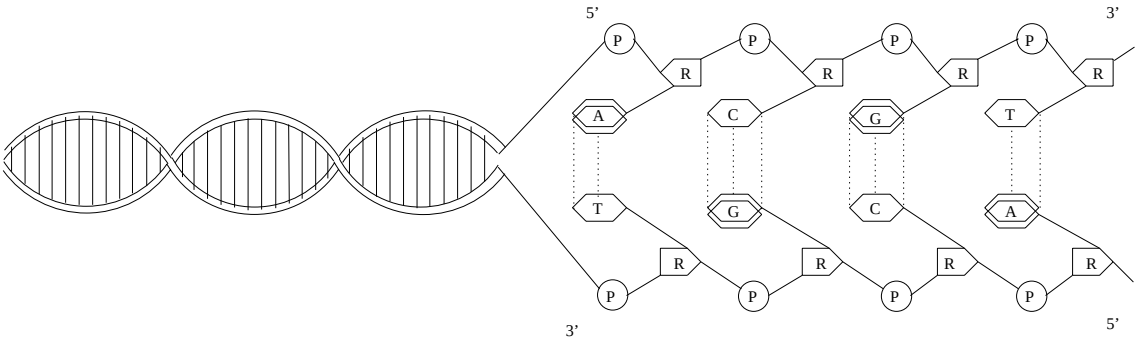

Figura 2.1: Uma representação das duas fitas componentes de uma molécula de DNA

Apesar da composição simples da molécula de DNA, descobriu-se mais tarde que nela estavam contidas todas as informações genéticas de um dado indivíduo que determinam suas características e de seus descendentes. Estas informações são copiadas ou transcritas nas moléculas de **RNA** (um ácido ribonucléico, que possui a base Uracila ao invés da base Timina em sua composição) e servirão como o "código" para a especificação dos aminoácidos que irão constituir as proteínas produzidas pelas células de um dado organismo.

Descrevendo de forma mais detalhada o processo de síntese de proteínas pelas células (também conhecido como **Dogma Central da Biologia**), antes da transcrição das

informações presentes no DNA para o RNA, as duas fitas complementares da primeira molécula se separariam por meio da quebra de suas pontes de hidrogênio. A partir daí, uma dessas fitas daria origem a um RNA (conhecido como **RNA-mensageiro**) transcrevendo neste o código de suas bases nitrogenadas. Esta transcrição não é literal, mas indireta, com cada base do RNA correspondendo a uma base complementar na fita de DNA da qual se originou.

Produzido o RNA-mensageiro, outro tipo de RNA entra em ação: o **RNA-transferidor** ou **RNA-transportador**. Esse RNA possui em uma de suas extremidades a seqüência CCA de bases, que é o local onde se encaixa um dado aminoácido. Em outro setor desta molécula existe um grupo diferente de três bases, que age como grupo de "reconhecimento" do aminoácido que será ligado às bases CCA do RNA. Encontrado o aminoácido específico, o RNA-transportador já está em condições de procurar ao longo do RNA-mensageiro uma seqüência de três bases (denominada de **códon**) que se complementa perfeitamente com o seu terceto do "grupo de reconhecimento de aminoácidos". Encontrada esta seqüência, o RNA-transportador encaixa-se então num ponto exato do RNA-mensageiro e o aminoácido que ele carrega liga-se ao códon correspondente por meio de estruturas celulares denominadas ribossomos, compostas por um terceiro tipo de RNA, denominado **RNA ribossômico**. Da mesma forma, outros RNAs-transportadores irão depositar seus aminoácidos ao longo da cadeia do RNA-mensageiro, formando as proteínas necessárias ao funcionamento do organismo. Vale salientar que nem toda a cadeia do DNA especifica proteínas, mas apenas partes delas conhecidas como **genes**. É exatamente nestas regiões da molécula que estão armazenadas todas as informações genéticas de um dado organismo, como cor dos olhos, espessura dos fios de cabelo etc.[12]

2.2.2 O problema e suas motivações

Da descrição acima, podemos concluir que quem comanda todo o processo de síntese de proteínas pelas células é o DNA através de seu RNA-mensageiro. Conhecendo-se então as bases componentes dessa molécula, principalmente aquelas que constituem seus genes, podemos obter informações valiosas a respeito da funcionalidade de cada proteína produzida, da evolução histórica da molécula e, mais ainda, identificar mutações em sua seqüência que, porventura, venham a ocasionar alguma doença de fundo hereditário no organismo da qual ela provém. Tudo isto tem motivado os cientistas a tentar resolver o **Problema do Seqüenciamento de DNA**, que corresponde exatamente à tarefa de determinar as bases componentes de uma determinada molécula. Para isso, vários métodos de seqüenciamento têm sido desenvolvidos, com alguns deles exigindo grande esforço computacional, como será visto na próxima seção.

Desde o seqüenciamento completo do primeiro organismo procarionte (bactéria *Haemophilus influenzae*, com aproximadamente 1830.10^3 bases) em 1995[16], já chega a

algumas centenas o número de seres vivos cujas bases componentes de suas moléculas de DNA estão totalmente determinadas, e vários outros projetos de seqüenciamento encontram-se atualmente em andamento. Talvez o mais importante deles seja o **Projeto Genoma Humano**[3], que se propõe a mapear todas as seqüências dos genes presentes no DNA humano, composto de, aproximadamente, três bilhões de bases. Outro projeto que merece destaque é o **Programa Genoma FAPESP**[2], que envolve vários laboratórios brasileiros na tarefa de seqüenciamento do DNA de vários organismos, dentre eles a *Xylella fastidiosa*[1]. Esta bactéria tem atacado de forma devastadora os laranjais de São Paulo, bloqueando o xilema da planta e prejudicando a circulação de sua seiva. Através da análise das bases que compõem o DNA da *Xylella*, os cientistas tentarão encontrar alguma forma de eliminar esta praga, conhecida como "praga do amarelinho".

2.3 Métodos de Seqüenciamento

Nos últimos anos, foram desenvolvidos vários métodos que permitem o rápido seqüenciamento de pequenas moléculas de DNA ou de partes de uma grande molécula. Um dos métodos mais tradicionais denomina-se **Método de Terminação de Cadeia de Sanger**[12], que se baseia na composição de fragmentos de DNA de diferentes comprimentos e constitui-se de três fases distintas: primeiramente, um conjunto de fragmentos, que podem ser vistos como pedaços da molécula contendo de 600 a 700 bases aproximadamente, é obtido de um clone do DNA a ser seqüenciado. Cada um destes fragmentos inicia-se num único ponto daquela molécula e termina em bases específicas dela. Após isso, os fragmentos são separados de acordo com seus tamanhos e, por último, determinam-se as bases finais de cada um deles. Quando ordenadas pelo tamanho dos fragmentos que elas terminam, essas bases finais provêm a seqüência do maior fragmento obtido da molécula. A Figura 2.2 mostra de forma mais detalhada as três etapas do Método de Terminação de Cadeia de Sanger. O DNA é copiado utilizando-se de uma enzima denominada *polimerase*, que atua sobre uma das fitas da molécula à qual está aderida uma seqüência inicial denominada *primer*. A *polimerase* catalisa o alongamento desta seqüência adicionando a ela bases complementares àquelas componentes do DNA sendo copiado. Este processo ocorre até que o *primer* fique do tamanho da molécula ou até que sejam adicionados dideoxinucleotídeos nesta seqüência, que nada mais são que nucleotídeos modificados que impedem a ligação de outros nucleotídeos no *primer*. Esta última forma de interrupção é realizada com os quatro dideoxinucleotídeos mais comuns a cada base (ddATP, ddGTP, ddCTP e ddTTP) e é a que permite a obtenção dos fragmentos de diversos tamanhos utilizados no Método de Sanger.

Para que a seqüência seja lida, os fragmentos obtidos são então submersos em uma

[1] O seqüenciamento total deste organismo foi recentemente concluído.

solução de gel de poliacrilamida, que separa os fragmentos de acordo com seus tamanhos. Em seguida, expõem-se este gel a um filme fotográfico de modo a visualizar o padrão de separação dos fragmentos. Cada um dos quatro sulcos do gel corresponde a um nucleotídeo e a seqüência pode ser lida diretamente a partir da posição das bandas.

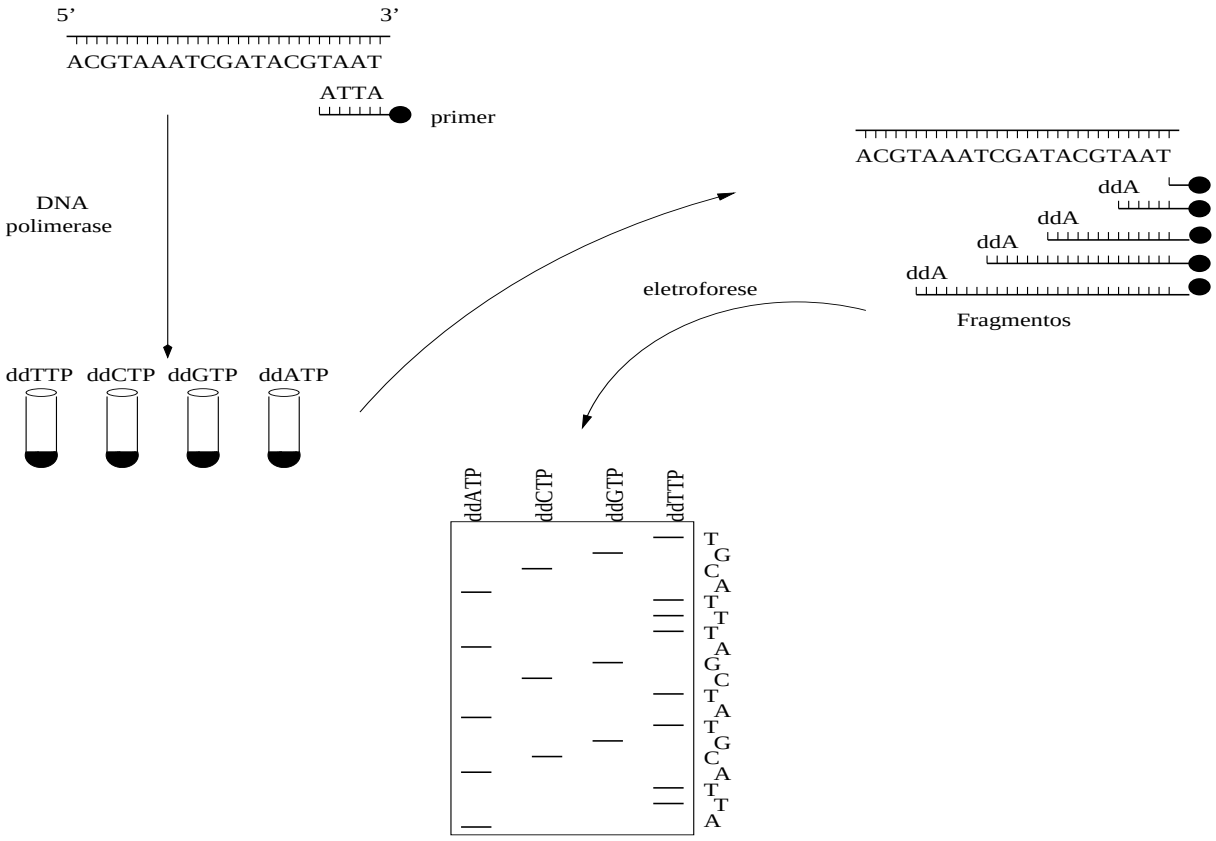

Figura 2.2: Esquema do Método de Terminação de Cadeia de Sanger

Outro método de seqüenciamento bastante utilizado foi desenvolvido por Maxam e Gilbert[24] e corresponde, na realidade, apenas a uma variação do Método de Sanger. No **Método Maxam-Gilbert**, os fragmentos não são obtidos por meio da enzima *polimerase*. Ao invés disso, preparam-se quatro reações que quebram a molécula de DNA em bases específicas (A, C, G, T, CT ou AG) e, a partir daí, o processo continua como descrito anteriormente.

O conjunto de reações componentes dos métodos acima (que podemos chamar de **Métodos diretos de seqüenciamento**) permite a determinação apenas de um número limitado de bases (300 a 1000), impedindo que eles sejam utilizados no seqüenciamento de longas cadeias de DNA. A tarefa de determinação das bases que constituem uma extensa molécula exige a quebra desta em vários fragmentos que, após seqüenciados, podem ser utilizados para a reconstituição da seqüência da molécula original. A

forma com que os fragmentos são obtidos e como a reconstituição do DNA original processa-se nos métodos de seqüenciamento de longas cadeias divide-os em duas categorias:

- **Métodos dirigidos**: os métodos incluídos nesta categoria permitem determinar de forma seqüencial e contínua as bases componentes de uma longa cadeia de DNA.

  Um desses métodos denomina-se *walking* ou *primer directed sequencing*[24], que inicia o processo de seqüenciamento de uma longa cadeia de DNA realizando-se o seqüenciamento de uma das extremidades da molécula (via o Método de Sanger, por exemplo), com o uso de um *primer* específico "universal". Após isso, deriva-se um novo *primer* da parte seqüenciada, que permitirá a obtenção de fragmentos pertencentes à região que vem logo em seguida àquela já determinada. Estes fragmentos são então seqüenciados e outros *primers* são produzidos, até que toda a seqüência do DNA seja determinada.

  Um outro tipo de método dirigido chama-se *nested deletion*[20], que se utiliza de uma enzima para remover as bases presentes na extremidade inicial da molécula sendo seqüenciada. Depois que as primeiras bases desta molécula são determinadas, a enzima remove a parte seqüenciada e o processo repete-se até que a porção final da molécula seja atingida.

  O problema dos métodos acima descritos, assim como de outros métodos dirigidos, é que eles têm se mostrado muito custosos no que diz respeito ao número de reagentes químicos utilizados e ao tempo gasto durante o processo de seqüenciamento[34], além de sua natureza totalmente seqüencial.

- **Métodos aleatórios**: estes métodos, também conhecidos como *shotgun*, permitem o seqüenciamento "desordenado" de uma molécula de DNA. Eles caracterizam-se pela utilização de fragmentos aleatoriamente obtidos da seqüência original e, além de não necessitarem da produção de *primers*, os processos que incluem são altamente paralelizáveis. Por correspondere a um dos nossos principais objetos de estudo, dedicaremos uma seção exclusiva a este método.

2.4 O Método *Shotgun* de Seqüenciamento

O Método *Shotgun* corresponde ao método mais utilizado atualmente nos grandes projetos de seqüenciamento. Assim como o Método de Sanger, o processo de seqüenciamento de uma molécula de DNA via Método *Shotgun* compreende três fases distintas, descritas a seguir:

1. Na primeira fase do seqüenciamento, realiza-se a clonagem da molécula de DNA a ser determinada. Isto pode ser feito através da inserção deste DNA no genoma de um dado organismo e, à medida que ele se reproduz, várias cópias da molécula inserida podem ser obtidas. DNAs produzidos desta forma são denominados **DNAs recombinantes** e os organismos que servem de hospedeiros para esta molécula são denominados organismos **vetores**, que incluem plasmídeos (pequenos DNAs circulares de replicação autônoma existentes nas bactérias), fagos (vírus) e cosmídeos (plasmídeos modificados com seqüências particulares de DNA).

2. Obtidas as várias cópias da molécula sendo seqüenciada, estas são quebradas em vários fragmentos, que podem ser seqüenciados a baixo custo por meio de algum método direto. A quebra dos vários clones de DNA pode ser feita utilizando-se de enzimas conhecidas como enzimas de restrição, que reconhecem e cortam seqüências específicas de nucleotídeos no DNA. Hoje são conhecidas perto de mil enzimas de restrição (também denominadas **endonucleases**), obtidas das mais diversas espécies de bactérias e que diferem entre si quanto à seqüência de nucleotídeos que reconhecem. Apesar de ser bastante utilizado, o processo de quebra de DNA via enzimas de restrição possui uma grande desvantagem: a necessidade da existência de seqüências específicas na molécula. Devido a isso, a grande maioria dos projetos de seqüenciamento utiliza-se de outro método de quebra das moléculas, conhecido como *shotgun* (daí o nome do método de seqüenciamento sendo abordado nesta seção). Aqui, as moléculas são submetidas a algum processo físico-químico de fragmentação, como elevados níveis de vibração, que as quebram em pontos aleatórios.

3. Lembrando-se que vários clones foram quebrados, os fragmentos obtidos possuem várias sobreposições entre si. A partir destas sobreposições, a terceira fase do método *shotgun* tenta reconstruir a seqüência original montando os fragmentos como se fossem as peças de um grande quebra-cabeça. Este problema é conhecido como o **Problema da Montagem de Fragmentos**, que será abordado com mais detalhes na seção 2.6. Nesta terceira fase do método *shotgun* entram vários algoritmos que têm por objetivo montar a seqüência original com a maior exatidão possível e servem como base para várias ferramentas de montagem utilizadas comercialmente. Estas ferramentas computacionais recebem como entrada um conjunto de seqüências, representando os fragmentos, e devolvem como saída uma seqüência cobrindo toda a molécula original ou várias seqüências cobrindo algumas de suas partes.

2.5 Alinhamento Entre Seqüências

Antes de falarmos um pouco mais a respeito do problema da montagem de fragmentos, iremos descrever de forma mais detalhada o que entendemos por sobreposição entre dois fragmentos.

Tomando-se os fragmentos como cadeias de caracteres sobre o alfabeto $\Sigma = \{A, C, G, T\}$, podemos pensar na sobreposição entre dois fragmentos A e B como sendo o melhor alinhamento entre um prefixo da seqüência relacionada a A e um sufixo da seqüência relacionada a B (ou vice-versa). Agora, o que significa o melhor alinhamento entre prefixos e sufixos de duas seqüências? Ou, de forma mais genérica, o que significa um alinhamento entre duas seqüências?

Grosseiramente falando, alinhar duas seqüências corresponde à tarefa de inserir buracos em quaisquer uma de suas posições de modo que elas fiquem do mesmo tamanho. Um alinhamento válido pode conter buracos inclusive nas extremidades das seqüências, mas nunca buracos emparelhados. Pegando-se como exemplo as seqüências $CAGCACTTGGATTCTCGG$ e $CAGCGTGG$, um possível alinhamento entre elas é mostrado abaixo, onde os buracos estão representados por '-'.

```
CAGCA-CTTGGATTCTCGG
---CAGCGTGG--------
```

Um outro alinhamentos entre as mesmas seqüências seria:

```
CAGCACTTGGATTCTCGG
CAGC-----G-T----GG
```

Como podemos ver através dos exemplos acima, existem vários possíveis alinhamentos entre duas seqüências e podemos atribuir uma certa pontuação para cada um deles de acordo com os caracteres presentes em suas colunas. Para exemplificar, consideraremos que cada coluna contendo dois caracteres iguais valerá 1 ponto. As que possuem dois caracteres diferentes valerão -1 ponto. Por último, aquelas contendo um buraco e um caracter receberão -2 pontos. Dessa forma, a pontuação do primeiro alinhamento acima será -19 e a do segundo -15, que corresponde à soma total dos pontos de cada coluna desses alinhamentos.

O **alinhamento ótimo** entre duas seqüências pode ser definido como aquele alinhamento de maior pontuação total, também chamada de **similaridade**, entre duas seqüências e este corresponde ao melhor alinhamento, citado acima, que representa uma sobreposição entre dois fragmentos. Através da estratégia de programação dinâmica é possível resolver de forma eficiente o problema de se encontrar a similaridade entre duas seqüências, assim como o alinhamento relacionado a ela. A idéia do algoritmo proposto

em 1981 por T. F. Smith e M.S. Waterman[38] é calcular a similaridade entre prefixos arbitrários das duas seqüências até que o resultado envolvendo as seqüências como um todo seja obtido.

O algoritmo inicia-se construindo uma matriz A com $(m+1)$ linhas e $(n+1)$ colunas, onde m e n representam o tamanho das seqüências s e t sendo comparadas e cada posição (i, j) de A armazena a similaridade entre os prefixos $s[1..i]$ e $t[1..j]$ destas seqüências. Assim, a primeira linha e a primeira coluna de A seriam inicializadas com múltiplos do valor associado com as colunas do alinhamento contendo buracos (-2 no nosso caso). Com relação às outras entradas (i, j), estas podem ser calculadas observando-se os valores nas posições $(i-1, j)$, $(i-1, j-1)$ e $(i, j-1)$ já calculados. De fato, para obter-se um alinhamento entre $s[1..i]$ e $t[1..j]$, temos somente as seguintes opções:

1. Alinhar $s[1..i]$ com $t[1..j-1]$ e emparelhar um buraco com $t[j]$ ou
2. Alinhar $s[1..i-1]$ com $t[1..j-1]$ e emparelhar $s[i]$ com $t[j]$ ou
3. Alinhar $s[1..i-1]$ com $t[1..j]$ e emparelhar $s[i]$ com um buraco.

Como conseqüência, a similaridade total, utilizando-se da pontuação descrita acima, pode ser obtida pela fórmula:

$$\text{sim}(s[1..i], t[1..j]) = \max \begin{cases} \text{sim}(s[1..i], t[1..j-1]) - 2 \\ \text{sim}(s[1..i-1], t[1..j-1]) + p(i,j) \\ \text{sim}(s[1..i-1], t[1..j]) - 2 \end{cases}$$

onde $p(i, j) = +1$ se $s[i] = t[j]$ e -1 se $s[i] = t[j]$.

Para obter-se o alinhamento correspondente à similaridade calculada, basta marcar na matriz A qual posição fora utilizada no cálculo de cada célula (i, j). Cada marca corresponderá a uma coluna do alinhamento e seguindo-a da posição (m, n) até a posição $(0, 0)$ obteremos este alinhamento por completo.

O algoritmo acima realiza o que chamamos de **alinhamento global** entre duas seqüências, mas às vezes estamos interessados apenas em comparar subseqüências de s e t. Esta variação do problema do melhor alinhamento pode ser resolvida utilizando-se a mesma matriz acima, mas agora cada posição (i, j) de A armazenará a pontuação do melhor alinhamento entre um sufixo de $s[1..i]$ e um sufixo de $t[i..j]$. Inicializando a primeira coluna e a primeira linha da matriz com zero, suas demais posições podem ser preenchidas de acordo com a seguinte fórmula:

$$A[i,j] = \max \begin{cases} A[i][j-1] - 2 \\ A[i-1][j-1]) + p(i,j) \\ A[i-1][j] - 2. \end{cases}$$

O maior valor dentro desta matriz corresponderá à pontuação do melhor alinhamento entre uma subseqüência de s e uma subseqüência de t, que chamamos de **alinhamento local** entre duas seqüências.

Uma outra variante do problema de alinhamento, conhecida como **alinhamento semi-global**, não leva em consideração os buracos situados nas extremidades das seqüências durante o cálculo da similaridade. Essa variante pode ser resolvida utilizando-se das seguintes idéias:

1. Se não desejamos cobrar pelos buracos no final de s, basta considerarmos a melhor pontuação entre s e um prefixo de t. Ou seja, basta tomarmos o maior valor presente na última linha da matriz.
2. Analogamente, podemos desconsiderar buracos no final de t pegando-se o maior valor na última coluna da matriz.
3. Se não desejamos considerar buracos iniciais nas seqüências, o segredo é inicializar a primeira linha e a primeira coluna da matriz com zeros e proceder como no algoritmo básico.

Combinando-se os três procedimentos acima, podemos encontrar o melhor alinhamento entre duas seqüências desconsiderando os buracos em suas extremidades.

A grande maioria das ferramentas de montagem utiliza-se do algoritmo de Smith-Watermam (ou de variações deste algoritmo) para encontrar sobreposições relevantes entre os fragmentos da entrada, sobreposições estas que podem ser de quatro tipos:

1. o alinhamento ocorre entre um sufixo de A e um prefixo de B.(Figura 2.3(a))
2. o alinhamento ocorre entre um prefixo de A e um sufixo de B.(Figura 2.3(b))
3. o alinhamento ocorre entre B e uma subseqüência de A.(Figura 2.3(c))
4. o alinhamento ocorre entre A e uma subseqüência de B.(Figura 2.3(d))

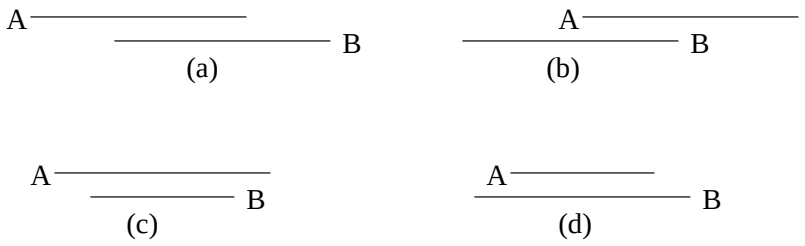

Figura 2.3: As quatro possíveis formas de sobreposição entre fragmentos

2.6 O Problema da Montagem de Fragmentos

Definindo formalmente, o problema da montagem de fragmentos corresponde à tarefa de determinar a seqüência de bases componentes de uma molécula de DNA D, de tamanho G, com base em vários fragmentos sobrepostos obtidos da mesma[34]. Conceitualmente, a molécula D pode ser vista como uma cadeia de caracteres sobre o alfabeto $\Sigma = \{A, C, G, T\}$ (onde A, C, G e T correspondem às iniciais das bases nitrogenadas que compõem o DNA) e cada fragmento f_i a ser montado corresponde a uma subcadeia de D, ou seja, a um trecho de caracteres consecutivos da molécula, contendo de 600 a 700 bases (ou caracteres) aproximadamente. O **complemento reverso** f^c de um fragmento f representa a seqüência de f na fita oposta da qual ele provém. Formalmente, $(a_1, a_2, ..., a_n)^c = w(a_n), ..., w(a_2), w(a_1)$, onde $w(A) = T$, $w(T) = A$, $w(C) = G$ e $w(G) = C$.

Tradicionalmente, o problema da montagem de fragmentos tem sido modelado com base no **Problema da Menor Super Cadeia Comum**[25], assim definido:

Problema da menor super cadeia comum: dado um conjunto finito S de cadeias sobre algum alfabeto Σ, encontrar a menor cadeia w tal que toda cadeia $x \in S$ é uma subcadeia de w, isto é, $w = uxv$, para algum u e v.

Apesar de parecer uma boa formalização para o problema da montagem de fragmentos, já que a seqüência a ser reconstruída precisa realmente incluir todos os fragmentos obtidos da molécula, o problema da menor super cadeia comum não leva em consideração fatos como possíveis erros no seqüenciamento dos fragmentos. Além disso, a necessidade de que a super cadeia seja a menor possível pode levar a resultados inesperados, principalmente no caso da existência de regiões repetidas dentro da seqüência sendo determinada. Todas estas falhas e também o fato da versão de decisão deste problema (isto é, dado S e um inteiro k, existe uma super cadeia comum de S com comprimento menor do que K?) ser NP-completa[25] têm levado a novas formalizações e, conseqüentemente, a novos algoritmos que tentam resolver da melhor forma possível o problema da montagem de fragmentos. Em [28], J. D. Kececioglu e E. W. Myers utilizam-se de uma formalização para o problema que lida com possíveis erros no seqüenciamento dos fragmentos. Em [34] é mostrada uma outra formalização para o problema de montagem que leva em consideração a existência de regiões repetidas dentro da seqüência. Além dessas, várias outras modelagens têm sido propostas[15, 27, 31, 36], dando origem a uma série de algoritmos utilizados como a base de vários sistemas de montagem disponíveis atualmente[10, 11, 17, 29]. Estas ferramentas podem facilmente ser encontradas na Internet[1, 7, 35] e algumas das mais conhecidas são o **PHRAP**[6] e o **TIGR**[8].

2.6.1 Fases do processo de montagem

Apesar de cada ferramenta possuir uma formalização própria para o problema da montagem de fragmentos, de forma geral, elas dividem a tarefa de reconstrução da seqüência original em três fases distintas, descritas abaixo:

Fase de sobreposição: nesta fase, todos os fragmentos f (e seus respectivos complementos reversos) do conjunto de entrada F são comparados entre si na busca por aqueles pares (f', f'') com um certo grau de sobreposição. Em outras palavras, buscam-se nesta fase pares de fragmentos cuja pontuação relacionada ao melhor alinhamento entre eles seja maior que uma dada constante l. Esta restrição no valor mínimo da similaridade é imposta com o objetivo de evitar que sejam consideradas como válidas aquelas sobreposições envolvendo fragmentos pertencentes a regiões não-contíguas da molécula, ou seja, falsas sobreposições entre fragmentos.

As informações relativas às sobreposições, como por exemplo seu tipo, o valor de sua similaridade e a quais fragmentos está associada podem ser armazenadas em um grafo dirigido G com peso nas arestas, conhecido como **grafo de sobreposições** (informações detalhadas a respeito de grafos e suas aplicações podem ser encontradas em [9]). Em um grafo de sobreposições, cada vértice corresponde a um fragmento $f \in F$, cada aresta representa uma sobreposição de um determinado tipo entre dois fragmentos e o peso associado a cada uma delas indica o valor da similaridade do alinhamento correspondente. A Figura 2.4 mostra um exemplo deste grafo, onde as arestas pontilhadas ligando dois vértices u e v representam uma sobreposição entre um sufixo da seqüência de u e um prefixo da seqüência de v (arestas *dovetail*) e as arestas contínuas entre u e v indicam uma sobreposição em que a seqüência de v corresponde a uma subseqüência de u (arestas *containment*).

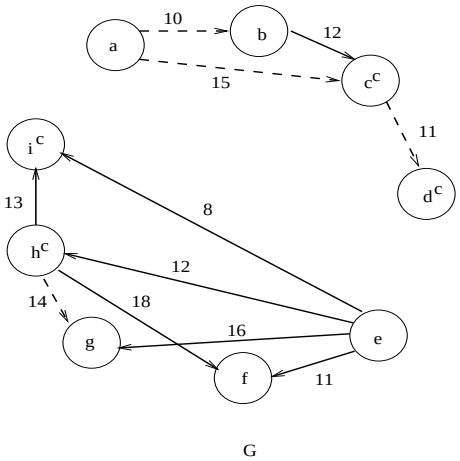

Figura 2.4: Um exemplo de grafo de sobreposição

Fase de *layout*: esta fase envolve dois passos de fundamental importância para o processo de montagem como um todo. O primeiro deles corresponde à tarefa de determinar uma orientação para todos os fragmentos $f \in F$, permitindo assim a criação de um subgrafo G' de G contendo apenas arestas representando sobreposições significativas à reconstrução da seqüência original. Por exemplo, se a mesma orientação for atribuída a dois fragmentos A e B, qualquer aresta de G representando uma sobreposição entre A e o complemento reverso de B pode ser eliminada do grafo G.

O segundo passo determina, através das sobreposições, a localização de cada um dos fragmentos f com relação à seqüência original s. Em outras palavras, podemos dizer que a fase de *layout* associa a cada fragmento $f \in F$ um intervalo $[i, j]$, relacionando-o com uma subseqüência de s que se inicia na posição i e termina na posição j desta cadeia. A um conjunto de intervalos associados aos fragmentos damos o nome de *layout*[28], que pode ser representado como abaixo para um conjunto de fragmentos A, B, C, D, E, F, G, H e I provenientes de s.

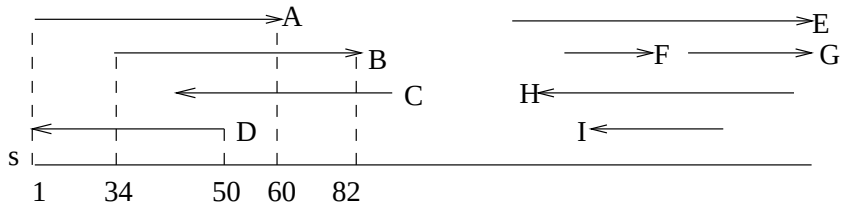

Figura 2.5: Representação de um *layout*

Cabe aqui uma definição de grande importância ao processo de montagem que é a definição de **contig**. Um contig pode ser definido como um conjunto maximal de fragmentos sobrepostos cobrindo um intervalo contíguo da seqüência original[28]. Cada fragmento pertence a um e somente um contig e cada contig possui pelo menos um fragmento[39]. Na figura acima, por exemplo, os fragmentos A, B, C e D correspondem a um contig, digamos c_1, de s, enquanto que E, F, G, H, I correspondem a outro contig, c_2, dessa seqüência. Ao final da fase de *layout* podemos identificar um único ou vários contigs, de acordo com a existência ou não de um número suficiente de fragmentos cobrindo toda a molécula sendo montada. É através destes contigs que as seqüências de bases ininterruptas da molécula de DNA são determinadas.

Fase de alinhamento múltiplo: Nesta última fase do processo de montagem de fragmentos todas as seqüências em um dado contig passam por um processo de alinhamento múltiplo, que possui como resultado a seqüência final correspondente ao intervalo contíguo da molécula coberto pelos contigs em questão. Assim como o alinhamento envolvendo apenas duas seqüências, existem vários alinhamentos

múltiplos dos fragmentos em um certo contig e o alinhamento desejado corresponde àquele de maior pontuação total que pode ser obtido. Suponhamos que os fragmentos A, B, C e D acima correspondem, respectivamente, às seqüências $CATAGTC$, $TAACTAT$, $AGACTATCC$ e $CAATGAC$. Então um possível alinhamento múltiplo destas seqüências seria:

```
CATAGTC-----
--TA-ACTAT--
---AGACTATCC
CAATGAC-----
CATAGACTATCC
```

Neste exemplo, a última linha representa o resultado do alinhamento ou, em outras palavras, a **seqüência consenso** cujos caracteres determinam a possível seqüência de bases componentes do intervalo de s coberto pelos fragmentos A, B, C e D.

2.6.2 Problemas relacionados ao processo de montagem

O processo de montagem de fragmentos envolve uma série de problemas advindos da obtenção dos fragmentos. Abaixo relacionamos alguns deles:

- **Erros no seqüenciamento**: Com a atual tecnologia de seqüenciamento, de $0,5$ a 5% das bases de um fragmento podem ter sido mal determinadas. Estes erros compreendem substituição, remoção e inserção de bases e, caso não sejam tratados pelos algoritmos de montagem, podem levar a resultados indesejáveis.

- **Fragmentos quimeras**: Durante o processo de quebra dos clones do DNA sendo seqüenciado, dois fragmentos provenientes de partes distintas dessa molécula podem se juntar, dando origem ao que chamamos de **fragmentos quimeras**. As ferramentas de montagem precisam reconhecê-los e eliminá-los do processo de montagem.

- **Contaminação dos clones pelo organismo vetor**: Durante a fase de clonagem da molécula sendo determinada, alguns clones podem ser contaminados pela seqüência do DNA do organismo vetor.

- **Orientação desconhecida dos fragmentos**: Como o DNA é composto de duas fitas orientadas em sentidos inversos e, no método *shotgun*, os fragmentos são obtidos de forma aleatória, não temos como saber de qual fita cada um deles provêm. Em outras palavras, não conhecemos a orientação dos fragmentos.

- **Falta de cobertura total da molécula**: Como o processo de obtenção dos fragmentos ocorre de forma aleatória, partes da seqüência original podem não estar cobertas por nenhum deles, ou cobertas de forma insuficiente (com um pequeno número de fragmentos). Isso impede o seqüenciamento total de uma molécula de DNA e as partes não cobertas precisam então receber um tratamento especial.

- **Regiões repetidas dentro das seqüências**: O seqüenciamento do DNA de vários organismos levou à descoberta de regiões repetidas dentro destas moléculas. Uma região repetida é um pedaço contíguo do DNA que possui uma grande semelhança com outra porção da molécula. Estas regiões podem ser totalmente idênticas, repetindo-se algumas vezes dentro da seqüência, (como por exemplo seis grupos de genes presentes no DNA da *Haemophilus influenziae*) ou diferirem em algumas bases, repetindo-se várias vezes dentro da molécula (como por exemplo uma região repetida presente no DNA humano conhecida com Alu, que pode ser encontrada a cada intervalo de cerca de 1000 bases). Estas regiões repetidas podem dar origem a falsas sobreposições envolvendo fragmentos pertencentes a partes distintas da molécula (Figura 2.6) e, neste caso, o processo de montagem ocasionalmente resultará em uma seqüência de bases contendo um alto grau de diferença com relação à molécula original (Figura 2.7). Talvez este seja um dos problemas de mais difícil solução e de maior influência na exatidão da seqüência final gerada pelas ferramentas de montagem.

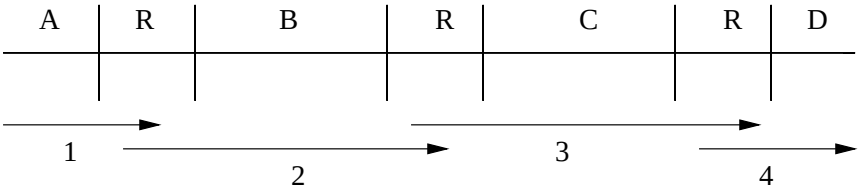

Figura 2.6: Uma molécula com três cópias da região R. Os fragmentos 1, 3 e 4 possuem sobreposições, mas pertencem a partes distintas da molécula.

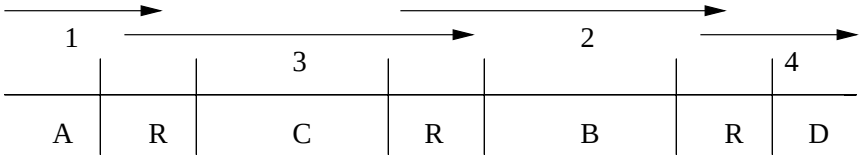

Figura 2.7: Uma possível montagem errada do DNA original, onde as regiões C e B tiveram suas posições trocadas.

As ferramentas avaliadas neste trabalho foram executadas sobre conjuntos de fragmentos que incorporam três dos seis problemas acima mencionados. Considerando-se que a maioria dos sistemas de montagem trata de forma parecida o problema de contaminação dos clones e orientação desconhecida dos fragmentos e que o problema da falta de cobertura total da molécula só pode ser resolvido com o aumento do número de fragmentos a serem processados, comparamos as ferramentas somente no que diz respeito ao tratamento de erros no seqüenciamento, fragmentos quimeras e regiões repetidas.

Capítulo 3

Ferramentas de Montagem

Todas as ferramentas de montagem recebem como entrada um conjunto de seqüências representantes dos fragmentos a serem montados e, ao final da execução, devolvem como saída a seqüência consenso relacionada aos contigs construídos por elas. Esta tarefa é realizada seguindo, de forma geral, as três fases citadas na seção 2.6.1, no entanto, cada uma das ferramentas possui uma formulação específica para o problema de montagem e incorporam características distintas para o tratamento dos problemas mencionados anteriormente. Neste capítulo daremos uma visão geral do funcionamento das quatro ferramentas avaliadas, descrevendo seus algoritmos e mostrando como cada uma delas tenta resolver os problemas de regiões repetidas, erros no seqüenciamento, fragmentos quimeras etc.

Vale aqui ressaltar que não dispomos da mesma quantidade de informações sobre as quatro ferramentas que temos em mãos. Para algumas delas, encontramos detalhes a respeito dos métodos utilizados durante o processamento dos fragmentos em artigos que descrevem o funcionamento destas ferramentas, enquanto que para outras, dispomos apenas de seus guias de utilização. Isto dificulta nossa tarefa de descrever todas as quatro de forma semelhante.

3.1 TIGR Assembler

O **TIGR Assembler**[40] foi desenvolvido pelo Instituto de Pesquisa Genômica (*The Institute for Genomic Research*, Rockville, Maryland) e seus autores são: Granger G. Sutton, Owen White, Mark D. Adams e Anthony R. Kerlavage. Este foi o programa utilizado no seqüenciamento do DNA da bactéria *Haemophilus influenziae* que, como já foi citado, corresponde ao primeiro organismo a ter seu genoma totalmente determinado.

O TIGR Assembler foi implementado utilizando-se da linguagem C e seu algoritmo

compõe-se dos seis passos básicos abaixo enumerados:

1. Compare todos os fragmentos entre si e crie uma lista de fragmentos sobrepostos.
2. Utilize a distribuição do número de potenciais sobreposições de cada fragmento a fim de rotulá-los como repetidos ou não-repetidos. Em outras palavras, para identificá-los como provenientes de regiões repetidas ou não.
3. Se existem fragmentos disponíveis, inicie a montagem utilizando-se, preferencialmente, um fragmento não-repetido. Senão, termine o algoritmo.
4. Utilize a lista de fragmentos sobrepostos na tentativa de intercalar à montagem corrente um fragmento não-repetido.
5. Caso não exista nenhum fragmento não-repetido a ser intercalado à montagem atual, tente realizar a fusão de algum fragmento repetido.
6. Se ainda existem fragmentos sobrepostos às extremidades da montagem corrente, vá para o passo 4. Senão, imprima informações relacionadas a essa montagem e vá para o passo 3.

Mostraremos informações mais detalhadas a respeito dos passos acima nas seções seguintes.

3.1.1 Comparação entre os fragmentos

O **TIGR Assembler** compara todos os fragmentos entre si na busca por pares que possuem sobreposições relevantes. Diferente da grande maioria das ferramentas, o TIGR não se utiliza do algoritmo de Smith-Waterman, considerado relativamente lento, para realizar essa tarefa. Ao invés disso, o que o programa faz é localizar todos os **n-mer oligonucleotídeos** compartilhados entre os pares de fragmentos, onde um n-mer oligonucleotídeos nada mais é que uma seqüência contendo n bases (com n variando entre 10 e 12). Aqueles fragmentos que compartilham alguma seqüência deste tipo, com um alto grau de similaridade, são então considerados como sobrepostos pelo TIGR.

3.1.2 Intercalando fragmentos com montagens

O **TIGR Assembler** utiliza-se de uma versão modificada do algoritmo de Smith-Waterman para determinar sobreposições relevantes entre a montagem corrente e um dado fragmento. Este fragmento é intercalado com a montagem se o melhor alinhamento obtido pelo algoritmo satisfaz os seguintes critérios de emparelhamento:

- Tamanho mínimo da sobreposição: na Figura 3.1, o tamanho da sobreposição entre o fragmento f e a montagem M é definido por $(b - a) + 1$.
- Pontuação mínima da região sobreposta com relação à melhor pontuação possível: pontuação da sobreposição entre f e M, determinada pelo algoritmo de Smith-Waterman modificado, dividida pela maior pontuação possível para qualquer fragmento.
- Número máximo de erros dentro da sobreposição: número de posições dentro do alinhamento contendo bases diferentes.
- Tamanho máximo da porção não alinhada pela versão modificada do algoritmo de Smith-Waterman: tamanho da maior porção não alinhada em ambos os extremos do alinhamento. Na Figura 3.1, o tamanho das porções não alinhadas (representadas por linhas diagonais) é $a - 1$ e $L - d$, onde L corresponde ao tamanho da montagem corrente.

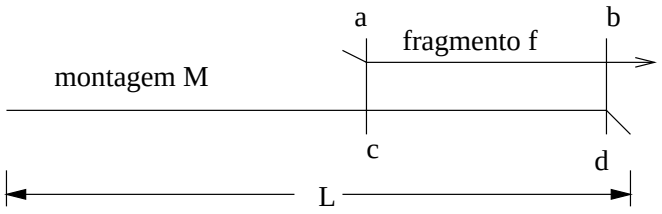

Figura 3.1: Um fragmento alinhado com a montagem corrente.

Todos estes critérios possuem um valor *default* determinado pelo TIGR, mas que podem ser alterados se passados como parâmetros ao programa. Vale salientar também que o rigor ao atendimento destes critérios muda de acordo com o tipo do fragmento sendo montado. Caso ele seja um fragmento repetido, melhores precisam ser os valores relacionados ao alinhamento deste fragmento com a montagem. Caso contrário, o rigor ao atendimento dos critérios diminui.

3.1.3 Construindo a seqüência consenso

A seqüência consenso para uma montagem é construída pelo **TIGR Assembler** através da intercalação de um fragmento por vez à seqüência consenso existente. Cada posição desta montagem é representada por meio de um *profile* que armazena as bases e os espaços alinhados com essa posição no passado. Por exemplo, a posição 10 da montagem pode ter 3 As e 1 espaço, provenientes do alinhamento iterativo de 4 fragmentos com a região da montagem contendo a posição 10. Este *profile* é usado pelo TIGR para dois propósitos: alinhar um fragmento com a montagem corrente e gerar a seqüência

consenso relacionada à montagem, sendo esta última tarefa realizada de acordo com as seguintes regras:

1. Se o caracter que aparece em maior quantidade dentro do *profile* surge um número de vezes maior ou igual a dois terços do tamanho total do *profile*, a saída corresponderá a um dos caracteres A, C, G, T ou espaço.

2. Se existem dois caracteres diferentes de espaço prevalecendo-se no *profile*, a saída corresponderá a uma letra minúscula, que representa código ambíguo.

3. Se o caracter que aparece em maior quantidade surge um número de vezes maior ou igual à metade do tamanho total do *profile* (e menor que dois terços deste tamanho), a saída corresponderá a um dos caracteres a, c, g, t ou espaço.

Caso nenhuma das condições acima seja satisfeita, a saída corresponderá ao caracter n. Além disso, caso o espaço seja o caracter prevalecente, mas aparecendo um número de vezes menor ou igual a dois terços do total, o próximo caracter da seqüência aparecerá em minúsculo. O esquema da Figura 3.2 mostra um exemplo de utilização de cada uma das regras citadas:

```
CATA-T
CGTA-A
CGTCGA
CAAG-A
AGGTAA
GACT-A
------
Crtn-a
```

Figura 3.2: Exemplo de construção da seqüência consenso pelo TIGR.

3.1.4 Tratando regiões repetidas

Como já fora mencionado, o problema de seqüências repetidas é um dos que mais dificulta a montagem correta dos fragmentos obtidos de uma determinada molécula. Sendo assim, uma das principais tarefas das ferramentas de montagem é determinar quais fragmentos provêm de regiões repetidas e tratá-los de forma especial.

O **TIGR Assembler** realiza a detecção destes fragmentos repetidos através do número de sobreposições relevantes que cada fragmento possui. Dada a mediana m do número de sobreposições relevantes existentes, qualquer fragmento com mais de $p * m$

(onde p corresponde a um valor entre 1.4 e 1.6) sobreposições é rotulado como um fragmento repetido pela ferramenta.

Definidos os fragmentos repetidos e não-repetidos, o TIGR utiliza-se de duas estratégias principais para minimizar o número de montagens erradas devido a falsas sobreposições envolvendo fragmentos provenientes de regiões repetidas da molécula. A primeira estratégia é aumentar o rigor ao atendimento dos critérios de emparelhamentos durante a busca de sobreposições relevantes envolvendo fragmentos repetidos e a montagem corrente. Esta estratégia evita montagens erradas quando a diferença média entre duas regiões repetidas é consideravelmente maior que o número de erros introduzidos em um dado fragmento durante seu seqüenciamento. Algumas regiões, porém, são idênticas, o que torna o simples aumento do rigor ao atendimento dos critérios de emparelhamentos uma estratégia ineficiente.

Para tratar de regiões com um alto grau de semelhança, o TIGR utiliza-se de duas informações principais a respeito da molécula sendo seqüenciada: o tamanho de seus clones e as seqüências de bases f e f' que constituem ambas as suas extremidades. Estes fragmentos são denominados "fragmentos companheiros" e, através de informações a respeito da clonagem da molécula sendo determinada, o TIGR sabe que eles devem estar em lados opostos da seqüência, com a primeira base de f situada a aproximadamente L bases de f', onde L corresponde ao tamanho dos clones. O uso destas informações para tratamento de regiões repetidas é mostrado na Figura 3.3. Sejam $2F$ e $2R$ dois fragmentos companheiros obtidos do clone 2. Como os fragmentos $2F$ e $8R$ foram obtidos de duas regiões idênticas da molécula, eles possuem a mesma seqüência de bases, ao contrário do que ocorre com os fragmentos $2R$ e $8F$, que pertencem a regiões distintas do DNA sendo determinado. Através das informações relacionadas aos fragmentos companheiros, o TIGR monta os fragmentos $2F$ e $8R$ em locais diferentes, mesmo eles possuindo seqüências idênticas. Isto é feito iniciando a montagem com um fragmento não-repetido e intercalando fragmentos até que ela tenha sobreposições relevantes somente com fragmentos repetidos. A partir daí, o TIGR dá prioridade aos fragmentos repetidos cujos fragmentos companheiros correspondentes já se encontram na montagem e a uma distância apropriada com base no tamanho do clone. Fragmentos repetidos cujos fragmentos companheiros não se encontram na montagem, mas que já deveriam estar nela, são excluídos do processo de montagem.

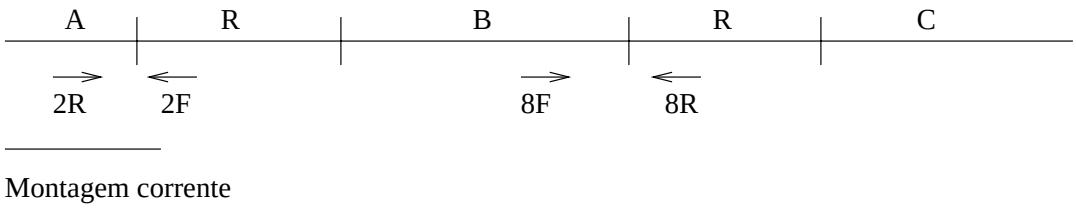

Figura 3.3: Tratamento de fragmentos repetidos pelo TIGR.

3.1.5 Identificando fragmentos quimeras

As informações a respeito dos fragmentos companheiros também permitem que o **TIGR Assembler** reconheça e elimine os fragmentos quimeras do processo de montagem. Na Figura 3.4, por exemplo, as montagens 1 e 2 não seriam intercaladas se dependessem única e exclusivamente do fragmento $2F$ (que corresponde a um fragmento quimera). Porém, a presença dos fragmentos companheiros dos clones 1, 3 e 4 justifica a fusão destas duas montagens e a identificação do fragmento $2F$ como sendo um fragmento quimera.

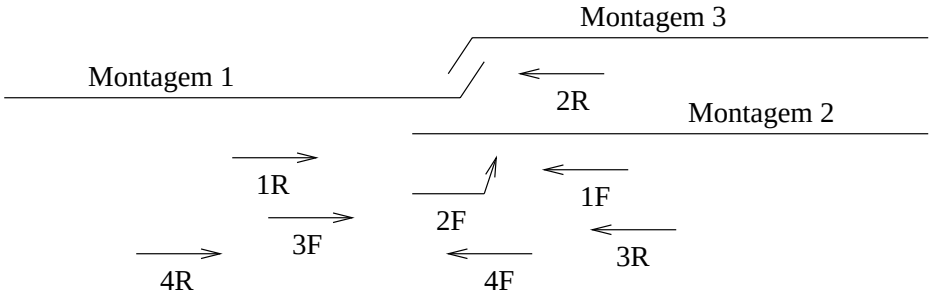

Figura 3.4: Utilização de fragmentos companheiros para identificação de fragmentos quimeras.

3.2 CAP2

O **CAP2**[23] foi desenvolvido por Xiaoqiu Huang na Universidade Tecnológica de Michigan (*Michigan Technological University*, Houghton, Michigan). Na verdade, esta ferramenta é uma versão mais atualizada do **CAP** [21] (*Contig Assembly Program*[1]) que incorpora uma série de melhoramentos, como estimação de uma taxa de erro de seqüenciamento para cada fragmento, identificação e remoção de fragmentos quimeras e resolução de incertezas relacionadas à montagem de fragmentos provenientes de regiões repetidas da molécula.

O processo de montagem no CAP2 consiste de três fases. Na primeira delas, cada par de fragmentos passa por uma espécie de filtro, que tem a finalidade de eliminar aqueles que, possivelmente, não se sobrepõem. Após a filtragem, determina-se então o melhor alinhamento para cada par de fragmentos aprovados, juntamente com seu valor de similaridade, por meio de uma versão modificada do algoritmo de Smith-Waterman. Calcula-se então, através das sobreposições, um vetor de erro para cada fragmento, que será utilizado para reconhecimento de fragmentos quimeras e falsas sobreposições envolvendo fragmentos repetidos. Na fase 2, uma montagem inicial dos fragmentos é obtida

[1]Vale salientar que já existe uma terceira versão deste programa, denominada CAP3.

por meio de um algoritmo baseado na estratégia gulosa. Sobreposições inconsistentes com esta montagem são então identificadas e utilizadas pelo CAP2 na determinação de fragmentos provenientes de cópias de uma região repetida. Para cada uma dessas regiões, um alinhamento dos fragmentos das cópias desta seqüência é construído. Posteriormente, por meio de diferenças neste alinhamento, os fragmentos são então particionados em grupos de forma que aqueles pertencentes à mesma cópia da seqüência fiquem no mesmo conjunto. Esta partição é utilizada para resolver incertezas presentes na montagem inicial. Na fase 3, um alinhamento dos fragmentos em cada contig é construído, refinado e a seqüência consenso produzida.

3.2.1 Detecção de fragmentos sobrepostos

Antes de determinar quais pares de fragmentos possuem sobreposições relevantes, o **CAP2** utiliza-se de um filtro que elimina rapidamente aqueles pares possivelmente não sobrepostos dentro da molécula, evitando assim uma comparação entre todos os fragmentos. Um par de fragmentos é considerado aprovado por este filtro se e somente se existe um alinhamento (sem espaços) de tamanho 20 entre eles, contendo pelo menos nove bases iguais consecutivas e no máximo duas bases diferentes. Todos os fragmentos aprovados são então comparados dois a dois, atribuindo-se valores a suas sobreposições de acordo com o algoritmo de Smith-Waterman. Através destes valores, as sobreposições são então classificadas em sobreposições de alta similaridade, sobreposições de baixa similaridade na extremidade $5'$ ou sobreposições de baixa similaridade na extremidade $3'$. Esta classificação é utilizada mais adiante pelo algoritmo para o reconhecimento de fragmentos quimeras.

Após determinar as sobreposições relevantes entre os fragmentos, o CAP2 estima uma taxa de erros de seqüenciamento para as regiões destes fragmentos. Estes valores são armazenados em um vetor de taxa de erros E_f para cada fragmento f da entrada, onde $E_f[i]$ armazena a taxa de erros da região de tamanho r de f iniciando-se na base i, para $1 \leqslant i \leqslant m_f - r + 1$, e a taxa de erros da região $3'$ de f iniciando-se na base i e terminando na base m_f, para $m_f - r + 1 < i \leqslant m_f$, com m_f correspondendo ao tamanho do fragmento f e r a uma constante entre 1 e o tamanho do maior fragmento.

A taxa estimada de erros de seqüenciamento para cada fragmento é calculada da seguinte maneira: cada um dos vetores E_f é inicializado com 1.0. Após o cálculo da melhor sobreposição entre dois fragmentos p e q, os vetores E_p e E_q são então refinados. Sejam $h(i,1) = i$, $k(i,1) = \min(i + r - 1, m_p)$, $h(i,0) = m_p - \min(i + r - 1, m_p) + 1$ e $k(i,0) = m_p - i + 1$. Então a região de p da base i à base $\min(i + r - 1, m_p)$ ocorre como a região de p^s da base $h(i,s)$ à base $k(i,s)$ no alinhamento entre p^s e q^t, onde s e t recebem o valor 0 ou 1, determinando a orientação dos fragmentos. Sejam *first* e *last* a posição inicial e final do alinhamento no fragmento p^s. Para cada i com $\mathit{first} \leqslant h(i,s)$ e $k(i,s) \leqslant \mathit{last}$, se $E_p[i]$ é maior do que a taxa de diferença x observada na porção do

alinhamento entre as bases $h(i,s)$ e $k(i,s)$, $E_p[i]$ recebe x. Calculado o vetor E_f de cada fragmento da entrada, o número de erros de uma região arbitrária de f que vai da base i à base i' pode ser determinado da seguinte maneira: particiona-se a região em subregiões de tamanho r. Estima-se então que o número de erros em cada uma dessas subregiões é r vezes a taxa de erros daquela subregião, valor este armazenado em E_f.

3.2.2 Detecção e eliminação de falsas sobreposições envolvendo fragmentos repetidos

Através dos valores associados às sobreposições e do vetor de taxas de erros calculado na fase de detecção de fragmentos sobrepostos, o **CAP2** é capaz de determinar falsas sobreposições envolvendo fragmentos provenientes de regiões repetidas da molécula e eliminá-las do processo de montagem.

Para isto, cada sobreposição é avaliada no intuito de verificar se ela atende a uma das duas condições seguintes:

(i) Sejam p e q os dois fragmentos sobrepostos. Se a taxa de diferença no alinhamento correspondente à sobreposição for maior que a soma das taxas de erros das seções de p e q envolvidas na sobreposição, mais uma constante $x = 0.03$, então a sobreposição não é mais considerada.

(ii) Sejam p e q os dois fragmentos sobrepostos. Se o alinhamento envolve uma longa região $3'$ ou $5'$ de baixa similaridade (informação esta que pode ser obtida da partição dos alinhamentos criada na fase de detecção de fragmentos sobrepostos) e sua taxa de diferença for maior que a soma das taxas de erros das seções de p e q envolvidas na sobreposição, mais uma constante $y = 0.10$, a sobreposição é eliminada.

A primeira condição tem o objetivo de eliminar sobreposições entre fragmentos provenientes de cópias relativamente similares de uma seqüência repetida, enquanto que a segunda elimina aquelas sobreposições envolvendo fragmentos provenientes das extremidades de cópias altamente similares de uma seqüência repetida.

3.2.3 Detecção de fragmentos quimeras

Para o reconhecimento dos fragmentos quimeras, o **CAP2** utiliza-se do vetor de taxas de erros calculado para cada fragmento e da definicão de ponto de separação. Uma posição de um fragmento quimera é chamada de ponto de separação *sp* se as regiões anteriores e posteriores a este ponto pertencem a partes distintas da molécula

de DNA. Considerando-se que não exista nenhum fragmento f' similar a um fragmento quimera f, as seguintes condições são verificadas pelo CAP2 a fim de identificar f como sendo um fragmento quimera:

(*i*) $\max\{E_f[i - \lfloor r/2 \rfloor] - E_f[i]/r < i < m_f - r\} > x$, onde x é um valor entre 0.15 e 0.25.

(*ii*) Existe algum fragmento f' com uma região similar à região de tamanho r de f terminando em sp (posição onde fora obtida a diferença máxima calculada acima). Além disso, existe um outro fragmento f'' que tenha uma região similar à região de tamanho r de f começando em sp.

(*iii*) Nenhuma das regiões similares acima estende-se além da posição sp.

Se (*i*), (*ii*) e (*iii*) forem satisfeitas, f será considerado um fragmento quimera e eliminado do processo de montagem.

3.2.4 Construção de contigs

Para construir os contigs que determinam a seqüência de bases componentes da molécula sendo determinada, o **CAP2** utiliza-se de uma estratégia gulosa que processa as sobreposições em ordem decrescente do valor de similaridade associado a elas.

Antes de iniciar o processo de montagem propriamente dito, o CAP2 constrói uma árvore T para cada fragmento f não-contido[2] em nenhum outro. A raiz de T é o próprio fragmento f, e um fragmento g é considerado filho de f em T se g está contido[3] em f. No caso de g estar contido em vários fragmentos, escolhe-se como pai de g aquele fragmento dentro do qual g melhor se encaixa, isto é, aquele fragmento com o qual g possui o melhor alinhamento (alinhamento de maior pontuação).

Construídas então todas as árvores, as sobreposições entre os fragmentos raízes são processadas da seguinte forma: suponhamos que p^s e q^t sejam os fragmentos cuja sobreposição possui a maior pontuação no momento. Consideremos que p^s pertença ao contig u^x e q^t ao contig v^y (de início, cada fragmento raiz corresponde a um contig). Se u^x e v^y são os mesmos contigs ou um deles é o complemento reverso do outro, o CAP2 ignora a sobreposição entre p^s e q^t. Senão, o CAP2 identifica o maior subcontig $u'^{x'}$ contendo p^s de u^x e o maior subcontig $v'^{y'}$ contendo q^t de v^y tal que:

[2]Dizemos que um fragmento f_1 não está contido em um fragmento f_2 ($f_1 \not\subseteq f_2$) se a cadeia de caracteres que representa f_1 não corresponde a nenhuma subcadeia da cadeia de f_2.

[3]Ao contrário da definição anterior, dizemos que um fragmento f_1 está contido em um fragmento f_2 ($f_1 \subseteq f_2$) se a cadeia de caracteres que representa f_1 corresponde a alguma subcadeia da cadeia de f_2.

(*i*) Os subcontigs $u'^{x'}$ e $v'^{y'}$ podem ser intercalados em um contig temporário d consistente com a sobreposição entre p^s e q^t.

(*ii*) O primeiro e o segundo fragmento da extremidade $5'$ do contig d correspondem aos fragmentos da extremidade $5'$ dos subcontigs $u'^{x'}$ e $v'^{y'}$.

(*iii*) O primeiro e o segundo fragmento da extremidade $3'$ do contig d correspondem aos fragmentos da extremidade $3'$ dos subcontigs $u'^{x'}$ e $v'^{y'}$.

Um exemplo destes contigs pode ser visto na Figura 3.5 abaixo:

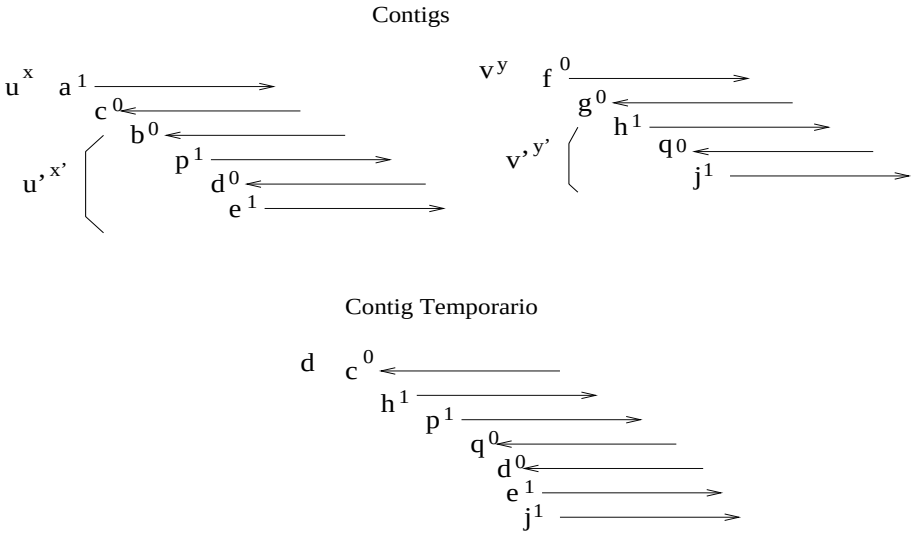

Figura 3.5: Processamento de sobreposições pelo CAP2.

Encontrados os subcontigs $u'^{x'}$ e $v'^{y'}$, se u^x e v^y são diferentes, o segundo fragmento da extremidade $5'$ de d corresponde ao fragmento da extermidade $5'$ de u^x ou v^y e o segundo fragmento da extermidade $3'$ de d corresponde ao fragmento da extremidade $3'$ de u^x ou v^y, o CAP2 intercala os contigs u^x e v^y em um novo contig z e seus complementos reversos no complemento reverso de z. O processo acima então repete-se sobre os novos contigs produzidos até que não existam mais sobreposições relevantes entre eles.

3.2.5 Construção do alinhamento múltiplo

Na fase de construção dos contigs, uma ordem parcial das árvores de fragmentos é determinada, onde cada fragmento aparece em alguma orientação na árvore, cada aresta na árvore representa um alinhamento entre dois fragmentos f e f', tal que $f \subseteq f'$ ou $f' \subseteq f$ e duas árvores adjacentes estão relacionadas por um alinhamento entre os seus

vértices (fragmentos) raiz. Nesta fase, estes alinhamentos são intercalados dentro de um alinhamento múltiplo dos fragmentos e a seqüência consenso é determinada.

O alinhamento múltiplo dos fragmentos é gerado por seções, onde cada seção consiste de um número fixo de colunas, por exemplo 60. Seja w o conjunto de fragmentos que aparecem na seção corrente. Um fragmento é incluído em w quando a primeira base do fragmento aparece na próxima coluna do alinhamento. Inicialmente, w consiste somente do fragmento raiz da primeira árvore de fragmento em um contig. Suponha que um fragmento f está em w e a próxima coluna do alinhamento envolve a base de f que se encontra na posição p. Então, verifica-se se o primeiro fragmento filho de f na árvore inicia na posição p de f. Caso isto ocorra, o fragmento filho é inserido em w e o segundo filho de f torna-se o primeiro filho deste fragmento. No caso em que f é a raiz de uma árvore T, a raiz da árvore adjacente a T também é verificada e pode ou não ser incluída em w. Uma coluna de alinhamento das bases dos fragmentos em w é construída de acordo com a correspondência entre pares destas bases. A construção do alinhamento múltiplo requer tempo proporcional à soma do tamanho dos fragmentos.

3.2.6 Refinamento do alinhamento dos fragmentos

Antes de determinar a seqüência consenso dos alinhamentos produzidos na seção anterior, o **CAP2** realiza um refinamento das regiões destes alinhamentos consideradas "pobremente" alinhadas. Dado o alinhamento múltiplo dos fragmentos em um certo contig, sua maior região contendo no máximo seis colunas perfeitas (compostas de apenas um tipo de base) e pelo menos uma coluna imperfeita é considerada pobremente alinhada pela ferramenta.

O refinamento das regiões pobremente alinhadas segue uma estratégia iterativa utilizada no refinamento de proteínas. Uma seqüência qualquer do alinhamento é escolhida e removida. Remove-se então os buracos desta seqüência assim como todas as colunas do alinhamento remanescente contendo apenas buracos. Um alinhamento de maior pontuação entre a seqüência e o alinhamento modificado é determinado e então repete-se o processo para todas as outras seqüências do alinhamento resultante.

3.3 FAKtory

O **FAKtory**[28, 35] foi desenvolvido pelo Departamento de Computação da Universidade do Arizona, e tem como principal responsável pelo seu projeto o professor Gene Myers. Esta ferramenta consiste de três partes fundamentais: uma biblioteca de software (**FAK**) contendo as funções que tratam do problema de montagem, uma *interface* baseada em Tcl/Tk e um software de suporte ao processamento *pipeline*. Como esta-

mos interessados apenas no comportamento do FAKtory diante da tarefa de montagem, concentraremos nossa atenção apenas no projeto das funções componentes do FAK. O problema da montagem de fragmentos é tratado pelo FAKtory utilizando-se de uma heurística para cada fase do processo descrita no capítulo anterior e esta ferramenta focaliza grande parte de sua atenção sobre o problema de erros no seqüenciamento.

No intuito de tentar resolver da melhor forma possível o problema da montagem de fragmentos, o FAKtory utiliza-se de uma variante do problema da menor super cadeia comum, assim definida:

O problema da reconstrução de uma seqüência de DNA: dado um conjunto F de fragmentos e uma taxa de erro $0 \leqslant e < 1$, encontre a menor seqüência s tal que para todo o fragmento $A \in F$ existe uma subcadeia B de s tal que:

$$\min(d(A,B), d(\bar{A},B)) \leqslant e|A|,$$

onde $\bar{A}$ representa o complemento reverso de A e $d(A,B)$ a distância de edição entre A e B, ou seja, o número mínimo de inserções, deleções e substituições necessárias para converter a seqüência correspondente ao fragmento A na seqüência correpondente a B.

O algoritmo implementado pelo FAKtory constitui-se de quatro fases distintas e, ao término de sua execução, mais de uma montagem dos fragmentos pode ter sido gerada, diferente do que ocorre com as outras ferramentas, que geram apenas uma solução para o problema. As quatro fases componentes do FAKtory consistem dos seguintes problemas combinatórios:

1. Construir um grafo representando todas as sobreposições relevantes entre os fragmentos.
2. Atribuir uma orientação para cada um dos fragmentos.
3. Selecionar um conjunto de sobreposições que levam a um *layout* consistente dos fragmentos orientados.
4. Intercalar as sobreposições adequadas através de um alinhamento múltiplo das seqüências dos fragmentos.

3.3.1 Determinação de fragmentos sobrepostos e construção do grafo de sobreposição

Dados dois fragmentos A e B, o **FAKtory** determina se eles possuem alguma sobreposição relevante verificando se, dentre os alinhamentos possíveis envolvendo estes

fragmentos, existe algum com uma probabilidade mínima de ocorrer por acaso. Isto diminui as chances de se considerar, durante o processo de montagem, falsas sobreposições entre dois fragmentos. Formalmente, a tarefa de verificação da existência de um alinhamento com probabilidade mínima de ocorrer por acaso pode ser assim descrita:

O problema de se encontar uma sobreposição relevante: Dado um par ordenado A, B de seqüências sobre um alfabeto de tamanho s, encontrar uma sobreposição entre A e B que minimize:

$$P_s(l,d) = \sum_{i=1}^{d} N_s(l,i)N_s(l,d-i)/(d+1)s^{l+d}$$

onde l representa o número de posições contendo caracteres iguais no alinhamento e d o número de posições contendo caracteres diferentes.

Na descrição acima, $P_4(l,d)$ determina um limite superior na probabilidade de um alinhamento contendo l casamentos perfeitos e d erros ocorrer, e

$$N_s(l,d) = \sum_{0 \le i \le d} \left(\begin{array}{c} l+d \\ i \end{array} \right) (s-i)^i$$

representa o número de seqüências de tamanho $(l+d)$, sobre um alfabeto de tamanho s, que contém uma dada subseqüência de tamanho l.

Um algoritmo do tipo força-bruta poderia muito bem ser utilizado na busca pelas sobreposições entre os fragmentos que minimize $P_4(l,d)$ (considerando que o alfabeto sobre o qual estamos trabalhando possui apenas quatro elementos), mas existe uma forma mais rápida de se fazer isso, proposta por Myers em [33]. A idéia deste algoritmo é calcular os melhores alinhamentos de forma incremental e representar as sucessivas soluções por meio de uma estrutura de dados que pode ser facilmente atualizada. Para $S = s_1 s_2 ... s_n$, denotemos por $S_{i,j}$ a subcadeia $s_i s_{i+1} ... s_j$, por $S_{i,*}$ o sufixo $S_{i,n}$ e por $S_{*,j}$ o prefixo $S_{1,j}$. Dadas duas seqüências A e B de tamanho m e n, Myers resolve uma série de problemas de alinhamento que compara incrementalmente os maiores sufixos $A_{k,*}$ com a seqüência de B. Para cada sufixo $A_{k,*}$, a distância de edição é obtida entre todos os $A_{k,i}$ e todos os $B_{*,j}$. Estas distâncias não são calculadas de forma explícita, mas estão representadas implicitamente por meio de uma estrutura de dados esparsa. Qualquer distância em particular pode ser obtida facilmente desta codificação.

Encontradas as sobreposições com probabilidade mínima de terem ocorrido por acaso, elas são armazenadas em um grafo de sobreposições $G = (E, V, w)$ como aquele descrito na seção 2.6.1. Neste grafo, V corresponde ao conjunto dos fragmentos, E representa o conjunto das sobreposições relevantes entre os fragmentos e $w = L(l,d) = -\log_4 P_4(l,d)$ define uma função que atribui pesos às arestas de G. Estes pesos representam o valor da probabilidade da sobreposição ocorrer. Vale salientar que, de início, G corresponde a um grafo de sobreposição contendo dois vértices para cada fragmento f: um deles representando a seqüência original de f e o outro seu complemento reverso.

3.3.2 Orientação dos fragmentos

Antes de determinar um *layout* consistente com as sobreposições obtidas na fase anterior, o **FAKtory** precisa atribuir a cada fragmento f, representado pelos vértices de G, uma determinada orientação. Em outras palavras, o FAKtory precisa determinar qual seqüência de cada fragmento está sendo usada na montagem, se a seqüência original ou seu complemento reverso. Ao término desta fase, teremos em mãos um subgrafo G' do grafo G obtido anteriormente, contendo apenas as arestas e os vértices consistentes com a orientação determinada.

Uma **orientação** de uma coleção F de fragmentos pode ser representada por uma partição $(O, \bar{O})$, onde $O \subseteq F$ é o conjunto de fragmentos com orientação original, enquanto que $\bar{O} = F - O$ determina o conjunto de fragmentos com orientação reversa.

Como algumas sobreposições podem ser eliminadas do grafo G após a determinação de uma orientação para os fragmentos, o FAKtory busca uma partição $(O, \bar{O})$ de peso máximo, ou seja, uma partição que maximize a soma dos pesos das arestas que permanecerem no grafo G'. Formalmente falando, o problema da orientação pode ser assim definido:

Problema da orientação : dado um conjunto de fragmentos F, encontrar uma orientação $(O, \bar{O})$ para F que maximize:

$$w((O, \bar{O})) = \sum \text{opp}(A, B) + \sum \text{same}(A, B).$$

Nesta definição, $w((O, \bar{O}))$ representa o peso da partição e as funções

$$\text{same}(A, B) = \max(w(A, B), w(B, A)) \text{ e } \text{opp}(A, B) = \max((w(A, \bar{B}), w(\bar{B}, A))$$

determinam o peso da melhor sobreposição quando A e B possuem a mesma orientação, e orientações diferentes, respectivamente.

De acordo com [26], o problema acima é NP-completo e Myers propõe um algoritmo guloso que calcula, em tempo polinomial, uma orientação $\mathcal{O} = (O, \bar{O})$ para F de peso próximo ao peso máximo. Esse algoritmo inicia-se com os conjuntos da partição $\mathcal{O}$ totalmente vazios. Dada uma ordenação $f_1, f_2, ..., f_n$ dos fragmentos em F, no passo i inclui-se o fragmento F_i em O ou em $\bar{O}$ de acordo com a seguinte regra: se $w((O \cup f_i, \bar{O})) \geq w((O, \bar{O} \cup f_i))$, f_i recebe a orientação 0 ($O = O \cup f_i$). Senão, f_i recebe a orientação 1 ($\bar{O} = \bar{O} \cup f_i$). De acordo com Myers, esta estratégia gulosa permite a obtenção de uma orientação com valor de peso correspondendo a pelo menos metade do valor do peso máximo.

3.3.3 Determinação de um *layout* para os fragmentos

A fase anterior de determinação de uma orientação para cada um dos fragmentos da entrada tem como resultado um subgrafo G' do grafo de sobreposições G inicial. Nesta penúltima fase do processo de montagem, o **FAKtory** precisa determinar um conjunto de arestas de G' consistente com uma interpretação dos fragmentos como intervalos $[i, j]$.

Para um dado *layout* L dos fragmentos, denotemos por $L(f)$ o intervalo associado ao fragmento f. O tamanho de L, denotado por $|L|$, pode ser definido como $|\bigcup_{f \in F} L(f)|$, ou seja, o tamanho do intervalo totalmente coberto por L.

A cada *layout* L, podemos associar um conjunto de arestas de um grafo de sobreposições da seguinte maneira: para todos aqueles fragmentos com intervalos idênticos, construa uma classe de equivalência destes fragmentos, selecione algum fragmento desta classe e então ligue-o a todos os outros da classe através de uma aresta *containment*. Desconsiderando-se os fragmentos não representativos, todos os demais podem ser ordenados em ordem crescente de i (início do intervalo) e decrescente de j (fim do intervalo). Para os fragmentos cujos intervalos estão contidos no intervalo de algum outro fragmento, pegue o menor intervalo que os contenha e, novamente, ligue o fragmento associado a este intervalo com aqueles que ele contém, através de uma aresta dirigida *containment* (como aquela descrita na subseção 2.6.1). Removendo-se todos os fragmentos contidos em outros, os fragmentos remanescentes encontram-se agora ordenados em ordem crescente tanto de i quanto de j. Se um fragmento possui uma sobreposição com seu sucessor na ordenação, ligue estes fragmentos através de uma aresta *dovetail* (como aquela descrita na subseção 2.6.1). O conjunto de arestas resultante deste processo satisfaz as quatro propriedades seguintes:

1. Todo vértice i é destino de, no máximo, uma aresta dirigida.
2. As arestas não formam um ciclo.
3. Não existem duas arestas *dovetail* partindo do mesmo vértice.
4. Nenhuma aresta *containment* é seguida por uma aresta *dovetail*.

A todo conjunto de arestas atendendo às propriedades acima chamamos de *dovetail-chain-branching*.

Assim como podemos associar a um dado *layout* um *dovetail-chain-branching*, a cada conjunto de arestas satisfazendo aquelas propriedades pode ser associado um *layout*. Para uma sobreposição entre dois fragmentos A e B, seja $\mathrm{Tam}(A, B)$ o tamanho do prefixo de A não alinhado com B. Dado um *dovetail-chain-branching* $\mathcal{B}$, pode-se construir

um layout para cada árvore $\mathcal{A}$ em $\mathcal{B}$ como se segue. Para um fragmento f em $\mathcal{A}$,

$$L(f) = [\text{left}(f), \text{left}(f) + |f| - 1],$$

onde

$$\text{left}(f) = \begin{cases} 0, \text{ se } f \text{ é a raiz de } \mathcal{A}. \\ \text{left}(A) + \text{Tam}(A, f), \text{ se } A \text{ possui uma sobreposição com } f \text{ em } \mathcal{A}. \end{cases}$$

Lembrando-se que os pesos associados às arestas do subgrafo G' representam a probabilidade $L(l, d)$ da sobreposição ocorrer, onde l é o número de posições com caracteres iguais no alinhamento e d o número de erros (posições com caracteres diferentes), para um alinhamento perfeito teremos $L(l, 0) = l$. Então, quando $e = 0$, o peso de uma aresta (A, B) é o tamanho do maior prefixo de B sobreposto com a seqüência de A. Considerando-se $w(\mathcal{B})$ como a soma dos pesos de todas as arestas em $\mathcal{B}$, então, o tamanho do *layout* induzido por $\mathcal{B}$ é:

$$|L| = \sum_{f \in F} |F| - w(\mathcal{B}).$$

Já que $\sum_{f \in F} |F|$ é uma constante para qualquer conjunto F de fragmentos, um *dovetail-chain-branching* de peso máximo resulta em um *layout* de tamanho mínimo. O problema de se encontrar um *layout* para os fragmentos pode então ser assim definido:

Problema do dovetail-chain-branching: Dado um grafo dirigido $G = (V, E)$ com peso nas arestas definido por uma função w, encontrar um *dovetail-chain-branching* $\mathcal{B} \subseteq E$ de $w(\mathcal{B})$ máximo.

J. D. Kececioglu and E. W. Myers[28] provaram que o problema acima é NP-Completo, e para encontrar um *dovetail-chain-branching*, o **FAKtory** começa a busca por um *branching* (conjunto de arestas que atendem às condições 1 e 2 acima) de peso máximo. Esta tarefa pode ser realizada em tempo polinomial e caso o *branching* encontrado atenda também às condições 3 e 4 especificadas anteriormente, este é tomado como uma solução para o problema. Senão, o FAKtory passa a buscar por outros *branchings*, em ordem decrescente de peso, até que um atendendo às quatro condições seja encontrado. Este *dovetail-chain-branching* é então utilizado para a determinação de um *layout* para os fragmentos.

3.3.4 Fase de alinhamento múltiplo

Nesta última fase do processo de montagem, o **FAKtory** determina as bases da cadeia de DNA sendo seqüenciada por meio de um alinhamento múltiplo envolvendo as

arestas do grafo condizentes com o *layout* obtido na fase anterior. Em outras palavras, o alinhamento incluirá aquelas sobreposições entre os fragmentos cujas posições relativas determinadas pelo *layout* e pelo alinhamento são as mesmas. Este conjunto de arestas é denominado de *closure* $\mathcal{C}$ do *branching* $\mathcal{B}$ e pode ser formalmente definido como:

$$\mathcal{C}(\mathcal{B}) = \{(A,B) \in G \ / \ [\mathrm{Tam}(A,B) - (\mathrm{left}_{\mathcal{B}}(B) - \mathrm{left}_{\mathcal{B}}(A))] \leq e|A| + e|B|\}.$$

Encontradas todas as sobreposições consistentes com o *layout*, estas precisam ser intercaladas por meio de um alinhamento múltiplo de suas seqüências. Como no caso de um alinhamento simples (envolvendo apenas duas seqüências), podem existir vários alinhamentos múltiplos envolvendo as seqüências representativas das sobreposições e, na busca por um resultado final bem próximo à cadeia de DNA sendo montada, o FAKtory precisa determinar um alinhamento condizente com as arestas em C. Dada uma aresta $e \in C$, resultado de um alinhamento entre $A = a_1a_2...a_m$ e $B = b_1b_2...b_n$, e pode ser representada por um conjunto de pares de sobreposições $(i_1, j_1), (i_2, j_2), ..., (i_k, j_k)$, onde um par (i, j) representa o emparelhamento dos caracteres a_i e b_j. Cada um destes pares é então tratado como uma restrição ao alinhamento buscado, no sentido de que ambos os caracteres do par devem aparecer na mesma posição do alinhamento.

Infelizmente, nem sempre é possível obter um alinhamento cujas posições contenham corretamente todos os pares das sobreposições presentes em $\mathcal{C}$ e busca-se então um subconjunto de restrições que possa ser satisfeito por um dado alinhamento e que possua peso total máximo, definido como sendo a soma dos pesos de cada par (i, j), peso este atribuído de acordo com a similaridade entre a_i e b_j.

Para verificarmos se um dado subconjunto de pares (i, j) pode ou não ser satisfeito por um alinhamento múltiplo, definimos um outro tipo de grafo $H = (V, E, \prec)$, denominado de **grafo de alinhamento**, onde os vértices em V correspondem a caracteres das seqüências sendo alinhadas e cujas arestas em E representam pares de caracteres emparelhados no alinhamento. A relação $\prec$ determina uma ordem parcial entre os vértices de H, onde $v \prec w$ se v e w são caracteres de uma mesma seqüência S e o caracter v precede w em S.

Em um grafo de alinhamento, qualquer subconjunto de arestas $T \subseteq E$ induz uma coleção de componentes conexas que particionam V. Dados dois componentes X e Y, $X \prec^* Y$ se existe algum $x \in X$ e algum $y \in Y$, tal que $x \prec y$. A relação $\prec^*$ pode não ser uma relação de ordem parcial, ou seja, é possível que $X \prec^* Y$ e $Y \prec^* X$ mesmo que Y seja diferente de X. Quando a relação $\prec^*$ sobre os componentes induzidos por T for de ordem parcial, então todas as restrições determinadas por este subconjunto de arestas podem ser satisfeitas: cada componente corresponde a uma posição do alinhamento, e qualquer ordenação topológica[4] dos componentes que respeite $\prec^*$ é uma ordenação

[4]Uma ordem topológica de um conjunto S com ordem parcial $\prec$ é uma ordenação total dos elementos de S que respeita $\prec$.

válida daquelas posições.

Uma ordem topológica existe quando a relação $\prec^*$ não contém um ciclo. Em outras palavras, as restrições em T podem ser satisfeitas se e somente se a relação $\prec^*$ sobre os componentes de T for acíclica. Dado então um conjunto de restrições T satisfazível, pode-se construir um alinhamento múltiplo correspondente através da determinação de suas componentes conexas e posterior ordenação topológica destas componentes.

O problema de se encontrar um alinhamento adequado a um certo *layout* pode então ser assim definido:

Problema da restrição de peso máximo Dado um grafo de alinhamento $H = (V, E, \prec)$, com função w de peso nas arestas, encontrar um conjunto de restrições $T \subseteq E$ satisfazíveis com $\sum_{e \in T} w(e)$ máximo.

O problema acima também é NP-completo[26] e o FAKtory utiliza-se de uma heurística que busca por uma árvore geradora de peso máximo em um grafo para encontrar uma solução próxima da ótima. Note que, no grafo de alinhamento H, as arestas ligando quaisquer duas seqüências formam um conjunto de restrições satisfazíveis. Já que qualquer árvore envolvendo estas arestas determina um alinhamento múltiplo das seqüências, poderíamos selecionar aquelas arestas em $\mathcal{C}$, atribuir um peso a elas de acordo com a similaridade dos caracteres emparelhados na sobreposição e, sobre estas arestas, determinar uma árvore geradora de peso máximo.

Tendo-se em mãos o alinhamento múltiplo das seqüências, a seqüência final s é obtida analisando-se cada uma de suas posições e escolhendo-se como o primeiro caracter de s aquele que aparece pelo menos $n/2$ vezes na primeira posição do alinhamento, onde n corresponde ao número de seqüências sendo alinhadas naquela posição. O segundo caracter de s é obtido da mesma forma, analisando-se desta vez a segunda posição do alinhamento, e assim por diante até que s esteja totalmente determinada.

3.4 Phrap

Último sistema de montagem a ser descrito, o **Phrap**[18, 19] foi desenvolvido por Phil Green na Universidade de Washington. Este programa é um dos mais utilizados atualmente e faz parte de um pacote denominado **swat/cross-match/phrap**, que inclui além do Phrap, os programas **Swat** e **Cross-match**, utilizados para comparação de duas os mais seqüências e busca de seqüências em bases de dados.

Infelizmente, não encontramos referências com informações detalhadas a respeito do(s) algoritmo(s) utilizados pelo Phrap durante o processo de montagem. Devido a isto, nesta seção daremos apenas uma breve descrição de algumas características da ferramenta e dos passos executados por ela até a geração final dos contigs.

3.4.1 Características do Phrap

As principais características do **Phrap** relacionadas com o processo de montagem são as seguintes:

1. Utilização de informações relacionadas à qualidade dos dados de entrada que permitem calcular a probabilidade das bases de cada fragmento terem sido mal determinadas. Isto auxilia o Phrap no reconhecimento de fragmentos provenientes de regiões repetidas e possibilita a geração de uma seqüência consenso mais próxima da seqüência original.

2. O fragmento por inteiro é utilizado pelo Phrap no processo de montagem. Outras ferramentas necessitam que os fragmentos da entrada sejam "podados" no intuito de eliminar seqüências de bases com baixa qualidade presentes neles. Este processo tende a excluir uma quantidade significativa de dados da montagem inicial, exigindo mais tarde um esforço manual na tentativa de juntar contigs distintos e resolver certas ambigüidades presentes na seqüência consenso. O Phrap minimiza ou elimina a necessidade de qualquer intervenção manual no processo de montagem e, mais importante que isso, a utilização dos fragmentos como um todo aumenta as chances da ferramenta identificar corretamente aqueles provenientes de regiões repetidas da molécula.

3. Utilização do algoritmo de Smith-Waterman para buscar um alinhamento local ótimo entre os dados da entrada.

4. Não se utiliza de nenhuma suposição biológica a respeito da seqüência sendo montada.

5. Fragmentos quimeras e seqüências do organismo vetor são identificados automaticamente.

6. As seqüências relacionadas aos contigs são determinadas como um "mosaico" das partes de melhor qualidade dos fragmentos.

7. Informações a respeito da exatidão e unicidade da montagem são geradas pelo Phrap. Possíveis locais onde alguma montagem errada tenha ocorrido ou onde dados adicionais fazem-se necessários também são sinalizados pela ferramenta.

3.4.2 Descrição geral do funcionamento do Phrap

Como as outras ferramentas já vistas, o **Phrap** inicia o processo de montagem dos fragmentos através da leitura dos fragmentos de entrada, juntamente com dados

referentes à sua qualidade. Tendo em mãos os fragmentos, o Phrap determina então seus respectivos complementos reversos e começa a buscar por pares de fragmentos sobrepostos entre si. Isto é realizado da seguinte maneira: primeiramente, qualquer região do início ou final de um fragmento consistindo de um único caracter é convertida para uma seqüência de 'N's. Estas regiões possuem uma alta probabilidade de terem sido mal seqüenciadas e precisam ser identificadas para evitar-se que falsas sobreposições sejam consideradas pelo algoritmo. Todas as sobreposições de tamanho $x \geq minmatch$ (onde $minmatch$ corresponde a um parâmetro de entrada definido pelo usuário) são então determinadas pelo Phrap através dos seguintes passos:

(i) Construção de uma lista de apontadores para cada posição das seqüências que inicia uma palavra com pelo menos $minmatch$ letras diferentes de 'N'.

(ii) Ordenação da lista.

(iii) Busca na lista ordenada por pares de palavras sobrepostas.

(iv) Definição de uma banda de tamanho específico para cada par de palavras sobrepostas.

(v) Intercalação de bandas sobrepostas.

(vi) Busca recursiva sobre as bandas por segmentos sobrepostos cuja pontuação do alinhamento seja maior ou igual a $minscore$ (outro parâmetro que pode ser definido pelo usuário).

Antes de iniciar o processo de construção dos contigs, o Phrap precisa encontrar seqüências dentro dos fragmentos que correspondem a seqüências da molécula de DNA do organismo vetor utilizado para replicação do DNA sendo montado. Encontradas estas seqüências, elas são marcadas para que não sejam utilizadas durante o processo de montagem. Além disso, antes de construir os contigs, o **Phrap** determina também os pares de fragmentos cujas sobreposições não são relevantes, evitando com isso montagens erradas ao final do processo.

Através dos valores de qualidade associados aos fragmentos de entrada e dos alinhamentos entre eles, o **Phrap** determina valores de qualidade para cada posição dos fragmentos, que serão utilizados posteriormente na determinação de sobreposições provenientes de regiões repetidas da molécula. O valor de qualidade para cada base é calculado da seguinte forma: se uma dada posição de um fragmento f é confirmada quando este é comparado a uma seqüência representante do complemento reverso de um fragmento f', àquela posição é atibuído um valor igual a soma das qualidades dos dois fragmentos. Caso uma dada posição seja confirmada por mais de uma seqüência, somente o maior valor de qualidade de todos os fragmentos é tomado como o valor de

qualidade daquela posição. Se considerarmos que os valores de qualidades estão relacionados com probabilidade de erros durante o seqüenciamento dos fragmentos, este procedimento pode ser interpretado como o cálculo de uma probabilidade de erro para cada base, que é o produto das probabilidades de erros dos dois fragmentos envolvidos.

Os valores de qualidade associados a cada base são então utilizados para o cálculo do valor LLR (*log-likelihood ratios*) de cada sobreposição entre dois fragmentos. Este valor corresponde a uma taxa utilizada para determinar se uma dada sobreposição é verdadeira (valor LLR positivo), ou se ela provém de uma região repetida da molécula (valor LLR negativo).

Após isso, prováveis quimeras são identificados pelo Phrap como fragmentos para os quais a parte confirmada pode ser separada em dois pedaços não sobrepostos, com no máximo *maxchimeragap* (outro parâmetro de entrada do programa) entre eles, tal que a parte confirmada por um dado fragmento encontre-se em apenas um dos pedaços. Além destas condições, cada pedaço precisa ter um alinhamento extenso com outro fragmento. Fragmentos com dois pedaços confirmados não sobrepostos, mas que não atendem à condição de alinhamento extenso com outro fragmento geralmente são a única ligação entre dois contigs não sobrepostos, e devido a isso precisam ser considerados durante o processo de montagem.

Através da estratégia gulosa, o Phrap inicia então a construção dos contigs utilizando-se das sobreposições relevantes em ordem decrescente dos valores LLR. Isto é feito construindo-se um grafo dirigido com peso nas arestas cujos vértices correspondem a determinadas posições dos fragmentos; existem arestas bidirecionais com peso 0 entre bases alinhadas dos fragmentos sobrepostos e arestas unidirecionais da posição $5'$ à posição $3'$ de um único fragmento, com peso igual à qualidade total da seqüência entre os dois vértices. Realiza-se então uma busca neste grafo por um caminho de peso máximo, cujos vértices constituirão o contig sendo criado.

Capítulo 4

Casos de Testes e Programas Auxiliares

Antes de mostrarmos os resultados obtidos por cada uma das ferramentas, falaremos neste capítulo a respeito dos casos de testes aos quais as ferramentas foram submetidas. Aqui, comentaremos também a respeito de três programas auxiliares utilizados durante o desenvolvimento do projeto, um para geração de fragmentos sobrepostos e inserção de erros em fragmentos e dois para comparação de seqüências.

4.1 Casos de Testes

Os sistemas de montagem foram avaliados tomando-se como entrada doze conjuntos de fragmentos sobrepostos, sendo que dez deles foram obtidos das seguintes seqüências que temos em mãos:

- Cosmídeo B1496 de *Mycobacterium leprae*
- Cosmídeo B2126 de *Mycobacterium leprae*
- Cosmídeo L247 de *Mycobacterium leprae*
- Cosmídeo L518 de *Mycobacterium leprae*

Algumas informações a respeito das seqüências acima podem ser vistas na tabela 4.1 e, vale salientar, que quatro dos dez conjuntos obtidos dos cosmídeos acima incluem fragmentos reais, ou seja, fragmentos que foram utilizados durante o projeto de

Seqüência	No. de bases	No. de regiões repetidas	Similaridade das cópias
B1496	41295	1	97.02
B2126	44566	1	99.87
L247	44438	1	99.84
L518	41412	1	99.93

Tabela 4.1: Informações a respeito das seqüências utilizadas

seqüenciamento daqueles DNAs. Os seis conjuntos restantes contêm fragmentos artificialmente obtidos das seqüências em questão por meio do programa *Genfrag*[14, 13], que veremos mais adiante.

Os dois últimos casos de testes incluem fragmentos artificiais provenientes da seqüência original do agregado *β-like* da globina do gene humano (73308 bp) e daquela descrita em [37] (36486 bp). Estas duas seqüências estão disponíveis no **GenBank**[4], uma base de dados contendo informações detalhadas a respeito de várias moléculas de DNA.

4.2 Programas Auxiliares

Para obtenção de um conjunto de fragmentos sobrepostos das seqüências acima citadas, utilizamos um pacote de programas conhecido como *Genfrag*, que inclui além do programa gerador de fragmentos (*enzyme*), um programa para inserção de erros nos fragmentos (*mutate*) e outro para inserção de repetições dentro das seqüências originais (*prepseq*). Cada um destes programas possui um conjunto de parâmetros de entrada específicos (tamanho médio dos fragmentos, taxa de erros, porcentagem de repetições etc.), permitindo um bom controle dos seus dados de saída.

O programa *enzyme* recebe como entrada uma dada seqüência de DNA e retorna como saída um conjunto de fragmentos sobrepostos daquela seqüência de acordo com certos parâmetros estabelecidos pelo usuário (tamanho médio dos fragmentos, cobertura etc.). Além do conjunto de fragmentos em si, faz parte também da saída do programa um arquivo contendo informações a respeito das orientações e posições dos fragmentos com relação à seqüência original. Chamando-se, por exemplo, o programa *enzyme* para um arquivo contendo a seqüência da Figura 4.1(a), com os valores 115 e 7 para os parâmetros de tamanho médio dos fragmentos e cobertura respectivamente, obtemos um total de 13 fragmentos sobrepostos. Três destes fragmentos são mostrados na Figura 4.1(b)[1] e suas orientações e posições com relação à seqüência original são gravadas pelo *enzyme* num arquivo de formato semelhante ao mostrado na Figura 4.1(c).

[1] A forma com que os fragmentos estão aí dispostos segue o formato conhecido como FASTA, que corresponde ao mais utilizado pelas ferramentas de montagem e comparação de seqüências.

(a)

```
GATCTTCCTGGAAAGGTGAGGAGATGGACACATAGTATATAAACACAGGTAATCAGCCATGTGGATGAATG
TAGATTAAAATAAATGGGTTATCTTAAATTGTGGTTCTAGTCAGAGAAGAGCCTAGCTATATGGTCAAGGT
GTTTGTAAATATATTTTGAGTCCTTGTCTTATTTCTGGTGAAGAGGTCAGCTCAGGACCTTTGGAAAA
```

(b)

```
>frag0000
TGTGGTTCTAGTCAGAGAAGAGCCTAGCTATA
TGGTCAAGGTGTTTGTAAATATATTTTGAGTC
CTTGTCTTAT
>frag0001
GATCTTCCTGGAAAGGTGAGGAGATGGACACA
TAGTATATAAACACAGGTAATCAGCCATGTGG
ATGAATGTAGATTAAAATAAATGGGTTATCTT
AAATTGTGGTTCTAGTCAGAGAAGAGCCTAGC
TATATGGTCAAGGTGTTTGT
>frag0002
CAAAATATATTTACAAACACCTTGACCATATA
GCTAGGCTCTTCTCTGACTAGAACCACAATTT
AAGATAACCCATTTATTTTAATCTACATTCAT
CCACATGGCTGATTACCTGTGTTTATATACTA
TGTGTCCATCTCCTCACCTTT
```

(c)

```
# ENZYME (1.12)
# Input parameters
enzyme.MeanFragment:  115
enzyme.MaxMinBound:  37.50
enzyme.DepthOfCoverage:  7.00
enzyme.RNGseed:  1
enzyme.FrequencyDistribution:  No
# Parent length
p 210
# Start Stop
n 101 174
n 1 148
r 160 12
r 210 117
r 110 1
```

Figura 4.1: Exemplo de entrada e saída do *enzyme*

O programa *mutate* recebe como entrada um conjunto de fragmentos e, conforme um arquivo de distribuição de erros, também fornecido como entrada ao programa, altera certas bases presentes nestes fragmentos. Os parâmetros desse programa incluem a taxa de erros desejada e o modo de alteração das bases (substituições e/ou inserções e deleções). Chamando-se, por exemplo, o programa *mutate* para os fragmentos da Figura 4.1(b), com uma taxa de erros de 15%, obtemos o resultado mostrado na Figura 4.2.

```
>frag0000
TGGGTTCTAGTCAGAGAAGGCCTAnGCTATAGtTtAGGTtcTTGTAtAcA
TAgTTTTcAGTCCTTGTCTTAT
>frag0001
aATCcTCCTGGAAGGTGAGGGATGGAACATAGTATATTtAACACaAGAtTA
ATCAaGCaAaTGTGGATGAATGTgGATTAcAAAAAATGGGTTATCaAAAT
TcGTGGTTCcAGTCAGaAGAAGgGCTAGCTgTATnGtTCAAGGTTTTGT
>frag0002
CAAAAatTaATTTACAAACgCCTGACaAnATAGCTAGGCTCTaCaTCaGA
gTAGAACCACAgTTAGATAgCCCATTTATaTTAATCTAaATTCATCaACA
TGGCTGATTACCTcTTTTATATACTATGTGTCtAgCTCCcCCTTT
```

Figura 4.2: Exemplo de saída do *mutate*

O programa *prepseq* recebe como entrada uma dada seqüência de DNA e um arquivo contendo certa subseqüência de bases a ser inserida de forma repetitiva em posições aleatórias daquela seqüência. A freqüência destas inserções é estabelecida dentro desse arquivo e o programa retorna como saída a seqüência original modificada, o número de repetições inseridas e a posição de cada uma delas dentro da seqüência original. Poderíamos então, através do *prepseq*, inserir o conjunto de bases *gtgtgtgtgt* na seqüência da Figura 4.1(a) e obtermos o seguinte resultado como saída do programa:

```
GATCTTCCTGGAgtgtgtgtgtAAGGTGAGGAGATGGACACATAGTATATAAACACAGGTAATCAGCCATGTGGATG
AATGTAGATTAAAATAAATGgtgtgtgtgtGGTTATCTTAAATTGTGGTTCTAGTCAGAGAAGAGCCTAGCTATATG
gtgtgtgtgtGTCAAGGTGTTTGTAAATATATTTTGAGTCgtgtgtgtgtCTTGTCTTATTTCTGGTGAAGAGGTCA
GCTCAGGACCgtgtgtgtgtTTTGGAAAA
```

Para avaliarmos a exatidão dos resultados gerados por cada ferramenta, comparamos as seqüências consensos relacionadas a cada um de seus contigs com a seqüência original de onde os fragmentos foram obtidos. Para isso, utilizamo-nos de um programa de alinhamento denominado **GAP**[22]. Este programa realiza um alinhamento global entre duas seqüências, mostrando como saída as posições iniciais e finais deste alinhamento, as colunas contendo erros e aquelas onde os espaços foram inseridos. Supondo que uma dada ferramenta monte os fragmentos da Figura 4.1(b) em um contig cuja seqüência é mostrada na Figura 4.3, parte do resultado de seu alinhamento com a seqüência original (Figura 4.1(a)) seria mostrada pelo GAP como indicado na Figura 4.4.

```
>Contig
ATAAGACAAGGACTCAAAATATATTTACAAACACCTTGACCATATAGCTAGGCTCTTCTCTGACTAGAACCACAAT
TTAAGATAACCCATTTATTTTAATCTACATTCATCCACATGGCTGATTACCTGTGTTTATATACTATGTGTCCATC
TCCTCACCTTTCCAGGAAGATC
```

Figura 4.3: Contig gerado por uma dada ferramenta

```
  1  ATAAGACAAG GACTCAAAATATATTTACAAACACCT TGACCATATAG
    -|| || || |-|  | ||  || ||| | | | |   -| | | || |
 51 AATCAGCCATGTGGATGAATGTAGATTAAAATAAATGGGTTATCTTAAAT
100    .    :    .    :    .    :    .    :    .    :
 48 CTAGGCTCTTCTCTGACTAGAACCACAATTTAAGATAACCCATTTATTTT
       || |||  || ||  ||| ||    | || | | |     ||-| |
101 TGTGGTTCTAGTCAGAGAAGAGCCTAGCTATATGGTCAAGGTGTT TGTA
```

Figura 4.4: Saída gerada pelo GAP

Para uma análise mais detalhada dos contigs gerados pela ferramentas, um segundo programa de alinhamento, denominado *cross_match*, fora utilizado por nós. Este

programa pertence ao pacote da ferramenta de montagem **Phrap** e realiza um alinhamento local entre seqüências, permitindo-nos identificar falhas na montagem decorrentes da presença de fragmentos quimeras e de regiões repetidas na seqüência sendo reconstruída. A saída do *cross_match* inclui a identificação dos contigs sendo comparados com a seqüência original, a posição inicial e final do alinhamento das subseqüências destes contigs com as subseqüências da seqüência original e as diferenças de bases observadas nestes alinhamentos (número de substituições, inserções e deleções). Pegando-se novamente o contig da Figura 4.3, o resultado de sua comparação, através do *cross_match*, com a seqüência mostrada na Figura 4.1(a), aparece como na Figura 4.5, onde o 'C' determina que o alinhamento fora realizado com o complemento reverso da seqüência original.

```
Contig1
166  0.00 0.00 0.00  Contig1  1   174 (0)  C exemplo.con   (36)   174     1
1 matching entries (first file).
Discrepancy summary:
sub del ins  total (\%)
0   0   0     0    (0.00)
```

Figura 4.5: Saída gerada pelo cross_match

Capítulo 5

Resultados dos Testes

Neste capítulo descrevemos os resultados da execução das ferramentas sobre os doze casos de testes que temos em mãos. Informações a respeito de cada um desses casos serão mostradas nas seções correspondentes e a avaliação das ferramentas será feita tomando-se como parâmetro, basicamente, o número de contigs gerados por cada uma delas, a porcentagem de cobertura da seqüência original por estes contigs e a quantidade total de erros (substituições) e buracos (inserções e remoções) presentes em seus alinhamentos com aquela seqüência.

Vale salientar que os resultados mostrados nesta seção foram obtidos executando-se todas as ferramentas com os valores *defaults* para seus parâmetros de entrada. Em outras palavras, executamos os programas cedendo-lhes apenas a identificação dos conjuntos de fragmentos a serem montados e deixando que eles utilizassem valores próprios para parâmetros do tipo tamanho mínimo de uma sobreposição válida, número máximo de buracos permitidos dentro de um sobreposição válida etc. Adotamos esta prática principalmente devido à falta de uniformidade no número e tipo dos parâmetros de entrada de cada sistema de montagem que temos em mãos.

Os valores dos resultados associados a cada ferramenta serão mostrados de duas formas. Primeiramente, através de tabelas, onde cada coluna representa um dos dados acima mencionados e, de acordo com a seção, outras informações como os fragmentos quimeras detectados ou as partes não cobertas da seqüência sendo reconstituída. Para uma melhor comparação dos resultados obtidos por cada ferramenta, os mostraremos também por meio de gráficos, cujos eixos das abscissas representam a seqüência original sendo montada e as setas horizontais situadas no primeiro quadrante destes gráficos representam os contigs produzidos por cada uma das ferramentas ao final do processo de montagem. A projeção destas setas no eixo das abscissas identifica a região da seqüência original coberta por eles, valendo salientar que aquelas apontando para a direita cobrem parte da seqüência em sua orientação normal, enquanto que aquelas

apontando para a esquerda cobrem parte da seqüência em sua orientação reversa.

5.1 Erros no Seqüenciamento

Apesar da tecnologia de seqüenciamento ter avançado muito nos últimos anos, de 0.5% a 5.0% do total das bases componentes de um dado fragmento podem ter sido mal determinadas ao término do processo de seqüenciamento via algum método direto. Estes erros compreendem substituições, inserções e remoções de bases e tendem a ocorrer nas extremidades dos fragmentos. Isto pode impedir o reconhecimento de sobreposições relevantes pelas ferramentas de montagem ou até mesmo levá-las a considerar como verdadeiras sobreposições entre fragmentos provenientes de regiões distintas da molécula. Estas possibilidades tornam os erros de seqüenciamento um problema merecedor de tratamento especial pelos sistemas de montagem.

Nesta primeira seção de testes avaliamos a capacidade que cada uma das ferramentas possui em lidar com a existência de "falsas" bases dentro das seqüências dos fragmentos. Para isso, contamos com os quatro casos de testes relacionados na Tabela 5.1, juntamente com o número de fragmentos que cada um deles possui (No. de fragms), o tamanho médio destes fragmentos (Tam. médio dos fragms) e a taxa de erros que incorporam (Taxa de erros). Estes casos foram obtidos através da inserção de erros, por meio da ferramenta *mutate*, nos conjuntos de fragmentos reais utilizados durante o projeto de seqüenciamento dos cosmídeos B1496, B2126, L247 e L518 (de acordo com [23], nenhum dos fragmentos componentes destes conjuntos possui erros de seqüenciamento).

Caso de teste	No. de fragms	Tam. médio dos fragms	Taxa de erros
Cosmídeo B1496	1811	233	4.45%
Cosmídeo B2126	1504	232	1.14%
Cosmídeo L247	1704	245	3.15%
Cosmídeo L518	1910	227	2.02%

Tabela 5.1: Testes utilizados para avaliação das ferramentas no tratamento de erros

Como foi dito anteriormente, o *mutate* utiliza-se de um arquivo de distribuição de erros para determinar as posições de inserção, remoção e substituição de bases dentro da seqüência. O arquivo *default* utilizado por este programa induz a uma taxa de erros de aproximadamente 1.5% e, no intuito de analisarmos o comportamento das ferramentas com relação a um número maior de erros, multiplicamos os valores presentes naquele arquivo por certas constantes, obtendo desta forma as taxas adicionais de 2.02%, 3.15% e 4.45% mostradas na tabela acima.

Comparando-se os resultados obtidos pelas ferramentas com a seqüência original dos cosmídeos acima citados tentaremos determinar quais delas tratam da melhor forma o problema de erros no seqüenciamento. Além disso, faremos também uma comparação entre os resultados gerados por cada sistema de montagem ao serem executados sobre o conjunto de fragmentos reais e sobre aqueles contendo erros. Com isto, tentaremos avaliar o quanto a presença de erros no seqüenciamento influi na exatidão dos resultados obtidos por cada uma das ferramentas.

5.1.1 Cosmídeo B2126 de Mycobacterium leprae

A execução das ferramentas sobre o conjunto de fragmentos reais utilizados no seqüenciamento do cosmídeo B2126 teve como resultados os valores mostrados na Tabela 5.2 abaixo.

Ferramentas	Contigs	Cobertura(%)	Erros	Buracos
CAP2	1	100	0	7
FAKtory	3	97.21	4	41
TIGR	3	97.54	0	6
Phrap	3	96.75	0	5

Tabela 5.2: Resultados das execuções sobre o conjunto de fragmentos reais do cosmídeo B2126

Após a inserção de um total de 3955 erros, distribuídos de acordo com a Tabela 5.3, nos fragmentos citados acima, as ferramentas obtiveram os resultados mostrados na Tabela 5.4 (Gráfico 5.1).

No. de substituições	No. de inserções	No. de remoções
2174	893	888

Tabela 5.3: Distribuição dos erros nos fragmentos do cosmídeo B2126

Ferramentas	Contigs	Cobertura(%)	Erros	Buracos
CAP2	3	97.40	2	5
FAKtory	3	97.39	22	881
TIGR	8	98.99	101	34
Phrap	3	97.25	7	10

Tabela 5.4: Resultados das execuções sobre o conjunto de fragmentos contendo erros do cosmídeo B2126

Esta tabela mostra que o CAP2 e o Phrap obtiveram os melhores resultados no que diz respeito ao número de erros e buracos presentes no alinhamento de seus contigs com a seqüência original. Diferente destas duas ferramentas, o TIGR e o FAKtory apresentaram taxas consideráveis de erros e buracos, respectivamente.

Apesar da existência de quase 4000 bases erradas dentro dos fragmentos deste caso de teste, o CAP2 conseguiu montá-los em três contigs cujos alinhamentos com a seqüência original acumulam apenas 2 erros e 5 buracos. Comparando-se os valores das tabelas 5.2 e 5.4, podemos ver que os erros inseridos mudaram de forma desprezível o total de erros e buracos apresentados pelo CAP2 quando da execução desta ferramenta sobre o conjunto de fragmentos reais, mas determinaram uma queda significativa no valor de cobertura apresentado por ela.

Com relação ao Phrap, seus três contigs também possuem bons alinhamentos com a seqüência original (que contêm apenas 7 erros e 10 buracos), mas a cobertura determinada por estes contigs é a menor dentre os valores apresentados pelas quatro ferramentas. O que mais nos chamou atenção, no entanto, foi o fato de que, apesar de ser o menor valor mostrado na Tabela 5.4, a cobertura obtida pelo Phrap ao término de sua execução sobre o conjunto contendo erros é maior que aquela obtida quando da execução dessa ferramenta sobre o conjunto real. Ao contrário do que ocorreu com o CAP2, os erros parecem ter influenciado de forma positiva o processo de montagem pelo Phrap no que diz respeito à cobertura obtida pela ferramenta.

Pela Tabela 5.4, podemos ver que o FAKtory corresponde à ferramenta cujos contigs determinam o maior número de buracos em seus alinhamentos com a seqüência original. Além disso, estes alinhamentos acumulam também uma quantidade de erros considerável quando comparada àquelas obtidas pelo Phrap e pelo CAP2. Estas falhas concentram-se principalmente nos contigs 1 e 2 construídos pela ferramenta e ocorrem devido a incertezas do FAKtory durante a fase de alinhamento múltiplo das seqüências dos fragmentos sendo montados, incertezas estas que, por sua vez, estão relacionadas com os erros presentes neste caso de teste. Assim como ocorreu com o Phrap, esses erros parecem também ter influenciado positivamente o FAKtory no que diz respeito à cobertura obtida pela ferramenta.

Com relação ao TIGR, esta ferramenta obteve a melhor cobertura dentre as quatro avaliadas, mas o alinhamento de seus contigs acumulam o maior número de erros frente aos valores obtidos pelo CAP2, Phrap e FAKtory. Comparando-se os valores das tabelas 5.2 e 5.4, podemos ver que o TIGR apresentou resultados bem diferentes após a inserção de erros nos fragmentos componentes do caso de teste em questão. Além da mudança significativa nos valores de erros e buracos apresentados, as "falsas" bases ocasionaram também uma variação considerável no número de contigs construídos pela ferramenta.

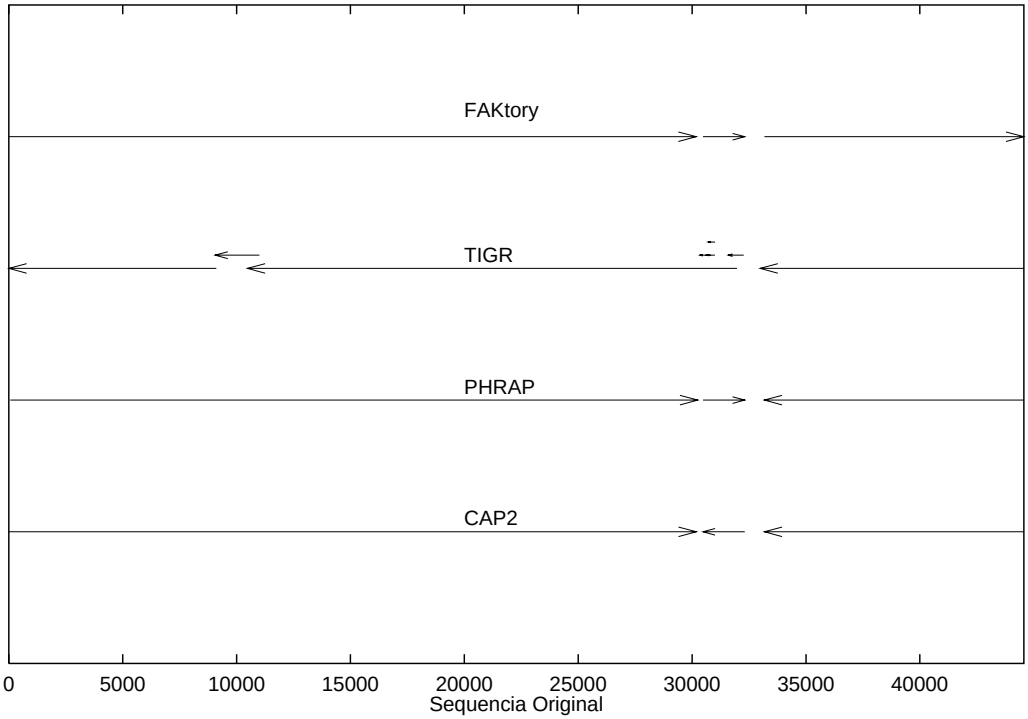

Figura 5.1: Gráfico da execução sobre o B2126

5.1.2 Cosmídeo L518 de Mycobacterium leprae

Na Tabela 5.5 mostramos os resultados da execução das ferramentas sobre o conjunto de fragmentos reais obtidos do cosmídeo L518 de *Mycobacterium leprae*.

Ferramentas	Contigs	Cobertura(%)	Erros	Buracos
CAP2	1	99.73	44	500
FAKtory	8	92.99	741	840
TIGR	#	#	#	#
Phrap	2	96.89	31	500

Tabela 5.5: Resultados das execuções sobre o conjunto de fragmentos reais do cosmídeo L518

Um fato estranho ocorrido durante essas execuções foi que, por algum motivo desconhecido, o TIGR ocasionou a geração de uma mensagem de *bus error!* pelo sistema operacional, impedindo-nos de verificar seus resultados.

Ao inserirmos um total de 8794 erros (distribuídos de acordo com a Tabela 5.6) nos

fragmentos deste caso de teste, poucas mudanças significativas ocorreram nos resultados gerados pelas ferramentas. Os valores obtidos após a execução sobre o conjunto de fragmentos contendo erros podem ser vistos na Tabela 5.7 (Gráfico 5.2).

No. de substituições	No. de inserções	No. de remoções
4856	1939	1999

Tabela 5.6: Distribuição dos erros nos fragmentos do cosmídeo L518

Ferramentas	Contigs	Cobertura(%)	Erros	Buracos
CAP2	1	99.72	45	502
FAKtory	6	82.04	2315	3115
TIGR	#	#	#	#
Phrap	2	96.89	33	520

Tabela 5.7: Resultados das execuções sobre o conjunto de fragmentos contendo erros do cosmídeo L518

De acordo com os resultados apresentados na Tabela 5.7, o CAP2 mostrou novamente ser a melhor ferramenta no que diz respeito ao tratamento de erros. Apenas um contig foi gerado pelo CAP2 ao final do processo de montagem, contig este que cobre quase 100% da seqüência original e cujo alinhamento com esta acumula 45 erros e 502 buracos. Apesar destes valores parecerem altos, eles somam apenas 1 erro e 2 buracos aos números apresentados pela ferramenta na Tabela 5.5, como se os 8794 erros inseridos nos fragmentos não tivessem nenhuma influência significativa no processo de montagem executado pelo CAP2.

Com resultados muito parecidos ao do CAP2, o Phrap também tratou de forma eficiente os erros presentes neste caso de teste. Apesar de seus dois contigs não cobrirem tão bem a seqüência original como o construído pelo CAP2, a quantidade de erros e buracos presentes em seus alinhamentos é baixa quando comparada ao total de erros inseridos nos fragmentos e ao número de erros e buracos existentes no alinhamento dos dois contigs gerados pela ferramenta quando da execução sobre o conjunto de fragmentos reais.

Além de gerar um número maior de contigs que o CAP2 e o Phrap, o FAKtory também apresentou uma quantidade bem maior de erros e buracos frente às taxas relacionadas a essas duas ferramentas. Assim como no caso anterior, este fato está diretamente relacionado às incertezas desta ferramenta durante a fase de alinhamento múltiplo e geração da seqüência consenso de seus contigs. Além de ocasionar um aumento considerável no número de erros e buracos, as bases erradas presentes neste

caso de teste também determinaram uma queda significativa na cobertura dos contigs gerados pelo FAKtory, como pode ser visto comparando-se os valores das tabelas 5.5 e 5.7.

Como no caso da execução sobre os fragmentos reais, novamente o TIGR não finalizou o processo de montagem. A mensagem de *bus error!* gerada na saída padrão e o conseqüente aborto do processo mais uma vez nos impediram de ver os resultados do programa.

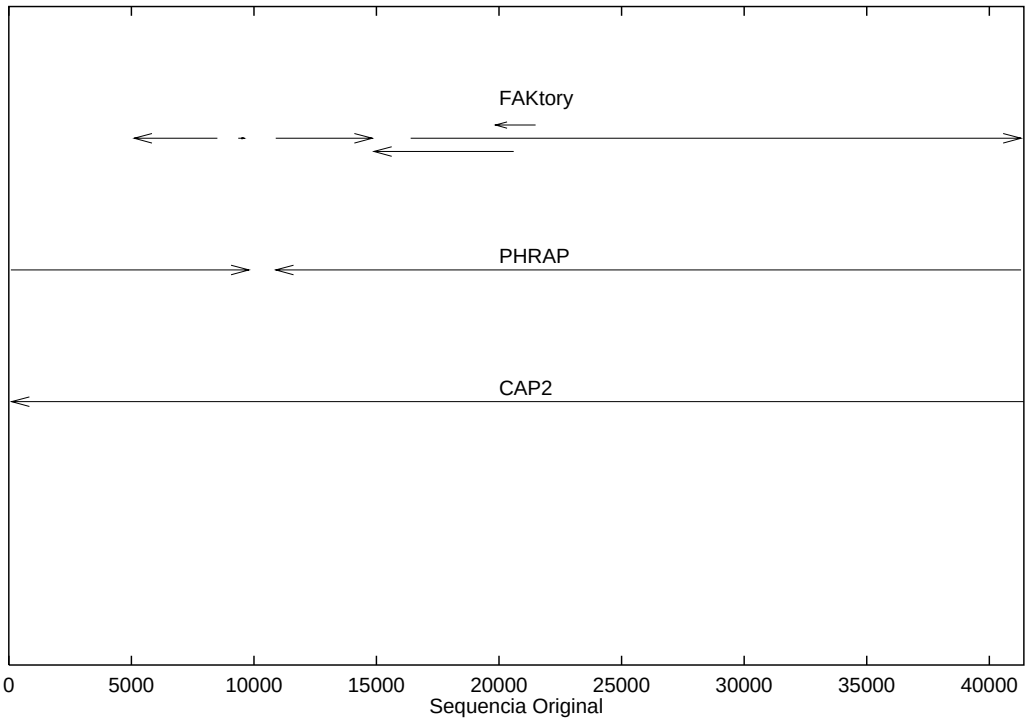

Figura 5.2: Gráfico da execução sobre o L518

5.1.3 Cosmídeo L247 de Mycobacterium leprae

Antes de inserirmos qualquer tipo de erro nos fragmentos relacionados com a seqüência do cosmídeo L247, as ferramentas obtiveram os resultados mostrados na Tabela 5.8.

Na Tabela 5.10 (Gráfico 5.3[1]) mostramos os resultados obtidos após a inserção de um total de 13158 erros, distribuídos de acordo com a Tabela 5.9, nos fragmentos do caso de teste em questão.

[1]Poupamo-nos representar graficamente o resultado obtido pelo TIGR devido ao grande número de contigs produzidos pela ferramenta.

Ferramentas	Contigs	Cobertura(%)	Erros	Buracos
CAP2	1	100	121	1391
FAKtory	2	99.21	81	1373
TIGR	2	100	115	1404
Phrap	1	99.98	114	1390

Tabela 5.8: Resultados das execuções sobre o conjunto de fragmentos reais do cosmídeo L247

No. de substituições	No. de inserções	No. de remoções
7314	2903	2941

Tabela 5.9: Distribuição dos erros nos fragmentos do cosmídeo L247

Ferramentas	Contigs	Cobertura(%)	Erros	Buracos
CAP2	1	100	122	1394
FAKtory	2	99.21	339	4010
TIGR	190	99.62	2160	4223
Phrap	1	99.99	152	1447

Tabela 5.10: Resultados das execuções sobre o conjunto de fragmentos contendo erros do cosmídeo L247

Mais uma vez, o CAP2 e o Phrap foram as ferramentas que melhor trataram o problema de erros no seqüenciamento, com cada um de seus contigs cobrindo praticamente toda a seqüência original a uma taxa de similaridade elevada. Bem diferente dos resultados obtidos por estas ferramentas, o TIGR gerou uma quantidade muito grande de contigs, cujos alinhamentos com a seqüência original acumulam um número elevado de erros e buracos.

Assim como no caso anterior, apenas um contig fora gerado pelo CAP2 ao final do processo de montagem. Este contig cobre toda a seqüência original e seu alinhamento com esta acumula 123 erros e 1394 buracos, valores estes muito bons se repararmos que a execução desta ferramenta sobre o conjunto de fragmentos reais gerou também um único contig que acumula praticamente o mesmo número de erros e buracos (Tabela 5.8)

Com resultados semelhentes ao do CAP2, o Phrap também se mostrou bastante eficiente no tratamento dos erros inseridos neste caso de teste. Como o CAP2, somente um contig foi gerado pelo Phrap, cobrindo 99.99% da seqüência original e cujo alinhamento acumula um número de erros um pouco maior que aquele do contig gerado pelo

CAP2 (mas também não muito diferente dos apresentados pela ferramenta na Tabela 5.8).

Quanto ao FAKtory, seus resultados comprovam a relativa ineficiência desta ferramenta no tratamento de erros de seqüenciamento. Apesar de gerar apenas dois contigs ao final do processo de montagem, o valor de cobertura obtido pelo FAKtory é o pior dentre os apresentados pelas ferramentas. Além disso, o alinhamento de seus contigs com a seqüência original acumula uma quantidade considerável de erros e, principalmente, de buracos, bem maiores que as apresentadas pela ferramenta após a execução sobre o conjunto de fragmentos reais.

Como ocorreu com a montagem dos fragmentos modificados do cosmídeo B2126, o TIGR também gerou o maior número de contigs ao final da execução sobre este caso de teste. O problema principal, no entanto, é que este número é bastante elevado, assim como a taxa de erros e buracos apresentados pela ferramenta. Além de diminuir a similaridade dos contigs com relação à seqüência original, os erros presentes nos fragmentos deste caso de teste também afetaram a cobertura desta seqüência por estes contigs. Como pode ser visto nas tabelas 5.8 e 5.10, o valor de cobertura é um pouco menor nesta última e isto pode ser devido ao fato de que, dos 1704 fragmentos existentes no caso de teste em questão, 715 não foram utilizados pelo TIGR por causa da quantidade de erros que apresentam.

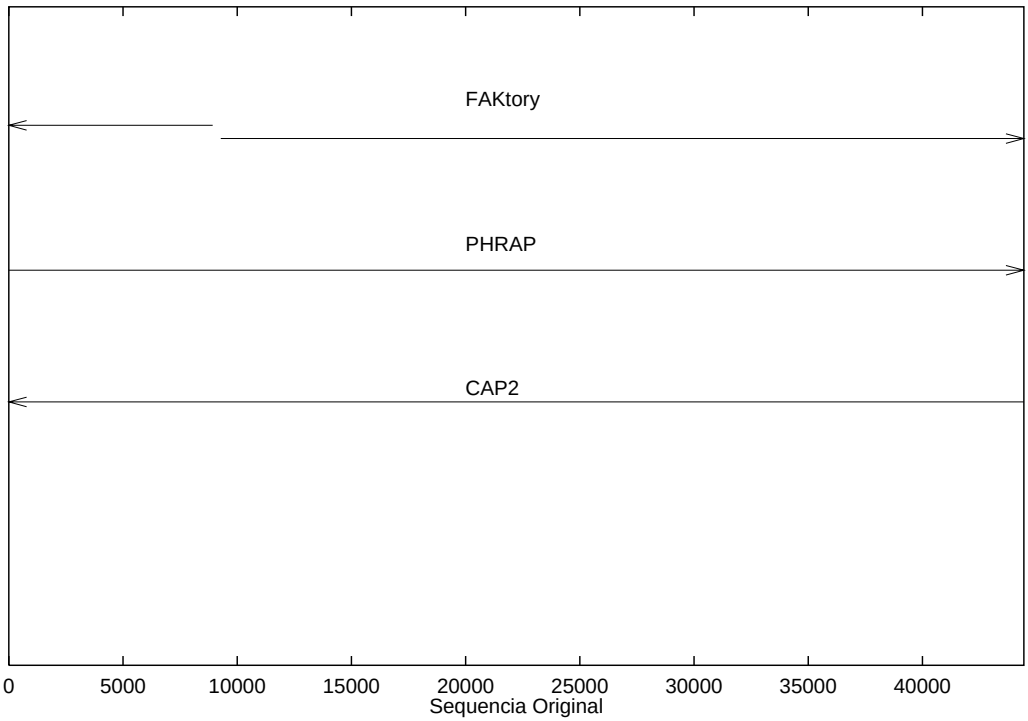

Figura 5.3: Gráfico da execução sobre o L247

5.1.4 Cosmídeo B1496 de Mycobacterium leprae

Último caso de teste utilizado para avaliar o comportamento das ferramentas frente ao problema de erros no seqüenciamento, os fragmentos relacionados ao cosmídeo B1496 possuem um total de 18758 erros, distribuídos de acordo com a Tabela 5.11 mostrada abaixo.

No. de substituições	No. de inserções	No. de remoções
10390	4132	4236

Tabela 5.11: Distribuição dos erros nos fragmentos do cosmídeo B1496

Os resultados da execução das ferramentas sobre o conjunto de fragmentos reais utilizado no seqüenciamento do cosmídeo B1496 e sobre aquele modificado por nós através da inserção de erros são mostrados nas tabelas 5.12 e 5.13 (Gráfico 5.4) abaixo.

Ferramentas	Contigs	Cobertura(%)	Erros	Buracos
CAP2	1	99.43	81	1087
FAKtory	1	99.43	70	1087
TIGR	#	#	#	#
Phrap	1	98.95	81	1080

Tabela 5.12: Resultados das execuções sobre o conjunto de fragmentos reais do cosmídeo B1496

Ferramentas	Contigs	Cobertura(%)	Erros	Buracos
CAP2	1	99.43	93	1091
FAKtory	1	99.42	571	4732
TIGR	#	#	#	#
Phrap	1	98.95	132	1163

Tabela 5.13: Resultados das execuções sobre o conjunto de fragmentos contendo erros do cosmídeo B1496

Comprovando a eficiência do CAP2 no tratamento de possíveis erros de seqüenciamento, a Tabela 5.13 mostra que, apesar do mesmo número de contigs gerados pelas ferramentas, o CAP2 destaca-se pelo fato de seu contig cobrir a maior parte da seqüência original com o menor número de erros e buracos (valores estes um pouco maiores que aqueles mostrados na Tabela 5.12).

Os resultados do Phrap não se distanciam muito daqueles gerados pelo CAP2, ao contrário do que ocorre com o FAKtory. O contig gerado por esta última ferramenta, apesar de cobrir bem a seqüência original, possui um alinhamento com ela que acumula um número de erros e buracos quase quatro vezes maior que a quantidade apresentada pelo CAP2 e pelo Phrap.

Como ocorreu no segundo caso de teste desta seção, não conseguimos avaliar os resultados da execução do TIGR sobre os fragmentos do cosmídeo B1496. Novamente, uma mensagem de *bus error!* foi gerada durante o processamento, por esta ferramenta, tanto do conjunto de fragmentos reais quanto do conjunto de fragmentos modificados.

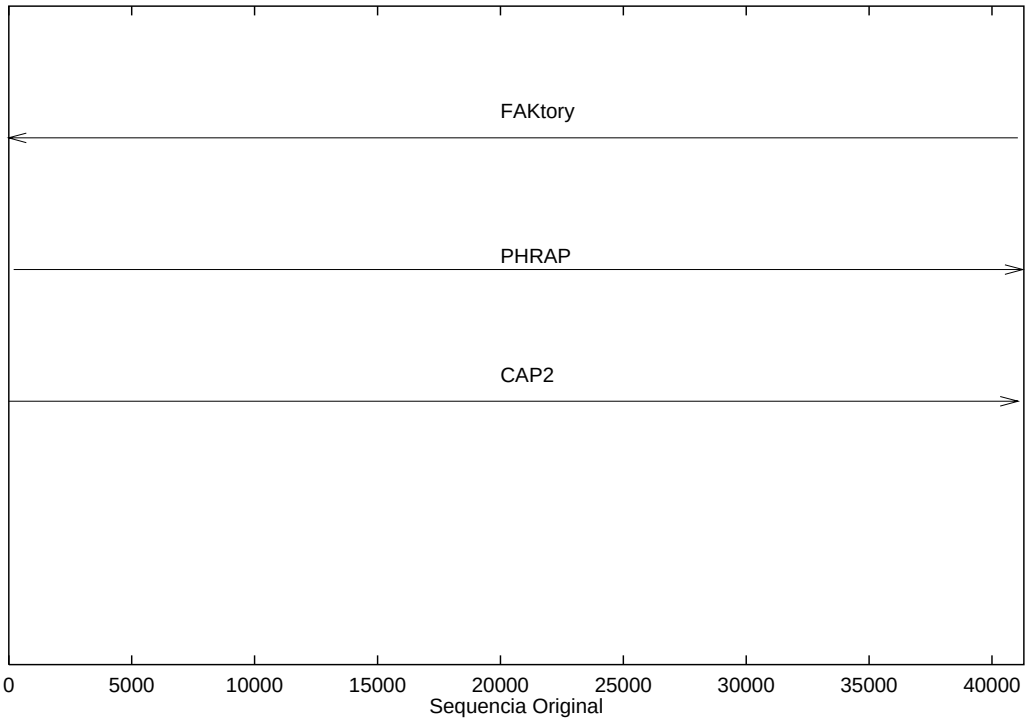

Figura 5.4: Gráfico da execução sobre o B1496

5.1.5 Efeitos dos erros de seqüenciamento sobre os resultados das ferramentas

Além dos testes acima, onde avaliamos a eficiência das ferramentas no tratamento de erros no seqüenciamento, realizamos outros testes no intuito de compararmos o comportamento delas frente a um aumento gradativo da taxa de erros de um conjunto de fragmentos.

Para realizarmos esta tarefa, pegamos o conjunto de fragmentos reais relacionados

com a seqüência do cosmídeo B2126 e inserimos nele as quantidades de erros mostradas na Tabela 5.14.

Taxa de erros	No. de substituições	No. de inserções	No. de remoções
1.14%	2174	893	888
3.16%	6086	2421	2501
5.35%	10305	4104	4213
7.34%	14011	5733	5816
9.69%	18654	7543	7637

Tabela 5.14: Distribuição dos erros nos fragmentos do cosmídeo B2126

Os resultados da execução de cada uma das quatro ferramentas sobre os cinco casos de testes criados podem ser vistos nas tabelas 5.15 , 5.16, 5.17, 5.18 e 5.19, cujos valores mostram que o CAP2 e o Phrap apresentam um crescimento menor do número de erros presentes no alinhamento de seus contigs que as outras ferramentas. Este fato pode ser visualizado no Gráfico 5.5 abaixo, onde vemos também que a curva representante da taxa de erros relacionada ao TIGR destaca-se das demais.

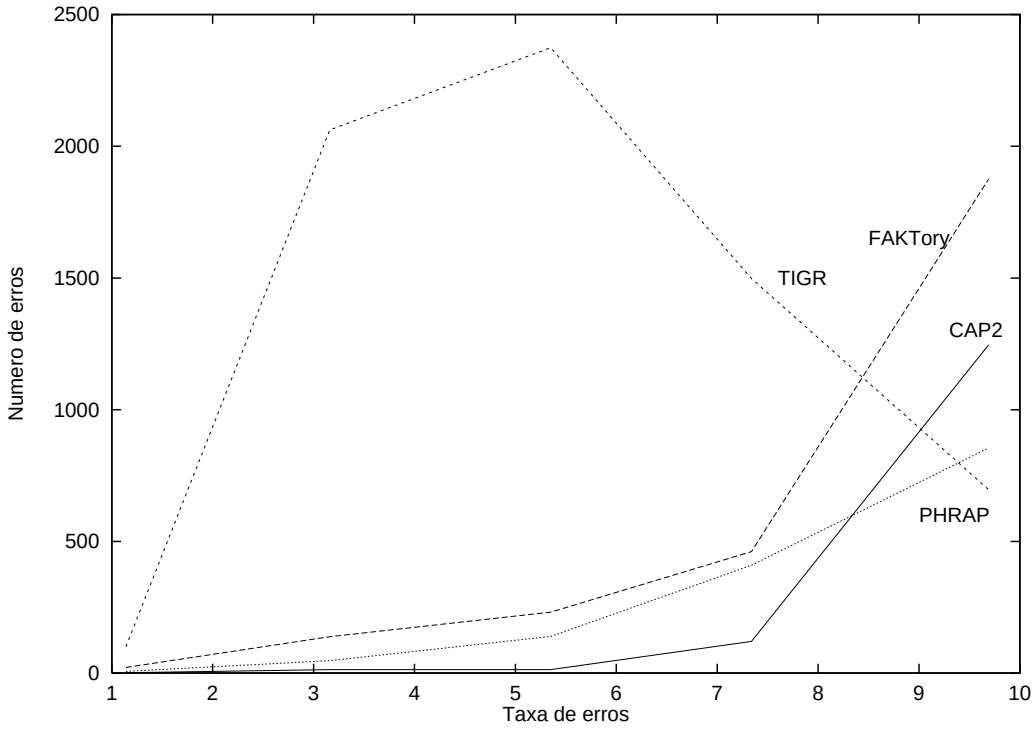

Figura 5.5: Gráfico de erros (Taxa de erros x Número de erros)

Inicialmente, esta curva apresenta um crescimento bem maior que aquelas representantes do número de erros das outras ferramentas, mas posteriormente ela começa a diminuir. Isto está relacionado com o fato de que, após o 4o. caso de teste, o TIGR deixa de utilizar uma quantidade considerável de fragmentos no processo de montagem. Este fato provavelmete ocorre devido aos erros que os fragmentos não utilizados apresentam e pode ser notado pelas baixíssimas coberturas apresentadas pela ferramenta nos dois últimos casos de testes.

Da mesma forma que ocorreu com o número de erros, o CAP2 e o Phrap correspondem às ferramentas que apresentam o menor crescimento no número de buracos presentes no alinhamento de seus contigs à medida que a taxa de erros vai aumentando. A curva que merece destaque aqui (Gráfico 5.6) é aquela que representa a taxa de aumento no número de buracos dos contigs produzidos pelo FAKtory. Esta ferramenta apresentou um crescimento no número de buracos bem maior que as demais, chegando a mais de oito mil buracos no último caso de teste. No Gráfico 5.6 podemos ver também que, como aconteceu com o número de erros, o TIGR apresentou uma queda na quantidade de buracos após a execução do terceiro caso de teste (também por não utilizar vários fragmentos do conjunto de entrada).

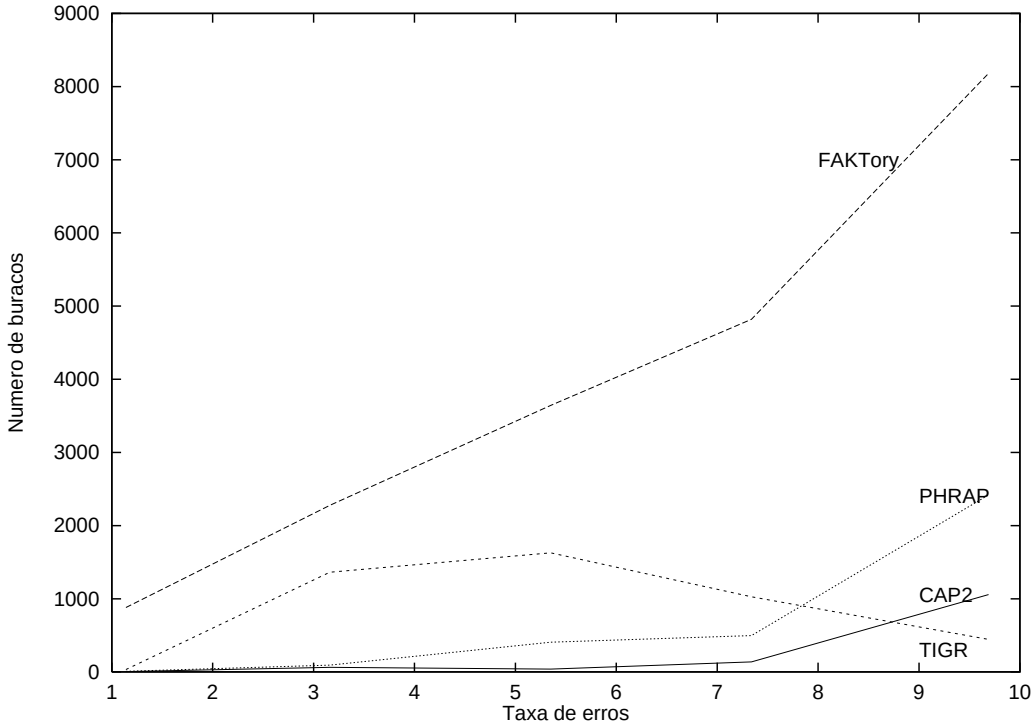

Figura 5.6: Gráfico de buracos (Taxa de erros x Número de buracos)

Com relação ao número de contigs produzidos pelas ferramentas, o FAKtory cor-

responde ao sistema que apresentou a menor variação deste parâmetro à medida que a taxa de erros aumenta. Em quase todos os casos, o FAKtory construiu o mesmo número de contigs, bem diferente do que aconteceu com o TIGR. No Gráfico 5.7, podemos ver que o TIGR apresentou números variados de contigs e, como vem ocorrendo até o momento, este número diminui nos dois últimos casos de teste. Assim como o FAKtory, o Phrap também não constrói uma quantidade muito diferente de contigs a cada caso de teste. Apenas quando a taxa de erros é de 9.69% é que o número de contigs eleva-se razoavelmente. Quanto ao CAP2, a quantidade de contigs produzidos por essa ferramenta apresenta até uma queda no meio da realização dos testes, mas volta a subir gradativamente chegando a 53 contigs no último caso.

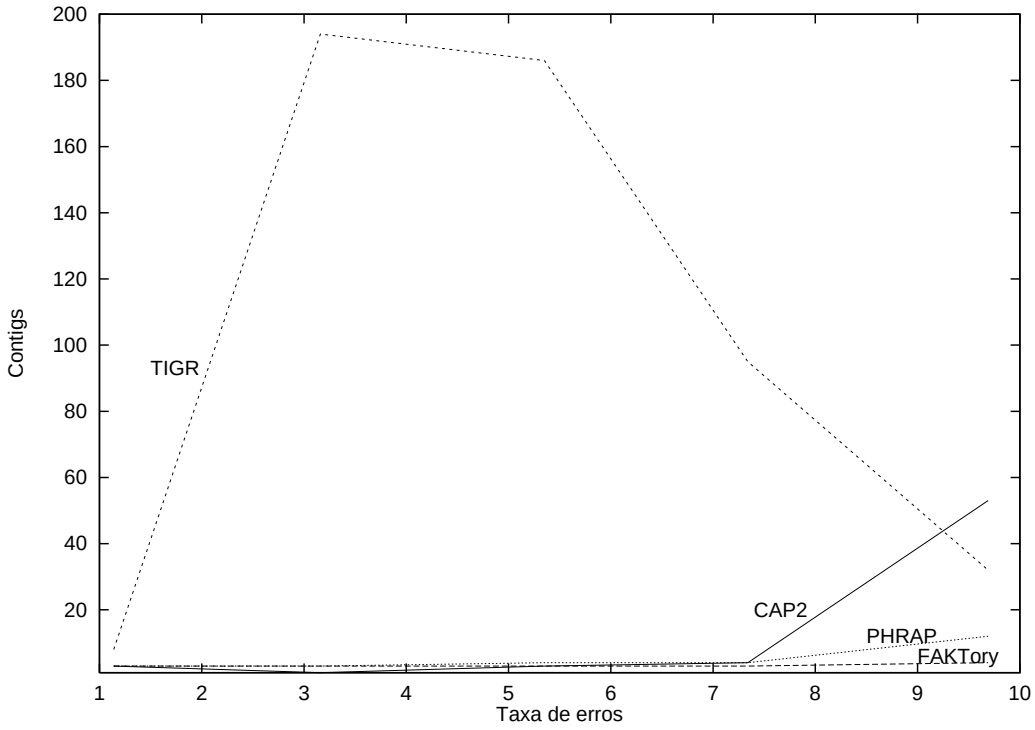

Figura 5.7: Gráfico de contigs (Taxa de erros x Contigs)

O Gráfico 5.8 mostra a relação entre as taxas de erros e a cobertura obtida por cada ferramenta. Como podemos ver, o TIGR apresenta uma queda brusca na cobertura à medida que a taxa de erros aumenta e, como no caso do número de erros, isso também está relacionado com o fato dessa ferramenta não utilizar uma série de fragmentos de "baixa qualidade" durante a execução sobre os cinco casos de testes. Assim como o FAKtory e o Phrap, o CAP2 apresenta uma curva não-uniforme de cobertura, que aumenta e diminui de forma alternada. O importante é observar que, no caso dessa ferramenta, a cada momento que ocorre um aumento da cobertura, o número de erros e buracos eleva-se consideravelmente, o que não acontece quando a cobertura diminui

(com exceção do último caso).

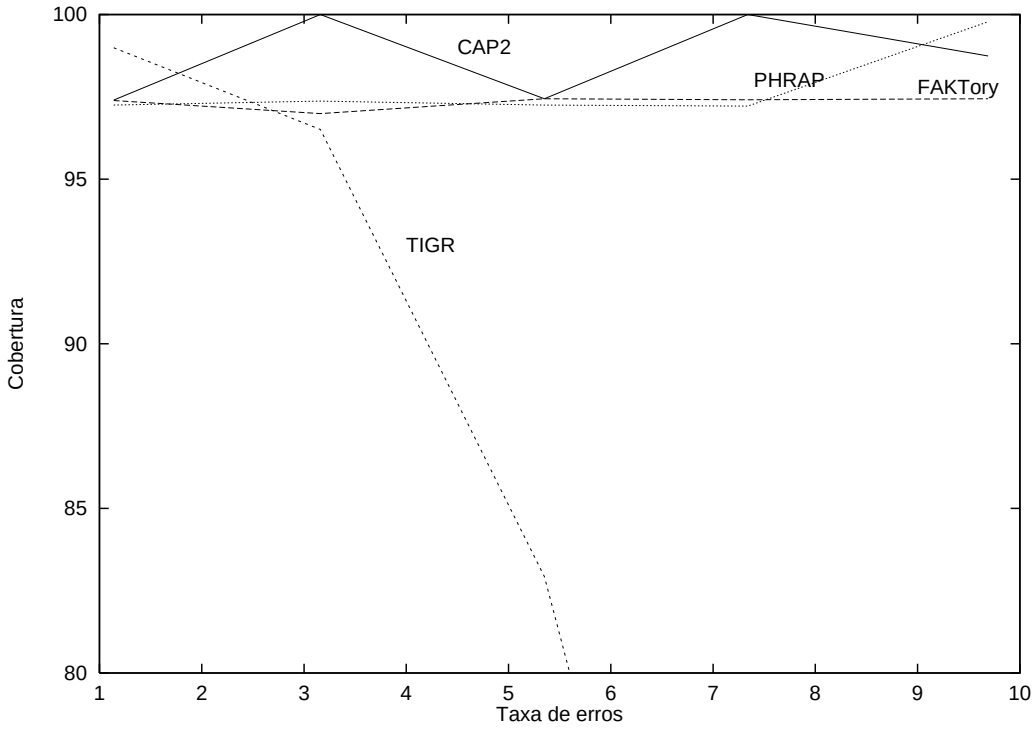

Figura 5.8: Gráfico de cobertura (Taxa de erros x Cobertura)

Ferramentas	Contigs	Cobertura(%)	Erros	Buracos
CAP2	3	97.40	2	5
FAKtory	3	97.39	22	881
TIGR	8	98.99	101	34
Phrap	3	97.25	7	10

Tabela 5.15: Resultados obtidos sobre o conjunto contendo uma taxa de erros de 1.14%

Ferramentas	Contigs	Cobertura(%)	Erros	Buracos
CAP2	1	100	14	64
FAKtory	3	96.99	139	2277
TIGR	194	96.51	2062	1363
Phrap	3	97.37	48	94

Tabela 5.16: Resultados obtidos sobre o conjunto contendo uma taxa de erros de 3.16%

Ferramentas	Contigs	Cobertura(%)	Erros	Buracos
CAP2	3	97.44	14	39
FAKtory	3	97.44	232	3642
TIGR	186	82.93	2374	1629
Phrap	4	97.25	140	406

Tabela 5.17: Resultados obtidos sobre o conjunto contendo uma taxa de erros de 5.35%

Ferramentas	Contigs	Cobertura(%)	Erros	Buracos
CAP2	4	100	121	137
FAKtory	3	97.41	462	4819
TIGR	95	59.50	1498	1028
Phrap	4	97.22	410	497

Tabela 5.18: Resultados obtidos sobre o conjunto contendo uma taxa de erros de 7.34%

Ferramentas	Contigs	Cobertura(%)	Erros	Buracos
CAP2	53	98.74	1246	1058
FAKtory	4	97.44	1875	8180
TIGR	32	24.88	697	444
Phrap	12	99.78	855	2419

Tabela 5.19: Resultados obtidos sobre o conjunto contendo uma taxa de erros de 9.58%

Algumas considerações a respeito dos resultados obtidos pelas ferramentas

Pudemos observar com estes casos de testes que a possível existência de erros ao final do processo de seqüenciamento de fragmentos pode realmente comprometer o processo de montagem caso uma ferramenta adequada não seja escolhida. Os vários contigs gerados pelo TIGR dificultam em muito a reconstituição total da seqüência original pelos biólogos e a elevada taxa de erros presente naqueles construídos pelo FAKtory (que nos surpreende pelo fato da ferramenta focalizar grande parte de sua atenção sobre este problema) exige uma análise bem detalhada das bases que o compõem.

O CAP2 demonstrou ser uma ferramenta bastante eficiente no tratamento de erros no seqüenciamento. Como pudemos ver pelos gráficos da subseção 5.1.5, esta ferramenta trata de forma adequada até quantidade de erros acima da taxa normal de 5%, levando-

nos a elogiar seu esquema de criação dos vetores de taxa de erros descrito na subseção 3.2.1.

Com relação ao Phrap, esta ferramenta também apresentou ótimos resultados durante esta seção de testes. Os gráficos da subseção 5.1.5 mostram que esta ferramenta obteve resultados até melhores do que os apresentados pelo CAP2 (com relação ao número de erros e contigs produzidos) no último caso de teste.

5.2 Fragmentos Quimeras

Durante a quebra dos vários clones de DNA, na segunda fase do processo de seqüenciamento via método *shotgun*, podem ocorrer junções de dois ou mais fragmentos provenientes de regiões completamente distintas daquela molécula. Estas junções dão origem ao que chamamos de **fragmentos quimeras**, fragmentos estes que podem comprometer o processo de seqüenciamento caso não sejam detectados pelas ferramentas e eliminados do processo de montagem.

Nesta seção de testes, avaliamos a capacidade que cada uma das ferramentas possui para detectar um certo número de fragmentos quimeras existentes em quatro casos de testes distintos. Esta avaliação será feita através da comparação dos resultados obtidos pelas quatro ferramentas, no intuito de determinar qual delas é a mais eficiente na detecção e tratamento de fragmentos quimeras, assim como por meio da comparação dos resultados obtidos por uma ferramenta tomando-se como entrada o conjunto original de fragmentos e o conjunto contendo fragmentos quimeras. Tentaremos, com esta última forma de análise, determinar o quanto a presença de fragmentos quimeras influi na exatidão dos resultados dos programas de montagem que temos em mãos.

Um pouco diferente da seção anterior, desta vez não utilizamos os conjuntos de fragmentos reais relacionados com as seqüências dos cosmídeos B1496, B2126, L247 e L518 para avaliarmos a eficiência das ferramentas no tratamento de fragmentos quimeras. Ao invés disso, usamos como casos de testes quatro conjuntos de fragmentos artificialmente obtidos, por meio do programa *enzyme*, das seqüências daqueles DNAs. A Tabela 5.20 mostra algumas informações a respeito destes conjuntos, como o número de fragmentos (No. fragms) presentes em cada um deles, o tamanho médio (Tam. fragms) destes fragmentos e a profundidade da cobertura (Prof. Cob.).

Adotamos esta metodologia por não sabermos com certeza se existem fragmentos quimeras nos conjuntos reais utilizados na seção anterior e, se existem, quais são eles. Além de nos garantir que nenhum dos fragmentos artificiais corresponde a um fragmento quimera, o *enzyme* fornece-nos também informações relacionadas às posições desses fragmentos dentro das seqüências, informações estas que nos possibilitam uma análise mais detalhada dos resultados obtidos nesta seção.

Testes	No. de fragms	Tam. médio dos fragms	Prof. Cob.
B1496.quim	723	400.91	7.02
B2126.quim	780	400	7
L247.quim	778	401.70	7.03
L518.quim	725	401.65	7.03

Tabela 5.20: Testes utilizados para avaliação das ferramentas no tratamento de fragmentos quimeras

Na tabela seguinte, identificamos os fragmentos quimeras inseridos por nós em cada um dos quatro casos de testes acima citados. Estes falsos fragmentos foram obtidos artificialmente juntando-se em três seqüências (quatro no caso do B1496) seis fragmentos aleatoriamente obtidos de seus respectivos conjuntos. Informações a respeito dos fragmentos componentes de cada fragmento quimera, como sua identificação e posições iniciais e finais na seqüência original antes da junção, serão mostradas nas subseções seguintes.

Caso de teste	Fragmentos quimeras
B1496.quim	frag0077, frag0097, frag0122 e frag0371
B2126.quim	frag0014, frag0134 e frag0598
L247.quim	frag0068, frag0124 e frag0466
L518.quim	frag0100, frag0250 e frag0697

5.2.1 Cosmídeo B1496 de Mycobacterium leprae

Na Tabela 5.21, mostramos os resultados da execução de cada uma das ferramentas sobre o conjunto original de fragmentos obtido da seqüência do cosmídeo B1496 de *Mycobacterium leprae*, ou seja, sobre o conjunto que não contém nenhum fragmento quimera.

Ferramenta	Contigs	Cobertura(%)	Erros	Buracos
CAP2	25	98.21	11887	6038
FAKtory	1	99.14	3	5
TIGR	2	99.72	3	3
Phrap	2	98.52	2	5

Tabela 5.21: Resultados da execução das ferramentas sobre o conjunto original de fragmentos obtido do cosmídeo B1496

Inserindo-se os quatro fragmentos quimeras listados na Tabela 5.22[2] (juntamente com os fragmentos que os compõem e suas posições iniciais e finais na seqüência original) no conjunto original de fragmentos, as ferramentas apresentaram como resultados os valores mostrados na Tabela 5.23 (Gráfico 5.9).

Fragmentos quimeras	Fragmentos componentes	Pos. inicial	Pos. final
frag0077	frag0077	878	571
	frag0105	880	629
frag0097	frag0097	36904	36649
	frag0116	9451	9737
frag0122	frag0122	7442	7166
	frag0296	27859	27556
frag0371	frag0371	36386	36108
	frag0437	10839	10525

Tabela 5.22: Fragmentos quimeras presentes no conjunto de fragmentos relacionados à seqüência do B1496

Ferramentas	Contigs	Quimeras detectados	Cobertura	Erros	Buracos
CAP2	25	frag0097 frag0122 frag0371	98.21%	11887	6348
FAKtory	4	frag0122 frag0077 frag0371	99.14%	84	39
TIGR	2	frag0077 frag0097 frag0122 frag0371	99.72%	3	3
Phrap	1	frag0077 frag0097 frag0122 frag0371	98.91%	2	4

Tabela 5.23: Resultados das execuções das ferramentas sobre o conjunto contendo quimeras

[2]Aqueles fragmentos cujas posições finais antecedem as posições inicias nesta tabela foram obtidos do complemento reverso da seqüência. Esta observação vale para as demais tabelas desta seção.

Tanto o Phrap quanto o TIGR conseguiram detectar a presença dos quatro fragmentos quimeras listados na Tabela 5.22, inclusive a do frag0077, composto de fragmentos que se sobrepõem dentro da seqüência original. O CAP2 e o FAKtory identificaram apenas três fragmentos deste tipo, como pode ser visto na Tabela 5.23.

O CAP2 falha ao considerar o fragmento frag0077 como sendo um fragmento real e sua montagem no interior do 11o. contig gerado por esta ferramenta determina um alinhamento deste com a seqüência original contendo um grande número de buracos internos e contíguos (310 no total). Estes buracos ocorrem da posição 3560 à posição 3869 do contig 11, que corresponde à região deste contig que inclui parte do fragmento frag0077. Além da falha na detecção de um fragmento quimera, outro problema relacionado com a execução do CAP2 sobre este caso de teste é que os contigs gerados por ele determinam a menor cobertura da seqüência original quando observamos os valores obtidos pelas outras ferramentas. De acordo com a Tabela 5.23, podemos ver que o CAP2 gerou o maior número de contigs ao final do processo de montagem, mas estes deixam quase 2% da seqüência descoberta. Em adição a isso, os contigs gerados por essa ferramenta determinam também um número bem maior de erros e buracos em seus alinhamentos com a seqüência original, mas é importante salientar que esta falha na similaridade já existia antes da inserção dos fragmentos quimeras (Tabela 5.21). Comparando-se os valores das tabelas 5.23 e 5.21, podemos notar que a montagem errada do fragmento frag0077 parece ter ocasionado apenas um aumento no número de buracos presentes no alinhamento dos contigs gerados pelo CAP2, deixando os valores de erros e cobertura praticamente inalterados.

Assim como o CAP2, o FAKtory também não detectou a presença dos quatro fragmentos quimeras existentes neste caso de teste. O fragmento frag0097 passou despercebido por esta ferramenta e foi montado no início de seu 1o. contig, cujo alinhamento com a seqüência original acumula uma quantidade considerável de erros e buracos em sua porção inicial. Como se pode ver pela comparação dos números das tabelas 5.23 e 5.21, a existência de fragmentos quimeras e a montagem errada do fragmento frag0097 não interferiu em nada na boa cobertura determinada pelos quatro contigs gerados pelo FAKtory neste caso de teste, mas parecem ter sido fator determinante do número considerável de erros e buracos apresentados pela ferramenta na Tabela 5.23.

O TIGR e o Phrap correspondem às ferramentas que obtiveram os melhores resultados frente a este caso de teste. A detecção de todos os fragmentos quimeras por estas ferramentas determinam um número de erros e buracos muito pequeno no alinhamento de seus contigs com a seqüência original, contigs estes que determinam também uma boa cobertura dessa seqüência. A eficiência destas duas ferramentas na detecção dos fragmentos quimeras presentes neste caso de teste resulta na grande semelhança dos valores de erros e buracos apresentados nas tabelas 5.23 e 5.21.

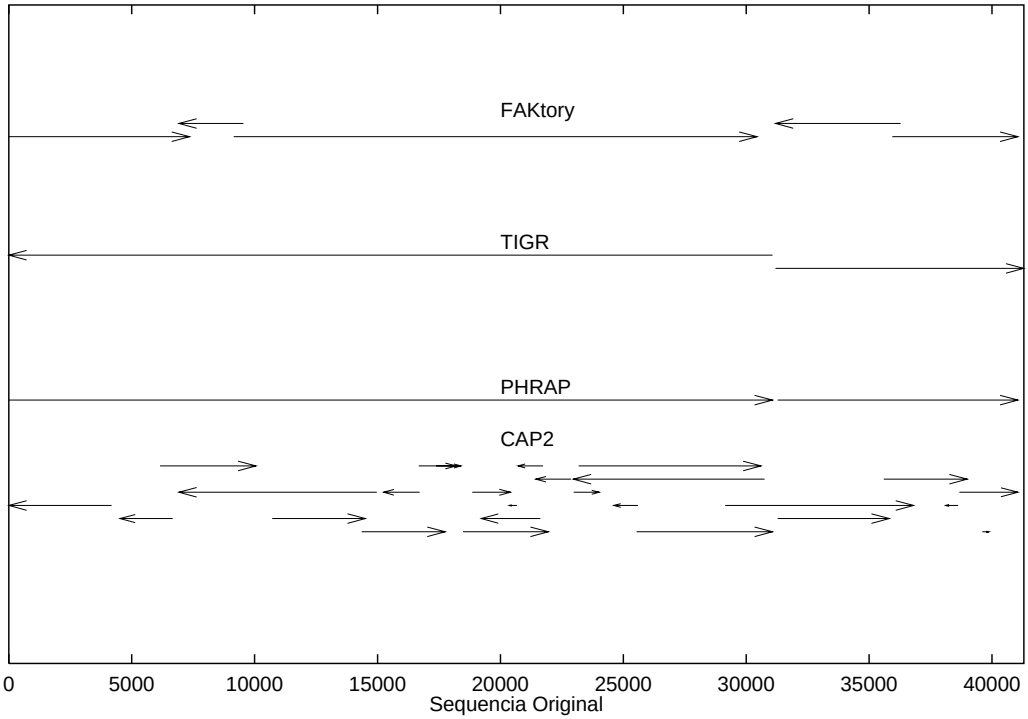

Figura 5.9: Gráfico da execução sobre o B1496

5.2.2 Cosmídeo B2126 de Mycobacterium leprae

Antes da inserção dos três fragmentos quimeras indicados na Tabela 5.25, os resultados da execução das ferramentas sobre o conjunto de fragmentos obtido da seqüência do cosmídeo B2126 foram os mostrados na Tabela 5.24 abaixo.

Ferramenta	Contigs	Cobertura(%)	Erros	Buracos
CAP2	23	94.84	16347	8520
FAKtory	4	98.01	18	65
TIGR	3	97.91	1	1
Phrap	2	96.57	1	0

Tabela 5.24: Resultados da execução das ferramentas sobre o conjunto original de fragmentos obtido do cosmídeo B2126

Considerando-se agora a existência dos fragmentos quimeras deste caso de teste, a Tabela 5.26 (Gráfico 5.10) mostra valores um pouco diferentes dos apresentados anteriormente.

Fragmentos quimeras	Fragmentos componentes	Pos. inicial	Pos. final
frag0014	frag0014	3367	3097
	frag0039	28570	28310
frag0134	frag0134	28047	27769
	frag0288	9909	9614
frag0598	frag0598	21528	21256
	frag0777	39989	40259

Tabela 5.25: Fragmentos quimeras presentes no conjunto de fragmentos relacionados à seqüência do B2126

Ferramentas	Contigs	Quimeras detectados	Cobertura	Erros	Buracos
CAP2	24	frag0014	95.35%	16457	15152
FAKtory	6	frag0014	98.02%	109	129
		frag0598			
TIGR	3	frag0014	97.93%	1	1
		frag0134			
		frag0598			
Phrap	2	frag0014	96.57%	1	0
		frag0134			
		frag0598			

Tabela 5.26: Resultados da execução das ferramentas sobre o conjunto contendo quimeras

Podemos ver pela Tabela 5.26 que, como no caso anterior, as únicas ferramentas que conseguiram detectar a presença de todos os fragmentos quimeras inseridos no caso de teste em questão foram o TIGR e o Phrap. Dentre os quatro programas, o CAP2 foi aquele que obteve o pior desempenho, conseguindo reconhecer apenas o fragmento frag0014 como quimera. Com relação ao FAKtory, dois (frag0014 e frag0598) dos três fragmentos quimeras existentes foram detectados por esta ferramenta.

O fragmento frag0134 fora montado pelo CAP2 dentro do seu 3o. contig. De acordo com o GAP, este contig possui um alinhamento de baixa qualidade com a seqüência original, com um total de 920 erros e 505 buracos que acabam contribuindo para a elevada taxa de erros e buracos mostrada na Tabela 5.26. Quanto ao fragmento frag0598, mesmo não sendo detectado pelo CAP2 como um fragmento quimera, ele não foi utilizado durante o processo de montagem, constituindo num único contig ao final da execução da ferramenta. Como no caso anterior, a cobertura da seqüência original pelos contigs gerados pelo CAP2 é a mais baixa dentre as obtidas pelas quatro ferramentas, deixando quase 5% da seqüência descoberta. Desta vez, a não detecção dos fragmentos quimeras

teve uma grande influência tanto sobre o número de erros presentes no alinhamento dos contigs gerados pelo CAP2 quanto sobre o número de buracos que eles acumulam, como pode ser observado comparando-se os valores das tabelas 5.26 e 5.24.

Quanto ao FAKtory, esta ferramenta não identificou o fragmento frag0134 como sendo um fragmento quimera, montando-o na porção final de seu 3o. contig. Esta montagem equivocada determina grande parte dos erros e buracos apresentados pelo FAKtory na Tabela 5.26, onde se pode ver também que esta ferramenta corresponde àquela que determina a melhor cobertura da seqüência original dentre as quatro sendo avaliadas. Pelos valores de erros e buracos apresentados na Tabela 5.24, o FAKtory parece ser uma ferramenta bastante influenciável pela presença de fragmentos quimeras, com a exatidão de seus resultados piorando muito na falha de detecção de alguns deles.

Assim como no caso de teste anterior, o TIGR e o Phrap obtiveram os melhores resultados neste caso de teste. Além de detectar os três fragmentos quimeras aí existentes, seus poucos contigs cobrem uma grande porção da seqüência original com uma quantidade de erros e buracos desprezível, o que implica em resultados muito parecidos com os da execução sobre o conjunto original de fragmentos.

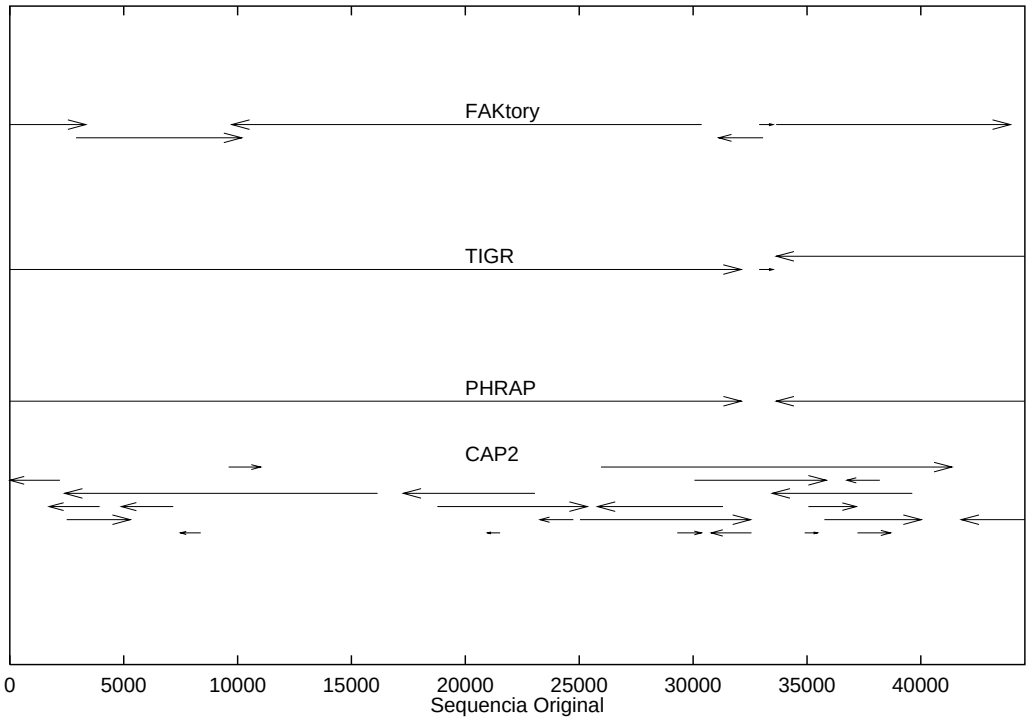

Figura 5.10: Gráfico da execução sobre o B2126

5.2.3 Cosmídeo L247 de Mycobacterium leprae

Os resultados da execução das ferramentas sobre o conjunto original de fragmentos obtido da seqüência do cosmídeo L247 de *Mycobacterium leprae* são mostrados na Tabela 5.27 abaixo.

Ferramenta	Contigs	Cobertura(%)	Erros	Buracos
CAP2	22	83.51	17809	9291
FAKtory	3	99.14	1036	507
TIGR	3	99.61	1044	467
Phrap	2	99.61	1	1

Tabela 5.27: Resultados da execução das ferramentas sobre o conjunto original

Assim como no caso de teste anterior, a inserção de fragmentos quimeras (Tabela 5.28) no conjunto original de fragmentos levou a mudanças nos resultados gerados por cada ferramenta, como pode ser visto na Tabela 5.29 (Gráfico 5.11).

Fragmentos quimeras	Fragmentos componentes	Pos. inicial	Pos. final
frag0068	frag0068	14523	14081
	frag0208	19186	18874
frag0124	frag0124	696	438
	frag0351	24520	24809
frag0466	frag0466	31408	30880
	frag0676	32317	32609

Tabela 5.28: Fragmentos quimeras presentes no conjunto de fragmentos relacionados à seqüência do L247

Ferramentas	Contigs	Quimeras detectados	Cobertura	Erros	Buracos
CAP2	22	frag0068 frag0124	84.30%	16629	8687
FAKtory	8		95.24%	2193	1535
TIGR	5	frag0124	99.61%	1249	605
Phrap	2	frag0068 frag0124 frag0466	99.61%	1	1

Tabela 5.29: Resultados da execução das ferramentas sobre o conjunto contendo quimeras

Pelos valores da Tabela 5.29, podemos ver que a única ferramenta que conseguiu detectar a presença dos três fragmentos quimeras listados na Tabela 5.28 foi o Phrap. O TIGR, que vinha identificando todos os fragmentos quimeras até o momento, reconheceu neste caso de teste apenas o fragmento frag0124, enquanto que o CAP2 detectou a presença de dois (frag0068 e frag0124) dos três fragmentos quimeras existentes. O FAKtory obteve os piores resultados, não conseguindo detectar a presença de nenhum dos três fragmentos quimeras inseridos neste caso de teste.

Um pouco melhor que no caso anterior, desta vez o CAP2 conseguiu detectar a presença de dois dos três fragmentos existentes neste caso de teste, montando o fragmento frag0466 no interior de seu 2o. contig. De acordo com os resultados do GAP, o melhor alinhamento deste contig com a seqüência original possui um total de 830 erros e 419 buracos, que se refletem nos valores mostrados na Tabela 5.29. Apesar de ter detectado um maior número de fragmentos quimeras que o TIGR e FAKtory, o CAP2 continua sendo a ferramenta de piores resultados. Além da elevada taxa de erros e buracos presentes no alinhamento de seus contigs, estes ainda cobrem a menor porcentagem da seqüência original com relação aos valores apresentados pelas outras ferramentas. De acordo com a Tabela 5.29, apenas 84.30% da seqüência original é coberta pelos 22 contigs gerados pelo CAP2. Vale salientar aqui, que a presença dos fragmentos quimeras neste caso de teste e a não detecção de um deles pelo CAP2 parece ter influenciado de forma positiva os resultados desta ferramenta. Pela comparação das tabelas 5.29 e 5.27, podemos ver que, além do valor de cobertura ter aumentado após a inserção dos fragmentos quimeras, o número de erros e buracos diminuíram, fatos estes, no mínimo, curiosos.

Os fragmentos frag0124 e frag0466 foram montados pelo FAKtory no início de seus 3o. e 7o. contigs respectivamente, enquanto que o fragmento frag0068 fora encaixado no interior do 6o. contig gerado por esta ferramenta. Uma análise detalhada deste contig, obtida através do *cross_match*, revela que suas partes possuem um bom alinhamento com duas porções não-contíguas da seqüência original. Mais especificamente, a primeira metade (da base 1 à base 3005) do 6o. contig gerado pelo FAKtory alinha-se com a seqüência original entre sua 16160ª e 19256ª base, enquanto que sua segunda metade (da base 3032 à base 5303) alinha-se com a porção da seqüência original que vai da base 14067 à base 16338. Estes dois alinhamentos possuem no total apenas 10 erros e mostram um exemplo típico de problema que pode ocorrer no processo de montagem em decorrência da falta de detecção de fragmentos quimeras. O que o FAKtory fez aqui foi exatamente juntar num único contig porções não contíguas da seqüência original por meio das sobreposições existentes entre elas e as extremidades do fragmento frag0068 (vide Tabela 5.28). As montagens erradas descritas acima determinam uma elevada taxa de erros e buracos nos resultados obtidos pelo FAKtory (Tabela 5.29), que correspondem quase ao dobro de erros e buracos determinados pelos contigs construídos a partir dos fragmentos originais (Tabela 5.27). A presença dos fragmentos quimeras neste caso

de teste e a falha em detectá-los parece também que custaram ao FAKtory uma queda significativa no valor de cobertura da seqüência original, que pode ser vista comparando-se os valores das tabelas 5.29 e 5.27.

Diferentemente dos resultados anteriores, desta vez o TIGR não conseguiu detectar a presença de todos os fragmentos quimeras existentes no caso de teste em questão. Os fragmentos frag0068 e frag0466 foram montados pelo TIGR no início do 1o. e 3o. contigs, respectivamente, impedindo um alinhamento de boa qualidade destes com a seqüência original. Apesar destas montagens erradas, vale salientar que os contigs produzidos pelo TIGR determinaram uma boa cobertura da seqüência original e grande parte dos erros e buracos mostrados na Tabela 5.29 também estão presentes no alinhamento dos contigs gerados pelo TIGR ao fim da execução sobre o conjunto original de fragmentos (Tabela 5.27).

Analisando os valores da Tabela 5.29, podemos confirmar os bons resultados obtidos pelo Phrap até o momento. Além da detecção de todos os fragmentos quimeras, os contigs gerados por esta ferramenta determinam uma boa cobertura da seqüência original, com uma taxa de similaridade de quase 100%, idêntica àquela obtida na execução sobre o conjunto original de fragmentos (Tabela 5.27).

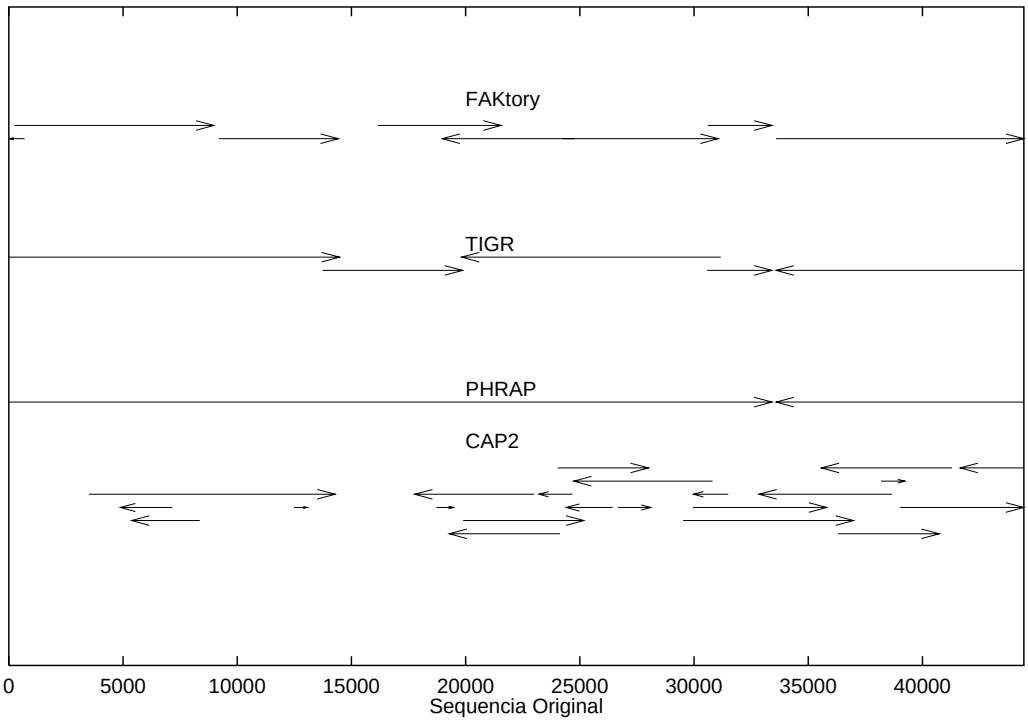

Figura 5.11: Gráfico da execução sobre o L247

5.2.4 Cosmídeo L518 de Mycobacterium leprae

As tabelas 5.30 e 5.31 (Gráfico 5.12) mostram, respectivamente, os resultados da execução de cada uma das ferramentas sobre o conjunto original de fragmentos e sobre o conjunto contendo os fragmentos quimeras listados na Tabela 5.32.

Ferramenta	Contigs	Cobertura(%)	Erros	Buracos
CAP2	24	78.68	13782	6877
FAKtory	6	87.34	337	155
TIGR	2	98.01	0	0
Phrap	3	95.76	164	144

Tabela 5.30: Resultados da execução das ferramentas sobre o conjunto original de fragmentos obtido do cosmídeo L518

Ferramentas	Contigs	Quimeras detectados	Cobertura	Erros	Buracos
CAP2	25	frag0100 frag0697	83.17%	10658	5424
FAKtory	9		85.66%	3998	1822
TIGR	3	frag0250 frag0697	90.47%	0	0
Phrap	3	frag0100 frag0250 frag0697	95.76%	164	114

Tabela 5.31: Resultados da execução das ferramentas sobre o conjunto contendo quimeras

Fragmentos quimeras	Fragmentos componentes	Pos. inicial	Pos. final
frag0100	frag0100	35452	35139
	frag0111	32768	33129
frag0250	frag0250	36977	36684
	frag0446	1569	1877
frag0697	frag0697	16228	16485
	frag0723	17014	16543

Tabela 5.32: Fragmentos quimeras presentes no conjunto de fragmentos relacionados à seqüência do L518

Com resultados muito parecidos aos do teste anterior, novamente a única ferramenta que conseguiu detectar a presença dos três fragmentos quimeras existentes neste caso de teste foi o Phrap. Mais uma vez, o FAKtory não conseguiu detectar a presença de nenhum dos três fragmentos quimeras inseridos e, um pouco melhor que no caso anterior, o TIGR reconheceu dois (frag0250 e frag0697) dos três fragmentos quimeras, da mesma forma que o CAP2.

O fragmento frag0250, não detectado pelo CAP2 como um fragmento quimera, fora inserido no 7o. contig gerado por esta ferramenta, contig este que, de acordo com o GAP, possui um alinhamento com a seqüência original contendo um total de 2884 erros e 1537 buracos. Novamente, o CAP2 corresponde à ferramenta cujos contigs cobrem a menor porção da seqüência original, com seus alinhamentos acumulando uma grande quantidade de erros e buracos. Mais uma vez nos surpreendemos com o fato de que a presença de fragmentos quimeras e a falha na detecção de um deles pelo CAP2 melhora os resultados obtidos por esta ferramenta no que diz respeito ao número de erros e buracos relacionados com o alinhamento de seus contigs e à cobertura da seqüência original por eles. Isto pode ser visto pela comparação dos valores mostrados nas tabelas 5.31 e 5.30.

Os fragmentos frag0100 e frag0697 foram montados pelo FAKtory no início do 4o. e 6o. contig, respectivamente, enquanto que o fragmento frag0250 fora encaixado no interior do 1o. contig gerado por esta ferramenta, contig este que acumula um total de 647 erros e 22 buracos com com relação à seqüência original. Novamente, uma análise mais detalhada do 1o. contig gerado pelo FAKtory mostra que sua porção que vai da base 7729 à base 14213 inclui duas partes não contíguas da seqüência original (1-1772 e 36673-41412), que se sobrepõem com as extremidades do fragmento frag0250. A comparação dos resultados obtidos pelo FAKtory nas tabelas 5.31 e 5.30 confirma nossa hipótese de que esta ferramenta é bastante sensível à presença de fragmentos quimeras. Como no caso anterior, a não detecção dos três falsos fragmentos presentes neste caso de teste parece determinar uma queda brusca da cobertura determinada pelos contigs gerados pelo FAKtory. Além disso, o número de erros presentes nos alinhamentos destes contigs com a seqüência original é bem maior que aqueles acumulados nos alinhamentos dos contigs construídos a partir dos fragmentos do conjunto original.

Com relação ao TIGR, esta ferramenta montou o fragmento frag0100 no início de seu 1o. contig e, diferente do que pensávamos que iria ocorrer, essa falha não resultou no aumento da quantidade total de erros e buracos acumulados no alinhamento deste contig com a seqüência original. Tanto na tabela relacionada com o conjunto original de fragmentos, quanto naquela contendo os resultados obtidos ao término da execução sobre o conjunto contendo quimeras, o número de erros e buracos é zero. Podemos ver, porém, que o valor de cobertura diminuiu significativamente e o fato de uma das regiões não cobertas da seqüência original (33119 - 35943) incluir partes dos fragmentos componentes do frag0100 (Tabela 5.32) leva-nos a crer que a sua inserção no conjunto

original de fragmentos tenha sido o fator determinante dessa queda.

O Phrap confirmou neste caso de teste sua eficiência com relação à detecção de fragmentos quimeras. Como vinha acontecendo até o momento, nenhum fragmento quimera passou despercebido por essa ferramenta, mas, ao contrário do que ocorreu com o TIGR, os contigs gerados pelo Phrap determinam uma quantidade de erros considerável em seus alinhamentos com a seqüência original. Esta falha porém não está relacionada com a presença de fragmentos quimeras no caso de teste em questão, pois já existiam ao final da execução do Phrap sobre o conjunto original de fragmentos (Tabela 5.30).

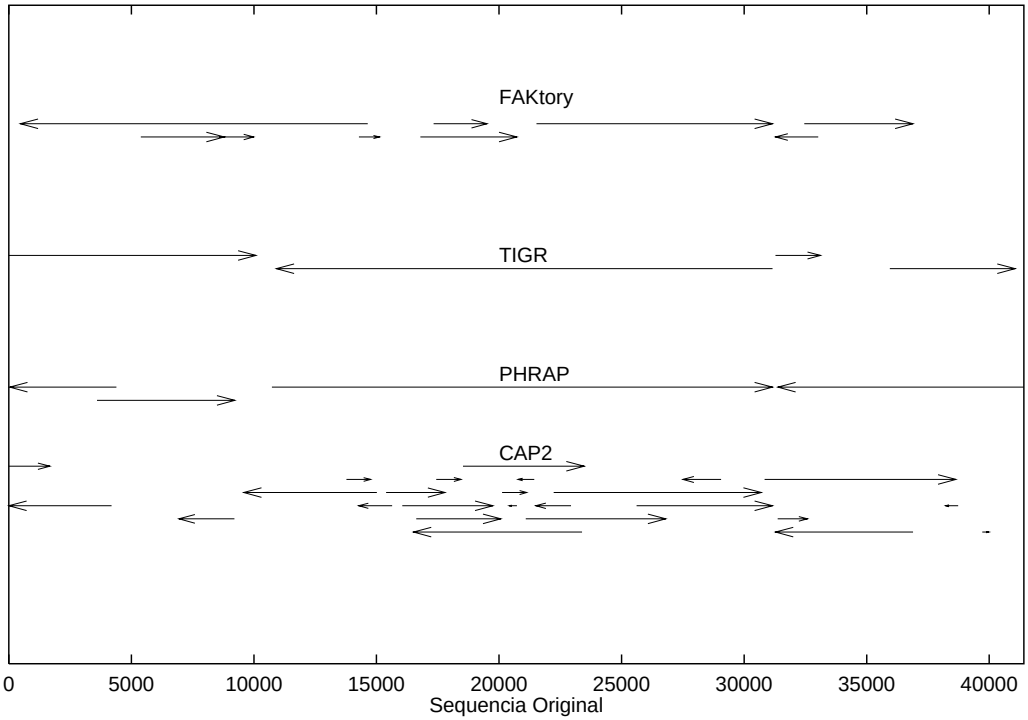

Figura 5.12: Gráfico da execução sobre o L518

Algumas considerações a respeito dos resultados obtidos pelas ferramentas

Através dos resultados obtidos pelas ferramentas nesta seção de testes, pudemos ver que a possível existência de fragmentos quimeras nos conjuntos de entradas dos programas não representa um problema tão sério para três das quatro ferramentas que avaliamos. Tanto o Phrap quanto o TIGR reconheceram a maioria dos falsos fragmentos existentes nos casos de testes e mesmo quando esta última ferramenta falhou na deteçcão de alguns deles, seus resultados continuaram razoáveis, assemelhando-se aos obtidos na

execução sobre o conjunto original de fragmentos.

O CAP2 falha ao considerar certos fragmentos quimeras como reais, mas o fato realmente intrigante é que estas falhas acabaram contribuindo para melhorar os resultados obtidos pela ferramenta. Isto nos levou a realizar repetidas observações de seus contigs e dos alinhamentos destes com a seqüência original para verificar se não havíamos cometido nenhum erro de análise, mas nenhuma falha foi observada.

Com relação ao FAKtory, esta ferramenta mostrou-se bastante ineficiente no tratamento de fragmentos quimeras. Além de não ter detectado nenhum dos falsos fragmentos existentes nos dois últimos casos de teste, o FAKtory montou alguns deles no interior de certos contigs, fazendo com que suas seqüências consensos apresentassem disparidades relevantes com as partes da seqüência original que cobrem.

5.3 Regiões Repetidas Dentro das Seqüências

A análise de várias moléculas de DNA tem levado ao descobrimento de regiões que se repetem dentro de suas seqüências, fato este que pode ocorrer algumas centenas de vezes, no caso das regiões serem apenas similares, ou dezenas de vezes quando elas se constituem exatamente das mesmas bases. O problema da existência de regiões repetidas pode comprometer o processo de montagem de várias formas. Primeiramente, a existência de regiões repetidas pode dar origem a falsas sobreposições envolvendo fragmentos provenientes dessas regiões. Estas sobreposições precisam então ser reconhecidas e tratadas de forma adequada pelas ferramentas no intuito de se evitar montagens erradas e, conseqüentemente, uma seqüência consenso muito diferente da seqüência original. Um outro problema relacionado com a existência de regiões repetidas dentro das seqüências surge quando temos um certo fragmento totalmente contido em uma delas. Como podemos determinar a localizacão deste fragmento, já que ele se encaixa bem em qualquer uma das regiões repetidas? No caso das regiões serem idênticas, este não seria um problema tão sério, já que poderíamos inserir o fragmento em qualquer uma delas. Agora, caso elas sejam apenas similares, o encaixe daquele fragmento em uma região diferente da qual ele provém poderia dar origem a uma seqüência consenso relativamente diferente da seqüência original.

Para avaliarmos a eficiência das ferramentas de montagem frente ao problema da existência de regiões repetidas dentro das seqüências, utilizamo-nos dos quatro casos de testes detalhados na Tabela 5.33.

Caso de teste	No. de fragms.	Tam. médio dos fragms.	No. Reg. Repetidas
B1496.rep	810	400	14
L247.rep	846	400	36
T-cell	639	400	3
Globina	1283	400	22

Tabela 5.33: Testes utilizados para avaliação das ferramentas no tratamento de regiões repetidas

Os conjuntos de fragmentos denominados B1496.rep e L247.rep foram obtidos artificialmente, através do programa *enzyme*, das seqüências originais dos cosmídeos B1496 e L247 adicionadas dos conjuntos de bases abaixo relacionados:

```
001 gaattcatgc cgaggaagga gaatagcttg acctcaggag gcggaggttg cagtgagctg
061 agattgcgcc actgcactcc agcctgggcg acagagcgag actccgtctc aaaaaaaaaa
121 aaaagaagaa gaaaaaaaag aagtggaggt tcagaggttt ggaaagaaac agaccaaagg
181 ctcataaccc cgtgggactg acctcacgag acacaggcac ctcctttcca aggggtctgg
241 ttttagtttt cttccttgag ccgaggcttg ccatctgatg cctagggatc cacatcttca
301 tttgtgaaaa ctgtacctgt caaatttgtg atttttcaat accaacaact cccatccaaa
361 acaatgaggc aaggagggag tcccgcagag gaggaagtgt gtgatttaac ccttttcttc
421 tggctcttta tagaaaaact tcaaaggaag ctt
```

```
gtgtgtgtgtgtgtgtgtgt
```

Estes conjuntos de bases foram inseridos nas seqüências daqueles DNAs por meio do programa *prepseq* e representam as regiões repetidas a serem tratadas pelas ferramentas de montagem (além daquelas presentes originalmente nas seqüências).

O conjunto de fragmentos detalhado na Tabela 5.33 com o nome de T-cell foi obtido da seqüência descrita em [37] e suas regiões repetidas correspondem à região da base 5000 à base 5999 duplicada por nós e inserida manualmente após as bases 20000 e 28000 daquela seqüência. O último caso de teste visto na Tabela 5.33 também foi obtido artificialmente por meio do *enzyme*, e a seqüência que usamos como entrada deste programa corresponde à seqüência original do agregado *β-like* da globina do gene humano. Esta seqüência contém várias regiões similares (13 cópias da seqüência ALU, com algumas delas contendo um grau de similaridade de 85% e 8 cópias parciais da seqüência LINE [23]) tornando-se assim um bom *benchmark* para nossas ferramentas de montagem.

5.3.1 B1496.rep

Foram inseridas um total de 11 seqüências de bases idênticas à primeira mostrada acima na seqüência original do cosmídeo B1496. As posições iniciais e finais destas

regiões podem ser vistas na Tabela 5.34, que mostra também informações a respeito de outras regiões repetidas existentes na seqüência original daquele DNA.

Tamanho da seqüência	Pos. inicial	Pos. final	Similaridade
453	242	695	100%
453	8907	9360	100%
453	10131	10584	100%
453	15429	15882	100%
453	18007	18460	100%
453	21948	22401	100%
453	26054	26507	100%
453	29926	30379	100%
453	31569	32022	100%
453	33071	33524	100%
453	33946	34399	100%
235	1	235	97%
235	41059	41294	97%

Tabela 5.34: Regiões repetidas presentes na seqüência modificada do cosmídeo B1496

Os resultados obtidos pelas ferramentas após a execução sobre o conjunto denominado B1496.rep são mostrados na Tabela 5.35 (Gráfico 5.13) abaixo.

Ferramentas	Contigs	Porções não cobertas	Cobertura(%)	Erros	Buracos
CAP22	27	27519 - 28713 42682 - 43169	96.34	12848	8812
FAKtory	16	1-820 2279-2505 33798-33904 37771-37808 38989-39107	97.16	219	272
TIGR	3	1-4816 46177-46278	89.37	8264	7225
Phrap	2	1-236 4859-5590	97.90	6277	9370

Tabela 5.35: Resultados das execuções sobre o B1496

Como podemos ver, o FAKtory corresponde à ferramenta que obteve os melhores resultados frente a este caso de teste, com seus 16 contigs cobrindo uma boa parte da

seqüência original e cujos alinhamentos com esta seqüência acumulam um número de erros e buracos bem menor que os presentes no alinhamento dos contigs gerados pelos outros sistemas.

Os 27 contigs gerados pelo CAP2 cobrem boa parte da seqüência original, mas, como nos casos de teste da seção anterior, seus alinhamentos com esta seqüência acumulam uma quantidade de erros e buracos bem maior que os determinados pelos contigs das outras ferramentas. Uma análise mais detalhada dos contigs gerados pelo CAP2 que possuem as menores similaridades com relação à seqüência original revela que suas partes possuem bons alinhamentos com porções não contíguas desta seqüência. Um exemplo disso ocorre com o 16o. contig gerado pela ferramenta. De acordo com o *cross_match*, as partes deste contig alinham-se com a seqüência original conforme a tabela seguinte, onde Pos. ini. do con. e Pos. ini. da seq. representam a posição inicial do contig e da seqüência respectivamente e Pos. fin. do con. e Pos. fin. da seq. suas posições finais.

Pos. ini. do con.	Pos. fin. do con.	Pos. ini. da seq.	Pos. fin. da seq.	Erros
1	1180	19545	20724	0
728	2857	9813	11942	0
2405	7249	20272	25116	0

Se compararmos as posições iniciais e finais da seqüência original mostradas acima com a localização de algumas regiões repetidas existentes na molécula, podemos ver que elas se assemelham ou estão próximas, levando-nos a crer que o 16o. contig tenha surgido a partir de montagens erradas envolvendo fragmentos provenientes dessas regiões (ou de regiões próximas a elas).

Com relação ao FAKtory, esta ferramenta gerou um número de contigs menor que o CAP2 ao final do processo de montagem mas que também determinam uma boa cobertura da seqüência original. Apesar do alinhamento com a seqüência original dos 16 contigs gerados pelo FAKtory apresentarem um número de erros e buracos bem menor que os alinhamentos relacionados aos contigs das outras ferramentas, o *cross_match* revela que o FAKtory também engana-se com as regiões repetidas presentes naquela seqüência. Três contigs construídos por esta ferramenta alinham-se com porções não-contíguas da seqüência original, como por exemplo o contig 3, cujas informações relacionadas ao seu alinhamento podem ser vistas na tabela abaixo:

Pos. ini. do con.	Pos. fin. do con.	Pos. ini. da seq.	Pos. fin. da seq.	Erros
1	90	33394	33482	1
234	701	694	1161	11
830	6237	12021	17428	1

Quanto ao Phrap, esta ferramenta gerou apenas dois contigs ao final do processo de montagem, contigs estes cujos alinhamentos com a seqüência original acumulam uma quantidade de erros e buracos bastante considerável, bem diferente dos resultados obtidos nos casos de testes anteriores. Uma análise mais detalhada dos contigs gerados por esta ferramenta revela que nenhum deles alinha-se de forma contígua com a seqüência original e, de acordo com os resultados gerados pelo *cross_match*, podemos notar que estes problemas de alinhamento também estão, de certa forma, relacionados à presença de regiões repetidas na seqüência original. Pegando-se por exemplo o 2o. contig gerado pelo Phrap, o alinhamento de duas de suas porções, cujas informações podem ser vistas na tabela seguinte, revela que esta ferramenta juntou em um único contig duas partes não contíguas da seqüência original. Aparentemente, esta falha de montagem deve-se ao fato da extremidade final do primeiro alinhamento e da extremidade inicial do segundo alinhamento estarem bem próximas de uma das regiões repetidas mostradas na Tabela 5.34, o que pode ter confundido a ferramenta durante a fase de detecção de sobreposições relevantes entre os fragmentos.

Pos. ini. do con.	Pos. fin. do con.	Pos. ini. da seq.	Pos. fin. da seq.	Erros
5719	11922	11490	17693	0
11470	14019	33550	36099	0

Dos três contigs gerados pelo TIGR, o 1o. alinha-se de forma não-contígua com 11 porções distintas da seqüência original, o que determina o elevado número de erros mostrados na Tabela 5.35. Assim como as outras três ferramentas, os problemas de alinhamento do 1o. contig gerado pelo TIGR também podem estar relacionados com a existência de regiões repetidas dentro desta seqüência. Observando-se as informações relacionadas ao alinhamento de duas porções contíguas deste contig, mostradas abaixo, concluímos que o TIGR considerou como verdadeira as sobreposições entre os fragmentos provenientes das regiões que incluem as posições mostradas na terceira (primeira linha) e quarta (segunda linha) colunas da tabela e montou-os de forma errada em um único contig. Além de ser a possível causa desta falha do TIGR, as regiões repetidas parecem também ter influenciado de forma negativa a cobertura da seqüência original pela ferramenta. Neste caso de teste, o TIGR deixou mais de 10% da seqüência descoberta, bem diferente do que vinha ocorrendo até o momento e o pior valor dentre os apresentados pelas ferramentas.

Pos. ini. do con.	Pos. fin. do con.	Pos. ini. da seq.	Pos. fin. da seq.	Erros
3712	8489	34002	29225	0
8037	9184	1147	1	3

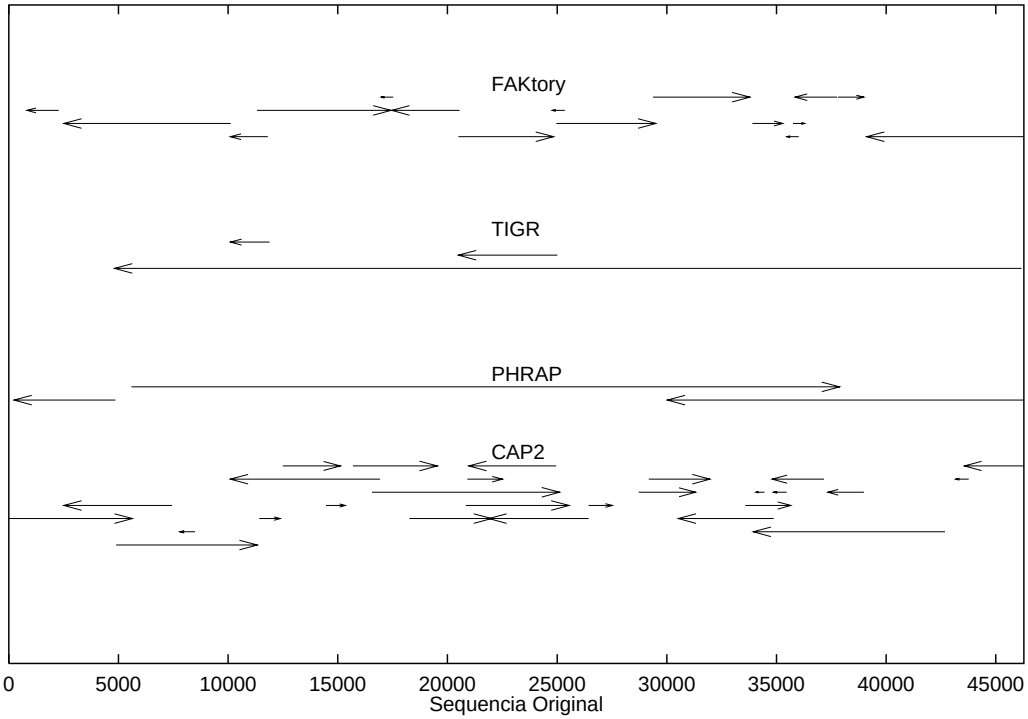

Figura 5.13: Gráfico da execução sobre o B1496

5.3.2 L247.rep

Um grande número de regiões repetidas foi inserido na seqüência original do cosmídeo L247 e as posições iniciais e finais de algumas delas podem ser vistas na Tabela 5.37, que também inclui informações relacionadas a uma região repetida real existente na seqüência desse DNA.

Na Tabela 5.36 (Gráfico 5.14) abaixo mostramos os resultados obtidos pelas quatro ferramentas.

Ferramentas	Contigs	Porções não cobertas	Cobertura(%)	Erros	Buracos
CAP2	33	12906-13041 19810-20547 45871-48316	92.37	19363	9832
FAKtory	4	9654-11109	97.19	427	310
TIGR	4	33368-33439	99.84	1186	715
Phrap	2	33368-33462	99.80	2	51

Tabela 5.36: Resultados das execuções sobre o L247

Tamanho da seqüência	Pos. inicial	Pos. final	Similaridade
20	675	695	100%
20	3973	3993	100%
453	7045	7498	81.64%
20	9942	9962	100%
453	9784	10237	79.65%
20	13676	13696	100%
453	14168	14621	78.98%
20	15762	15782	100%
20	16535	16555	100%
20	20476	20496	100%
453	23391	23844	80.75%
20	23952	23972	100%
640	5246	5886	99.84%
640	8781	9421	99.84%

Tabela 5.37: Algumas regiões repetidas presentes na seqüência modificada do cosmídeo L247

Diferente do que ocorreu no caso de teste anterior, desta vez a ferramenta que apresentou os melhores resultados foi o Phrap. Além da excelente cobertura obtida pelos seus dois contigs, o número de erros e buracos presentes em seus alinhamentos com a seqüência original impressiona quando comparado aos valores das outras ferramentas.

Os trinta e três contigs gerados pelo CAP2 deixam uma porção considerável da seqüência original descoberta e o elevado número de erros e buracos presentes em seus alinhamentos pode novamente estar relacionado com a presença de regiões repetidas na seqüência original. Como ocorreu na seção de teste anterior, alguns contigs gerados pelo CAP2 foram construídos de forma errada, sendo compostos de partes que possuem bons alinhamentos com porções não contíguas da seqüência original.

Novamente, o FAKtory determinou uma boa cobertura da seqüência original (a única parte descoberta inclui uma das regiões repetidas mostradas na Tabela 5.37), com o alinhamento de seus contigs acumulando um número de erros e buracos bem menor que os determinados pelos contigs do CAP2 e do TIGR. Estes valores provêm do fato de que apenas o 4o. contig gerado pela ferramenta possui problemas de alinhamento, cujas informações mostradas na tabela a seguir não nos permitem determinar claramente se as falhas em sua construção foram determinadas pelas regiões repetidas existentes na seqüência.

Pos. ini. do con.	Pos. fin. do con.	Pos. ini. da seq.	Pos. fin. da seq.	Buracos
2	1763	11096	9354	63
1080	5451	5326	9653	54

Como já estávamos acostumados a ver, o Phrap novamente obteve excelentes resultados. Ao contrário do que ocorreu na subseção anterior, desta vez o Phrap não gerou nenhum contig com problemas de alinhamento e, além disso, eles cobrem quase 100% da seqüência original. Estes valores obtidos pelo Phrap levam-nos a crer que esta ferramenta trata de forma adequada regiões repetidas da molécula que não sejam 100% similares e também aquelas contendo um pequeno número de bases, como as 29 seqüências de tamanho 20 inseridas neste caso de teste.

Com relação ao TIGR, esta ferramenta também obteve bons resultados frente a este caso de teste, apesar do alinhamento de seus contigs apresentar uma quantidade bem maior de erros e buracos que os alinhamentos dos contigs gerados pelo Phrap. Isto se deve ao fato do 1o. contig gerado pelo TIGR apresentar problemas de alinhamento semelhantes aos dos ocorridos na seção anterior, mas ao invés de suas porções alinharem-se com regiões distantes uma da outra na seqüência original, elas alinham-se com regiões contíguas desta seqüência, o que não chega a comprometer tanto os resultados obtidos pela ferramenta.

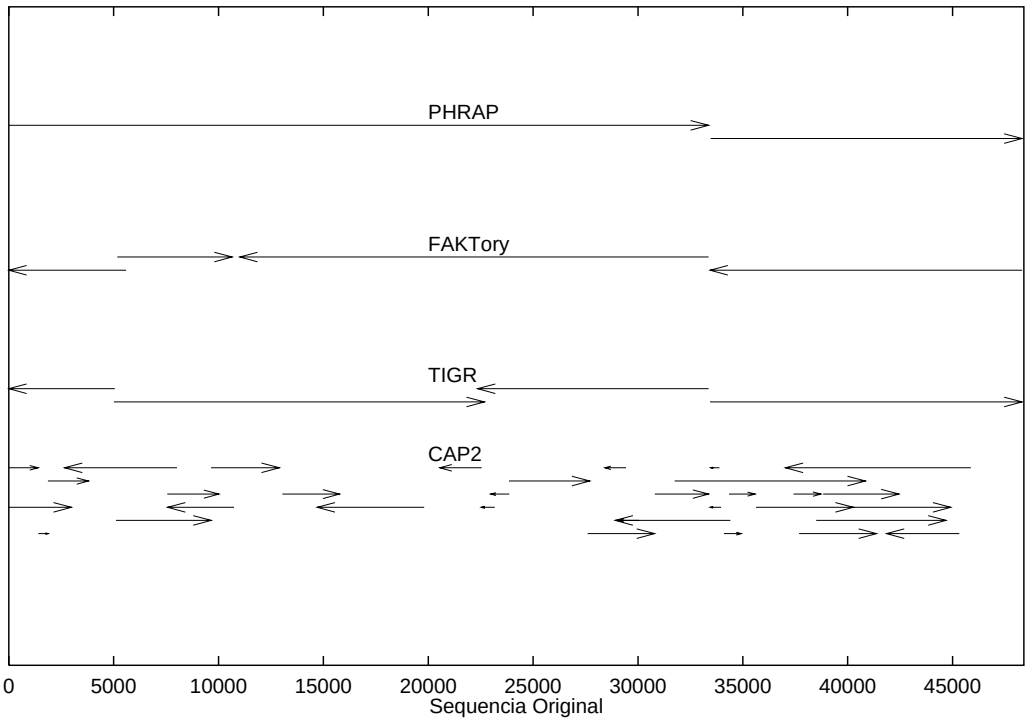

Figura 5.14: Gráfico da execução sobre o L247

5.3.3 T-cell

A seqüência relacionada a este conjunto de fragmentos possui um total de 3 regiões repetidas, manualmente inseridas por nós e cujas posições iniciais e finais podem ser vistas na Tabela 5.38.

Tamanho da seqüência	Pos. inicial	Pos. final	Similaridade
1000	5000	5999	100%
1000	20000	20999	94.29%
1000	29120	30119	93.00%

Tabela 5.38: Regiões repetidas presentes na seqüência modificada do T-cell

Analisando-se os números da Tabela 5.39 (Gráfico 5.15), que contém os resultados da execução das quatro ferramentas sobre este caso de teste, podemos ver que, novamente, o TIGR e o Phrap foram aquelas que obtiveram os melhores resultados frente ao problema de existência de regiões repetidas dentro das seqüências. Cada um dos contigs gerados por elas alinham-se com uma única porção da seqüência original e cobrem praticamente 100% desta seqüência.

Ferramentas	Contigs	Porções não cobertas	Cobertura(%)	Erros	Buracos
CAP2	29	5252-6069 25803-26275 35143-36468	92.57	12197	5942
FAKtory	7	5252-5674 10986-11014 35144-35208	95.04	1958	1551
TIGR	4	10986-11014 35142-35208	99.65	0	3
Phrap	4	-	100	0	3

Tabela 5.39: Resultados das execuções sobre o T-cell

O CAP2 mostrou-se novamente como a pior ferramenta no que diz respeito ao tratamento de regiões repetidas. Além de determinar a menor cobertura da seqüência original, o alinhamento de seus contigs com esta seqüência acumula uma grande quantidade de erros e buracos, com um deles alinhando-se com duas porções não-contíguas da seqüência original, de acordo com o *cross_match*.

Com relação ao FAKtory, esta ferramenta também não obteve bons resultados frente a este caso de teste, ao contrário do que ocorreu anteriormente. Apesar de seus contigs

cobrirem boa parte da seqüência original, o alinhamento de dois deles com esta seqüência apresenta uma grande quantidade de erros e, principalmente, buracos, que se reflete nos números apresentados na Tabela 5.39. Uma análise mais detalhada de um destes contigs leva-nos a acreditar que o FAKtory confundiu-se com as repetições artificiais existentes neste caso de teste, juntando em um único contig duas porções não-contíguas da seqüência original que se sobrepõem com duas cópias da região repetida aí existente (tabela abaixo).

Pos. ini. do con.	Pos. fin. do con.	Pos. ini. da seq.	Pos. fin. da seq.	Buracos
1	3364	17685	20998	87
2336	8469	29069	35143	17

Os resultados gerados pelo TIGR e pelo Phrap são muito parecidos. As duas ferramentas determinaram uma excelente cobertura da seqüência original e o alinhamento de seus contigs apresentam uma quantidade de erros e buracos desprezível com relação a essa seqüência. Nenhuma das repetições existentes neste caso de teste parece ter confundido alguma das duas ferramentas, que geraram contigs possuindo um excelente alinhamento com uma única porção da seqüência original.

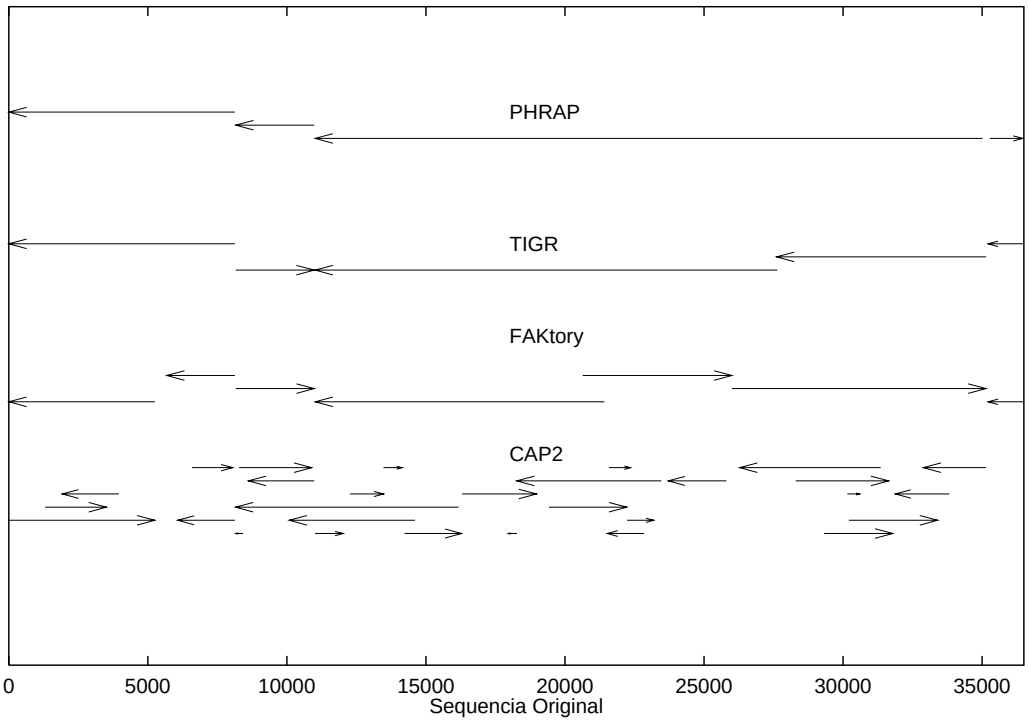

Figura 5.15: Gráfico da execução sobre o T-cell

5.3.4 Globina

As regiões repetidas existentes na seqüência original da Globina estão relacionadas na Tabela 5.40.

Tamanho da seqüência	Pos. inicial	Pos. final	Similaridade
3200	33380	36578	93.02%
3200	38317	41478	100%

Tabela 5.40: Regiões repetidas presentes na seqüência da Globina

A Tabela 5.41 (Gráfico 5.16) abaixo mostra o resultado das execuções das ferramentas sobre este caso de teste.

Ferramentas	Contigs	Porções não cobertas	Cobertura(%)	Erros	Buracos
CAP2	43	32867-35269 43032-43134 43445-46234 60727-65443 71827-73308	84.31	24793	14732
FAKtory	12	1-4284 23505-27325 36818-37719 38567-41279 43139-44896 46089-52067 62244-62427 72372-73308	71.93	4256	2681
TIGR	3	39279-40205	99.87	2	0
Phrap	3	34378-36526 41593-44205	93.50	130	73

Tabela 5.41: Resultados das execuções sobre a Globina

Como no caso de teste anterior, o TIGR e o Phrap mostraram ser as melhores ferramentas para tratamento de regiões repetidas dentro das seqüências. Ambas geraram apenas três contigs ao final do processo de montagem e todos eles alinham-se com porções únicas da seqüência original. O fato de não terem realizado montagens erradas, devido às regiões repetidas existentes dentro da seqüência original, levou as duas ferramentas (principalmente o TIGR) a construir contigs de alta similaridade com esta.

Vale observar também que as porções não cobertas pelos seus contigs correspondem às porções da molécula original contidas nas regiões repetidas presentes.

Os resultados obtidos pelo CAP2 frente a este caso de teste comprovam a relativa ineficiência desta ferramenta no tratamento de regiões repetidas. Além de não cobrirem várias partes da seqüência original, alguns dos 43 contigs gerados pelo CAP2 foram construídos de forma errada, apresentando os mesmos problemas de alinhamento das seções anteriores e determinando o elevado número de erros e buracos mostrados na Tabela 5.41.

Com resultados piores do que os obtidos nas subseções anteriores, o FAKtory também não pode ser considerado uma boa ferramenta para tratamento de regiões repetidas dentro das seqüências. Três, dos doze contigs gerados por ela ao final do processo de montagem, determinam uma elevada quantidade de erros e buracos em seus alinhamentos com a seqüência original, e um deles (o contig 3) corresponde exatamente àquele que cobre a 1ª cópia da região repetida presente nesta seqüência. Uma análise mais detalhada de outro contig gerado pelo FAKtory que possui problemas de alinhamento (o contig 1) revela que suas partes alinham-se com quatro porções não contíguas da seqüência original, falha esta que pode estar relacionada com a existência de outras regiões repetidas nesta seqüência.

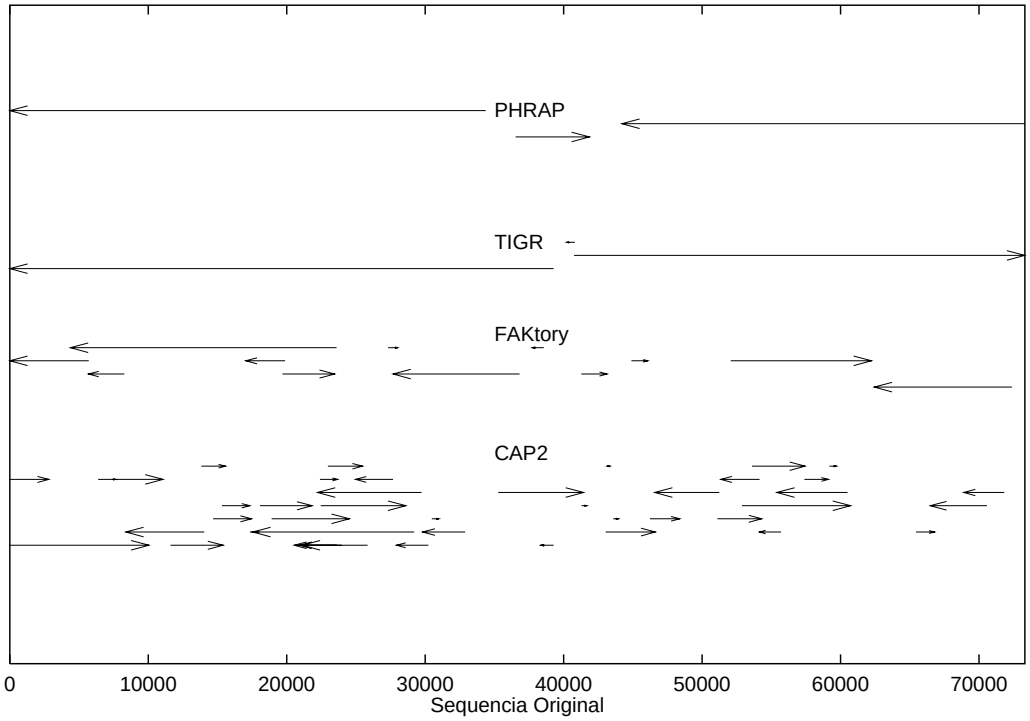

Figura 5.16: Gráfico da execução sobre a Globina

Algumas considerações a respeito dos resultados obtidos pelas ferramentas

Através dos resultados acima, podemos ver que o problema da existência de regiões similares dentro das seqüências das moléculas constitui-se na maior barreira a ser superada pelas ferramentas de montagem. Fragmentos quimeras e erros de seqüenciamento possuem também uma grande influência na exatidão do processo de montagem mas nada comparado ao problema de repetições, principalmente quando consideramos a presença de regiões idênticas dentro das seqüências e de tamanhos consideráveis.

O TIGR e em especial o Phrap demonstraram ótimos resultados nesta seção de testes, onde se confundiram apenas no tratamento do primeiro conjunto de fragmentos, advindos de uma seqüência contendo várias regiões 100% similares.

Bem diferente do que ocorreu na seção do problema de erros no seqüenciamento, desta vez o CAP2 apresentou resultados regulares, sendo enganado pelas regiões repetidas nos quatro casos de testes, inclusive naqueles cujas regiões repetidas não são totalmente idênticas.

Com relação ao FAKtory, esta ferramenta apresentou resultados bastante diferentes. Curiosamente, ela tratou de forma melhor os fragmentos do primeiro conjunto que, devido à alta similaridade de suas regiões, supunhamos que teria maior influência na exatidão dos resultados das ferramentas, enquanto que os dois últimos casos de testes foram tratados de forma regular pelo FAKtory. Isto nos leva a concluir que, mais do que a similaridade das regiões repetidas, o tamanho destas possui uma influência considerável nos resultados gerados pelo FAKtory.

5.4 Considerações Finais

Como pudemos ver, realmente existe uma grande disparidade entre os resultados gerados pelas ferramentas de montagem e cada uma delas parece tratar um problema em especial de melhor forma que as outras. Isto, adicionado ao fato de termos executado todas as ferramentas com os valores *defaults* para seus parâmetros de entrada, dificulta o estabelecimento de um "*ranking*" para os quatro sistemas avaliados e nos leva a concluir que a escolha da melhor ferramenta a ser utilizada em um projeto de seqüenciamento depende das informações que temos em mãos a seu respeito (taxa de erros na seqüência de seus fragmentos, tamanho estimado da molécula sendo determinada, se a probabilidade desta conter regiões repetidas é elevada etc.).

Os resultados gerados pelo CAP2 mostram que esta ferramenta realiza um excelente tratamento do problema de erros no seqüenciamento, mas apresenta falhas ao ser executada sobre conjuntos de fragmentos contendo quimeras e sobre aqueles obtidos de seqüências contendo regiões repetidas. Ao contrário do CAP2, o TIGR parece ser uma

boa ferramenta para tratamento de regiões repetidas, mas não é recomendável utilizá-lo em projetos de seqüenciamento cujos fragmentos a serem processados apresentam taxa de erros consideráveis. Além de ter construído vários contigs cujos alinhamentos com a seqüência original apresentam uma quantidade considerável de erros e buracos, a mensagem de *bus error!* gerada durante a execução do TIGR sobre alguns casos de teste da seção 5.1 deixou-nos bastante intrigados. Não encontramos nenhum problema nestes casos de teste que pudesse ocasionar o aborto do programa e o fato dos fragmentos desta seção corresponderem a fragmentos reais aumenta ainda mais nossas dúvidas a respeito do que pode ter causado este comportamento inesperado no TIGR.

Com relação ao FAKtory, podemos considerar que a qualidade de seus resultados encontra-se num ponto intermediário dos conseguidos pelas quatro ferramentas. Apesar desta ferramenta ter apresentado apenas resultados regulares ao final da execução sobre os casos de testes descritos anteriormente, durante a seção de erros no seqüenciamento e regiões repetidas eles foram bem melhores que os obtidos pelo TIGR e CAP2, respectivamente. Talvez a utilização do FAKtory seja mais propícia nos casos em que não possuímos nenhuma informação relevante a respeito do projeto de seqüenciamento sendo efetuado e da molécula sendo determinada.

Os resultados obtidos pelo Phrap são, de longe, os melhores dentre os gerados pelos quatro sistemas, mas mesmo essa ferramenta apresentou algumas falhas de montagem no decorrer da execução dos testes. Apesar da aparente supremacia do Phrap, acreditamos que doze não seja um número de casos de testes suficiente para considerarmos esta ferramenta como a melhor dentre as quatro sendo comparadas, mas com certeza não deixaríamos de recomendá-la para utilização em projetos de seqüenciamentos que incluem os três problemas abordados nas seções 5.1, 5.2 e 5.3.

Capítulo 6

Análise de Desempenho

No intuito de avaliarmos o desempenho de cada uma das ferramentas no processo de montagem de fragmentos, executamos cada uma delas sobre nove conjuntos distintos de fragmentos, artificialmente obtidos da seqüência do cosmídeo B2126. Estes casos de testes possuem tamanhos variados e foram criados atribuindo-se diferentes valores para o parâmetro de profundidade da cobertura do programa *enzyme*. Na tabela abaixo mostramos o número de fragmentos existente em cada conjunto e alguns dos parâmetros utilizados para sua criação.

Saídas	Tam. médio dos fragm.	Núm. de fragms.	Prof. da Cobertura
b2126.365	232	365	1.90
b2126.750	232	750	3.90
b2126.1057	232	1057	5.50
b2126.1503	232	1503	7.82
b2126.1854	232	1854	9.65
b2126.2114	232	2114	11.00
b2126.2517	232	2517	13.10
b2126.2882	232	2882	15.00
b2126.3266	232	3266	17.00

A variação do parâmetro profundidade da cobertura como mostrado acima permitiu-nos obter conjuntos de fragmentos com tamanhos variados (365, 750, 1057, 1503, 1854, 2114, 2517, 2882 e 3266 fragmentos), o que nos possibilita um análise mais detalhada do tempo de CPU ocupado por cada uma das ferramentas.

6.1 Resultado Geral dos Testes

Executando-se quatro vezes as ferramentas para cada um dos nove casos de teste acima citados, obtivemos os tempos médios de utilização da CPU mostrados na Tabela 6.1 e representados graficamente na Figura 6.1.

Casos de testes	Phrap	TIGR	FAKtory	CAP2
b2126.365	0min5s	0min16s	0min28s	1min30s
b2126.750	0min15s	0min40s	1min33s	6min39s
b2126.1057	0min28s	1min3s	2min47s	8min28s
b2126.1503	0min51s	1min5s	5min41s	11min13s
b2126.1854	1min21s	2min4s	8min44s	15min16s
b2126.2114	1min45s	2min30s	11min37s	22min57s
b2126.2517	5min18s	2min55s	16min22s	27min37s
b2126.2882	8min12s	3min18s	21min51s	36min9s
b2126.3266	11min35s	3min45s	28min21s	46min12s

Tabela 6.1: Tempo de utilização da CPU por cada uma das ferramentas ao serem executadas sobre os nove conjuntos de entrada

.

Estes valores foram tomados por meio do comando *time* do UNIX, com as ferramentas sendo executadas em uma *Sparcstation*1000 com 2304Mb de memória RAM compartilhada, e mostram que existe uma grande disparidade entre os tempos que cada uma das ferramentas leva para terminar seu processamento.

À primeira vista, a ferramenta que parece possuir o melhor desempenho é o Phrap. Este programa ocupou menos de um segundo do tempo total da CPU nos primeiros quatro casos de teste, onde se inclui um conjunto contendo mais do que 1500 fragmentos. Apesar destes bons resultados iniciais, o tempo total de CPU ocupado pelo Phrap começa a crescer de forma considerável após sua execução sobre o caso de teste contendo 2517 fragmentos, fato este que também ocorre com o CAP2 e que pode ser visto na Figura 6.1.

Com resultados iniciais um pouco piores que os do Phrap mas também apresentando um bom desempenho, o TIGR ocupa um tempo bem menor de CPU que suas concorrentes CAP2 e FAKtory, gastando menos de dois minutos do tempo total deste recurso durante todo o processo de montagem dos fragmentos presentes nos quatro casos iniciais acima citados. Mais importante que isso é o fato de que o TIGR corresponde à ferramenta cuja curva que representa o tempo de ocupação da CPU é a que possui o menor crescimento em função do tamanho das instâncias de entrada (Figura 6.1).

O FAKtory e o CAP2 correspondem às ferramentas que obtiveram os piores desem-

penhos dentro desta seção de testes. Ao término da execução sobre a instância contendo 365 fragmentos, o FAKtory havia ocupado, em média, apenas 28s do tempo total da CPU, porém este tempo triplica ao fim da montagem dos fragmentos pertencentes ao segundo caso de teste, chegando a mais de 5min ao serem considerados os 1502 fragmentos do 4o. conjunto. Com resultados bem piores que os obtidos pelo FAKtory, o CAP2 compromete-se ao ocupar, em média, mais de 45min do tempo total da CPU no último caso de teste. Além do ruim desempenho sobre esta instância de dados, os fragmentos pertencentes aos conjuntos menores são montados pelo CAP2 também em um tempo bem maior que aqueles requisitados pelas outras ferramentas. Os 1057 fragmentos da 3ª instância, por exemplo, são montados pelo CAP2 em um tempo quase quatro vezes maior que o necessitado pelo FAKtory e comparando-o com os tempos requisitados pelo TIGR e pelo Phrap obtemos uma disparidade ainda mais relevante.

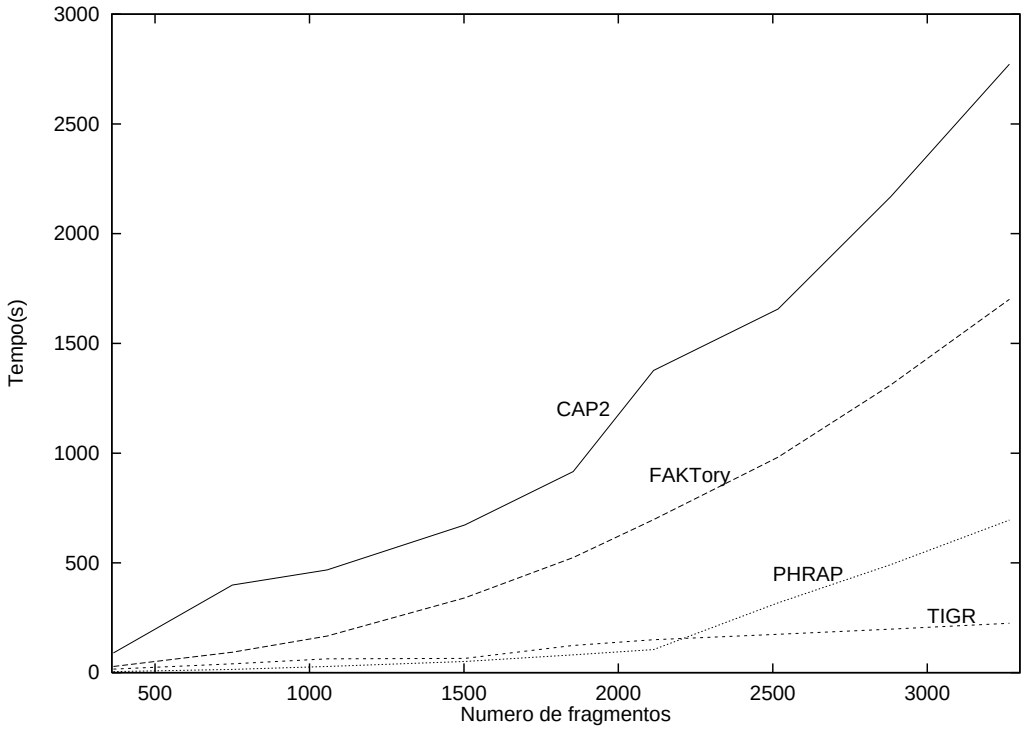

Figura 6.1: Gráfico de desempenho (Número de fragmentos x Tempo)

Capítulo 7

Conclusão

Ao fim de um trabalho como este, muitas vezes nos deparamos com mais dúvidas do que tínhamos antes do início do empreendimento. Neste capítulo iremos relatar o que consideramos mais relevante em cada um dos capítulos anteriores, ousando sugerir tópicos que, acreditamos, possam abrir caminhos para novas pesquisas no futuro.

Os assuntos abordados no primeiro capítulo desta dissertação permitiram-nos estabelecer um contato com vários conceitos da área de Biologia Molecular, como aqueles relacionados à descrição do DNA, e nos deixaram a par dos vários métodos utilizados para seqüenciamento dessa molécula. Tudo isto, além de ampliar o pouco conhecimento que tínhamos a respeito destes assuntos, levou-nos a estreitar a nova relação existente entre duas áreas de pesquisa aparentemente antagônicas.

Apesar dos vários métodos utilizados por cada ferramenta para tratamento dos problemas relacionados ao processo de montagem possuírem um forte embasamento teórico, como pôde ser visto no Capítulo 3, constatamos no Capítulo 5 que a prática condena alguns deles. Os dados obtidos mostram que nenhuma das ferramentas consegue tratar de forma totalmente correta todos os três problemas a que foram submetidas. Similaridades e coberturas pouco abaixo de 100% são aceitáveis, mas, em certos casos de testes, algumas ferramentas falharam de forma surpreendente, chegando a construir contigs que cobrem menos de 50% da seqüência original ou que determinam quantidades muito grandes de erros e buracos em seus alinhamentos com esta seqüência.

O Capítulo 5 mostra também que realmente existe uma grande disparidade entre os resultados obtidos por cada sistema de montagem, fato este que justifica nossa preocupação em compará-los e a proposta de novos trabalhos, onde um grupo diferente de ferramentas (em especial ferramentas comerciais) seja avaliado frente a outros casos de testes (como feito em [32]). Estes poderiam permitir uma comparação das ferramentas no que diz respeito ao tratamento de outros problemas relacionados ao processo de montagem como também no que diz respeito à quantidade total de memória exigida

por cada uma delas durante suas execuções.

Apresentando uma disparidade também relevante do tempo de CPU gasto por cada ferramenta durante sua execução, o Capítulo 6 leva-nos à conclusão de que, em períodos de concorrência entre empresas de seqüenciamento, como o que vivemos hoje, a escolha de uma ferramenta de montagem que apresente apenas bons resultados não corresponde a uma boa prática. Com o aumento crescente do número de fragmentos a serem processados, a escolha de um sistema com baixo desempenho pode comprometer o projeto como um todo, principalmente no caso da existência de prazos a serem cumpridos.

Apesar de algumas falhas apresentadas pelas ferramentas avaliadas neste projeto, acreditamos que o problema da montagem de fragmentos já tenha sido bastante estudado pelos pesquisadores, o que se reflete no grande número de sistemas de montagem disponíveis atualmente. Este fato desviou-nos de um dos objetivos principais deste trabalho e, adicionado ao crescente número de organismos já seqüenciados, torna a predição de genes um campo de pesquisa bastante promissor dentro da área de Biologia Computacional.

Tendo a seqüência completa do DNA de um certo organismo em mãos e os seus genes totalmente reconhecidos, os cientistas estarão prontos para realizar o estudo das funcionalidades destes genes, outro campo de pesquisa promissor e de fundamental importância para a resolução de várias questões que envolvem este antro de mistérios que é a vida.

Referências Bibliográficas

[1] *Dr. Xiaoqiu Huang.* [ftp://cs.mtu.edu/pub/huang/].

[2] *FAPESP Genome Program.* [http://watson.fapesp.br/Genoma.htm].

[3] *Human Genome Project.* [http://www.ornl.gov/TechResources/Human_Genome/home.html].

[4] *NCBI.* [http://www.ncbi.nlm.nih.gov/Genbank/index.html].

[5] O ácido desoxirribonucléico: DNA. Apostila do Instituto de Biologia da Universidade de São Paulo.

[6] *The Phred/Phrap/Consed System.* [http://bozeman.mbt.washington.edu/].

[7] *Staden Package Program.* http://www.mrc-lmb.cam.ac.uk/pubseq/overview.html.

[8] *TIGR.* [http://www.tigr.org/softlab/n_license.html].

[9] M. Behzad and G. Chartrand. *Introduction to the Theory of Graphs.* Allyn & Bacon, Boston, 1971.

[10] J. K. Bonfield, K. F. Smith, and R. Staden. A new DNA sequence assembly program. In *Nucleic Acids Research*, volume 23, pages 4992–4999. 1995.

[11] S. Dear and R. Staden. A sequence assembly and editing program for efficient management of large projects. In *Nucleic Acids Research*, volume 19, pages 3907–3911. 1991.

[12] De Robertis e De Robertis Jr. *Bases da Biologia Celular e Molecular.* Guanabara Koogan, Rio de Janeiro, 1993.

[13] M. L. Engle and C. Burks. Artificially generated data sets for testing DNA sequence assembly algorithms. In *Genomics*, number 16, pages 286–288. 1993.

[14] M. L. Engle and C. Burks. Genfrag 2.1: New features for more robust fragment assembly benchmarks. In *Comput. Appl. Biosci.*, volume 10, pages 567–568. 1994.

[15] C. E. Ferreira, C. C. de Souza, and Y. Wakabayashi. Rearrangement of DNA fragments: a branch-and-cut algorithm. Technical Report 9701, Universidade de São Paulo, Departamento de Ciência da Computação, 1997.

[16] R. D. Fleischmann, M. D. Adams, O. White, R.A. Clayton, E. F. Kirkness, A. R. Kerlavage, C. J. Bult, J. F. Tomb, B. A. Dougherty, J. M Merrick, and et al. Whole-genome random sequencing and assembly of *haemophilus influenzae rd*. In *Science*, volume 269, pages 496–512. July 1995.

[17] T. R. Gingeras, J. P. Milazzo, and R. J. Roberts. Computer programs for the assembly of DNA sequences. In *Nucleic Acids Research*, volume 7, pages 529–545. 1979.

[18] P. Green. *Documentation for Phrap and Cross_Match (Version 0.990319).* [http://bozeman.mbt.washington.edu/phrap.docs/phrap.html].

[19] P. Green. *General documentation on swat, crossmatch, phrap, and phrapview.* [http//www.phrap.org/phrap.docs/general.html].

[20] D. Gusfield. *Algorithms on Strings, Trees, and Sequences.* Cambridge University Press, Cambridge, 1997.

[21] X. Huang. A contig assembly program based on sensitive detection of fragment overlaps. In *Genomics*, number 14, pages 18–25. 1992.

[22] X. Huang. On global sequence alignment. In *Comput. Appl. Biosci.*, volume 10, pages 227–235. 1994.

[23] X. Huang. An improved sequence assembly program. In *Genomics*, number 33, pages 21–31. 1996.

[24] T. Hunkapiller, R. J. Kaiser, B. F. Koop, and L. Hood. Large-scale and automated DNA sequence determination. In *Science*, volume 254, pages 59–67. October 1991.

[25] T. Jiang and M. Li. DNA sequencing and string learning. In *Mathematical Systems Theory*, number 29, pages 387–405. 1996.

[26] J. D. Kececioglu. *Exact and approximation algorithms for DNA sequence reconstruction.* PhD thesis, Department of Computer Science, The University of Arizona, tucson, 1991.

[27] J. D. Kececioglu, M. Li, and J. Tromp. Reconstructing a DNA sequence from erroneous copies. In *Theoretical Computer Science*, volume 185, pages 3–13.

[28] J. D. Kececioglu and E. W. Myers. Combinatorial algorithms for DNA sequence assembly. In *Algorithmica*, number 13, pages 7–51. 1995.

[29] C. B. Lawrence, N. W. Parrot, T. C. Flood, L. Gu, L. Zhang, M. Jain, S. Larson, and E. W. Myers. The genome reconstruction manager: A software environment for supporting high-throughput DNA sequencig. In *Genomics*, number 23, pages 192–201. 1994.

[30] J. Meidanis and J. C. Setubal. *Uma Introdução à Biologia Computacional.* IX Escola de Computação – SBC, Recife, 1994.

[31] J. Meidanis and J. C. Setubal. *Introduction to Computational Molecular Biology.* PWS Publishing Co., Boston, 1997.

[32] M. J. Miller and J. I. Powell. A quantitative comparison of DNA sequence assembly programs. *Journal of Computational Biology*, 1(4), 1994.

[33] E. W. Myers. Incremental alignment algorithms and their application. Technical Report 86-2, University of Arizona, 1986.

[34] E. W. Myers. Toward simplifying and accurately formulating fragment assembly. *Journal of Computational Biology*, 2(2):275–290, 1995.

[35] E. W. Myers, S. Miller, and E. Anson. *FAKtory: A software environment for DNA Sequencing.* [http://www.cs.arizona.edu/faktory/].

[36] H. Peltola, H. Soderlund, and E. Ukkonen. Seqaid: a DNA sequence assembly program based on a mathematical model. In *Nucleic Acids Research*, number 12, pages 307–321. 1983.

[37] D. Seto, B. F. Koop, and L. Hood. An experimentally derived data set constructed for testing large-scale DNA sequence assembly algorithms. In *Genomics*, number 15, pages 673–676. 1993.

[38] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, (147):195–197, 1981.

[39] R. Staden. Automation of the computer handling of gel reading data produced by the shotgun method of DNA sequencing. In *Nucleic Acids Research*, volume 10, pages 4731–4751. 1982.

[40] G. G. Sutton, O. White, M. D. Adams, and A. R. Kerlavage. Tigr assembler: A new tool for assembling large shotgun sequencing projects. In *Genome Science & Technology*, pages 9–19. 1995.