Resumo do artigo "CAP3: a terceira versão de montagem de sequências de DNA"

Rodney Rick
email: rodneyrick@gmail.com

***Resumo*—O artigo descreve a terceira geração do programa CAP[1] o qual novos recursos, como análise de baixa qualidade de *reads* gerados e como verificar sobreposições mais eficientes para melhor aproveitamento de construções múltiplas de alinhamento. Outro ponto é a comparação com PHRAP utilizando conjuntos BAC.**

I. Introdução

A abordagem de sequenciamento utilizando a estratégia de *shotgun* tem sido amplamente utilizada em projetos de sequenciamento do genoma. Com o intuito de reunir estruturas curtas (chamadas de *reads*) e combiná-los a fim de produzir longas sequências. Neste artigo, o autor comenta vários programas de outros pesquisadores que enfrentam o mesmo desafio e buscam resolvê-lo, por exemplo, aplicando bibliotecas para enquadramento de rotinas para cada fase do processo de montagem. O desenvolvimento e aperfeiçoamento contínuos de programas de montagem de sequências são necessários para enfrentar os desafios dos projetos de genomas de animais e plantas.

Ainda no artigo descreve-se o novo processo do CAP3 que apresenta uma série significativa de melhorias e novas abordagens perante os recursos propostos como: aumento da capacidade de agregar sequências de baixa qualidades; sobreposição entre *reads* de *reads*; e geração de sequências utilizando a metodologia de consenso.

Os resultados de CAP3 em quatro conjuntos de dados de BAC (*Bacterial Artificial Chromosome*) são apresentados e neste artigo realiza uma comparação com o abordagem de PHRAP (*Phil's Revised Assembly Program*). Onde, por exemplo, CAP3 produz sequências mais curtas com menos erros, enquanto PHRAP funciona opostamente, com *reads* mais longos com mais erros.

II. Metodologia

O algoritmo de montagem consiste em três etapas principais, conforme demonstra a figura 1 do artigo. Na primeira fase, as regiões desprovidas de qualidades, como sobreposição de *reads* nas fitas 5' e 3' de cada *read* são identificadas e removidas, falsas sobreposições são removidas também, através de cálculos da porcentagem de qualidade que deseja-se obter. Numa segunda etapa, os *reads* são unidos para formar *contigs* e ordenados em ordem decrescente de pontuações de sobreposição. Após isso, as restrições *forward-reverse* são usadas para fazer correções de possíveis bugs na construção dos *contigs*. Na terceira etapa o foco é concentrado no alinhamento múltiplo de sequências e no cálculo da qualidade para cada base de *contig* alinhado.

Nas próximas subseções estão citados algumas etapas em mais detalhes.

A. Rápida identificação de pares de reads utilizando sobreposições

Projeto para ser um método rápido para encontro de pares de sequências que se sobrepõe, trabalhando com as seguintes definições:

- sejam f_1, f_2, ... f_n os reads de entrada, conforme orientação selecionada;
- e sejam r_1, r_2, ... r_n, os complementos reversos de cada f_x.

De acordo com os autores, "a abordagem consiste em encontrar pares entre f_x e f_y ou r_x e r_y que se sobrepõe, sempre considerando $x < y$. Um detalhe a ser ressaltado é que cada par de *reads* identificadas representa duas sobreposições simétricas devido a uma relação complementar inversa. Uma sobreposição entre as *reads* f_x e f_y é simétrica para uma entre as *reads* r_x

e r_y e uma sobreposição entre as *reads* r_x e f_y para uma entre as *reads* f_x e r_y".

Então, a fim de determinar se duas cadeias se sobrepõe, alinha-se-as, antes já ordenados por critério de tamanho e utilizando a técnica como BLAST, verifica-se quais *reads* de mesmo tamanho são mais próximos, ou seja, uma maior pontuação. A similaridade entre duas cadeias é considerada em potencial caso a pontuação atingida seja maior que a pontuação de corte [1] desejada.

Com a separação dos *reads* (f_1,f_2,...) são combinados, delimitando os pontos necessários para estudo com os caracteres especiais. O próximo passo consiste em executar os *reads* da sequencia combinada com f_x e f_y, respeitando sempre a condição $x < y$. E, em paralelo, alinhar reversos r_x para encontrar pares de *reads*, como r_x e f_y.

Assim, para cada novo *read* sobreposto e considerado potencialmente interessante, cria-se uma matriz com diagonais, utilizando programação dinâmica, determinando possíveis pontos a serem cobertos pelas cadeias de pontuação maiores que a pontuação mínima de corte desejada.

B. Recorte de regiões de baixa qualidade

Regiões da sequencia combinada que possam apresentar baixa qualidade ou não são devidamente localizadas nos *contigs*, são removidas. O resultado dessa estratégia de recorte oferece a um *read* a chance de melhorar sua qualidade, assim como os autores descrevem "qualquer região suficientemente longa de valores de alta qualidade que seja altamente similar a uma região de outra *read* é definida como boa". Na figura 2 do artigo, ilustra-se o cálculo das posições de recorte $5'$ e $3'$ de um *read*. Com valores maiores de qualidade, torna-se possível obter um alinhamento local ótimo através de sobreposição potencialmente de alta qualidade. Aproximando para a diagonal principal da matriz de comparação entre duas sequências, ou pelo menos, expandido a faixa entre a diagonal principal e tornando o alinhamento mais próximo do ótimo. Como especificado, o algoritmo de alinhamento local de Smith e Waterman (descrito em 1981), busca qualidade de possíveis bases alinhadas localmente. O valor de qualidade de uma base é $q = -10 * log_{10}(p)$, onde p é a probabilidade estimada de erro para a base. Qualquer incompatibilidade ou penalidade de diferença no alinhamento são ponderadas pelos valores de qualidade das bases presentes.

Logo, trabalhando com valores de qualidades de base, torna-se possível obter duas abordagens. Para resultados positivos, exibem maiores qualidades e o inverso para resultados negativos. Os autores descrevem como fatores de correspondência, adotando:

- m, n e g como inteiros positivo, negativo e fator de penalidade de extensão de espaçamento, respectivamente;
- q_1 e q_2 são bases de valores de qualidade de uma coluna no alinhamento e compõem as pontuações para os cálculos $m * min(q_1, q_2)$ para valores de boa qualidade, $n * min(q_1, q_2)$ para valores com discrepância e $-g * min(q_1, q_2)$ para extensões.

Assim, tem-se que para cada *read*, uma boa região de qualidade, caso contrário, pode ser removida.

C. Computação e Avaliação de Sobreposições

Um dos pontos fortes deste método é identificar falsas sobreposições, podendo assim gerar um *read* de forma mais limpa ou um *read* significativamente bom para uma região boa, excluindo regiões pobres entre $5'$ e $3'$. A definição para uma sobreposição de um alinhamento global ideal descreve quando dois *reads* apresentam um alinhamento local ideal, eliminando sobreposições falsas. Ou seja, quando algumas regiões boas de um alinhamento global entre os *reads* não são semelhantes, indica que a sobreposição é falsa, enquanto que um alinhamento local ideal mostra apenas regiões semelhantes. Assim, com essa abordagem, verifica-se formação de bandas diagonais centradas no posição de alinhamento inicial e expandido para uma possível pontuação de valor maior, conforme exibido na figura 3 do artigo. Facilitando a computação no momento do alinhamento, tanto local quanto global entre duas sequências, e atuando com técnica de divisão-e-conquista para execução dos cálculos das banda das faixas.

Os autores avaliam que para cada sobreposição deve-se considerar cinco medidas. Quando a sobreposição falhar em qualquer uma das regras, então a mesma é desconsiderada na construção de *contigs*.

- As três primeiras medidas determinam se a sobreposição satisfaz os requisitos mínimos de comprimento, porcentagem de identidade e pontuação de similaridade;

[1] Uma pontuação de corte pode ser adotada utilizando como o tamanho do alinhamento mínimo desejado ou uma adaptação deste corte conforme a necessidade apresentada pela qualidade dos dados dos *reads* da base de dados.

- A quarta medida examina possíveis diferenças entre sobreposições em bases de valores de alta qualidade. Se a sobreposição contiver um número suficiente de diferenças em bases de valores de alta qualidade, então a sobreposição é provavelmente falsa;
- Para a quinta medida, a taxa de diferença da sobreposição é examinada em relação às taxas de erro de sequenciamento das duas regiões envolvidas na sobreposição. A taxa de erro de qualquer região de um *read* é estimada usando o método do vetor de erro. Para qualquer sobreposição verdadeira, a taxa de diferença da sobreposição é próxima da soma das taxas de erro das duas regiões envolvidas na sobreposição. Caso contrário, desconsidera-se.

D. Uso de Restrições na Construção de Contigs

De forma resumida, todo o procedimento para o uso de restrições na construção de *contigs* consiste em quatro etapas principais. Na primeira etapa, um layout inicial de *reads* é considerado usando um método guloso em ordem decrescente de pontuações de cada sobreposição. Na segunda etapa, a qualidade da apresentação dos *contigs* é avaliada pela verificação de restrições, assim as informações podem satisfazer as restrições para cada parte dos *reads* coletado. Na terceira etapa, uma região é considerada com maior número de restrições insatisfeitas se está localizada satisfaz os critérios de avaliação para formação de um *contig*. Se tal região existir, são feitas correções para a região e os passos 2 e 3 são repetidos. Caso contrário, o procedimento é terminado. Na etapa 4, *contigs* são combinados com as devidas restrições.

E. Construção de Alinhamentos e Sequências de Consenso

Um alinhamento de sequências múltiplas de *reads* é construído para cada *contig*, assim a construção é feita através do alinhamento repetido até o próximo *reads* com o alinhamento atual.

Para uso da metodologia de algoritmo guloso, todos os *reads* são ordenados crescentemente em suas posições no *contig*. Depois de construído um alinhamento, calcula-se uma sequência de consenso juntamente com um valor de qualidade para cada base para o *contig*. Para cada coluna do alinhamento, calcula-se uma soma ponderada dos valores de qualidade para cada tipo de base e, caso o tipo da base com a maior soma dos valores de qualidade, aceita-se como consenso para a coluna. O valor de qualidade para a base de consenso é descrita pela soma dos valores de qualidade para cada tipo base do consenso menos a soma dos valores de qualidade para cada outro tipo de base. Caso uma coluna apresente dois tipos de base predominantes, cada um com uma soma grande de valores de qualidade, então um valor de qualidade muito baixa é atribuído à base de consenso. A atribuição do valor de baixa qualidade indica um problema potencial frequentemente causado por polimorfismo ou colapso de cópias altamente semelhantes de um elemento repetitivo.

Os valores de qualidade CAP3 para bases de consenso estão relacionados às taxas de erro estimadas para cada uma das bases. Um conjunto adequado de pesos será elaborado no futuro para que os valores de qualidade CAP3 correspondam com precisão às taxas de erro.

Um alinhamento global do bloco e do *read* que apresentar pontuação máxima é computado em espaço linear usando uma técnica do algoritmo de Hirschberg. Como a computação de alinhamento *pairwise* é realizada no máximo uma vez para cada *read*, é acessível para realizar a computação em toda a matriz de programação dinâmica para obter melhores resultados. Os valores de qualidade médios para o bloco são pré-computados de modo que cada entrada na matriz de programação dinâmica é calculada em um tempo constante.

III. AJUSTES

Os métodos descritos anteriormente foram incorporados na nova versão do programa CAP3. A nova versão foi desenvolvido, refinado e ajustada baseados em conjuntos de dados reais e conjuntos de dados artificiais gerados por GenFrag[2] (Engle e Burks 1993).

IV. COMPARAÇÃO ENTRE CAP3 E PHRAP

Este artigo realiza a comparação entre as metodologias CAP3 e PHRAP focando em dois tipos de dados: de passa-baixa com restrições *forward-reverse* e de várias coberturas sem restrições *forward-reverse*. A tabela 4 do artigo demonstra resultados de performance entre as técnicas. No entanto, verificando em detalhes: (a) valores provindo do CAP3 quase sempre menores para o comprimento dos *contigs*; (b) valores de erros internos menores comparado ao PHRAP; e (c) com quantidade de *gaps* consideravelmente maiores.

[2]http://genfrag.rubyforge.org/

REFERÊNCIAS

[1] X. Huang and A. Madan. Cap3: A dna sequence assembly program. pages 868–877, 1999.