Um Estudo dos Algoritmos de Montagem de Fragmentos de DNA

Melissa Lemos
e-mail: melissa@inf.puc-rio.br

Antônio Basílio
Departamento de Bioquímica e Biologia Molecular, Fundação Oswaldo Cruz
Rio de Janeiro - RJ
e-mail: antonio@gene.dbbm.fiocruz.br

Marco Antônio Casanova
e-mail: casanova@inf.puc-rio.br

PUC-RioInf.MCC05/03 Fevereiro, 2003

Abstract

The human genome project is a program to map and sequence the entire human genome. A number of model organisms were selected for complete sequencing, partly in order to develop new technology for mapping, sequencing and sequence analysis. In addition, the sequences from these genomes were expected to facilitate the elucidation of the functions of genes and sequences in the human genome.
One of the main problems of DNA sequencing on a large scale is that its methods only obtain a small part of the DNA. After breaking a sequence into many fragments, cloning them and sequencing them, there is a set of fragments which needs to be merged for the reconstruction of the original DNA sequence. This monograph presents the biological and computational context of DNA sequence assembly.

Keywords: Sequence Assembly Algorithm, Bioinformatic, Shotgun, DNA, Genome.

Resumo

O Projeto Genoma Humano tem como objetivo o mapeamento e sequenciamento do genoma humano inteiro. Além do genoma humano, alguns organismos modelo foram selecionados para serem sequenciados e ajudarem a desenvolver novas tecnologias e a elucidar o genoma humano.
Um dos principais problemas do sequenciamento em larga escala é que seus métodos sequenciam somente uma pequena parte do DNA. Depois de quebrar uma sequência em vários fragmentos, cloná-los e sequenciá-los, há um conjunto de fragmentos que precisam ser montados para reconstrução da sequência de DNA original. Este trabalho apresenta o contexto biológico e computacional da montagem de fragmentos de sequências de DNA.

Palavras-chave: Algoritmo de Montagem de fragmentos, Bioinformática, Shotgun, DNA, Genoma.

Sumário

1. Introdução

O Projeto Genoma Humano iniciou-se oficialmente em 1990 como um programa para mapear e sequenciar o genoma humano inteiro; um esforço que revolucionaria a biologia e a medicina. Além do genoma humano, alguns organismos modelo foram selecionados para serem sequenciados. Isto foi feito para que novas tecnologias de mapeamento, sequenciamento e análise de sequências fossem desenvolvidas e, além disso, porque as sequências destes organismos facilitariam a elucidação de funções de genes e sequências do genoma humano [SL00].

Um dos grandes objetivos do Projeto Genoma Humano é sequenciar o genoma humano inteiro, obtendo as 3 bilhões de bases da cadeia que representam o DNA humano. Um dos principais problemas deste sequenciamento em larga escala é que seus métodos (Maxam-Gilbert ou Sanger [Coo94, WHR+87]) sequenciam somente uma pequena parte do DNA. Isto é, mesmo que o sequenciamento seja feito por uma máquina automatizada ou por um método mais manual, a maior subcadeia de DNA com qualidade que pode ser determinada em um procedimento em laboratório possui cerca de 400 bases. Desta forma, para sequenciar cadeias mais longas ou um genoma inteiro, o DNA precisa ser dividido em vários fragmentos pequenos que são sequenciados individualmente e depois montados para obter a cadeia inicial completa. Entre as principais diferenças dos métodos de sequenciamento em larga escala estão a forma com que a sequência original do DNA completo é dividido em pequenos fragmentos e a maneira como é feita a montagem destes fragmentos de aproximadamente 400 bases na sequência original [Gus97].

Apesar do aumento de esforço no sequenciamento, as técnicas básicas de sequenciamento de DNA desenvolvidas há mais de 20 anos atrás ainda estão em uso. Ao invés da utilização de novas tecnologias, um aumento no número de amostras processadas foi obtido através do desenvolvimento de instrumentos de sequenciamento automatizados, da preparação de amostras robotizadas, da construção de enzimas de sequenciamento e da utilização de corantes com alta sensibilidade [SL00].

Depois de quebrar uma sequência em vários fragmentos, cloná-los e sequenciá-los, há um conjunto de fragmentos que precisam ser montados para obtenção da sequência original. Este é um problema algorítmico conhecido como montagem de fragmentos ou *sequence assembly*.

A molécula original de DNA a ser sequenciada possui L pares de bases. A sequência de uma das fitas é representada por $\mathbf{a} = a_1a_2\ldots a_L$. No sequenciamento por *shotgun, N* pequenos fragmentos são gerados randomicamente com $l << L$ pares de bases de comprimento que são determinadas experimentalmente. Todos estes fragmentos estão contidos em **a**, mas a ordem relativa deles é desconhecida. Além disso, existem outras complicações, porque cada fragmento f_i é conhecido aproximadamente e sua orientação é desconhecida. Ou seja, há 50% de chance de f_i ser escrito com a polaridade de **a** e 50% de chance de ser o complemento reverso, f_{ir}, de sua posição em **a**. E, ainda, muitas sequências contém regiões exatamente, ou aproximadamente, iguais, conhecidas como repetições. Se uma região repetitiva for muito grande, não existirá nenhum fragmento que contenha a região inteira e não será possível posicionar uma região repetitiva corretamente em relação às outras regiões. Todas estas características complicam a montagem de fragmentos [IW95].

Para amenizar estas complicações, a sequência **a** é copiada várias vezes, em uma média de $c = Nl/L$. Os experimentos escolhem tipicamente $c \in [3, 10]$ [IW95].

Se existir conhecimento prévio da sequência **a**, é possível determinar a ordem relativa e a orientação dos fragmentos e, então, encontrar erros. No entanto, na falta deste conhecimento, a ordem dos fragmentos deve ser inferida apenas através das bases das sequências dos fragmentos. Uma maneira natural de se fazer isso é usar a informação da sobreposição entre o extremo da direita de um fragmento e o extremo da esquerda de outro. A idéia do sequenciamento por *shotgun* é solucionar o quebra-cabeça de **a** procurando por sobreposições de seus fragmentos [IW95].

Este trabalho apresenta o contexto biológico e computacional da montagem de fragmentos de sequências de DNA.

No capítulo seguinte está o contexto biológico, onde são explicadas as técnicas de sequenciamento e o processo de montagem de fragmentos. Ao mesmo tempo, são apresentadas as dificuldades encontradas nesta área e as vantagens e desvantagens relacionadas.

Os próximos capítulos tratam do contexto computacional onde são descritos o método tradicional de montagem de fragmentos, os programas principais e mais utilizados pelos biólogos, como Phrap, TIGR Assembler e CAP3, e um método novo de montagem baseado em grafos.

2. Contexto Biológico

As duas técnicas mais importantes de sequenciamento de DNA são o método de terminação enzimática de cadeia [SNC+77] e o método de degradação química [MG77].

Atualmente quase todo o sequenciamento de DNA é feito usando o método de terminação enzimática de cadeia. Inicia-se o processo com DNA alvo purificado e um iniciador oligonucleotídico complementar a um sítio específico neste DNA. Para cada uma das quatro bases (A,C, G, T), é feita uma reação na qual a DNA polimerase sintetiza uma população de fragmentos fita simples de comprimentos variados, cada um dos quais é complementar a um segmento do DNA alvo que se estende do iniciador até a ocorrência desta base. Estes fragmentos são separados de acordo com o seu tamanho por eletroforese em gel, onde os seus tamanhos relativos, juntamente com a identidade da base final de cada fragmento, permitem que a sequência de base do DNA alvo seja inferida. Para cada reação de uma determinada base, um nucleotídeo modificado (terminador) é adicionado em pequenas quantidades fazendo com que a síntese pela DNA polimerase seja interrompida em uma pequena proporção para cada ocorrência desta base, gerando fragmentos de tamanhos distintos.

Inicialmente estes fragmentos eram marcados com átomos radioativos e o gel era exposto em um filme fotográfico que apresentava as bases do fragmento. Cada base era representada por uma mancha escura no filme em uma coluna apropriada. Cada coluna indicava uma base diferente. Veja na Figura 1.

Hoje em dia, métodos não radioativos têm a preferência. Corantes fluorescentes podem ser adicionados tanto ao iniciador quanto ao terminador. Tipicamente um corante distinto é usado para cada uma das quatro reações, de tal forma que elas possam ser combinadas em um único canal do gel. No final do gel, um laser excita os corantes fluorescentes nos fragmentos conforme eles passam, e detetores coletam as intensidades de emissão em quatro diferentes comprimentos de onda. Isto é feito de forma contínua de tal maneira que uma imagem do gel onde cada canal (coluna) possui um padrão de bandas de quatro cores diferentes (cada cor indica uma base) seja construída, com cada banda correspondendo aos fragmentos de um determinado tamanho.

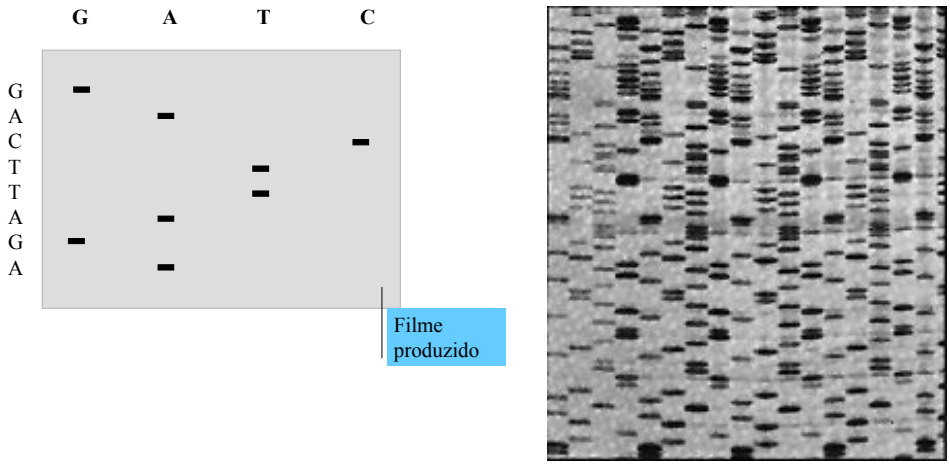


Figura 1. Filme fotográfico com as bases do fragmento.

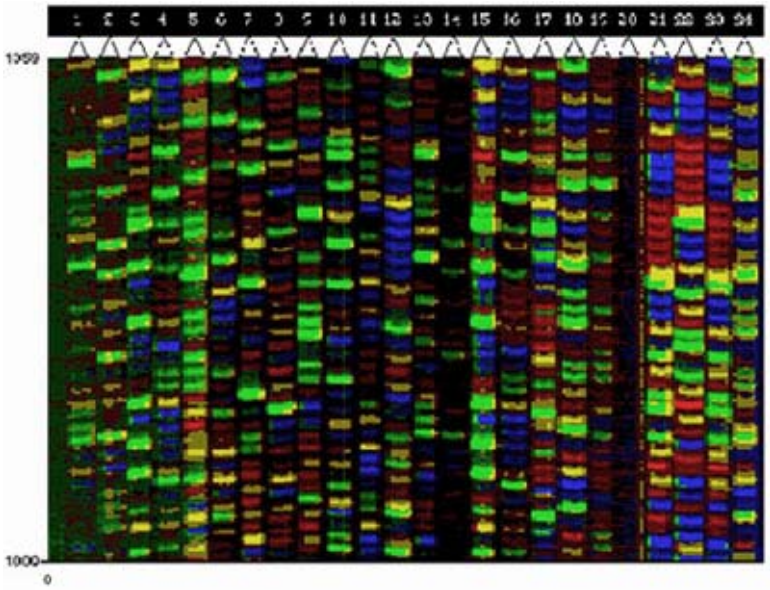


Figura 2. Sequenciamento não radioativo.

Após a realização do gel, é feita uma análise por computador para converter a imagem (padrões de absorção do laser) em uma sequência de bases (leitura), que irá representar a sequência de bases do DNA alvo.

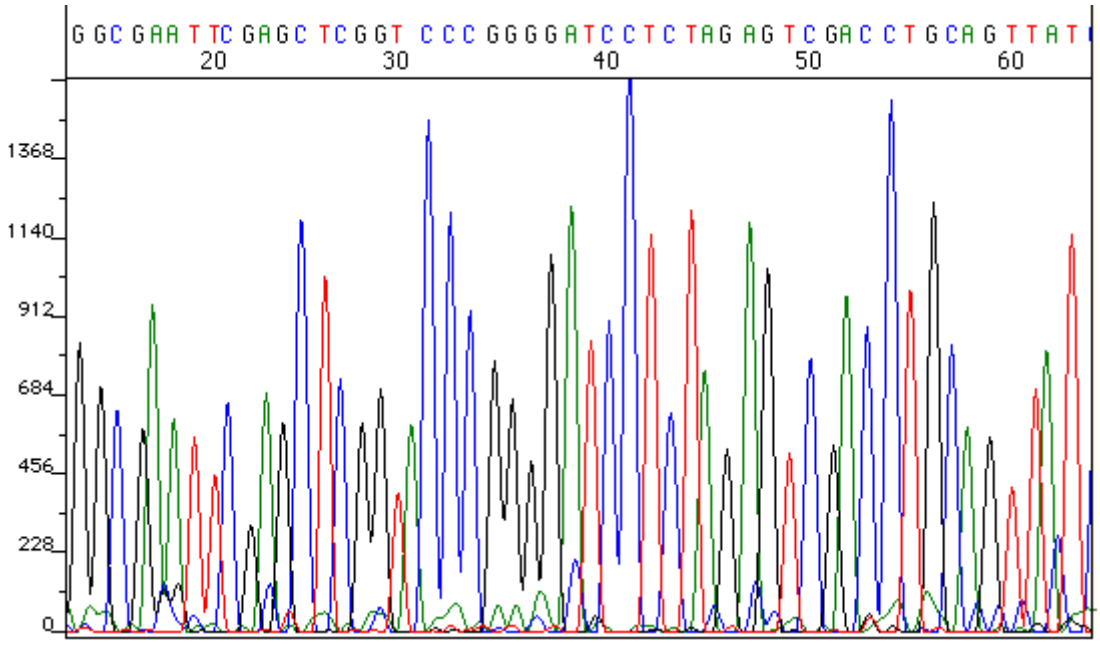


Figura 3. Parte de uma leitura típica de sequenciamento de DNA.

Dois tipos de problemas podem ocorrer na leitura de um determinado fragmento de DNA:

- leitura errada de uma marca, ocorre quando a marca é fraca demais, quando é difícil distinguir entre n e n+1 marcas consecutivas na mesma coluna ou quando existem marcas fantasmas; e
- perda de resolução no final do filme [MS94].

Para diminuir os erros, várias cópias de um fragmento são geradas e sequenciadas. O processo para obtenção de cópias de um fragmento é chamado de *clonagem* ou de *amplificação*.

2.1. Clonagem

A clonagem é um processo de produção de um grupo de clones, todos geneticamente idênticos, a partir de um único ancestral [DOE92].

Este processo depende de um *vetor de clonagem*. Um vetor de clonagem é uma molécula de DNA originada de vírus, bactérias ou células de tecidos na qual um fragmento de DNA de um comprimento apropriado pode ser integrado sem que ela perca sua capacidade de replicação [DOE92].

Um vetor é utilizado em um processo de clonagem porque é possível integrá-lo ao fragmento de DNA que se deseja copiar. Este fragmento de DNA é chamado de *inserto* ou *DNA exógeno*.

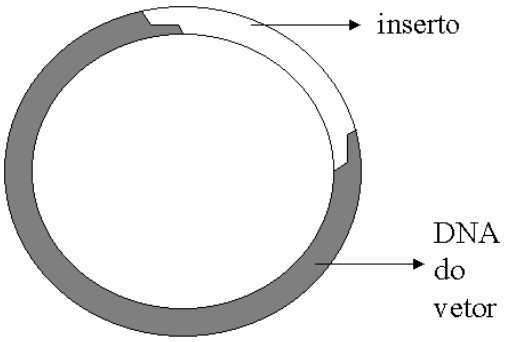


Figura 4. Inserto ou DNA exógeno.

Existem vários tipos de vetores, cada qual é capaz de integrar um fragmento de DNA de comprimento diferente. A tabela seguinte apresenta o nome de alguns vetores e a variação do comprimento dos insertos que são capazes de acolher.

Vetor	Comprimento do inserto
Yeast Artificial Chromossome (YAC)	300Kb a 1Mb
Bacterial Artificial Chromosome (BAC)	25 Kb a 300Kb
P1-derived Artificial Chromosome (PAC)	25 Kb a 300Kb
Smaller constructs (Cosmídeos)	40Kb
Plasmídeos, M13	1 a 10kb

Há dois tipo de clonagem: *in vivo* e *in vitro*.

Clonagem in vivo

A clonagem *in vivo* envolve o uso da tecnologia de DNA recombinante para colocar os fragmentos de DNA dentro de um hospedeiro. O DNA que se deseja clonar é geralmente isolado do seu cromossomo usando enzimas de restrição e é unido ao vetor. Esta união forma a molécula de DNA recombinante que deve ser inserida em células de hospedeiros. Os hospedeiros são então cultivados e consequentemente o inserto é clonado.

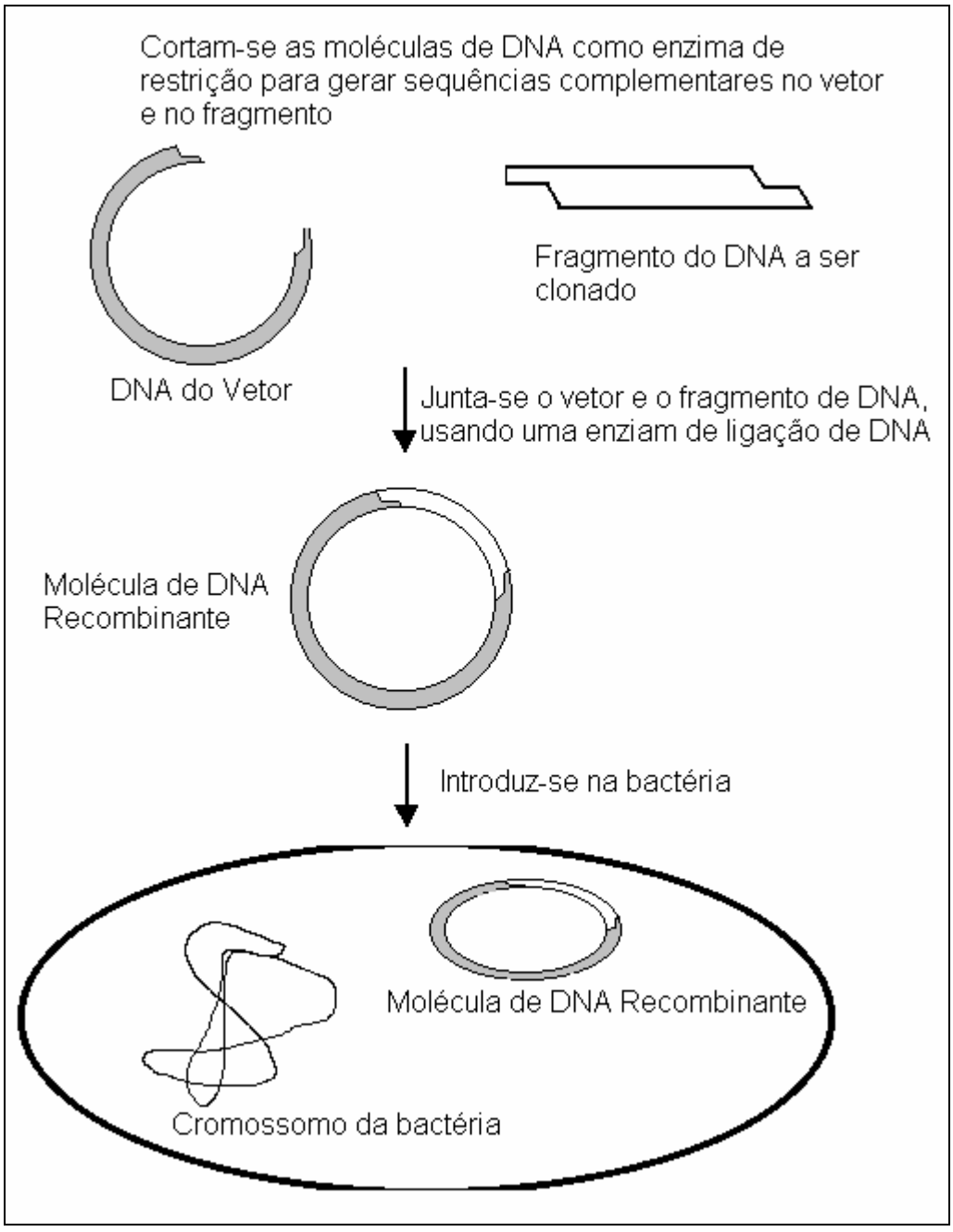


Figura 5. Esquema da clonagem in vivo.

Clonagem in vitro

A técnica de clonagem *in vitro*, também conhecida como PCR (*Polymerase Chain Reaction*) pode amplificar a sequência de DNA desejada de qualquer origem (vírus, bactéria, planta ou humano) centenas de vezes em horas; uma tarefa que necessitaria

de vários dias para ser feita pela tecnologia recombinante, explicada anteriormente [DOE92].

PCR é um processo baseado em uma enzima polimerase especializada que pode sintetizar uma fita complementar de um dado DNA (localizado em um vetor) que esteja presente em uma mistura que tenha dois fragmentos de DNA, conhecidos como primers. Veja na Figura 6 a explicação do processo, que é repetido para cada vez que deseja clonar um fragmento.

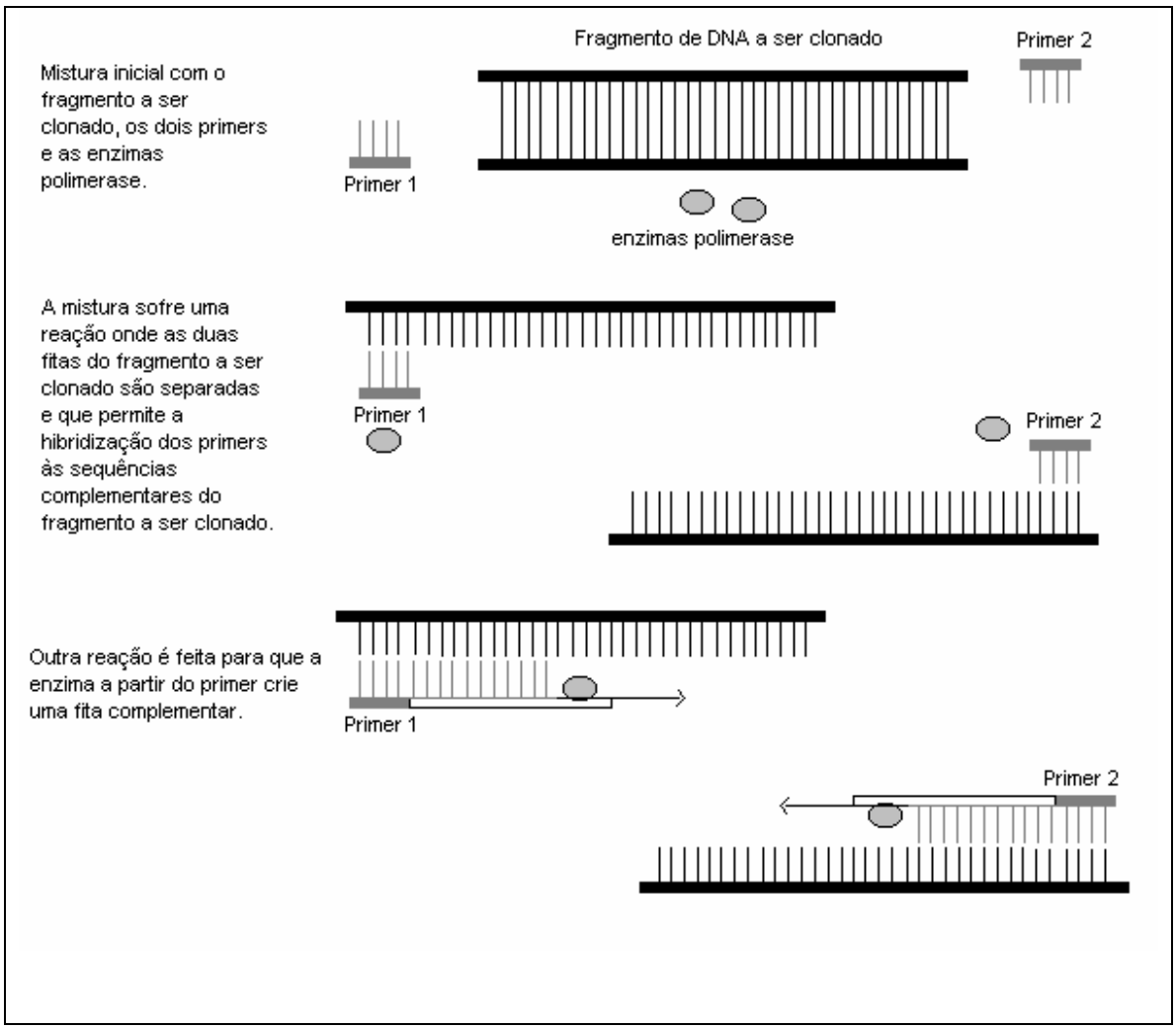


Figura 6. Esquema da clonagem in vitro.

Construção de Clones para Sequenciamento

As moléculas de DNA clonadas precisam ser divididas em tamanhos cada vez menores para estarem adequadas às tecnologias de sequenciamento atuais. Cada divisão do fragmento deve ser clonado utilizando um vetor que corresponda a seu

comprimento. Veja na Figura 7 a divisão do cromossomo em fragmentos cada vez menores.

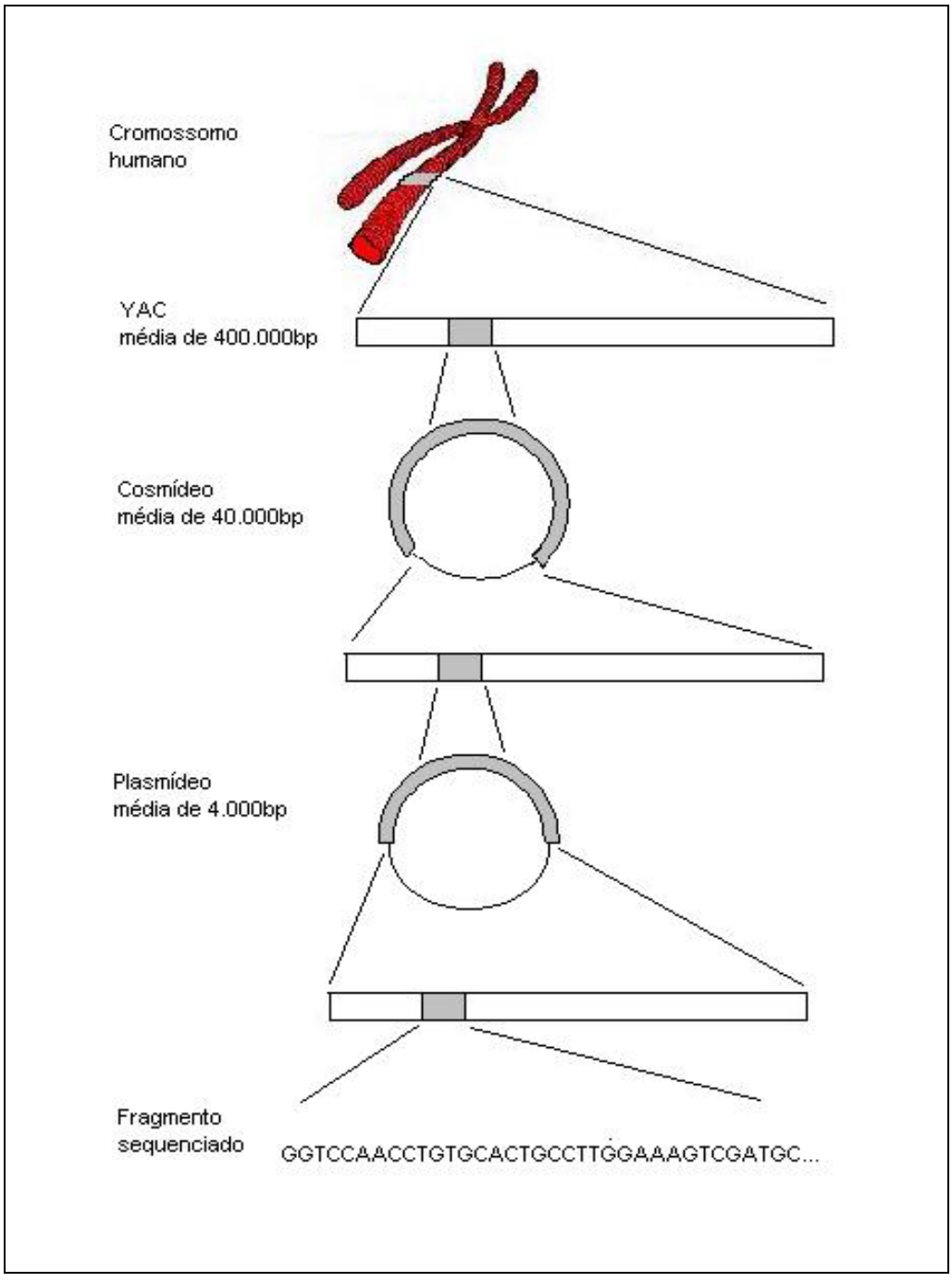


Figura 7. Esquema da construção de clones.

2.2. Sequenciamento Dirigido e Sequenciamento Randômico

Existem duas principais estratégias de sequenciamento de DNA, a dirigida e a randômica. Elas se diferem no processo de clonagem, tamanho dos insertos e na maneira de sequenciamento [SL00]. Seja qual for a estratégia, é importante conseguir

uma cobertura adequada, isto é, não deixar nenhuma parte do DNA original sem representação no conjunto de fragmentos, e assegurar que haja sobreposição suficiente entre os fragmentos para permitir uma montagem segura [MS94]. Veja na

o DNA original e os fragmentos obtidos a partir dele, demonstrando dois casos ruins e um bom para montagem de fragmentos.

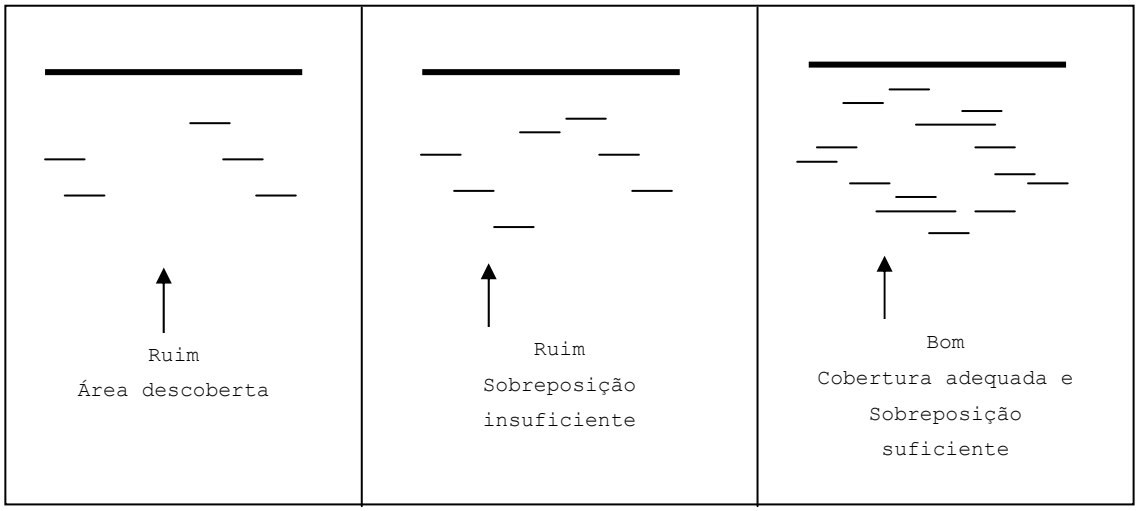


Figura 8. Área de cobertura e sobreposição de fragmentos.

Sequenciamento Dirigido

As técnicas do sequenciamento dirigido executam reações em posições conhecidas da sequência. Esta estratégia possui pouca redundância e um número insignificante de reações comparado ao randômico [SL00].

O método mais comum de sequenciamento dirigido é o *primer walking*. Este método utiliza a técnica PCR apresentada anteriormente. Ela envolve a hibridização de um primer em uma sequência conhecida e a sintetização de uma pequena fita complementar a partir desta sequência conhecida. Ao sintetizar e sequenciar um fragmento, usa-se o seu final como o primer para o sequenciamento do próximo fragmento da sequência. Desta maneira "anda-se" por toda a sequência e o cromossomo é sistematicamente sequenciado do início ao fim. Como os primers são sintetizados quimicamente, uma desvantagem deste método é o grande número de diferentes primers que se precisa para se andar por longas distâncias [DOE92].

O principal problema do sequenciamento dirigido é ser sequencial e consequentemente lento. Uma melhora pode ser adotada se vários pontos diferentes

forem conhecidos no DNA o que permitiria que o sequenciamento dirigido pudesse ser iniciado nestes pontos paralelamente [Gus97].

O segundo problema crítico deste tipo de sequenciamento é que se, por qualquer razão, o sequenciamento dirigido for impedido de prosseguir em algum ponto, a sequência após este ponto não poderá ser obtida, a menos que um novo ponto de partida seja descoberto. Um exemplo que frequentemente impede ou confunde o prosseguimento do sequenciamento dirigido é o surgimento de regiões com cadeias repetitivas. Se o primer selecionado for uma subcadeia repetitiva, ele pode hibridizar em uma região não desejada da sequência e neste caso o PCR sintetizará a parte errada da sequência [Gus97].

Sequenciamento Randômico

Neste método, também conhecido como *shotgun* [MW96, Mye99, WM97, Gre97], o DNA a ser sequenciado é submetido a algum processo que causa a quebra das moléculas em pontos aleatórios [MS94]. Isto pode ser feito através de diversas técnicas que utilizam por exemplo enzimas de restrição ou ondas ultrassônicas [SL00]. Estes fragmentos obtidos, geralmente de 1 a 2kb, podem ser guardados em uma biblioteca de insertos.

Como qualquer pedaço de DNA para ser sequenciado tem que estar presente em quantidade suficiente e, com este método, cada fragmento deve ser único na mistura obtida após a quebra, é necessário produzir muitas cópias de cada fragmento selecionado (amplificar ou clonar o fragmento).

Depois da clonagem e sequenciamento dos clones, obtem-se vários clones espalhados randomicamente pelo fragmento original. Estes clones são então combinados e montados formando contigs [SL00]. Veja o esquema na Figura *7*.

Geralmente o sequenciamento randômico é feito até que 75% a 95% do fragmento original seja coberto, e é seguido do sequenciamento dirigido para cobrir os buracos [SL00].

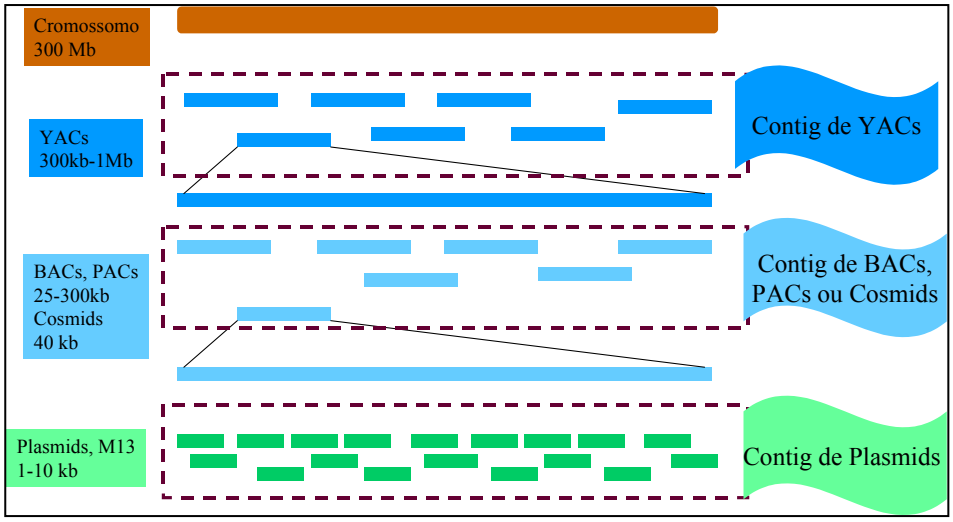


Figura 9. YACs, BACs, PACs, Cosmids e Plasmids.

2.3. Dificuldades

Alguns problemas podem ocorrer durante estes processos.

Primeiro, pode ocorrer falta de cobertura ou falta de sobreposição adequada que impedirão a montagem da sequência de DNA original [MS94].

Segundo, mutações espontâneas no processo de clonagem dos fragmentos podem fazer com que o fragmento sequenciado seja ligeiramente diferente do original. Além disso, há a possibilidade de contaminação dos fragmentos por trechos de DNA do organismo hospedeiro. Em geral, este problema pode ser controlado pois conhece-se de antemão a sequência completa do hospedeiro, de modo que uma busca de pedaços idênticos ao DNA do hospedeiro pode ser feita em todos os fragmentos antes do procedimento de montagem propriamente dito [MS94].

Há ainda a questão da orientação das sequências de acordo com suas fitas. Cada fita de DNA tem duas extremidades livres, chamadas de 5’ e 3’. A extremidade 3’ de uma fita corresponde a extremidade 5’ da outra. Por isso costuma-se dizer que as fitas são antiparalelas. Um A em uma fita corresponde a um T na outra, e um C corresponde a um G. Dizemos que A e T são bases complementares, assim como C e G. Com isto vemos que a sequência de nucleotídeos numa das fitas determina completamente a molécula de DNA. A sequência da outra poderá ser determinada invertendo a ordem das bases e complementando cada uma delas, ou seja, substituindo A por T, T por A, C por G e G por C. O complemento reverso de uma sequência de bases é a sequência

obtida realizando-se estas operações na sequência. Veja por exemplo a sequência CTATCG. Invertendo sua ordem tem-se GCTATC, que possui como complemento CGATAG. Logo o complemento reverso de CTATCG é CGATAG.

```
5′ - - - C T A T C G - - - 3′
3′ - - - G A T A G C - - - 5′
```

Apesar das moléculas de DNA terem duas fitas complementares e antiparalelas, o sequenciamento é feito com o DNA de fita simples. Assim os fragmentos obtidos provêem das duas fitas da molécula original e, via de regra, não há como saber de qual delas. Cada fragmento pode vir de uma das duas fitas independentemente. Por outro lado, é sempre possível distinguir entre as extremidades 3' e 5' de um fragmento sequenciado e, portanto, a orientação correta destes fragmentos é conhecida. A convenção universalmente adotada é escrever sequências começando da extremidade 5'. A consequência destes fatos para a montagem é que, para saber se dois fragmentos se sobrepõem, é necessário levar em conta as duas possíveis situações [MS94]:

- Ambos provêem da mesma fita; neste caso, se eles realmente se sobrepõem, uma semelhança será notada entre as sequências nas orientações dadas;
- Os fragmentos provêem de fitas distintas; neste caso, se eles realmente se sobrepõem, uma semelhança será notada entre uma das sequências e o complemento reverso da outra.

Portanto há a necessidade de duas comparações, uma envolvendo as sequências nas orientações dadas e a outra com uma delas sendo invertida e complementada [MS94].

Por último, após a montagem dos fragmentos podem existir buracos na sequência original que não foram descobertos. Isto pode ocorrer quando por exemplo, os *reads* (fragmentos sequenciados) possuem baixíssima qualidade. Como resultado, teríamos dois ou mais contigs. Na fase final, subclones adicionais devem ser obtidos e sequenciados para cobrir tal parte. O objetivo é permitir que todos os dados sejam unidos em um único contig com uma taxa de erro de 10^{-4} ou menos [KR98].

3. Método Tradicional de Montagem de Fragmentos

Existem diferentes implementações de algoritmos de montagem de fragmentos mas quase todos seguem a mesma idéia descrita em [PST+83, PSU84]. Esta idéia contém três passos: *detecção de sobreposição*, *layout dos fragmentos* e *decisão da sequência*

consenso [Gus97][Mye94] e é conhecida como o paradigma "overlap-layout-consensus" [PTW01]. O primeiro e o último passo correspondem a problemas de cadeias bem definidos [Gus97].

3.1. Passo 1 - Detecção de Sopreposição

O primeiro passo [Gus97] consiste em descobrir, para cada par de cadeias (fragmentos), quanto o sufixo da primeira casa-se com o prefixo da segunda. Se as sequências não possuíssem erros, este problema se resumiria a encontrar para cada par de cadeias S1 e S2, o maior sufixo de S1 que casasse com o prefixo de S2. Mas, na realidade, as sequências possuem erros e por isso é necessário o uso de casamentos aproximados. Com isso, a maioria das implementações definem o casamento entre sufixo-prefixo quando for resolvido o seguinte problema para cada par de cadeias S1 e S2:

- encontre o sufixo de S1 e o prefixo de S2 cuja *similaridade* é a maior entre todos os pares de sufixos e prefixos de S1 e S2.

Neste caso, a similaridade é baseada em um critério de pontuação onde o casamento exato de caracteres representa um valor positivo, e desigualdades e buracos representam valores negativos. Este problema pode ser resolvido por programação dinâmica e possui complexidade quadrática.

Para aumentar a velocidade deste passo, existem heurísticas com o objetivo de encontrar os melhores casamentos entre sufixos e prefixos, ou seja, encontrar os pares cujos casamentos possuam similaridade alta e consequentemente sejam candidatos de pares que se sobrepoem (ou seja, que são vizinhos). Estes candidatos são usados no próximo passo do algoritmo de montagem de fragmentos.

Além disso, é possível aumentar a velocidade deste passo se forem identificados os pares de cadeias cuja melhor similaridade sufixo-prefixo não seja claramente suficiente para serem considerados candidatos atrativos. Nestes casos, a programação dinâmica não precisaria ser utilizada.

Para identificar quais pares são ou não atrativos existem vários métodos. Por exemplo, para que duas cadeias fossem consideradas atrativas, elas deveriam ter uma subcadeia comum significativamente longa, sendo que o comprimento é considerado significativo de acordo com algum argumento probabilístico. O ponto principal aqui é descobrir e excluir muitos pares de cadeias que não aparentam serem vizinhos na sequência original.

Uma heurística parecida com a do BLAST poderia encontrar pares de cadeias vizinhas ao encontrar regiões de alta similaridade entre pares de cadeias. Novamente, a idéia seria de que duas sequências seriam consideradas vizinhas, quando elas possuíssem regiões com comprimento bom de similaridade local alta.

3.2. Passo 2 - Layout dos Fragmentos

Existem diversos tipos de obtenção de layouts, alguns obtidos através de algoritmos bastante complicados, mas a maioria segue uma variação do método guloso [Gus97]. Primeiro, o par de cadeia com sobreposição sufixo-prefixo com maior pontuação é escolhido e intercalado. Depois, o próximo par com pontuação maior é escolhido e intercalado, resultando em um contig com três fragmentos ou em dois contigs com quatro fragmentos (Figura 10).

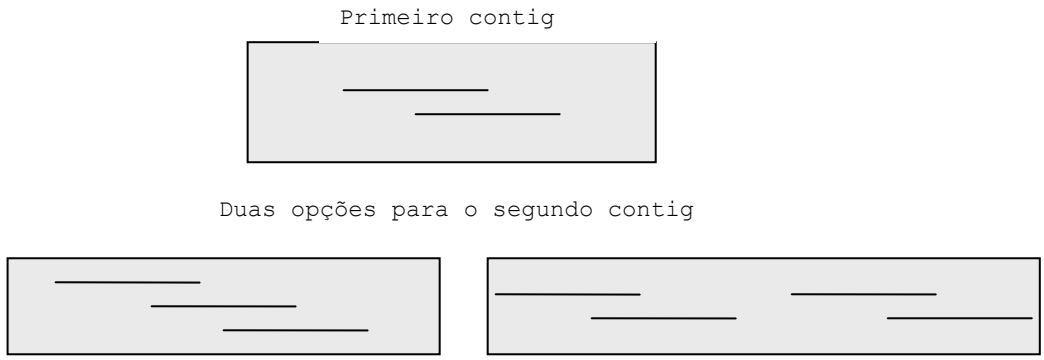


Figura 10. Layout de fragmentos.

Como a sobreposição de sufixo e prefixo é determinada por um critério de similaridade, são permitidos buracos nos casamentos de fragmentos. Consequentemente, como vários fragmentos são adicionados em um contig, buracos adicionais podem ser inseridos na cadeia que está sendo adicionada ao contig para que ela fique consistente com os buracos das cadeias intercaladas previamente. Este é exatamente o mesmo conceito de alinhamento múltiplo, e de fato, o que está sendo construído aqui é uma estrutura de alinhamento múltiplo.

Note que este método pode não calcular a melhor sobreposição sufixos-prefixos entre os fragmentos de um determinado contig e entre os contigs. Provavelmente existirá um melhor alinhamento múltiplo, mas geralmente seria computacionalmente caro e por isso na prática não é calculado. Em vez disso, todas as decisões de layout dos fragmentos são baseadas em pontuações dos pares sufixos-prefixos calculados no

passo 1, e as inconsistências no layout são resolvidas no passo 3. Além disso como o passo 2 não envolve programação dinâmica, ele é calculado rapidamente.

3.3. Passo 3 - Decisão da Sequência Consenso

Cada contig criado no passo 2 é utilizado aqui para criar um único fragmento (sequência consenso), ou seja, para definir as bases deste fragmento [Gus97].

Existem diferentes formas de calcular a sequência consenso, mas todas seguem um método similar. Cada fragmento que compõem um contig possui um caracter em uma posição particular. Se os caracteres de uma determinada posição em todos os fragmentos forem iguais, então é definido que aquele caracter é a base daquela posição no fragmento. Mas, se isto não ocorrer, deve ser escolhido um caracter ou deve ser indicado que há uma divergência muito grande para sua determinação.

Existem diversas formas de determinar a sequência consenso; algumas das quais serão aqui ilustradas.

A maneira mais simples seria fazer um relatório com a frequência de cada caracter em cada coluna e deixar o usuário decidir como usar tal informação. Alternativamente, uma sequência consenso pode ser calculada obtendo o caracter que mais se repete em cada coluna. É possível também determinar as *janelas* no contig que possuem muitas divergências e utilizar um método de alinhamento múltiplo nelas para calcular os seus caracteres. Fazendo isso, vários caracteres podem ser mudados ou deslocados. Em [Kec91, KM95, Kec93] estão descritos métodos mais rigorosos de ajustar a disposição e determinar as bases da sequência consenso.

Além disso, um método intermediário seria reintercalar os fragmentos dispostos nos contigs obtidos no passo 2 da mesma forma que foi feito no passo 2, só que agora cada fragmento seria intercalado apenas no contig que já faz parte usando um alinhamento do novo fragmento com um perfil dos fragmentos já dispostos no contig.

Este tipo de intercalação não é tão intenso computacionalmente quanto o alinhamento múltiplo completo. Ele requer apenas n-1 alinhamentos, sendo n igual ao número de fragmentos. No entanto, ele deve gerar um alinhamento melhor que o obtido originalmente, já que ele leva em consideração os fragmentos apenas de um determinado contig. Depois que todas os fragmentos são reintercalados, uma sequência consenso é criada a partir do novo layout.

Por exemplo, veja na figura seguinte o primeiro contig que foi formado no passo 2. Neste alinhamento de pares de cadeias foi inserido um espaço.

Continuando o passo 2, o próximo melhor casamento sufixo-prefixo casava o sufixo ATCT com o prefixo ATCG, como apresentado a seguir. O alinhamento do par não contém buracos, mas um buraco é inserido nesta terceira cadeia quando ela é adicionada ao contig. O novo espaço é herdado do alinhamento entre as primeiras duas cadeias.

No passo 3, é feito novamente o alinhamento entre as duas primeiras cadeias obtendo um alinhamento entre um buraco e T e os outros alinhamentos entre caracteres idênticos.

Em seguida, a terceira cadeia é alinhada às duas já alinhadas anteriormente (em vez de somente com a primeira cadeia); neste caso nenhum espaço é inserido. Este alinhamento torna mais convincente que o caracter T deve ser a base da coluna que possui um buraco.

3.4. Principais Programas Existentes

Os algoritmos de montagem de fragmentos começaram a surgir com [Sta80], algoritmo este que foi estendido e elaborado por muitos outros. A seguir está uma tabela de referências dos vários programas existentes.

Programas de Montagem de Fragmentos	
Referência	Nome
[PSU84]	SEQAID
[Kec91]	-
[Hua92]	CAP
[Mei92], [Mei93]	-
[PFB93]	-
[Gre94]	Phragment Assembly Program (Phrap), que faz parte de um pacote que possui mais dois outros programas, o Phred [EHW+98][EG98] e o Consed [GAG98];
[KM95]	-
[KS99]	-
[IW95]	-
[Mye95]	-
[BSS95]	Genome Assembly Program – GAP
[SWA+95]	TIGR Assembler
[WDH+98]	Genome Automated Sequence Preprocessor - GASP
[HM99]	CAP3
[AM99][MSD+00] [ACH+00][HRM01]	Celera assembler
[KY01]	-
[PTW01a] [PTW01b]	-
[FSW96]	-

4. Método Novo de Montagem de Fragmentos

Nos últimos vinte anos a montagem de fragmentos de DNA seguiu o paradigma "overlap-layout-consensus" descrito anteriormente. Apesar deste método clássico ter

resultado em excelentes ferramentas de montagem de fragmentos (Phrap, CAP3, TIGR e Celera assemblers) ele possui alguns pontos fracos. Por exemplo, eles utilizam comparação entre pares de fragmentos, em vez de comparação múltipla, que não é adequado para decidir se dois reads similares pertencem à mesma região (ou seja, suas diferenças são ocasionadas por erros de sequenciamento) ou se são repetições encontradas em regiões diferentes do genoma [PTW01b].

Em [PTW01b] há uma proposta nova para montagem de fragmentos baseada na idéia de sequenciamento por hibridização (SBH do inglês *Sequence By Hybridization*). Conceitualmente SBH é similar a montagem de fragmentos. De fato, os primeiros métodos [DLB+89, LFK+88] para solucioná-lo seguiram o paradigma "overlap-layout-consensus". No entanto, mesmo no caso mais simples onde não existiam erros nos dados do SBH, o layout correspondente levava ao problema NP-completo do caminho Hamiltoniano.

Em [Pev89] foi proposto um método diferente que reduzia SBH a um problema fácil de se resolver, o problema do caminho Euleriano utilizando o grafo *de-Bruijn*, abandonando o paradigma "overlap-layout-consensus".

Em [IW95] foi mostrado que o método do caminho Euleriano poderia ser aplicado à montagem de fragmentos. Infelizmente esta idéia não foi bem sucedida porque os erros de sequenciamento transformavam um grafo de-Bruijn simples em um emaranhado de arcos errados.

Em [PTW01] há uma redução do problema de montagem de fragmentos para uma variação do problema clássico do caminho Euleriano, chamada *Supercaminho Euleriano*. Esta redução abriu novas possibilidades para resolução de repetições, gerou ótimas soluções em projetos de montagem de fragmentos de larga escala e provou que pode melhorar a precisão de programas como Phrap, CAP3 e TIGR.

A idéia principal deste artigo está baseada na correção de erros que usa implicitamente alinhamento múltiplo em subcadeias pequenas que modificam os reads originais. Esta correção consegue distinguir se dois reads similares fazem parte da mesma região ou se são repetições localizadas em diferentes partes do genoma. Além disso, como ela faz com que os reads fiquem quase sem erros, consegue transformar o grafo grande original em um grafo com poucas arcos errados, criando uma nova instância do problema de montagem de fragmentos com uma grande redução dos erros.

Tanto [IW95] quanto [Mye95], que utilizam o paradigma convencional, tentaram tratar os erros e repetições via reduções de grafos. Mas ambos não exploraram alinhamento múltiplo dos reads para consertar os erros de sequenciamento em um estágio de pré-processamento. É claro que o alinhamento múltiplo é muito custoso e o alinhamento entre pares foi a única solução realista no estágio de sobreposição dos algoritmos de montagem de fragmentos convencionais. No entanto, o alinhamento múltiplo torna-se viável quando se trata de casamentos de tuplas perfeitas ou quase perfeitas, que é o caso do método SBH para montagem de fragmentos.

4.1. Idéias Novas

O método clássico de montagem de fragmentos baseia-se na noção de um grafo de sobreposições. A sequência de DNA da Figura 11*a* consiste de quatro segmentos únicos A, B, C e D e uma repetição R. Cada read corresponde a um vértice do grafo de sobreposições e dois vértices são conectados por um arco se os reads correspondentes se sobrepuserem (Figura 11*b*). O problema de montagem de fragmentos é um exemplo de como se encontrar um caminho em um grafo de sobreposições visitando cada vértice uma única vez, um problema do caminho Hamiltoniano. O problema do caminho Hamiltoniano é NP-Completo e não se conhece algoritmos eficientes para se resolver este problema. Isto mostra porquê montagem de fragmentos de genomas altamente repetitivos é um problema difícil de se resolver [PTW01a].

Em [PTW01a, PTW01b] sugere-se um método de montagem de fragmentos baseado no grafo *de-Bruijn,* chamado **Euler**. De uma maneira informal, pode-se visualizar a construção deste grafo representando uma sequência de DNA como um "linha" com regiões repetidas cobertas por uma "cola" que "liga" tais regiões (Figura 11*c*). O grafo *de-Bruijn* (Figura 11*d*) consiste em 5 arcos. Neste método, cada repetição corresponde a um arco ao invés de uma coleção de vértices no grafo de sobreposições (Figura 11*b*).

É possível notar que o grafo de-Bruijn (Figura 11d) representa as repetições de uma forma muito mais simples que o grafo de sobreposições (Figura 11b). Cabe ainda perceber o mais importante: agora a montagem de fragmentos trata-se de um problema de encontrar um caminho que visite todos os arcos do grafo exatamente uma vez, um problema do caminho Euleriano. No exemplo existem dois caminhos Eulerianos: um corresponde a reconstrução da sequência ARBRCRD, enquanto o

outro corresponde a reconstrução da sequência ARCRBRD. Ao contrário do problema de caminho Hamiltoniano, o caminho Euleriano é fácil de se resolver mesmo em grafos com milhões de vértices porque existem algoritmos com tempo linear para solucioná-lo [Flei90].

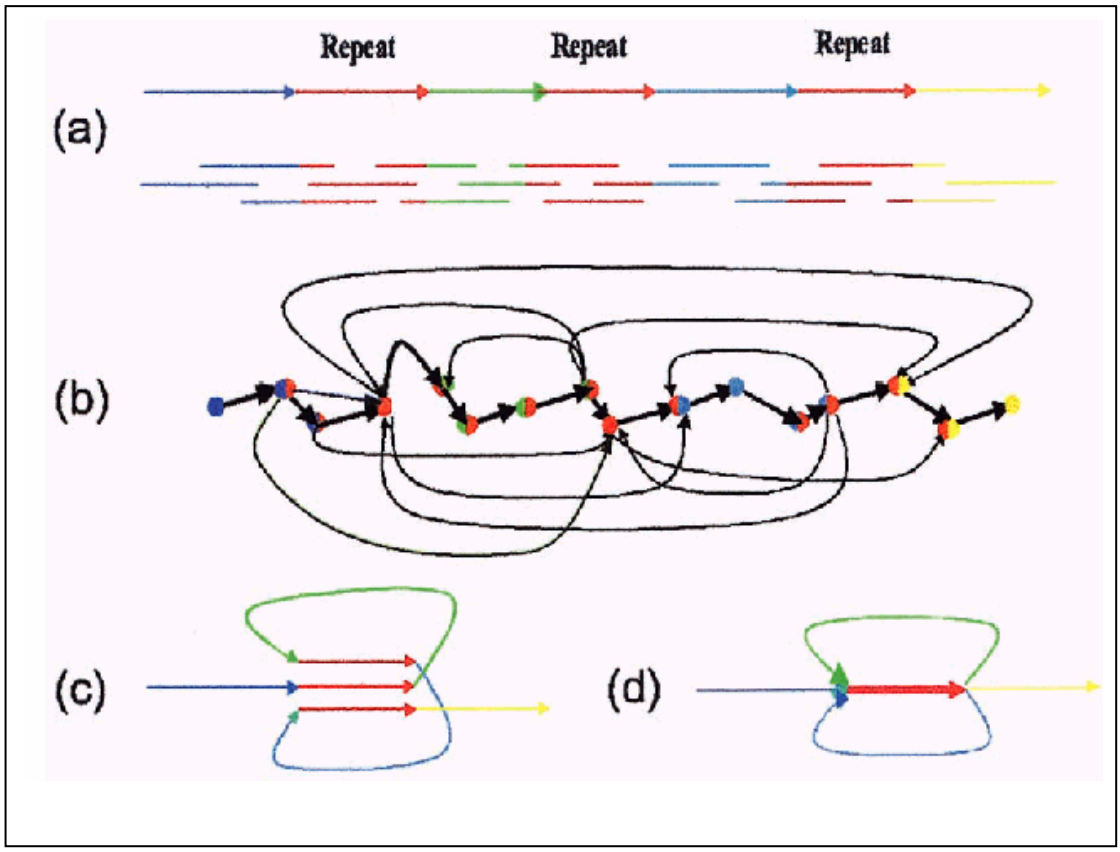


Figura 11. (a) Sequência de DNA com três repetições R; (b) o grafo do layout; (c) construção do grafo de-Bruijn colando as repetições; (d) o grafo de-Bruijn.

A construção do grafo de-Bruijn é feita a partir da coleção dos reads sequenciados. Mas esta tarefa não seria fácil pois a idéia de "colar" requereria conhecimento da sequência de DNA completa, que só ficaria disponível no último passo da montagem. A seguir será mostrado como construir o grafo de-Bruijn sem conhecimento prévio da sequência de DNA completa.

Além destas idéias novas, o Euler executa em primeiro lugar um procedimento similar ao terceiro passo do método convencional. Este procedimento corrige os erros dos reads e transforma o grafo de-Bruijn em um grafo com poucos arcos errados.

4.2. Correção de Erros

Os algoritmos existentes adiam o passo de consenso (correção de erros nos reads) para o final da montagem. O Euler inverte esta idéia, tratando disso em primeiro lugar.

Seja s um read com erros derivado de um genoma G. Se a sequência de G fosse conhecida, a correção de s poderia ser feita alinhando-a a G. Na realidade isto não acontece porque G é desconhecido. Neste caso, assume-se que o conjunto G_l de todas as *l-tuplas* (pequenas cadeias contínuas com comprimento fixo l) presentes em G é conhecido. É claro que G_l também é desconhecido, mas pode ser aproximado sem o conhecimento de G. A idéia da correção de erros usa esta aproximação para corrigir os erros nos reads. Isto é feito através do alinhamento múltiplo de subcadeias pequenas que modificam os reads originais e criam uma nova instância do problema da montagem de fragmentos com um número reduzido de erros.

Dada uma coleção de reads (cadeias) $S=\{s_1, s_2, ..., s_n\}$ de um projeto de sequenciamento e um inteiro *l*, o *spectrum* de S é o conjunto S_l de todas as l-tuplas dos reads $s_1, s_2, ..., s_n$ e $s_{1r}, s_{2r}, ..., s_{nr}$, onde s_r indica o complemento reverso de s.

Seja Δ o limite superior para o número de erros em cada read.

Problema de Correção de Erro: Dados S, Δ e *l*, realize no máximo Δ correções em cada read de S de tal forma que $|S_l|$ seja minimizado.

Um erro em um read afeta no máximo *l* das l-tuplas em s e *l* das l-tuplas em s_r (veja exemplo a seguir). e geralmente cria *2l* l-tuplas erradas que apontam para o mesmo erro de sequenciamento (ou *2d* para posições dentro de uma distância $d < l$ do extremo dos reads). Consequentemente uma estratégia gulosa para a correção de erros seria corrigir um erro em um read que reduzisse o tamanho de S_l em *2l* (ou *2d* para posições próximas aos extremos dos reads).

Exemplo: Suponha o read s CTATCG cujo complemento reverso s_r é CGATAG, que *l* seja 3 e que a base A de s esteja errada.

	s C T A T C G	s_r C G A T A G
l-tuplas	C T A	C G A
	T A T	G A T
	A T C	A T A
	T C G	T A G

Veja que 6 3-tuplas ficaram erradas.

Esta estratégia gulosa eliminaria 86,5% dos erros nos reads. Uma proposta melhor, que elimina 97,7% dos erros, será apresentada a seguir.

Duas l-tuplas são chamadas de *vizinhas* se elas se diferenciarem por uma mutação.

Exemplo: As seguintes 3-tuplas se diferenciam apenas em uma base e portanto são vizinhas.

C T A

C T G

A *multiplicidade m(a)* de uma l-tupla *a* é definida como o número de reads em S que contém tal l-tupla. Uma l-tupla é chamada de *órfã* se

(i) ela tem multiplicidade pequena, ou seja, $m(a) \leq M$, sendo M um valor determinado,

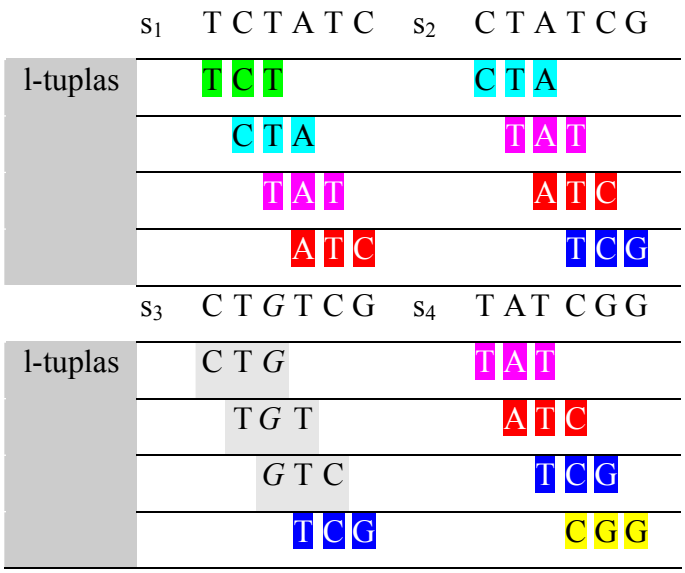
(ii) ela tem um único vizinho *b*; e

(iii) $m(b) \gg m(a)$.

A posição em que um vizinho e um órfão se diferenciam é chamada de *posição órfã*.

Uma observação importante é que cada l-tupla que contenha um erro de sequenciamento geralmente não aparece em outro read e é diferente dos reads corretos por apenas uma mutação (para um *l* escolhido apropriadamente). Consequentemente, uma mutação em um read geralmente cria 2l órfãos. Esta observação leva a um método que corrige erros em posições órfãs dentro de reads, se o número total de correções em um dado read para torná-lo sem órfãos seja no máximo Δ. O método de eliminação de órfãos para o problema de Correção de Erros começa nas correções de erros de posições órfãs que reduzem o tamanho de S_l em 2l (ou 2d para posições próximas aos extremos dos reads). Depois de corrigir todos estes erros com "condição 2l", a estratégia passa a tratar da condição mais fraca $2l - \delta$.

Exemplo: Imagine que a sequência completa de um genoma (depois de montar os fragmentos corretamente) possua o seguinte fragmento TCTATCGG. Suponha que os reads obtidos para este fragmento sejam os elementos de S = {TCTATC, CTATCG, CT***G***TCG, TATCGG }. As 3-tuplas obtidas estão listadas na tabela seguinte. Note que a base G em itálico do read CT*G*TCG representa um erro de sequenciamento; o read correto seria CTATCG.



A multiplicidade das 3-tuplas estão listadas na tabela seguinte.

tupla-3	m(tupla-3)
TCT	1
CTA	2
TAT	3
ATC	3
TCG	3
CTG	1
TGT	1
GTC	1
CGG	1

Para o exemplo serão descartados as tuplas TCT e CGG porque se localizam no extremo do pedaço do genoma que está sendo tratado e será suposto que existiam outros reads que possuíam estas tuplas.

Considerando isso, note que as tuplas CTG, TGT e GTC, que possuem um erro de sequenciamento, não aparecem em outro read e são diferentes dos reads corretos por apenas uma mutação. Uma mutação no read CT***G***TCG criou 2l = 6 órfãos, são eles: CTG, TGT e GTC e seus respectivos complementos reversos. Veja que cada órfão tem multiplicidade pequena e que tem apenas um vizinho com multiplicidade maior que ele. CTG (multiplicidade 1) tem o vizinho CTA (multiplicidade 2), TGT (multiplicidade 1) tem o vizinho TAT (multiplicidade 3), GTC (multiplicidade 1) tem o vizinho ATC (multiplicidade 3).

Ao detectar estas tuplas, trocando G por A, o read CT**A**TCG substituíria o CTCTCG. Desta maneira S ficaria igual a {TCTATC, CTATCG, TATCGG }. As tuplas CTG, TGT e GTC e seus respectivos complementos reversos não pertenceriam mais a S_l, diminuindo o tamanho de S_l de 2l = 2 x 3 = 6.

4.3. Grafo de-Bruijn

Sejam S o conjunto de reads $\{s_1, \ldots, s_n\}$, S_l o conjunto de todas as l-tuplas de S e S_{l-1} o conjunto de todas as (l-1)-tuplas de S. O grafo *de-Bruijn* é representado por $G(S_l)$. Seus vértices são os elementos de S_{l-1} e seus arcos são as l-tuplas de S_l. Uma tupla-(l-1) $v \in S_{l-1}$ é unida por um arco com a tupla-(l-1) $w \in S_{l-1}$, se S_l contiver uma l-tupla que tiver os seus l-1 primeiros nucleotídeos coincidentes com *v* e os seus l-1 últimos nucleotídeos coincidentes com *w*.

Um vértice *v* é chamado de *origem* se grau de chegada(*v*) = 0, de *fim* se grau de saída(*v*) = 0 e de *ramificação* se grau de chegada (*v*).grau de saída(*v*) > 1.

Um caminho $v_1 \ldots v_n$ em um grafo de-Bruijn é chamado de repetição se grau de chegada(v_1)>1, grau de saída(v_n)>1 e grau de chegada(v_1)=grau de saída(v_i) = 1 para $1 \leq i \leq n-1$. Veja a Figura 12.

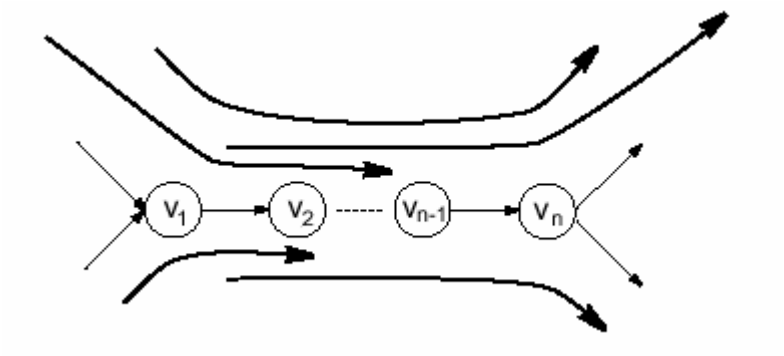


Figura 12. Uma repetição $v_1...v_n$ e um sistema de caminhos que se sobrepõem com esta repetição. O caminho mais acima contém a repetição e define o par correto entre a entrada e a saída. Se este caminho não existisse, a repetição $v_1...v_n$ seria um emaranhado.

Os arcos que entram no vértice v_1 são chamadas *entradas* da repetição, enquanto os que saem do vértice v_n são chamadas *saídas* da repetição. Um caminho Euleriano visita uma repetição poucas vezes e a cada visita define um par de entrada e saída. As repetições criam problemas na montagem de fragmentos porque, apesar de existirem poucas entradas e saídas, não fica claro qual saída é visitada depois de qual entrada no caminho Euleriano. Um read cobre uma repetição quando ele contém uma entrada e uma saída da repetição. Cada read que cobre uma repetição revela alguma informação sobre os pares corretos entre entradas e saídas. Uma repetição é chamada de *emaranhado* quando não existe nenhum read que a contenha. Os emaranhados causam problemas na montagem de fragmentos porque os pares entrada-saída não são resolvidos pela análise dos reads. Para tratar disso foi formulado a generalização do problema de caminho Euleriano: problema de Supercaminho Euleriano.

4.4. Problema de Supercaminho Euleriano

O problema do Supercaminho Euleriano é definido como "Dado um grafo de-Bruijn e uma coleção de caminhos neste grafo, encontre um caminho Euleriano neste grafo que contenha todos estes caminhos como subcaminhos".

Para resolver este problema, transforma-se o grafo G e o sistema de caminhos P deste grafo em um novo grafo G_1 com um novo sistema de caminhos P_1. Tal transformação é chamada de equivalente se existir uma correspondência um para um entre os Supercaminhos Eulerianos em (G, P) e (G_1, P_1). O objetivo é fazer uma série de transformações equivalentes

$$(G, P) \rightarrow (G_1, P_1) \rightarrow \ldots. \rightarrow (G_k, P_k)$$

que leve a um sistema de caminhos P_k, onde cada caminho é representado por um único arco. Como todas as transformações entre (G, P) até (G_k, P_k) são equivalentes, todas as soluções do problema de caminho Euleriano em (G_k, P_k) provêem uma solução para o problema do Supercaminho Euleriano em (G, P).

A seguir é descrita uma transformação simples que soluciona o problema do Supercaminho Euleriano em um grafo G que não tem arcos múltiplos. Sejam dois arcos consecutivos $x = (v_{in}, v_{mid})$ e $y = (v_{mid}, v_{out})$ do grafo G, e seja $P_{x,y}$ a coleção de todos os caminhos de P que incluem estes arcos como subcaminhos. Informalmente, o isolamento-*x,y* desvia os arcos x e y para um novo arco z e direciona todos os caminhos em $P_{x,y}$ para z, simplificando o grafo. No entanto, esta transformação afeta outros caminhos e precisa ser definida cuidadosamente. Define-se $P_{\rightarrow x}$ como a coleção de caminhos de P que terminam em x e $P_{y\rightarrow}$ como a coleção de caminhos em P que iniciam em y. O isolamento-*x,y* é uma transformação que adiciona um novo arco $z = (v_{in}, v_{out})$ e remove os arcos x e y de G (Figura 13).

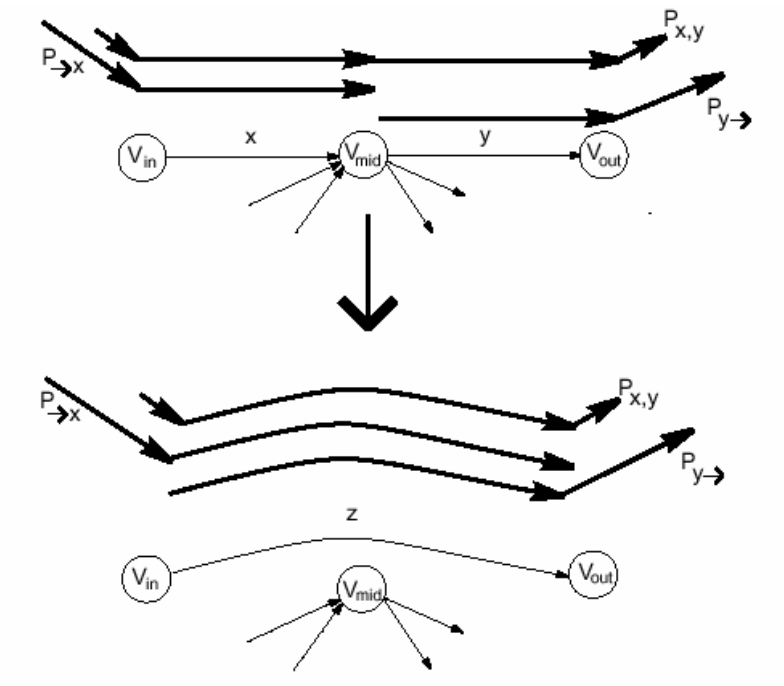


Figura 13. Isolamento-x,y é uma transformação equivalente que reduz o número de arcos do grafo.

Este isolamento altera o sistema de caminhos de P como se segue:

(i) substitua x, y por z em todos os caminhos de $P_{x,y}$;

(ii) substitua x por z em todos os caminhos de $P_{\rightarrow x}$, e

(iii) substitua y por z em todos os caminhos de $P_{y\rightarrow}$.

Como os isolamentos reduzem o número de arcos de G, eles diminuirão os caminhos de P para arcos únicos e reduzirão o problema do Supercaminho Euleriano ao problema do caminho Euleriano.

No entanto, no caso de grafos com múltiplos arcos, o procedimento de isolamento pode levar a transformações não equivalentes. Neste caso, o arco x pode ser visitado várias vezes no caminho Euleriano e pode ou não ser seguido do arco y em algumas destas visitas. Desta forma direcionar todos os caminhos do conjunto $P_{\rightarrow x}$ para o arco z pode não ser uma transformação equivalente. No entanto, se o vértice v_{mid} só tiver x como arco de entrada e y como arco de saída, o isolamento-x,y é uma transformação equivalente mesmo se x e y forem arcos múltiplos. Os isolamentos podem ser usados para reduzir todas as repetições em único arco.

É importante notar que, mesmo quando o grafo G não possui arcos múltiplos, o isolamento pode criá-los nos grafos $G_1,\ldots, G_k$. No entanto, eles não trazem problemas, já que nestes casos é sabido qual instância do arco múltiplo é usada em cada caminho.

Para ilustrar, considere o vértice v_{mid} da Figura 14 que tem apenas um arco de entrada $x = (v_{in}, v_{mid})$ com multiplicidade 2 e dois arcos de saída $y_1 = (v_{mid}, v_{out1})$ e $y_2 = (v_{mid}, v_{out2})$ com multiplicidade 1. Neste exemplo o caminho Euleriano visita o arco x duas vezes: em um caso é seguido por y_1 e no outro caso por y_2. Considere o isolamento-x,y_1 que adiciona um novo arco $z = (v_{in}, v_{out1})$ depois de remover o arco y_1 e uma das duas cópias do arco x. O isolamento (i) diminui todos os caminhos em $P_{x,y1}$ pela substituição de x,y_1 pelo arco simples z e (ii) substitui y_1 por z em todos os caminhos $P_{y\rightarrow}$. Este isolamento é uma transformação equivalente se o conjunto $P_{\rightarrow x}$ estiver vazio. Caso contrário não fica claro se o último arco do caminho $P \in P_{\rightarrow x}$ deve ser z ou x (a cópia que restou).

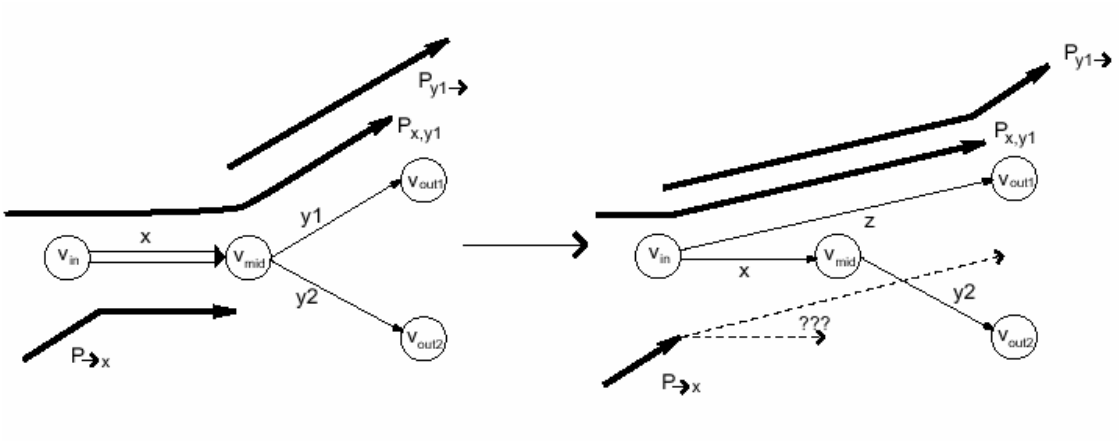


Figura 14. No caso em que x é um arco múltiplo, o isolamento-x,y1 é uma transformação equivalente se $P_{\rightarrow x}$ é vazio. Se $P_{\rightarrow x}$ não é vazio, não é claro se o último arco do caminho $P \in P_{\rightarrow x}$ deve ser z ou x.

Para resolver este dilema, deve-se analisar todos os caminhos $P \in P_{\rightarrow x}$ e decidir se ele está "relacionado" a $P_{x,y1}$ (neste caso deve ser direcionado através de z) ou a $P_{x,y2}$ (neste caso deve ser direcionado através de x). Entende-se como relacionamento a $P_{x,y1}$ ($P_{x,y2}$) todos os Supercaminhos Euleriano que visitam y_1 (y_2) logo depois de visitar x.

Dois caminhos são chamados de consistente se a união deles formar um caminho, ou seja não possuir ramificações. Um caminho P é consistente com um conjunto de caminhos P se ele for consistente com <u>todos</u> os caminhos em P, senão ele é considerado inconsistente. Existem três possibilidades (Figura 15)

- P é consistente com exatamente um dos conjuntos $P_{x,y1}$ e $P_{x,y2}$.
- P é inconsistente com ambos: $P_{x,y1}$ e $P_{x,y2}$.
- P é consistente com ambos: $P_{x,y1}$ e $P_{x,y2}$.

No primeiro caso (Figura 15a), o caminho é chamado *resolvível* porque ele está relacionado a $P_{x,y1}$ ou a $P_{x,y2}$. Se P é consistente com $P_{x,y1}$ e inconsistente com $P_{x,y2}$ então P deve ser atribuído ao arco z depois do isolamento-x,y_1 (substituindo x por z em P). Se P é consistente com $P_{x,y2}$ e inconsistente com $P_{x,y1}$ então P deve ser atribuído ao arco x. Um arco x é chamado de resolvível se todos os caminhos em $P_{\rightarrow x}$ são resolvíveis. Se o arco x é resolvível então o isolamento-x,y é uma transformação equivalente depois das atribuições corretas aos últimos arcos em cada caminho de $P_{\rightarrow x}$.

O segundo caso (P é inconsistente com $P_{x,y1}$ e $P_{x,y2}$) (Figura 15b) implica que o problema de Supercaminho Euleriano não tem solução, i.e. os dados sequenciados estão inconsistentes. Informalmente, neste caso P, $P_{x,y1}$ e $P_{x,y2}$ impõem 3 cenários diferentes para apenas duas visitas ao arco *x*.

O último caso (P é consistente com $P_{x,y1}$ e $P_{x,y2}$) (Figura 15c) corresponde a situação mais difícil. Se esta condição acontece em apenas um caminho em $P_{\rightarrow x}$, o arco *x* é chamado de não resolvível e a análise deste arco é adiada até que todos os arcos resolvíveis sejam analisados. Pode acontecer que transformações equivalentes em outros arcos resolvíveis tornem o arco *x* resolvível. A Figura 16 ilustra um caso em que transformações equivalentes resolveram arcos previamente não resolvíveis.

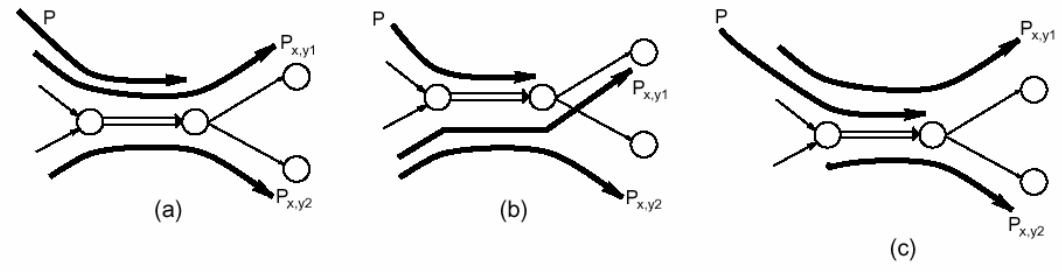


Figura 15. (a) P é consistente com $P_{x,y1}$, mas inconsistente com $P_{x,y2}$. (b) P é inconsistente com $P_{x,y1}$ e com $P_{x,y2}$. (c) P é consistente com $P_{x,y1}$ e com $P_{x,y2}$.

No entanto, alguns arcos não podem ser resolvidos mesmo depois dos isolamentos de todos os arcos resolvíveis. Tais situações geralmente correspondem a emaranhados e devem ser tratadas por outra transformação equivalente chamada de *corte*.

Considere um fragmento do grafo G com 5 arcos e 4 caminhos y_3-*x*, y_4-*x*, *x*-y_1 *e* *x*-y_2, com dois arcos (Figura 17). Neste caso $P_{\rightarrow x}$ consiste em dois caminhos *y3-x* e *y4-x* e cada um dos dois caminhos é consistente com ambos os caminhos: $P_{x,y1}$ e $P_{x,y2}$. Nesta situação simétrica, *x* é um emaranhado e não existe informação disponível para relacionar qualquer um dos dois caminhos y_3-*x*, y_4-*x* com nenhum dos outros dois caminhos *x*-y_1 e *x*-y_2. Consequentemente pode acontecer de não ter nenhum isolamento que seja uma transformação equivalente neste caso. Para tratar deste problema, introduz-se outra transformação equivalente que afeta o sistema de caminhos *P* e não afeta o grafo G.

Um arco *x = (v,w)* é removível se (i) ele for o único arco de saída de *v* e o único arco de entrada em *w* e (ii) *x* é um arco inicial ou terminal para todo caminho P $\in$ *P* que

contenha x. Um *x-corte* transforma *P* em um novo sistema de caminhos removendo *x* de todos os caminhos em $P_{\rightarrow x}$ e $P_{x\rightarrow}$.

Neste caso da Figura 17, x-cortes diminuíram os caminhos y_3-x, y_4-x, x-y_1 e x-y_2 para caminhos com arcos simples *y1, y2, y3* e *y4*. É fácil checar que um x-corte de um arco removível é uma transformação equivalente, i.e., todo Supercaminho Euleriano em (G,*P*) corresponde a Supercaminho Euleriano em (G, P_1) e vice-versa.

Os cortes provaram ser uma técnica poderosa para análise de emaranhados que não são tratáveis por isolamentos. Os isolamentos reduzem os emaranhados para arcos não resolvíveis simples que revelam-se removíveis na análise de genomas de bactérias feitas por [PTW01b]. Isto permitiu que o problema de Supercaminho Euleriano fosse reduzido para o problema de caminho Euleriano no genoma de todas as bactérias estudadas.

Apesar dos isolamentos e cortes serem suficientes para reduzir um problema no outro, ainda existe um buraco na análise teórica do problema de Supercaminho Euleriano quando os caminhos não são tratáveis por nenhum dos dois métodos (isolamentos ou cortes).

5. Comentários Finais

Este trabalho apresentou os algoritmos e programas de montagem de fragmentos.

Foi feita uma descrição detalhada do método convencional de montagem de fragmentos e apresentado os principais programas que utilizam este método, como Phrap e CAP3. Os detalhes destes programas não estão descritos em seus artigos, que geralmente dão ênfase a apresentação de resultados obtidos na execução.

Foi descrito um método diferente do convencional apresentado em [PTW01a, PTW01b]. A idéia principal deste método é uma etapa de correção de erros que modificam os fragmentos iniciais e criam uma nova instância de fragmentos para o algoritmo de montagem de fragmentos. Com esta correção, o grafo utilizado para representar os fragmentos se simplifica muito e permite obter melhores resultados.

Em uma situação ideal onde todos os fragmentos não possuem erros, os programas principais de montagem de fragmentos, Phrap, CAP3 e TIGR Assembler geram erros de montagem, enquanto este novo método [PTW01a, PTW01b] não produz erros e gera menos contigs com dados reais do que os outros programas produzem com dados sem erros. Uma das idéias propostas neste artigo é que os programas como Phrap,

CAP3 e TIGR poderiam utilizar seu método de correção de erros antes de executar sua montagem. A desvantagem é que a execução ficará mais lenta.

Os diversos programas disponíveis, apesar de se basearem no método tradicional de montagem de fragmentos, são diferentes pois, geralmente, eles são criados para diferentes aplicações. Por exemplo, o CAP3 é mais utilizado para montagem de fragmentos onde os biólogos não esperam montar o genoma completo do organismo. Isto acontece, por exemplo, quando se sabe que o genoma é tão grande e complexo que não vale a pena, atualmente, querer obter o genoma completo. Por outro lado, quando os biólogos estão interessados em montar todo o genoma de um organismo, o Phrap é utilizado.

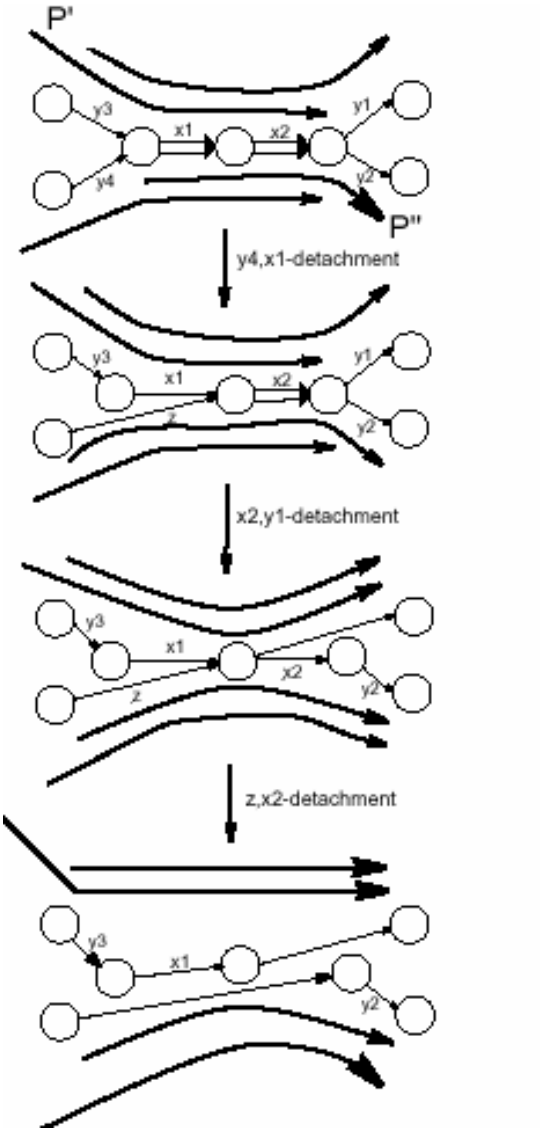


Figura 16. No grafo do topo, o arco x2 é não resolvível (P' $\in P_{\rightarrow x2}$ é consistente com $P_{x2,y1}$ e $P_{x2,y2}$) e o arco x1 é resolvível (P'' $\in P_{x1\rightarrow}$ é consistente com $P_{y4,x1}$ e inconsistente com $P_{y3,x1}$). Consequentemente o isolamento-y4,x1 é uma transformação equivalente que substitui y4 e x1 por z (segundo grafo). Esta transformação torna x2 resolvível e abre uma porta para o isolamento-x2,y1 (terceiro grafo). Finalmente, o isolamento-z,x2 trata de uma simplificação no grafo original.

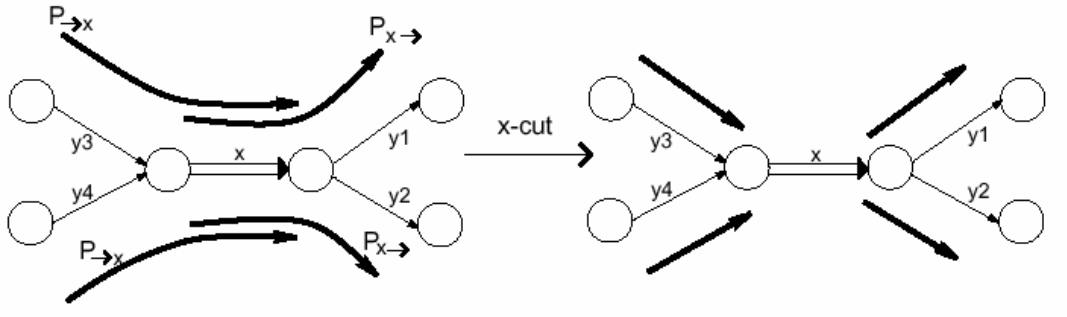


Figura 17. Se x é um arco removível, então o x-corte é uma transformação equivalente que diminui os caminhos em P.

Referências

[ACH+00] Adams, M.D., Celniker, S.E., Holt, R.A. The genome sequence of *Drosohila* melanogaster. Science, 287: 2185-2195, 2000.

[AGM+90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, e D. J. Lipman. A basic local alignment search tool. *J. of Molecular Biology* 215, pp. 403-410, 1990.

[AM99] Anson, E., Myers, E.W.. Algorithms for whole genome shotgun sequencing. Proceedings of the Third Annual International Conference on Research in Computational Molecular Biology, 1-9, 1999 (RECOMB1999).

[BSS95] Bonfield, J.K., Smith, K.F., Staden, R. A new DNA sequence assembly program. *Nucleic Acids Research*, 23: 4992-4999, 1995.

[Coo94] Cooper, N. *The Human Genome Project*. Univ. Science Books, Mill Valey, CA, 1994.

[DLB+89] Drmanac, R., Labat, I., Brukner, I., Crkvenjakov, R.. Sequencing of megabase plus DNA by hybridization: theory of the method. Genomics, 4: 114-128, 1989.

[DS91] Dean, S., Staden, R. A sequence assembly and editing program for efficient management of large projects. Nucleic Acids Research, 10: 4731-4751, 1982.

[DOE92] Human Genome Program, U.S. Department of Energy, *Primer on Molecular Genetics*, Washington, D.C., 1992. Disponível em http://www.ornl.gov/hgmis/publicat/primer/intro.html.

[EHW+98] Ewing, B., Hillier, L., Wendl, M.C., Green, P. Base-Calling of Automated Sequencer Traces using Phred. I. Accuracy Assessment. Genome Research, 8: 175-185, 1998.

[EG98] Ewing, B, Green, P. Base-Calling of Automated Sequencer Traces using Phred. II. Error Probabilities. Genome Research, 8: 186-194, 1998.

[Flei90] Fleishner, H.. *Eulerian Graphs and Related Topics*. Elsevier Science, London, 1990.

[FSW96] Ferreira, C.E., Souza, C.C., Wakabayashi, Y.. Rearrangement of DNA Fragments: a branch-and-cut algorithm. *Proceedings of the 2nd Workshop on Solving Practical Combinatorial Optimization Problems*, 1996.
Disponível em http://www.ime.usp.br/~yw/html/nodetudo.html e http://citeseer.nj.nec.com/338477.html.

[GAG98] Gordon, D., Abajian, C., Green, P. Consed: A Graphical Tool for Sequence Finishing. Genome Research, 8: 195-202, 1998.

[Gre94] Green, P. Documentation for Phrap.
http://bozeman.mbt.washington.edu/phraps.docs/ phrap.html, 1994.

[Gre97] Green, P. Against a Whole-Genome Shotgun. Genome Research, 7:410-417, 1997.

[Gus97] Gusfield, D. Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology. Cambridge University Press, 1997.

[HM99] Huang, X., Madan, A. CAP3: A DNA sequence assembly program. *Genome Research*, 9: 868-877, 1999.

[HRM01] Huson, D.H., Reinert, K., Myers, E.W.. The greedy path-merging algorithm for sequence assembly. Proceedings of the Fifth Annual International Conference on Research in Computational Biology, 157-163, 2001 (RECOMB2001). ACM, 2001.

[Hua92] Huang, X. A contig assembly program based on sensitive detection of fragment overlaps. *Genomics*, 14: 18-25, 1992.

[IW95] Idury, Ramana M., Waterman, Michael S. A new algorithm for DNA Sequence Assembly. *Journal of Computational Biology*, 2:291-306, 1995.

[Kec91] Kececeioglu, J. D. Exact and approximation algorithms for DNA sequence reconstruction problem in computational biology. *PhD thesis*, Department of Computer Science, The University of Arizona, 1991.

[Kec93] Kececeioglu, J. D.. The maximum weight trace problem in multiple equence alignment. *Proc.4th Symp. on Combinatorial Pattern Matching*. *Springer LNCS* 684, 106-19, 1993.

[KM95] Kececeioglu, J. D., Myers, E.W. Combinatorial algorithms for DNA sequence assembly. *Algorithmica*, 13:7-51, 1995.

[KR98] Karp, R., Ruzzo, L. Algorithms in Molecular Biology. CSE 590BI, 1998. Disponível em http://www.cs.washington.edu/people/faculty/karp.html.

[KS99] Kim, S., Segre, A.M.. AMASS: A Structured Pattern Matching Approach to Shotgun Sequence Assembly. Journal of Computational Biology, vol6 (2),163-186,1999. Disponível em http://citeseer.nj.nec.com/85142.html.

[KY01] Kececeioglu, J. D., Yu, J.. Separating repeats in DNA sequence assembly. Proceedings of the Fifth Annual International Conference on Research in Computational Biology, 157-163, 2001 (RECOMB2001). ACM, 2001.

[LFK+88] Lysov, Y., Florent'ev, V., Khorlin, A., Khrapko, K., Shik, V., Mirzabevok, A.. DNA sequencing by hybridization with oligonucleotides. Doklady Academy Nauk USSR, 303: 1508-1511, 1988.

[MG77] Maxam, A.M., Gilbert, W., 1977. A new method for sequencing DNA. Proc. Natl. Acad. Sci. USA 74, 560-564, 1977.

[Mei92] João Meidanis. Algorithms for Problems in Computational Genetics. PhD Thesis, University of Wisconsin-Madison, 1992.

[Mei93] João Meidanis. Rethinking the DNA Fragment Assembly Problem. Relatório Técnico Departamento de Ciência da Computação DCC 23-93. Universidade Estadual de Campinas, São Paulo, Brasil, 1993.

[MS94] J. Meidanis e J. C. Setúbal. *"Uma Introdução à Biologia Computacional"*. IX Escola de Computação. Recife, 1994.

[MSD+00] Eugene W. Myers, Granger G. Sutton, Art L. Delcher, Ian M. Dew, Dan P. Fasulo, Michael J. Flanigan, Saul A. Kravitz, Clark M. Mobarry, Knut H. J. Reinert, Karin A. Remington, Eric L. Anson, Randall A. Bolanos, Hui-Hsien Chou, Catherine M. Jordan, Aaron L. Halpern, Stefano Lonardi, Ellen M. Beasley, Rhonda C. Brandon, Lin Chen, Patrick J. Dunn, Zhongwu Lai, Yong Liang, Deborah R. Nusskern, Ming Zhan, Qing Zhang, Xiangqun Zheng, Gerald M. Rubin, Mark D. Adams, J. Craig Venter. A whole-genome assembly of *Drosophila*. *Science*, 287: 2196-2204, 2000.

[Mye94] Myers, E.W.. Advances in sequence assembly in Automated DNA Sequencing and Analysis Techniques (C.Ventner, ed.), Academic Press Limited (London, 1994), 231-238.

[Mye95] Myers, E.M. Toward simplifying and accurately formulating fragment assembly. Journal of Computational Biology, 2: 275-290, 1995.

[Mye99] Myers, E.M. Whole-Genome DNA Sequencing. *IEEE Computational Engineering and Science 3*, 1:33-43, 1999.

[MW96] Myers, E.M., Weber, J.L. Is whole genome shotgun sequencing feasible? In Computational Methods in Genome Research (S.Suhai, ed.), Plenum Press, 73-89, 1996.

[Pev89] Pevzner, P.. l-tuple DNA sequencing: computer analysis. *Journal of Biomolecular Structure and Dynamics*, 7:63-73, 1989.

[PFB93] Parsons, R.J., Forrest, S., Burks, C.. Genetic Algorithms, Operators, and DNA fragment assembly. Proceedings First Conf. Intelligent Systems for Molecular Biology, AAAI Press, Menlo Park, Calif., 1993, pp. 310-318.

[PST+83] Peltola, H., Soderlund, H., Tarhio, J., Ukkonen, E.. Algorithms for some string matching problems arising in molecular genetics. *Proc. of the 9th IFIP World Computer Congress*, 59-64, 1983.

[PSU84] Peltola, H., Soderlund, H., Ukkonen, E.. SEQAID: A DNA sequence assembling program based on a mathematical model. *Nucleic Acids Research*, 12, 307-321, 1984.

[PTW01a] Pevzner, P.A., Tang, H., Waterman, M.S.. An Eulerian path approach to DNA fragment assembly. Proceedings of the National Academy of Sciences, vol. 98, no. 17, 9748-9753, 2001.

[PTW01b] Pevzner, P.A., Tang, H., Waterman, M.S.. A new approach to Fragment Assembly in DNA Sequencing. Proceedings of the Fifth Annual International Conference on Research in Computational Biology, 256-267, 2001 (RECOMB2001). ACM, 2001.

[SL00] Sterky, F., Lundeberg, J. Sequence analisys of genes and genomes. Journal of Biotechnology. 76, 1 - 31, 2000. Disponível em www.elsevier.com/locate/jbiotec.

[SNC+77] Sanger, F., Nicklen, S., Coulson, A.R.. DNA Sequencing with chain-terminating inhibitors. Proc. Natl. Acad. Sci. USA 74, 5463-5467, 1977.

[Sta79] Staden, R. A strategy of DNA sequencing employing computer programs. *Nucleic Acids Research*, 6: 2601-2610, 1979.

[Sta80] Staden, R. A new computer method for the storage and manipulation of DNA gel reading data. Nucleic Acids Research, 8 (16): 3673-3694, 1980.

[Sta82] Staden, R. Automation of the computer handling of gel reading data produced by the shotgun method of DNA sequencing. Nucleic Acids Research, 10: 4731-4751, 1982.

[SWA+95] Sutton, G., White, O., Adams, M., Kerlavage. TIGR assembler: A new tool for assembling large shotgun sequencing projects. Genome Science & Technology, 1: 9-19, 1995.

[Wat95] Waterman, M.S. Introduction to Computational Biology - Maps, sequences and genomes. Chapman & Hall, 1995.

[WDH+98] Wendl, Michael C., Dear, Simon, Hodgson, Dave, Hillier, LaDeana. Automated Sequence Prepocessing in a Large-Scale Sequencing Environment. Genome Research, 8: 975-984, 1998.

[WHR+87] Watson, J., Hopkins, N., Roberts, J., Steitz, J., Weiner, A.. *Molecular Biology of the Gene*, 4th ed. Benjamin Cummings, Menlo Park, CA, 1987.

[WM97] Weber, J.L., Myers, E.W.. Human Whole-Genome Shotgun Sequencing. Genome Research, 7:401-409, 1997.

Anexo

Esta seção está dedicada a definições da teoria de grafos necessárias para o entendimento do texto anterior.

Um *grafo* consiste em um conjunto de vértices finito e não vazio e um conjunto de pares desordenados de vértices, chamados de arcos.

Um *caminho* em um grafo é uma sequência não vazia de vértices tais que dois vértices consecutivos são unidos.

O *grau* de um vértice é o número de arcos que terminam e que se iniciam em um vértice. O *grau de chegada* é o número de arcos que terminam em um vértice e o *grau de saída* é o número de arcos que se iniciam em um vértice.

Um *caminho Euleriano* é um caminho que se inicia em um vértice, termina em outro vértice e que passa por cada arco exatamente uma vez. É também conhecido como *complete edge path.*

Um *Circuito Euleriano* é um caminho Euleriano que começa e termina no mesmo vértice.

Um *Caminho Hamiltoniano* é um caminho que se inicia em um vértice, termina em outro vértice e que passa por cada vértice exatamente uma vez. É também conhecido como *complete vertex path*.

Um *Circuito Hamiltoniano* é um caminho Hamiltoniano que começa e termina no mesmo vértice.

Sendo *s* o vértice de início e t o vértice final, o *Teorema de Euler* diz que existe um caminho Euleriano se e somente se

$$\text{grau de chegada}(v) = \text{grau de saída}(v) \qquad \text{para } v \neq s,t$$

$$\text{grau de saída}(s) - \text{grau de chegada}(s) = 1,$$

$$\text{grau de saída}(t) - \text{grau de chegada}(t) = -1.$$