3
Montagem de Fragmentos

A montagem de fragmentos é o passo seguinte ao sequenciamento do genoma. É a etapa onde os dados gerados são processados e obtém-se como resposta o mapeamento do genoma.

A montagem de fragmentos é um dos principais problemas estudados em biologia computacional[13] e a complexidade para a realização da montagem está relacionada ao tamanho do genoma, à quantidade de material genético sequenciado e a cobertura realizada.

Existem basicamente duas abordagens para a montagem de fragmentos:

1. Mapeamento (*Mapping*): quando já existe um genoma montado e pode-se usá-lo como referência na tomada de decisão ao encontrar ambiguidades e erros de sequenciamento.
2. *De Novo*: a montagem do genoma é feita exclusivamente a partir das sequências geradas, sem usar como base um genoma de referência.

Nesta dissertação analisaremos a abordagem De Novo, que é a abordagem utilizada pelo Velvet, programa estudado neste trabalho e descrito no capítulo 4.

3.1.
Abordagens *De Novo* para montagem de fragmentos

Para se realizar a montagem de fragmentos sem o genoma de referência, os métodos existentes se dividem principalmente em 3 grupos de abordagem: gulosos, sobreposição e os baseados em grafos de Bruijn [14].

3.1.1.
De Novo gulosos

Neste tipo de abordagem, o algoritmo sempre faz a escolha com o maior benefício imediato. Especificamente, a montagem sempre une os fragmentos com a melhor sobreposição, desde que esta não se contraponha ao que já fora montado. As decisões são sempre locais e não levam em conta a relação global entre os fragmentos.

O seu uso não é amplamente difundido por conta desta dificuldade em analisar as ligações globais entre os fragmentos. Esta análise é essencial para solucionar os erros de sequenciamento e identificar os erros de genes repetitivos. A maioria das implementações trabalha com heurísticas para contornar essa característica e desta forma evitar as sequencias repetidas[15]. Entretanto, este trabalho adicional torna esta abordagem menos eficiente que as demais para os fragmentos curtos[15].

3.1.2. De Novo por sobreposição

Esta abordagem é geralmente utilizada para os métodos tradicionais de sequenciamento Sanger e é composta de três fases[16]:

1. *Overlap*: inicia identificando os pares de fragmentos que tenham uma boa sobreposição. Organiza-os em um grafo, onde cada fragmento é um nó e as arestas são a ligação entre os fragmentos que possuem esta sobreposição.
2. *Layout:* o algoritmo calcula o caminho Hamiltoniano no grafo, que é um problema NP-Completo, com uma variação gulosa que tende a ser tratável para este tipo de grafo.
3. *Consensus*: compara as sobreposições que não se encaixam perfeitamente afim de decidir qual a provável posição no genoma.

Os algoritmos que usam esta abordagem se demostraram bem-sucedidos apenas para os casos onde a montagem é de pequenos genomas com fragmentos grandes. Isso se deve, principalmente, a problemas de escalabilidade em função do uso de caminhos Hamiltonianos e à natureza do grafo, onde as ambiguidades não são explícitas, ficando as sobreposições reais junto às sobreposições induzidas pela repetição dos fragmentos, conforme mostrado na Figura 4.

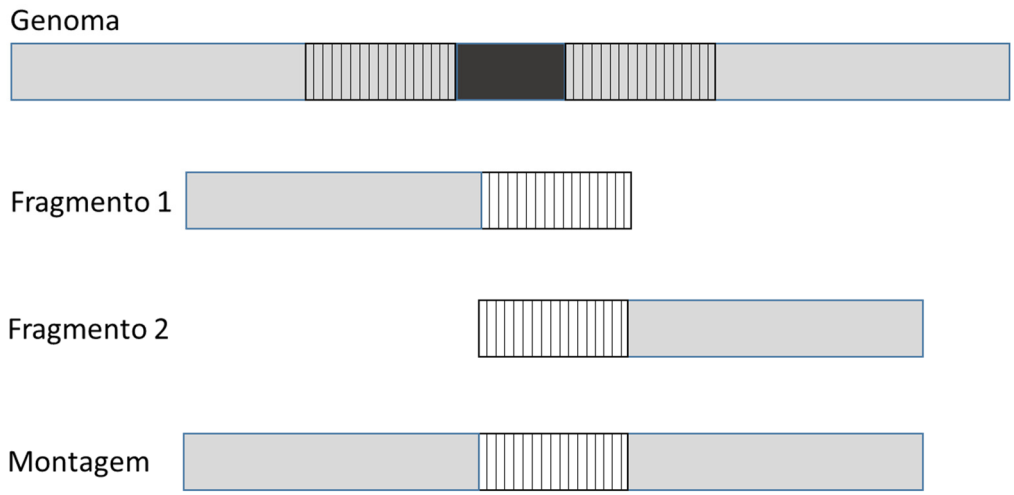

Figura 4 Erro na montagem de uma abordagem de sobreposição

Os dois segmentos riscados na Figura 4 são idênticos, portanto, na técnica de sobreposição pode-se cometer o erro de unir o segmento inicial com o segmento final, ignorando o segmento do meio. Estas implementações utilizam heurísticas para evitar este tipo de problema, aumentando a complexidade de computar todas as sobreposições.

3.1.3. De Novo baseados no grafo de Bruijn

Esta estrutura de grafo obteve este nome em homenagem ao matemático Nicolaas Govert de Bruijn que, em 1946, adaptou as ideias de Euler sobre o caminho Euleriano para encontrar uma sequência cíclica de letras tiradas de um determinado alfabeto[17]. Estas abordagens modelam os relacionamentos entre subsequências exatas de tamanho k extraídas dos fragmentos de modo semelhante à abordagem de sobreposição, através de grafos. Os nós representam os k-mers e as arestas indicam os k-mers adjacentes com a sobreposição exata de k-1, conforme ilustrado na Figura 5.

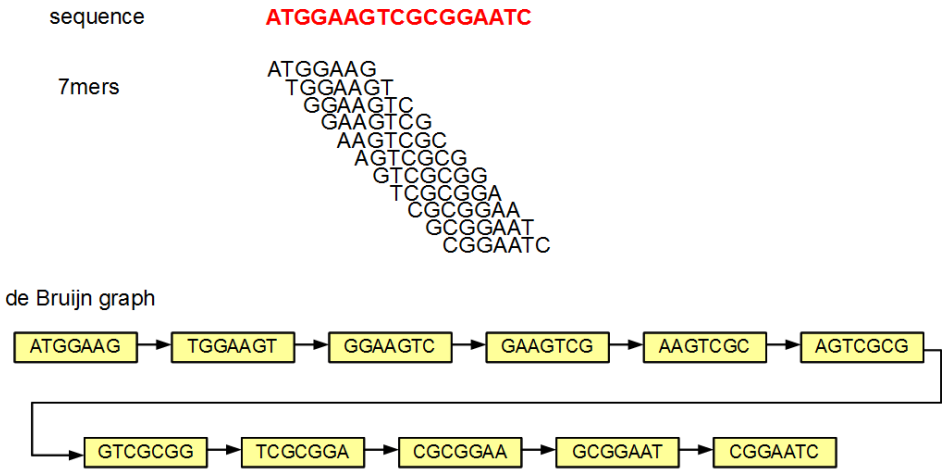

Figura 5 Extração dos k-mers e a construção do grafo de Bruijn

O *workflow* de execução dos algoritmos baseados neste método, seguem basicamente os mesmos passos, conforme apresentado na figura 6 que são:

- Geração de todas as *substrings* de tamanho k a partir dos fragmentos
- Construção do grafo de Bruijn.
- Redução do grafo, colapsando caminhos e retirando componentes não-conexos através de heurísticas especificas para grafos.
- Execução do caminho Euleriano no grafo, visitando cada aresta apenas 1 vez.
- Construção dos *contigs*.

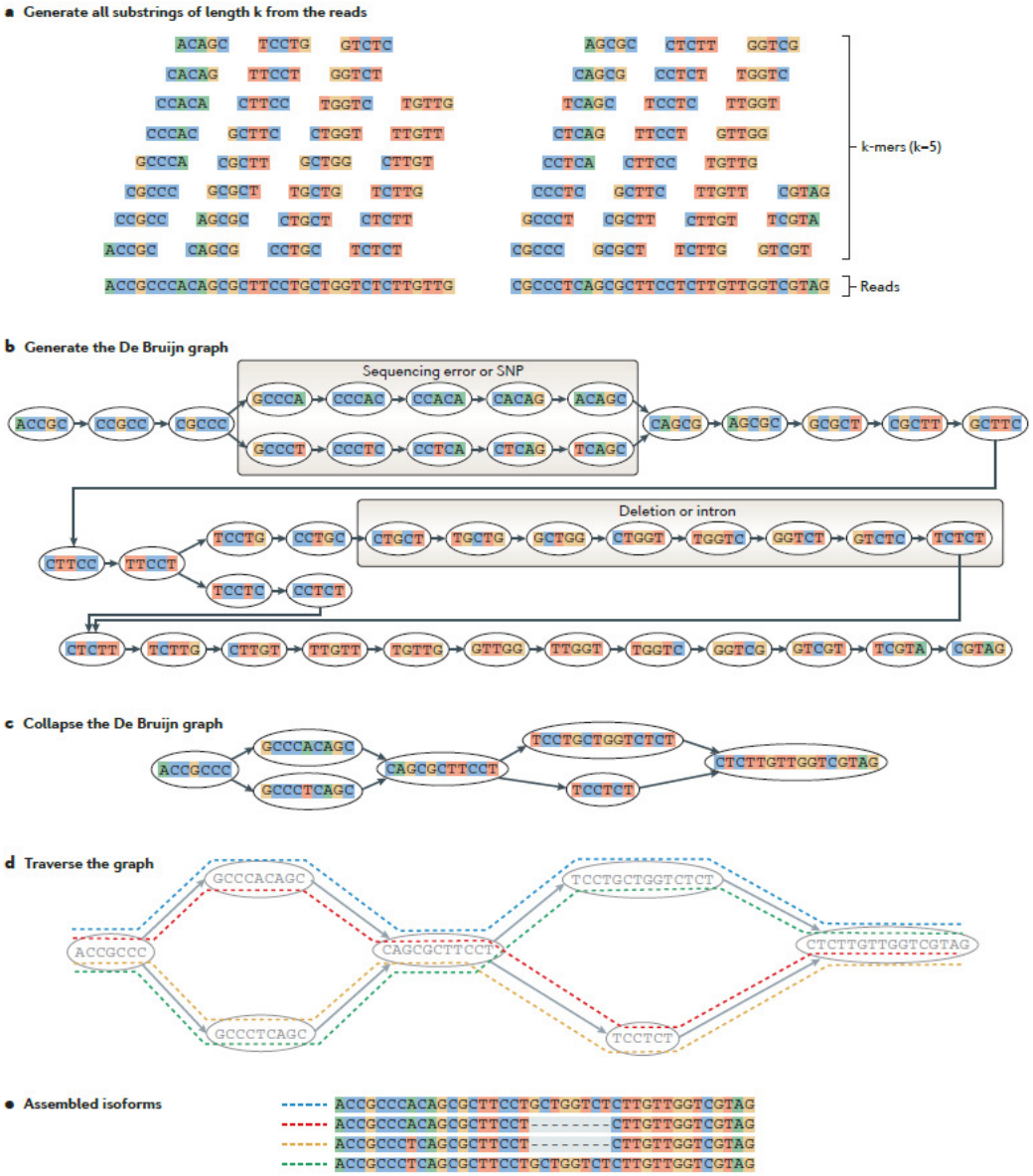

Figura 6 *Workflow* dos algoritmos baseados nos grafos de Bruijn[18]

A questão de qual deve ser o tamanho de K é algo que, em termos práticos, só é descoberto executando-se o programa diversas vezes, com diferentes tamanhos para k e escolhendo ao final melhor resultado analisando a quantidade e a qualidade dos *contigs* gerados[19].

Se utilizarmos K com um valor muito pequeno, induzimos um custo de processamento desnecessários dentro do grafo de *Bruijn*, se colocarmos um tamanho de K muito grande, a quantidade de nós distintos se torna muito grande, tornando o trabalho para a remoção de erros, que veremos mais a frente mais difícil.

3.2. Problema da montagem de fragmentos NGS

Dois fatores influenciam para o aumento na complexidade da resolução do problema de montagem de fragmentos da NGS[12]. O primeiro está relacionado ao tamanho dos fragmentos, como estes são pequenos, a biblioteca de saída possuiu uma quantidade muito maior de fragmentos a serem montados, dificultando a solução do problema. O segundo, está relacionado a quantidade de erros de sequenciamento, que dado as características do processo, não é baixa, conforme veremos na secção 3.2.1.

Como vimos na secção 2.3, para solucionar este problema é necessário aumentar a cobertura de sequenciamento. Isso resulta no aumento da quantidade de fragmentos na biblioteca de saída, tornando o problema ainda mais complexo para solucionar.

3.2.1. Erros de sequenciamento

Com uma quantidade de fragmentos muito maior dada a característica da NGS e a necessidade de uma cobertura muito alta, mesmo que o percentual de erros seja pequeno para a metodologia de sequenciamento escolhida, o volume de dados total torna o problema de montagem muito maior.

Através de um genoma completamente mapeado, é possível demonstrar a relação do número de erros com o aumento da quantidade de cobertura. A Figura 7, apresenta um gráfico com duas linhas traçadas. A primeira, marcada com círculos, indica a quantidade de k-mers distintos encontrados nas bibliotecas a cada sequenciamento gerado com um certo número de cobertura. A segunda linha, marcada com losangos, mostra a quantidade de k-mers distintos encontrados nas bibliotecas que de fato fazem parte do genoma.

A diferença entre a quantidade de k-mers distintos encontrados nas bibliotecas versus a quantidade de k-mers distintos existentes no genoma é o total de erros de sequenciamento existente. Este número aumenta de forma quase linear a cobertura.

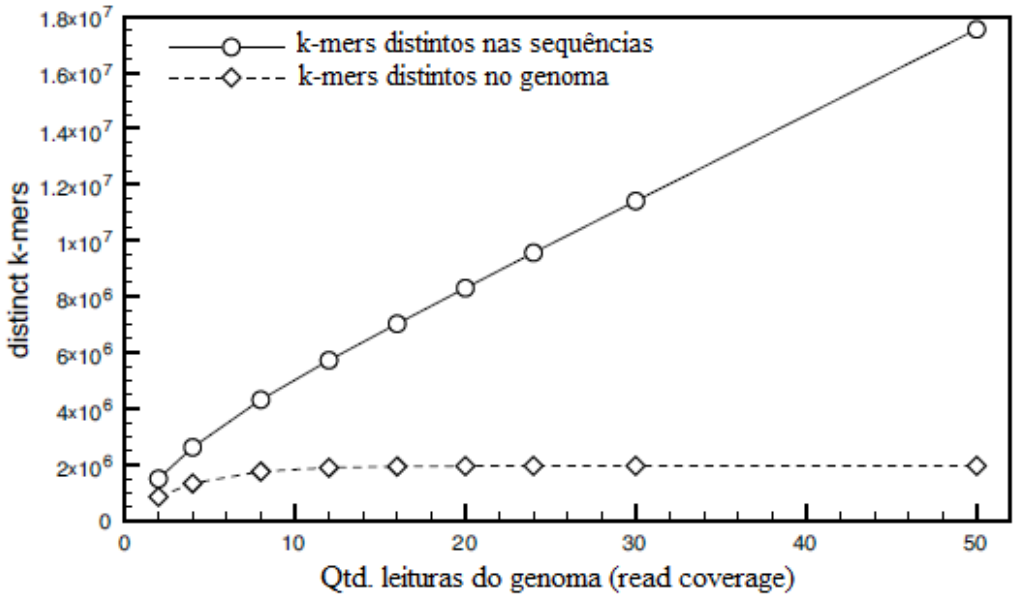

Figura 7 Quantidade de k-mer distintos no genoma versus quantidade de k-mer distintos sequenciados. Genoma: S. suis P1/7, k=21 [12].

É neste cenário que os programas baseados no grafo de Bruijn ganharam destaque pois, na sua implementação, diversas rotinas de tratamento de erro são aplicadas e o algoritmo para achar a sequência correta dos fragmentos, caminho Euleriano, tem complexidade O(n). Diferente de implementações antigas que em sua maioria utilizavam algoritmos NP-Completos para o problema de montagem no grafo, como caminhos Hamiltonianos em grafos de sobreposição de fragmentos [17].

3.2.2. Consumo de memória principal

A remoção dos erros de leitura, após um grafo ser construído, é relativamente simples do ponto de vista de processamento computacional. Porém, quanto maior o universo de dados a se trabalhar, maior será a necessidade de uso de memória principal para a montagem do grafo inicial.

Considerando-se uma estrutura de dados simples para o armazenamento dos nós no grafo, cada aresta armazena um valor de texto com o conteúdo de k e um

numérico com a identificação do fragmento correspondente {Char * k; int idSeq}, para um k=21 onde cada posição de k faz parte do alfabeto ∑={A,C,T,G}:

- Universo possível de k-mers distintos: [12]
  - 4^{21} = 4,3 Trilhões
  - Necessário 100 Terabytes de memória para armazenamento
- Dados da Cana-de-açúcar (Bibliotecas UFRJ)
  - Fragmentos na biblioteca = 275 Milhões
  - Tamanho médio dos fragmentos = 90bp
  - Quantidade de k-mers distintos possíveis = 13 bilhões
  - Necessário 449 Gigabytes de memória para armazenamento

Como visto na Figura 7, a quantidade de k-mers distintos obtidos pelo sequenciamento pode chegar a ser 7 vezes maior que a quantidade de k-mers distintos reais no genoma. No genoma utilizado na Figura 7, mesmo sendo de um genoma considerado pequeno, o Streptococcus Suis P1/7, o consumo de memória exigido muda de 100-200MB para mais 1GB com a inclusão dos erros. Em genomas maiores, como o da cana-de-açúcar, o consumo de memória se torna ainda maior.

3.3. Conclusão

Os conceitos da montagem de fragmentos, necessários para contextualizar a pesquisa apresentada foram explicitados, focando o interesse nas abordagens comumente estudadas. Abordou-se o problema computacional envolvendo a montagem de fragmentos e explicamos a relação entre a quantidade de erros nos dados com o alto consumo de memória RAM.

A seguir, será apresentado a implementação do Velvet, um montador de fragmentos, em especial o módulo VelvetH, onde a aplicação realiza operações de pré-processamento nos fragmentos afim de retirar os erros e reduzir o consumo de memória principal na fase de construção do grafo, assunto principal desse estudo.