Influência de Regiões Anômalas em Genomas de Referência Utilizados para Montagem Guiada

Dissertação de Mestrado
Programa de Pós-Graduação em Ciência da Computação

Bárbara Purkott Cezar

Faculdade de Computação
Universidade Federal de Mato Grosso do Sul

Orientadora: Profa. Luciana Montera Cheung

29 de outubro de 2015

Agradecimentos

Primeiramente à Deus, pois acredito que cada acontecimento é guiado e abençoado por Ele, assim como a realização desse sonho.

Aos meus pais, pelo apoio financeiro mas principalmente pelo carinho e atenção constantes.

À minha orientadora Luciana Montera, pela paciência e ensinamentos. Graças a ela foi possível concluir esse projeto.

Ao professora Flábio, por ter disponibilizado as cepas para serem estudados no desenvolvimento do projeto.

Aos meus companheiros de apartamento, Marcelo, Alex e Guilherme, pelo companheirismo, pela amizade e principalmente pelos momentos difíceis em que eles estavam sempre próximos para me ajudar e incentivar.

Aos demais amigos, tanto os próximos como os que estão longe, pelos momentos de alegria quando houve desânimo.

À CAPES pelo apoio financeiro.

Resumo

Cepas do complexo *Mycobaterium bovis* foram analisadas segundo uma estratégia desenvolvida nesse projeto, a qual analisa o quanto características biológicas, como por exemplo, transferência horizontal de genes, regiõs de alta frequência CG e regiões variáveis ou repetitivas, influenciam no resultado de um mapeamento de *reads*. Este trabalho investiga a existência de regiões que potencialmente representam tais características, chamadas regiões anômolas, e sua influência nos resultados deste mapeamento objetivando estabelecer uma relação entre elas e regiões não mapeadas. O agente *M. bovis* analisado é o causador da tuberculose em bovinos e outros mamíferos, incluindo os seres humanos e é uma doença de relevância econômica no contexto da pecuária, já que afeta diretamente a produtividade dos animais. Os resultados mostraram forte relação entre regiões não mapeadas, ou pouco mapeadas pelos *reads* e a potencialidade destas serem regiões anômalas, segundo análise feita por uma ferramenta existente para esta finalidade.

Palavras-Chave: Regiões anômalas, *Mycobacterium bovis*, Mapeamento de *reads*.

Abstract

Mycobaterium bovis strains were analyzed following a strategy proposed in this work which analyses the impact of biological features such as horizontal transfer genes, high GC content and repetitive regions in the mapping reads context. This work investigates the existence of alien regions that are likely to represent such features and their influence in the mapping reads context aiming the establishment of a relation between them and low coverage regions. *M. bovis* is the agent that causes tuberculosis in cattle and other mammals including humans. Tuberculosis is a relevant disease in the livestock economy context since it affects directly the productivity of the animals. The results shown a strong relation between low coverage regions and potentially alien regions that were predicted by a tool developed for this purpose.

Keywords: Alien regions, *Mycobacterium bovis*, reads mapping.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O sequenciamento de moléculas de DNA consiste na leitura de amostras de DNA e na determinação da sequência de nucleotídeos que as constituem. Os nucleotídeos podem ser Adenina, Citosina, Guanina e Timina, representados pelos caracteres A, C, G, T, respectivamente. Frederick Sanger foi um dos primeiros a propor um método de sequenciamento de DNA, na década de 1980 [67].

Em 1986 o primeiro sequenciador automático de DNA foi lançado, o ABI 370, e em 1998 o primeiro sequenciador de eletroforese capilar, o ABI 3700 [80], o qual foi criado a partir do desenvolvimento de duas técnicas. A primeira é a eletroforese capilar, que acontecia em tubos com espessura de um cabelo humano, por onde o DNA seria guiado por uma corrente elétrica. A segunda é a marcação de didesoxinucleotídeos utilizados para sequenciamento do DNA com as moléculas fluorescentes, as quais permitiam que cada base fosse marcada com um diferente fluorocromo capaz de emitir luz em um diferente comprimento de onda, se excitado por um laser. Tal luz, lida por um detector, identificava o nucleotídeo que passava em diferentes momentos da eletroforese [50].

A união dessas duas técnicas no ABI 3700 permitiu o rápido desenvolvimento da produção de sequências de DNA, pois o equipamento apresentava 96 colunas (ou capilares para a eletroforese), que permitiam o sequenciamento de cerca de 550 bases em cada coluna, possibilitando o sequenciamento de até 1 milhão de pares de bases por dia [50].

Por volta de 2005 começaram a surgir técnicas de sequenciamento baseadas em métodos diferentes do proposto por Sanger. Deu-se então início a uma nova era, na qual os sequenciadores desenvolvidos foram chamados de sequenciadores de nova geração (*Next Generation Sequencing*, em inglês), ou simplemente NGS. As tecnologias NGS aumentaram a velocidade e a capacidade de sequenciamento e, como resultado, reduziram drasticamente os custos globais de sequenciamento [90].

Diversas tecnologias atuam com sucesso hoje no mercado, dentre elas, Roche GS-FLX 454 Genome Sequencer [13], Illumina/Solexa Genome Analyzer [6] e ABI SOLiD [1]. Apesar de diferenciarem consideravelmente entre si, todos os sequenciadores de NGS se baseiam no processamento paralelo massivo de fragmentos de DNA ou RNA. Enquanto um sequenciador de eletroforese, que utiliza tecnologia Sanger, processa no máximo 96

fragmentos de DNA por vez, os sequenciadores de nova geração podem ler até bilhões de fragmentos ao mesmo tempo [80].

Dados publicados pelo *National Human Genome Research Institute* dos Estados Unidos, apresentados pelo gráfico da Figura 1.1, mostram que em 2004 houve uma queda no custo por megabase de DNA sequenciado, sendo que em 2008 tal redução foi mais significativa. Nota-se também que a evolução das técnicas de sequenciamento foi muito mais acelerada em comparação à capacidade de processamento dos processadores, de acordo com a Lei de Moore [59]. Assim, mais megabases de DNA sequenciado são geradas por menor custo. Analisando a comparação da capacidade de processamento com a quantidade de dados gerados que devem ser processados, a tarefa de lidar com esses dados se tornou mais intensa, pois esses crescimentos ocorreram de forma desproporcional.

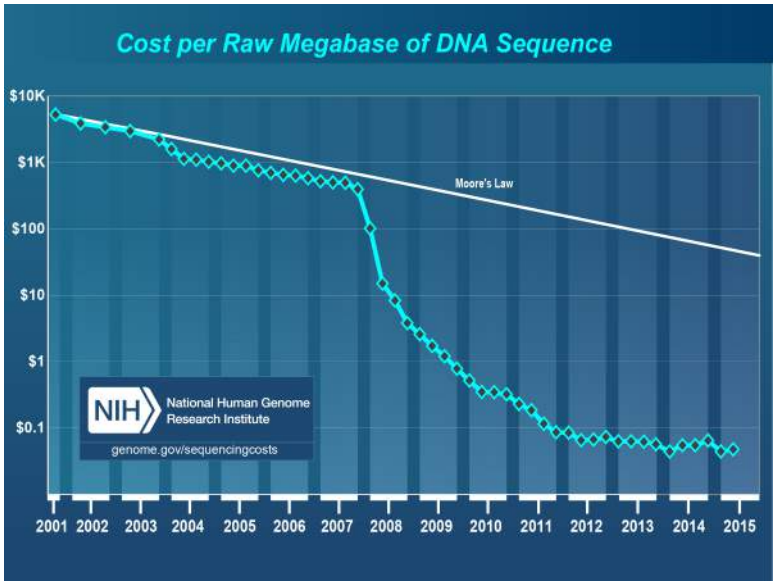


Figura 1.1: Custo de sequenciamento por megabase de DNA ao longo dos anos. Fonte: http://www.genome.gov/sequencingcosts/ [10]. Acessado em 15-09-2015.

O sequenciamento de DNA possui diversas aplicações, já que este permite diversas análises de organismos através do computador. Essas análises podem apresentar diversos objetivos, como por exemplo, estudar regiões codificantes e/ou genes presentes nos organismos sequenciados, capturar informações de transcriptomas em situações em que o RNA é sequenciado, ao invés do DNA, e também encontrar possíveis diferenças entre um organismo (referência) comparado a outro organismo de interesse.

Como resultado do sequenciamento de DNA, tem-se milhares de pequenos fragmentos do DNA, chamados *reads* do organismo de interesse que precisam ser *organizados* para que se obtenha a sequência original. Tal organização dos *reads* ocorre por meio de processos denominados Montagem de *Reads* ou Mapeamento de *Reads*. Para o mapeamento, utiliza-se uma sequência de referência que deve ser muito próxima evolutivamente da sequência que se deseja montar, enquanto que tal referência não existe no processo de montagem. Neste último caso temos o processo de montagem *de novo* e no primeiro, montagem

guiada.

Com o objetivo de explicar por que algumas regiões do genoma de referência são pouco mapeadas no processo de montagem guiada, neste trabalho propomos a avaliação da sequência utilizada como referência pela ferramenta Alien Hunter, que é um software capaz de identificar regiões do genoma de composição *anômala*. Tais anômalias podem ser devido à alta concentração de bases CG, regiões repetitivas ou variantes além de representarem regiões onde genes foram adquiridos de outros organismos por eventos de transferência lateral. Seguido da análise do genoma de referência, é feito a análise das regiões de baixa cobertura obtidas do mapeamento a fim de identificar a correlação existente entre a baixa cobertura e regiões anômalas no genoma de referência.

Como caso de estudo, 21 cepas de *Mycobacterium bovis*, foram utilizadas, além do genoma de referência AF2122/97. As cepas são o resultado de um projeto de genômica de *Mycobacterium bovis* que envolve o Instituto Nacional de Tecnologia Agropecuária da Argentina (INTA), o Departamento de Agricultura dos Estados Unidos (USDA), o Ministério da Agricultura Pecuária e Abastecimento do Brasil (MAPA), o Instituto Biológico de São Paulo (IB) e a Universidade Federal de Mato Grosso do Sul (UFMS).

O restante do texto está organizado como segue. No Capítulo 2 são apresentados os conceitos básicos computacionais incluindo estratégidas de montagem e mapeamento de *reads* e a descrição do software Alien Hunter utilizado para fazer a análise de regiões anômalas em genoma. A proposta de estratégia de montagem de genoma guiada por mapeamento em referência considerando regiões anômalas é brevemente descrita no Capítulo 3. O capítulo 4 descreve a avaliação das regiões anômalas em um genoma do complexo *M. bovis* e os mapeamentos de *reads* de cepas do mesmo complexo na referência. No Capítulo 5 é apresentado a montagem *de novo* dos *reads* não mapeados. As conclusões e propostas de trabalhos futuros são apresentados no Capítulo 6.

Capítulo 2

Conceitos Básicos

Este capítulo apresenta conceitos necessários para a compreensão dos problemas de montagem e mapeamento de *reads* tais como alinhamento, sobreposição e consenso. As principais estratégias de montagem e mapeamento também são apresentadas.

2.1 Alinhamento e sobreposição de *reads*

Para que os processos de montagem e mapeamento de *reads* sejam descritos, faz-se necessário definições prévias dos termos alinhamento, *match*, *mismatch*, *gap*, *score*, sobreposição e consenso.

Um alinhamento de duas sequências s e t construídas sobre um alfabeto $\sum$, transforma s em s' e t em t', sendo s' e t' construídas sobre o alfabeto $\sum' = \sum \cup$ '-', onde '-' é chamado de espaço. O tamanho de s' e t', denotado por $|s'|$ e $|t'|$, são iguais, portanto é possível avaliar a correspondência entre cada coluna do alinhamento, e assim identificar as igualdades, bem como as diferenças entre elas. As colunas do alinhamento que apresentam caracteres idênticos são ditas *matches*; as colunas que apresentam caracteres diferentes são ditas *mismatches*; as colunas que apresentam um caractere com um ífem são ditas espaços. As ocorrências de diversos espaços consecutivas são chamados de *gaps*. A Figura 2.1 apresenta um possível alinhamento de as sequências s = ACGTACCTAGCTGCACG e t = TGGTATCTGCTAGCACG, onde ocorrem 13 *matches*, 3 *mismatchs* e 2 espaços.

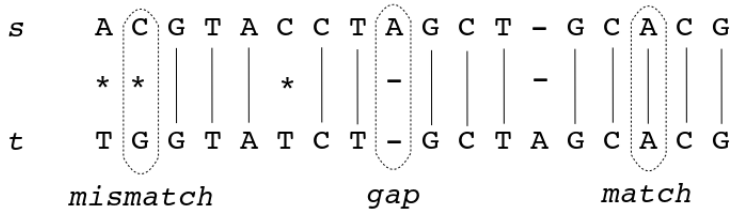


Figura 2.1: Representação de um alinhamento de as sequências s e t.

Diversos alinhamentos são possíveis entre duas sequências. Entretanto, os mais infor-

mativos são aqueles que evidenciam as igualdades entre elas. Desta forma, a fim de se avaliar, ou mesmo comparar alinhamentos, um esquema de pontuação é definido para que uma pontuação, chamada de *score* seja associado a cada alinhamento. Assim dada uma função de pontuação f qualquer, a pontuação ou *score* de um alinhamento $A(s,t)$ entre s e t, denotada por $\text{Score}_f(s,\, t)$, é definida como: $\text{Score}_f(s,\, t) = \sum_{i=1}^{|s'|} f(s'_i, t'_i)$ onde s'_i, t'_i correspondem o i-ésimo caractere de s', t', respectivamente.

Considerando uma função de pontuação f que atribui valor 1 para *matches*, -1 para *mismatches* e -2 para espaços, podemos calcular o *score* do alinhamento mostrado na Figura 2.1, cujo valor é $\text{Score}_f(s,\, t)$ = (-1-1+1+1+1-1+1+1-2+1+1+1-2+1+1+1+1+1) = 6. Para o cálculo do *score* de sobreposições, os espaços das extremidades não são considerados, portanto, a sobreposição mostrada na Figura 2.2 é $\text{Score}_f(s,\, t)$ = (0+0+0+0+0+0+0+0+0+1+1+1+1+1+1+1+1+1+1+0+0+0+0+0+0+0+0+0) = 11.

Para a definição de sobreposição é necessária a definição dos termos prefixo e sufixo. O prefixo de s é qualquer subcadeia de s tal que, $s[1..j]$ onde $0 \leq j \leq |s|$. Admite-se $j = 0$ e defina $s[1..0]$ como sendo cadeia vazia. Note que t é um prefixo de s, se e somente se, existe uma outra cadeia u tal que $s = tu$ [70]. De forma análoga, o sufixo de s é uma subcadeia tal que $s[i..|s|]$ para qualquer i sendo $1 \leq i \leq |s| + 1$. Admite-se $i = |s| + 1$ no caso em que $s[|s| + 1..|s|]$ denota cadeia vazia. Uma cadeia t é um sufixo de s, se e somente se, existe u tal que $s = ut$ [70].

A sobreposição entre dois fragmentos, ou duas sequências s e t, é dada pelo alinhamento de um prefixo de s e um sufixo de t, ou entre um sufixo de s e um prefixo de t, como exemplificado na Figura 2.2, onde s = CCGATATGCGCTAATGCTAG e t = GCTAATGGTAGGGCTAGCTAC.

```
s   C C G A T A T G C G C T A A T G C T A G - - - - - - - - - -

t   - - - - - - - - - - G C T A A T G C T A G G G C T A G C T A C
```

Figura 2.2: Sobreposição entre um sufixo de s e um prefixo de t.

Pode-se definir, de forma análoga à definição de alinhamento de duas sequências, o alinhamento de três ou mais sequências, neste caso, tem-se um alinhamento múltiplo de sequências. Todo alinhamento de duas ou mais sequências pode ser representado por uma sequência única nomeada consenso, obtida pela simples votação do caractere mais frequente em cada coluna do alinhamento múltiplo. Tome como exemplo o alinhamento mostrado na Figura 2.3 entre as sequências s_1 = ACTGG, s_2 = GACGCTG, s_3 = CGCT e s_4 = GACTGG, cujo consenso c = GACGCTGG foi obtido pela identificação de caractere mais frequente em cada uma das colunas do alinhamento múltiplo.

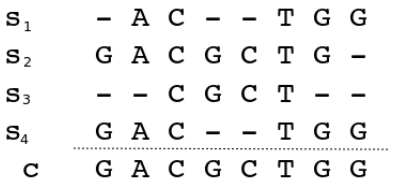


Figura 2.3: Alinhamento múltiplo entre quatro sequências e o consenso correspondente.

2.2 Montagem de *reads* de DNA

Como dito no capítulo anterior, para se determinar a sequência completa do organismo sequenciado, ou mesmo do conjunto de RNAs sendo expressos em determinado momento, é necessária a organização ou a determinação da ordem dos *reads*. Uma maneira de se determinar a ordem deles é através do processo de montagem.

Pode-se definir a montagem da seguinte maneira: seja G um genoma e R o conjunto de *reads* gerados pelo sequenciamento de G. Tem-se então $R = \{r_1, r_2, ..., r_n\}$ formado por *reads* construídos sobre o alfabeto $\sum = \{A, C, G, T\}$. A montagem em si refere-se ao alinhamento de *reads* que se sobrepõem com objetivo de construir uma sequência consenso, a qual define a ordem dos elementos de R.

Dessa forma, dado o conjunto de *reads*, a montagem objetiva alinhá-los a fim de reconstruir uma sequência alvo. A Figura 2.4 exemplifica o processo de montagem de um conjunto de *reads* resultantes do sequenciamento de um genoma G.

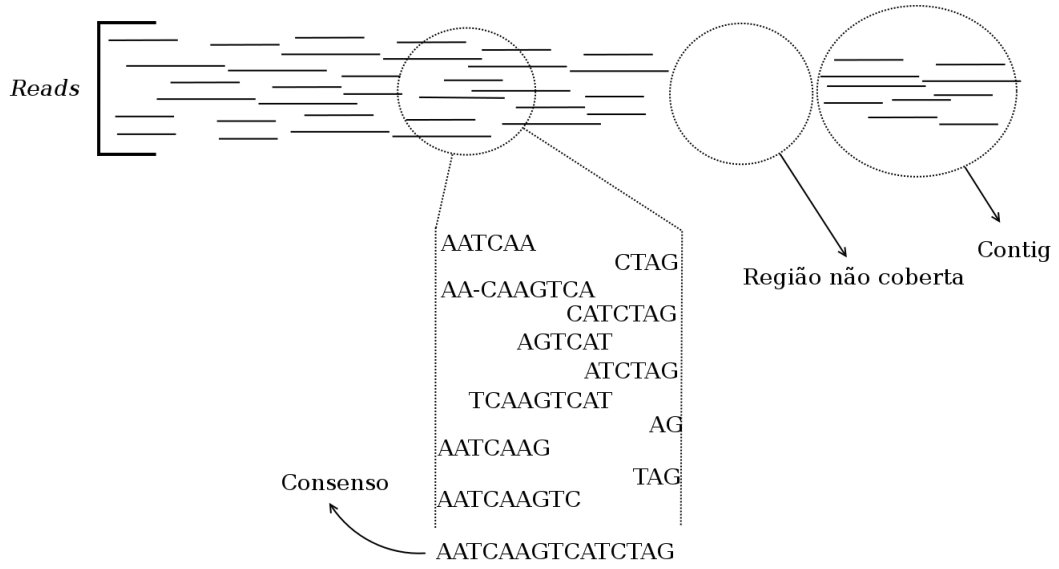


Figura 2.4: Ilustração do processo de montagem de *reads* do genoma G.

A montagem agrupa *reads* em *contigs* e *contigs* em *scaffolds*. Os *contigs* são o resultado do alinhamento de múltiplas sequências de *reads* formando uma única sequência, que

também chamamos de consenso. Os *scaffolds* definem a ordem e a orientação dos *contigs* [57].

Descreveremos, de maneira breve, três estratégias para o problema da montagem de *reads*. São elas Grafo de Bruijn, Grafo de Sobreposição e Algoritmos Gulosos.

2.2.1 Grafo de Bruijn

O grafo de Bruijn é construído tendo como rótulos das arestas todos os distintos k-*mers* obtidos dos *reads* do conjunto de *reads* a serem remontados e, como vértices, todos os prefixos e sufixos de tamanhos k-1, ou seja, para um 3-*mer* igual a CGT, existe um vértice de rótulo CG e outro de rótulo GT. Uma aresta é inserida no grafo entre os vértives v_i e v_j se existe uma sobreposição entre um sufixo do vértice v_i e um prefixo do vertice v_j. Tome como exemplo os reads $CGTGCAA$, $ATGGCGT$, $CAATGGC$, $GGCGTGC$ e $TGCAATG$ e todos os possíveis e distintos 3-*mers* CGT, GTG, TGC, GCA, CAA, ATG, TGG, GGC, GCG, AAT. Para cada 3-*mer*, um vértice rotulado com seu prefixo de tamanho 2 é inserido no grafo (caso ainda não exista tal vértice) bem como um vértice rotulado com seu sufixo de tamano 2 (caso este vértice não exista no grafo). Uma aresta ligando este vértices deve existir no grafo. A Figura 2.5 apresenta grafo gerado para os prefixos e sufixos do conjunto de 3-mers apresentado.

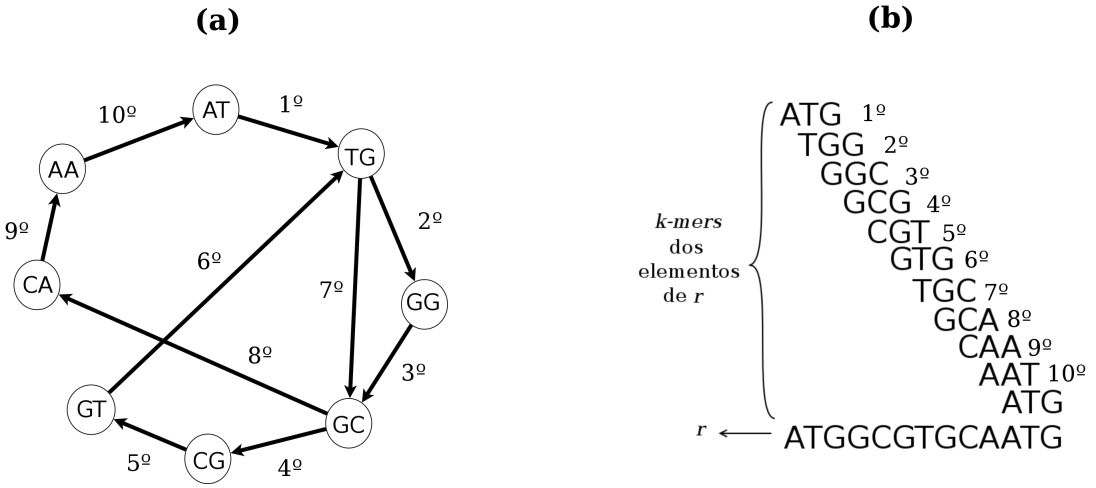


Figura 2.5: Grafo de Bruijn construído para os prefixos e sufixos de tamanho 2 presentes no 3-*mers* dos *reads* $CGTGCAA$, $ATGGCGT$, $CAATGGC$, $GGCGTGC$ e $TGCAATG$.

Dada a construção do grafo como descrita, pode-se rotular as arestas como os k-*mers* obtidos do conjunto de *reads*. A determinação de um caminho Euleriano, o qual visita cada aresta apenas uma vez [17], no grafo de Bruijn, ou seja, representa uma possível montagem do conjunto de *reads* [19]. Exemplos de *softwares* que utilizam dessa estratégia são Velvet [89] e ABySS [72].

2.2.2 Grafo de Sobreposição

A estratégia que utiliza o Grafo de Sobreposição baseia-se na utilização de um grafo que contém todos os *reads* bem como as sobreposições entre eles. A aplicação dessas estruturas para o alinhamento de *reads* é utilizada da seguinte maneira:

- Identificação das sobreposições: nessa etapa ocorre o alinhamento de todos os *reads*. O resultado desses alinhamentos gera um grafo de sobreposição G, definido da seguinte maneira: dado um conjunto de *reads* R, cria-se o grafo de sobreposição R_G cujo conjunto de vértices v_1, v_2, ..., v_n representa os *reads* r_1, r_2, ..., r_n em R. As arestas de G possuem pesos que correspondem aos *scores* obtidos pela sobreposição dos *reads* que estão ligados em suas extremidades, tal que, dado um vértice x, a aresta dirigida dele para o vértice y indica a sobreposição do sufixo de x com o prefixo de y.

  O objetivo é encontrar os pares de bases que apresentam melhor casamento entre os *reads*. Nesse processo os complementos reversos dos *reads* também são considerados. Para a geração dessas arestas há um valor mínimo onde somente *scores* acima dele permitem criação de arestas novas, garantindo assim a qualidade dos alinhamentos. A Figura 2.6 (a) ilustra um exemplo de sobreposição para o conjunto com os *reads*: r_1 = GATCACGAA, r_2 = CGAAAGCAC, r_3 = AGATAGCGAA, r_4 = CGATTTAGAT e r_5 = AGATTACGAT.

- Manipulação do Grafo: etapa que consiste encontrar um ciclo hamiltoniano, ou seja, um ciclo que contém cada vértice do grafo exatamente uma vez [77], de maior peso H em G. A Figura 2.6 (b) ilustra o resultado dessa etapa para o grafo R_G que contém os *reads* r_1, r_2, r_3, r_4 e r_5.

- Alinhamento das sequências: geração do consenso obtido pelo caminho hamiltoniano H, por meio do alinhamento múltiplo dos *reads* representados por tal caminho. A Figura 2.6 (c) ilustra a execução dessa etapa.

Dentre os *softwares* que utilizam essa estratégia podemos citar Celera [60] e Arachne [16].

2.2.3 Algoritmos Gulosos

A técnica que utiliza a estratégia de algoritmos gulosos sempre faz a escolha que parece ser a melhor no momento em questão. Isto é, faz uma escolha localmente ótima na esperança de que essa escolha leve a uma solução globalmente ótima [49]. A montagem de *reads* baseada na estratégia gulosa pode ser feita da seguinte maneira: seja um grafo G, formado pelo conjunto de vértices V que representam os *reads*, e o conjunto de arestas A, rotuladas com o número das sobreposições dos vértices que ela interliga. Montadores baseados no uso de algoritmos gulosos sempre fazem a escolha com a maior contribuição imediata para resolver o problema de montagem de *reads*. Segue-se a mesma operação

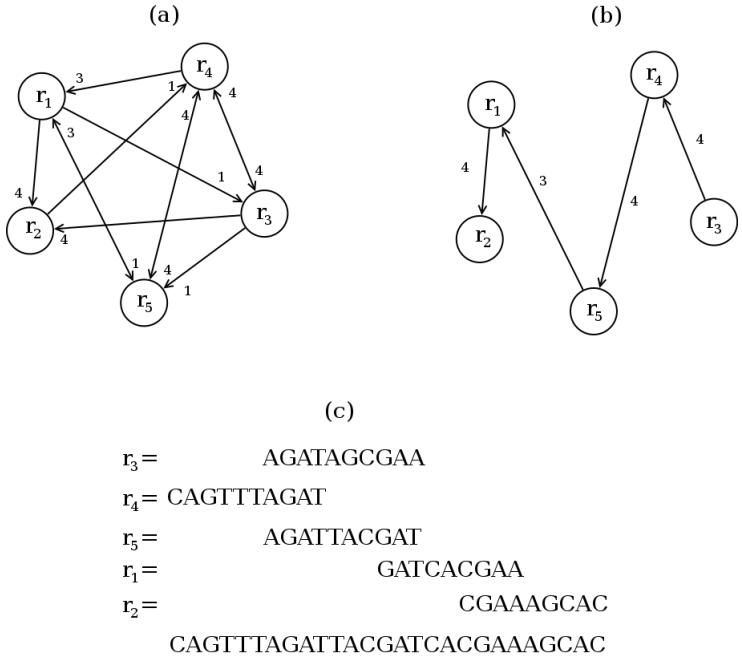


Figura 2.6: Ilustração dos passos a serem executados para construção do Grafo de Sobreposição. Modificado de [26].

básica: seja v um vértice de G e n arestas partindo de v, que formam o conjunto de arestas A_v. Dentre as arestas de A_v, será escolhida uma aresta a, tal que, a possui o maior peso dentre todas as arestas em A_v, ou seja, ele faz a escolha do vértice vizinho ligado à aresta que mais contribui no comprimento da sobreposição do vértice atual [22]. A técnica gulosa não é muito utilizada pois não considera a informação global dada pelos *reads*, não sendo a melhor escolha para certa aplicações das montagens. Os montadores que utilizam da técnica gulosa são adequados apenas para genomas pequenos.

O algoritmo 1 demonstra um exemplo utilizando a técnica gulosa para a montagem de *reads*, onde a entrada é um grafo G com todas as arestas que ligam os *reads*, e retorna como saída o consenso dado pela sobreposição entre eles.

Considere como exemplo o conjunto $R = \{ r_1, r_2, r_3, r_4, r_5 \}$ onde r_1 = ACTGAATGCA, r_2 = CACTAAACTG, r_3 = GCAGAGCATG, r_4 = ATGCATTGCA e r_5 = TGCAAACCAC. A Figura 2.7 demonstra a aplicação do algoritmo nesse exemplo. Na Figura 2.7 (a) é possível observar um grafo que contém todas as ligações entre os *reads* do conjunto R. A Figura 2.7 (b) mostra o caminho encontrado pela técnica gulosa, e a Figura 2.7 (c) mostra as sobreposições seguindo o caminho encontrado.

Algumas ferramentas que realizam a montagem por meio de algoritmos guloso: Cap3 [37], SSAKE [84], SHARCGS [20] e VCAKE [39].

Algorithm 1 Algoritmo Guloso

```
1: função GULOSO(G, S)        ▷ G é o grafo dado como entrada, S é o consenso gerado como saída
2:     i ← 1
3:     tam ← Tamanho do grafo G
4:     V[i] ← Escolha algum vértice em G
5:     Maior ← −∞
6:     enquanto i ≤ tam faça
7:         x ← V[i]
8:         enquanto não passar por todo vértice y adjacentes à x faça
9:             se Maior < ValorSobreposição (x,y) então
10:                Maior ← ValorSobreposição (x,y)
11:                V[i + 1] ← Vértice(G(y))
12:            fim se
13:        fim enquanto
14:        i ← i + 1
15:        Maior ← −∞
16:    fim enquanto
17:    Consenso(S,V)
18:    devolve (S)
19: fim função
```

2.3 Mapeamento de *reads* de DNA

Assim como o processo de montagem de *reads*, o mapeamento também objetiva determinar a sequência completa de um organismo sequenciado através da organização ou determinação da ordem de seus *reads*. Porém, diferentemente da montagem, o mapeamento usa técnicas que utilizam uma sequência como referência para basear-se em seus alinhamentos e sobreposições. O problema do mapeamento de *reads* de DNA em um genoma de referência pode ser definido da seguinte maneira:

Considere G um genoma e $R = \{ r_1, r_2, ..., r_n \}$ o conjunto de *reads* gerados pelo sequenciamento de G. O mapeamento em si refere-se ao alinhamento de *reads* com um genoma de referência, denomidado M, objetivando construir uma sequência consenso, a qual define a ordem dos elementos de R.

Tome como exemplo um *read* r = CGATTCGATGC. Temos $|r| = 11$ e $r[i]$, para $1 \leq i \leq |r|$, o caractere que ocupa a posição i do *read*. Por exemplo, $r[4]$ = T e $r[7]$ = G. Dado um genoma M, dito genoma de referência e um *read* r, denotamos por M_r a lista de posições iniciais em que r se alinha em M. Sendo M = ATGCCGATTCGATGCAGGACGATTTGATGCAGCCGATTCGATGCTC temos $M_r = \{5, 34\}$.

No exemplo dado foi considerado apenas os alinhamentos exatos entre o *read* e a referência, porém pode-se considerar também as ocorrências não exatas de um *read* r em G. Estas também serão inseridas na lista de ocorrências de r. Tais ocorrências podem estar relacionadas à marcadores moleculares, que são sequências de DNA que revelam

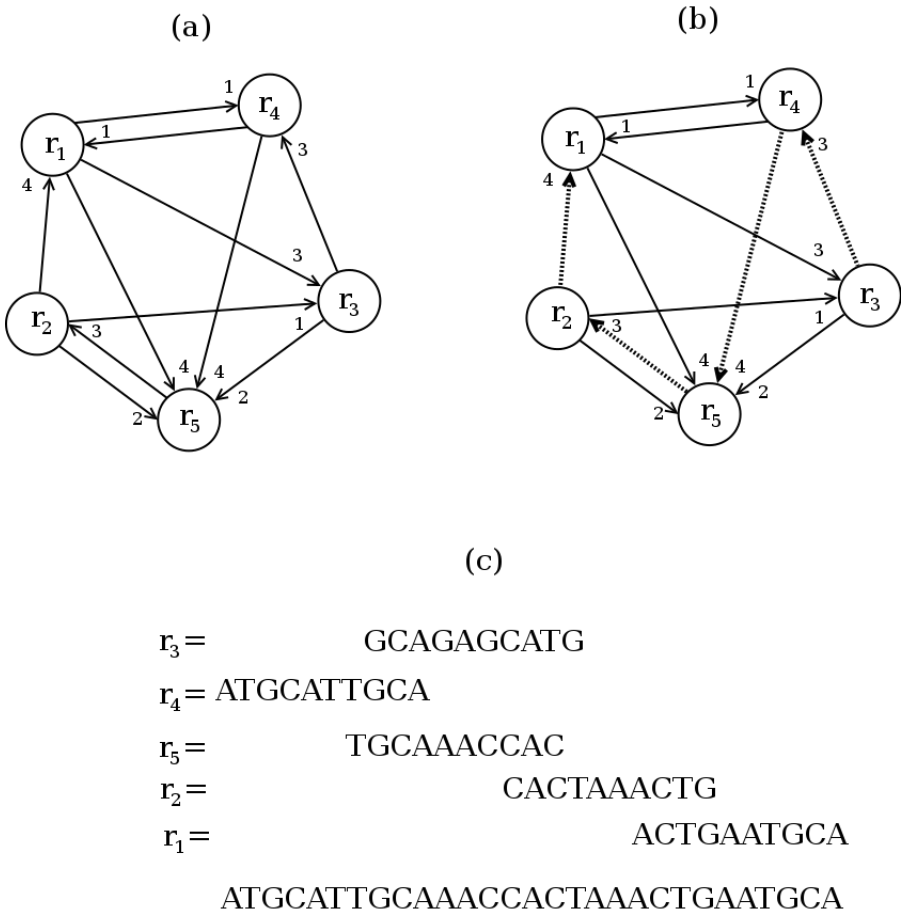


Figura 2.7: Ilustração dos passos a serem executados utilizando a estratégia de Algoritmos Gulosos. Modificado de [26].

polimorfismos entre indíviduos geneticamente relacionados. Entre os polimorfismos existentes, o que mais interessa aos nossos estudos são os polimorfismos de nucleotídeo único (*single nucleotide polymorphism* - SNP), os quais são a mudança de uma única base em uma sequência de DNA, e os *Indels* [83].

A Figura 2.8 exemplifica possíveis posições de mapeamento para um *read* em um genoma G e representa quatro situações a serem consideradas durante o mapeamento de *reads* de DNA: casamento exato, quando todas as bases conseguem ser alinhadas perfeitamente ao genoma de referência; deleção, quando bases do *read* estão ausentes na referência; inserção, onde bases na referência estão ausentes no *read*; e substituição ou SNP, quando alguma base do *read* é diferente da referência.

É fato que um *read* é representante de uma única região do genoma como resultado do sequenciamento. Entretanto, durante o mapeamento, mais de uma única ocorrência para um *read* pode ser encontrada no genoma de referência. Esse fato pode ocorrer devido à existência de diversas bases consecutivas de um mesmo tipo (*run*) encontradas na referência que possuem tamanho maior que o tamanho do *read*. Além disso, algumas com-

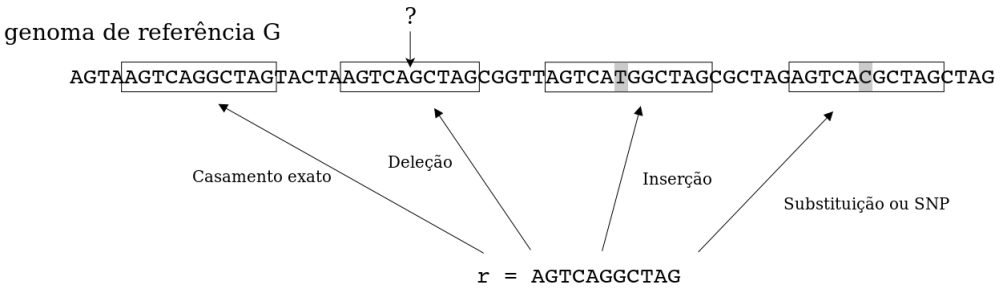


Figura 2.8: Ilustração de possíveis situações de mapeamentos de um *read* r em uma referência G.

binações específicas de bases (*repeats*), podem ocorrer em diversas localizações ao longo da referência. Por exemplo, para uma referência que apresenta um trecho "AAAAAA-AAAAAAAAAAAAAAAAAAAAAAAA", o *read* r_1= AAAAAA, será encontrado em mais de uma posição na referência, e nesse caso temos a caracterização de um *run*. No caso do trecho "AAACTCGATCATGCTTATCAACTCGGTAGCTAG"estar presente na referência, o *read* r_2 = AACTCG pode ser mapeado nas posições 2 e 20, o que caracteriza um *repeat*.

Dessa maneira algumas heurísticas são implementadas pelos softwares mapeadores para escolher uma dentre duas ou mais posições de mapeamento de um *read*, como por exemplo a escolha do casamento de maior *score*. Outra estratégia também utilizada por alguns softwares é o uso da informação sobre a qualidade das bases[1] de um *read* fornecido pelos sequenciadores.

Diversas ferramentas estão disponíveis para a execução do mapeamento de *reads*. Podemos diferenciá-las de acordo com a estratégia utilizada para o mapeamento. Dentre as principais estratégias estão a estratégia Ingênua, a estratégia baseada na construção de Tabela de *Hash*, a estratégia baseada na construção de Árvore de Sufixo e a estratégia de *Burrows-Wheeler*, as quais são descritas a seguir.

2.3.1 Estratégia Ingênua

Um algoritmo para comparação de duas sequências, por exemplo, um genoma G e um *read* r, utilizando a estratégia ingênua, ou força bruta, realiza a comparação de todas as posições de uma sequência com todas as posições da outra. A ideia da Estratégia Ingênua se baseia na verificação de todo deslocamento $d = 0, ..., n - m$ se $G[d + 1..d + n] = r[1...m]$. Dessa forma, compara-se caractere a caractere da subcadeia de G que contém m símbolos e começa na posição $d + 1$ e termina na posição $d + m$, com a sequência r. Caso $G[d+1..d+m] = r[1...m]$ então r ocorre na posição $d+1$ em G. Podemos visualizar no Algoritmo 2 como a estratégia funciona.

[1]A qualidade associada a cada base de um *read* indica uma estimativa de confiança em que a base presente no *read* é a mesma base presente no organismo.

Algorithm 2 Estratégia Ingênua

```
função FORÇA BRUTA(G, n, r, m)    ▷ G de tamanho n, onde será pesquisado r de tamanho m
    para i ← 0 até n-m faça
        j ← 0
        enquanto j < m e r[i] = G[i + j] faça
            j ← j + 1
        fim enquanto
        se j = m então
            devolve i
        fim se
    fim para
    devolve −1
fim função
```

Na situação onde não encontra-se a sequência r em G, configura o pior caso do algoritmo cuja complexidade é de O(mn), já que é necessário comparar todos os caracteres de r, ou seja $|r| = m$, em cada uma das $(n - m + 1)$ janelas de busca. Devido a sua complexidade e considerando tamanho do genoma que pode apresentar bilhões de bases e a quantidade de *reads* a serem mapeados, que também pode ultrapassar a quantidade de milhões, esta estratégia não é utilizada na prática.

2.3.2 Árvore de Sufixos

Uma árvore de sufixos, de acordo com [70], é uma estrutura que armazena todos os sufixos de uma sequência s. Formalmente, a árvore de sufixo para sequência $s = s_1$, s_2, ..., s_n é uma árvore enraizada T com $n + 1$ folhas tal que $n = |s|$. Essa árvora respeita as seguintes propriedades:

- As arestas de T são direcionadas da raiz para as folhas e cada aresta é rotulada por uma subcadeia de s.
- Cada nó interno possui ao menos dois nós filhos.
- Quaisquer arestas distintas que saem de um determinado nó estão rotuladas com subcadeias de prefixos distintas.
- Cada folha é rotulada com um inteiro i, tal que $1 \leq i \leq n$, que representa uma posição de s. A concatenação dos rótulos das arestas pertencentes ao caminho da raiz até a folha i resulta no sufixo de s, que começa na posição i.

A Figura 2.9 ilustra um exemplo de árvore de sufixo para G = GTATCTAGG\$. O símbolo \$ marca o fim da sequência e está presente sozinho em algumas arestas. Isso se deve ao fato de não ser possível representar todos os sufixos de uma sequência que tenha

dois ou mais sufixos que compartilham de um mesmo prefixo, evitando também problemas com sufixos vazios.

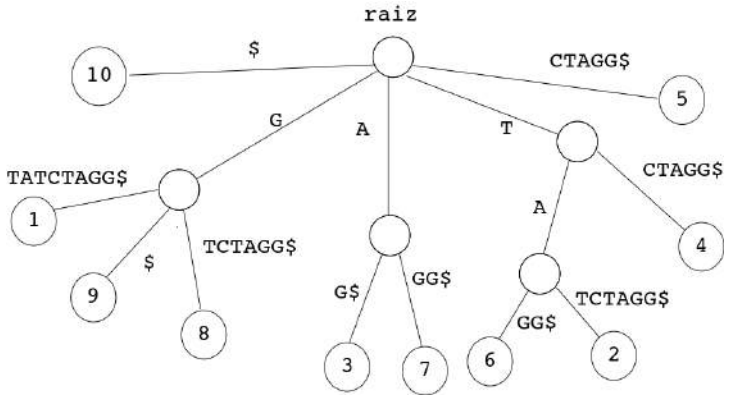


Figura 2.9: Árvore de sufixos para a sequência GTATCTAGG. Modificado de [70].

Na construção de uma árvore de sufixos utiliza-se a busca das ocorrências dos sufixos que estão sendo adicionados, portanto, se faz necessária a explicação de como ocorre a busca de uma sequência na árvore e em seguida como é construída uma árvore de sufixos.

Dada a árvore de sufixos T_G, construída a partir de G, a busca em uma árvore de sufixos determina todas as ocorrências de uma sequência s de tamanho m em um texto G de tamanho n. Qualquer sufixo $[i...m]$ de G é representado na árvore por um único caminho da raiz até a folha rotulada com i. Porém, qualquer subcadeia $G[i..j]$ de G, $i \leq j \leq m$, é um prefixo do sufixo $G[i..m]$, portanto, precisa rotular a parte inicial desse caminho único. Para descobrir a ocorrência de s em G, percorre-se esse caminho, comparando os símbolos de s com os símbolos que rotulam as arestas do caminho, até que s seja encontrado ou nenhum casamento seja mais possível. No caso em que s é encontrado, os nós folhas na subárvore que estão abaixo do nó onde a busca terminou, rotulam as posições onde s acontece em G. Caso s não seja encontrado, então s não ocorre em G.

A árvore de sufixos T_G, ilustrada na Figura 2.9 possui 10 nós folha numerados de 1 a 10, os quais correspondem ao sufixos s_i com $1 \leq i \leq 10$. Considere o exemplo de busca onde $s^{'} = TA$. A busca por $s^{'}$ em T_G termina em um nó interno entre as folhas que rotulam 2 e 6, portanto s ocorre nas posições 2 e 6. Observa-se também que a busca pode terminar em uma folha, caracterizando a situação em que s apresenta somente uma ocorrência em G, como por exemplo a busca por $s = CTAGG\$$.

Após a determinação da ocorrência de s em G, através da execução de um algoritmo de percurso em subárvore enraizada pelo vértice dado na busca, os rótulos das folhas são coletados e é obtido as ocorrências de s em G em tempo $O(m + k)$, onde k é o número de ocorrências.

Diversos algoritmos atuais são capazes de realizar a construção de árvores de sufixo com espaço de armazenamento e tempo de execução proporcionais ao tamanho de G, ou seja, n. Weiner [85] em 1973 apresentou o primeiro algoritmo linear no tamanho de G

para a construção de uma árvore de sufixos. Em seguida, no ano de 1976, McCreight [55] apresentou um algoritmo também linear mas que apresentava mais economia em termos de espaço. Em 1995, Ukkonen [79] apresentou uma versão também linear, com vantagens do algoritmo de McCreight, porém mais didático.

Em [29] é possível encontrar uma explicação detalhada sobre algoritmo de Ukkonen. Porém, apresentaremos aqui uma descrição mais simples da construção, que apresenta complexidade $O(n^2)$, objetivando apenas o entendimento da estrutura.

Na construção da árvore a estrutura se inicia como uma árvore T_G^1, a qual contém somente a raiz, uma aresta rotulada com sufixo $s[1..n]$ e uma folha rotulada com 1. Seja então T_G^i a árvore intermediária que possui todos os sufixos $G[j..n]$ de G, com $1 \leq j \leq i$.

Na etapa da construção de T_G^{i+1}, a cada passo o sufixo $s[i+1...n]$ é acrescentado à T_G^i. O processo realiza-se da seguinte forma, busca-se $G[i+1..n]$ em T_G^i. Seja C o caminho percorrido nessa busca. Note que C sempre termina antes que $G[i+1..n]$ seja integralmente encontrado, pois não há sufixos que sejam prefixos de outros sufixos maiores. Seja então $G[r]$ o útimo símbolo encontrado no caminho C, $i+1 \leq r < n$, ou seja, o sufixo $G[r+1..n]$, tem duas partes: a primeira, $G[i+1..r]$, foi encontrada em C e a segunda parte, $G[r+1..n]$, não foi encontrada. A segunda parte deve então ser inserida na árvore, logo após o símbolo de $G[r]$ em C. Há duas situações que podem acontecer: a primeira acontece quando logo após $G[r]$ em C, temos um nó interno v. Neste caso uma nova aresta (v, w) será inserida em T_G^i, rotulada com a sequência $G[r+1..n]$, e w é uma nova folha rotulada com $i+1$; a segunda acontece quando $G[r]$ e $G[r+1]$ estão na mesma aresta de T_G^i. Neste caso um novo nó interno v é inserido exatamente entre $G[r]$ e $G[r+1]$, uma nova folha w também é criada e rotulada com $i+1$, e uma nova aresta (v, w) é criada com o rótulo $G[r+1..n]$ [78]. A Figura 2.10 ilustra os passos da construção da árvore de sufixo para a sequência s = GTATCTAGG$, cuja árvore é apresentada na Figura 2.9.

Uma vez construída uma árvore de sufixo T para um genoma G, a tarefa de mapear *reads* em G, desconsiderando a ocorrência de *gaps* e *mismatches*, é simples e tem custo proporcional ao tamanho do *read*, por meio da busca na árvore de sufixo.

Uma das principais ferramentas que utilizam dessa estratégia é a MPScan [65], a qual faz a busca em um texto (genoma G) para um conjunto de palavras (*reads*) em um único computador, sem utilização de paralelização ou hardware especial.

2.3.3 *Hashing*

A tabela de *hash* é uma estrutura de dados para o armazenamento de informação utilizando a ideia de divisão de um universo U de dados. Esses dados são organizados em subconjuntos de forma mais facilmente gerenciável, visando permitir o armazenamento e a procura rápida de grande quantidade de dados. Essa técnica baseia-se em dois conceitos fundamentais:

- Tabela de *hash* (H), estrutura que permite o acesso aos subconjuntos. Para o armazenamento nessa estrutura, cria-se um critério para dividir o universo em sub-

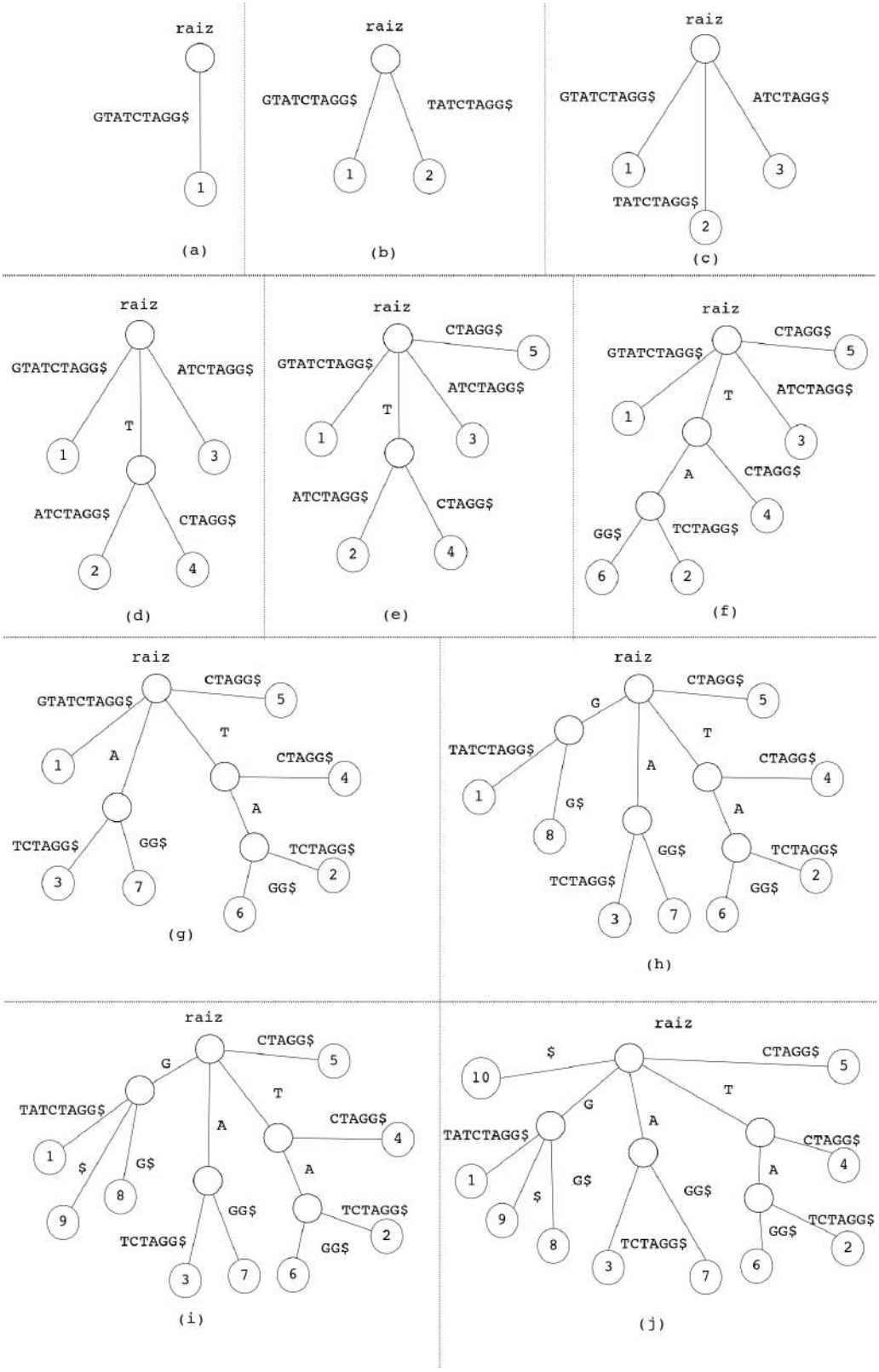


Figura 2.10: Passos da construção da árvore de sufixos para a cadeia GTATCTAGG$.

conjuntos com base em alguma qualidade de domínio dos elementos pertencentes a U. Esses subconjuntos são denominados chaves;

- Função de *hashing* (f), é uma função capaz de realizar um mapeamento entre valores de chaves e as entradas na tabela de *hash*. Após saber quantos subconjuntos são necessários, cria-se uma regra de cálculo que responde, dada uma chave, em qual subconjunto procurar pelos dados com esta chave, ou onde inserir este dado na estrutura caso seja um novo elemento.

Considere a tabela de *hash*, denotada por $H[0...m-1]$. Cada elemento pertencente a U com a chave k é armazenado na posição $f(k)$, ou seja, utiliza-se a função de *hash* f para calcular a posição da chave k. A função f mapeará o universo U de chaves para as posições da tabela H.

O resultado da função de *hash* f pode gerar o mesmo mapeamento para duas chaves diferentes, o que chamamos de colisão. Uma solução para o problema de colisão é o encadeamento, onde os elementos resultantes da função de *hash*, que mapeiam na mesma posição na tabela, são dispostos em uma lista encadeada, esquematizado na Figura 2.11.

De acordo com estudo feito em [33], os métodos para mapeamento de *reads* baseados em *hash* são divididos em dois tipos: o *hashing* de *read* e o *hashing* do genoma. Em geral, a idéia principal para ambos os tipos é a construção de uma tabela *hash* para subsequencias de *read*/genoma. A chave de cada entrada é uma subsequência, enquanto que o valor é uma lista de posições onde a subsequência pode ser encontrada. Algumas ferramentas baseadas em *hash* sobre o genoma de referência são: SOAP [53], Bfast [35], FastHASH [88], GSNAP [87], Novoalign [12], mrFAST [15], SRMapper [27]. Algumas ferramentas baseadas em *hash* sobre os *reads* são: MAQ [52], RMAP [74].

A estrutura de *hashing* apresenta palavras de comprimento k indexadas, que chamamos de *k-mers*. A tabela armazena a posição em que cada *k-mer* ocorre no genoma de referência nos casos das ferramentas Bfast, Novoalign, SOAP, mrFAST, ou no *read* em si, no caso da ferramenta MAQ, ou em algum casos ambos, como mrsFAST [30]. O alinhamento de *reads* inicia-se identificando as posições no genoma com correspondentes *k-mer* ou *k-mers* intervalados com espaços que permitem *mismatches* e continua o alinhamento de uma correspondência inexata.

A Figura 2.11 apresenta um exemplo da execução do mapeamento através da estratégia de *hashing*, onde a tabela armazena as posições do genoma de referência. Inicialmente, o genoma de referência G é quebrado através de sobreposições de 3-mers, e as suas respectivas posições no genoma são armazenadas (Figura 2.11 (a)). Em seguida, o *read* é quebrado em 3-mers (ver Figura 2.11 (b)) e cada *k-mer* do *read* é comparado com cada *k-mer* da referência, utilizando a tabela de *hashing*.

No exemplo da Figura 2.11 o primeiro *k-mer* do *read* ocorre nas posições 3 e 8 e o segundo *k-mer* nas posições 1 e 6. É fato que um *k-mer* fica em seguida do outro, portanto, para que o casamento do *read* aconteça, as ocorrências do segundo *k-mer* devem acontecer nas posições do primeiro adicionando um deslocamento à direita com o tamanho definido para os *k-mers*, nesse exemplo 3. Dessa forma, sabendo que as posições do primeiro *k-mer* são 3 e 8, as do segundo *k-mer* devem acontecer nas posições 6 e 11, para que

o *match* do *read r* seja verdadeiro. Porém o segundo *k-mer* apresenta posições 1 e 6, portanto haverá *match* somente na posição 6 para o *read r* (ver Figura 2.11 (c)). Após as comparações e classificações, somente as posições compatíveis, como o caso da posição 6 do segundo *k-mer* no exemplo, são as que resultam *match* (ver Figura 2.11 (d)).

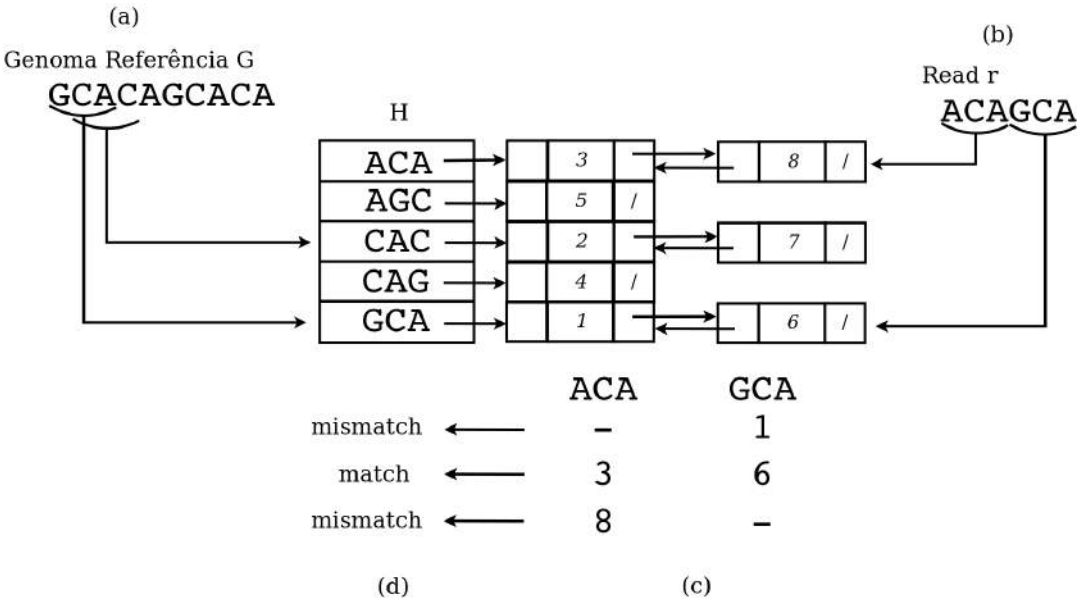


Figura 2.11: Exemplo mapeamento de *read r* sendo mapeado na referência G, através da estratégia de *hashing*. Adaptado de [69].

2.3.4 *Transformada de Burrows-Wheeler*

Baseada em uma estratégia originalmente desenvolvida para fins de compressão de dados, a Transformada de Burrows-Wheeler [18] ou apenas BWT (do inglês *Burrows-Weeler Transform*) foi aplicada ao problema de mapeamento de *reads*. Descreveremos a seguir como essa estratégia é aplicada no problema de mapeamento de *reads*.

Considere o alfabeto $\sum = \{A, C, G, T\}$ e o símbolo \$, tal que \$ $\notin \sum$ e \$ é lexicograficamente menor que todos os outros símbolos em $\sum$. Seja a sequência $x = x_1x_2x_3x_4...x_n\$$, tal que $x_i \in \sum$, com $1 \leq i \leq n$. Seja uma matriz quadrada M, de ordem $|n + 1|$, cujas linhas correspondem às permutações obtidas de x, pelo deslocamento à direita dos seus caracteres, um a um. Como exemplo tome x = AGCTACTAT\$. A primeira linha da matriz M correspondente deve representar a sequência \$AGCTACTAT. A segunda linha de M deve representar a sequência T\$AGCTACTA, a terceira linha de M deve representar a sequência AT\$AGCTACT, e assim por diante, como mostra a Figura 2.12.

Após construída, as linhas da matriz M são lexicograficamente ordenadas. A Figura 2.13 mostra o resultado da ordenação das linhas da matriz M mostrada na Figura 2.12.

Com uma matriz M ordenada, apenas a primeira coluna (F) e a última coluna (L) (ver Figura 2.13) são consideradas como estruturas de busca. Note que os caracteres

```
$ A G C T A C T A T
T $ A G C T A C T A
A T $ A G C T A C T
T A T $ A G C T A C
C T A T $ A G C T A
A C T A T $ A G C T
T A C T A T $ A G C
C T A C T A T $ A G
G C T A C T A T $ A
A G C T A C T A T $
```

Figura 2.12: Matriz M que armazena todas as permutações cíclicas da sequência AGCTACTAT$.

```
F                   L
$ A G C T A C T A   T
A C T A T $ A G C   T
A G C T A C T A T   $
A T $ A G C T A C   T
C T A C T A T $ A   G
C T A T $ A G C T   A
G C T A C T A T $   A
T $ A G C T A C T   A
T A C T A T $ A G   C
T A T $ A G C T A   C
```

Figura 2.13: Matriz M ordenada lexicograficamente com indicação das colunas F e L.

da coluna L e F representam permutações da sequência x, visto que todos os caracteres de x aparecem uma única vez na primeira coluna (F) e na última coluna (L) de cada deslocamento à direita de x. Assim, a primeira coluna pode ser obtida pela ordenação lexicográfica da última. Diante dessa característica das colunas F e L, obtém-se o Lema 1 definido em [25] por Ferragina e Manzini, apresentado a seguir:

Lema 1 (*Last-First Mapping*) *Seja M a matriz cujas linhas armazenam todas as permutações cíclicas de x, ordenados lexicograficamente, e seja L_i o caractere na última coluna da linha i e seja F_i o primeiro caractere nessa mesma linha. Então:*

1. *Na linha i de M, L_i precede F_i no texto original, ou seja, $x = \ldots\ L_i\ F_i\ \ldots$*
2. *Seja c um caractere de x, a j-ésima ocorrência de c em L corresponde ao mesmo caractere de texto que a j-ésima ocorrência de c em F.*

Antes da descrição da utilização de F e L como estruturas de busca de subsequência de x (lembre-se que $|x| = n + 1$, já que o caractere \$ é adicionado ao final de x), é necessária a definição de *rank* para os elementos das colunas L e F, como segue: seja L = L_1 L_2...L_{n+1} e F = F_1 F_2...F_{n+1}. Para todo símbolo $\ell \in \sum$ em L verifica-se sua quantidade de ocorrências até sua posição dentro de L, ou seja, para o caractere que está na posição L_i, com $1 \leq i \leq n + 1$, conta-se quantos caracteres iguais a ele aparecem em L_1, L_2, ..., L_i. Essa quantidade de vezes é definida como $rank(\ell, L, j)$ para $1 \leq j \leq n+1$. O mesmo se aplica para F. A Tabela 2.1 apresenta os valores dos *ranks* para as colunas L e F da Figura 2.13.

Tabela 2.1: Valores dos *ranks* para os vetores L e F.

rank	F	L	*rank*
rank(\$, F, 1) = 1	\$	T	rank(T, L, 1) = 1
rank(A, F, 2) = 1	A	T	rank(T, L, 2) = 2
rank(A, F, 3) = 2	A	\$	rank(\$, L, 3) = 1
rank(A, F, 4) = 3	A	T	rank(T, L, 4) = 3
rank(C, F, 5) = 1	C	G	rank(G, L, 5) = 1
rank(C, F, 6) = 2	C	A	rank(A, L, 6) = 1
rank(G, F, 7) = 1	G	A	rank(A, L, 7) = 2
rank(T, F, 8) = 1	T	A	rank(A, L, 8) = 3
rank(T, F, 9) = 2	T	C	rank(C, L, 9) = 1
rank(T, F, 10) = 3	T	C	rank(C, L, 10) = 2

Os caracteres de $x'[i]$, para $1 \leq i \leq 3$, são buscados nas colunas F e L, da direita para esquerda e aos pares, como descrito no procedimento a seguir.

- Passo 1: Inicialização de $i = |x'|$;
- Passo 2: Busque as posições em F onde $x'[i]$ aparecem. Para as m possíveis ocorrências de $x'[i]$ em F, as defina como k_1, k_2, ..., k_m, formando a lista k.
- Passo 3: Para todas as m posições k_1, k_2, ..., k_m encontradas no passo anterior, verifique se $L[k_j] = x'[i-1]$, tal que $1 \leq j \leq m$. Quando houver a igualdade, defina as novas posições encontradas como l_1, l_2, ..., l_z, tal que z é o número total de posições encontradas, formando a lista l.
- Passo 4: $i = i - 1$
- Passo 5: Apague as posições da lista k. Encontre todas as posições em F que possuem o mesmo *rank* das posições l_1, l_2, ..., l_z encontradas no Passo 3, ou seja, $rank(x'[i], L, l_j) = rank(x'[i], F, w)$ tal que tal que $1 \leq j \leq z$ e $1 \leq w \leq n$.
- Passo 6: Nesse momento, se $i = 1$, todas as posições encontradas no Passo 5 serão as devidas ocorrências de x' na matriz M. Se $i > 1$, então para todas as posições encontradas no Passo 5, redefina-as como $k_1, k_2, ..., k_m$, tal que m se tornará número de posições encontradas no Passo 5 e volte ao Passo 3.

A Figura 2.14 ilustra a execução dos passos descritos anteriormente para x = AGCTACTAT e $x^{'} = TAC$. É importante observar que os passos descritos anteriormente determinam a posição inicial do alinhamento de $x^{'}$ na coluna F. Portanto, para encontrar a posição do alinhamento de $x^{'}$ em x, deve-se utilizar uma estrutura que possa armazenar cada caractere em F juntamente com seu correspondente em x.

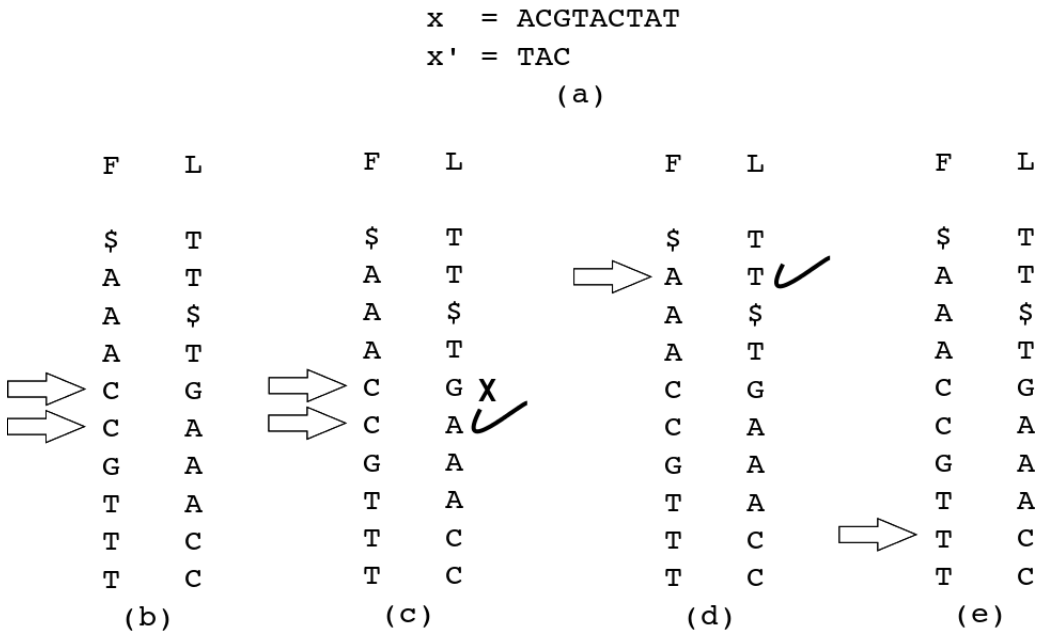


Figura 2.14: Exemplo de execução da estratégia Burrows-Wheeler Transform.

2.4 Transferência horizontal de genes

Transferência horizontal de genes, ou simplesmente HGT, do inglês *Horizontal Gene Transfering*, pode ser definida como a transferência de regiões genomas de DNA entre diferentes espécies, sem uso de mecanismos sexuais, mitose ou meiose, ou seja, transmissão do material genético entre organismos sem o uso de mecanismos de herança genéticas (transmissão do tipo pai para filho) [66]. Porém, uma vez que os fragmentos genômicos são incorporados no genoma, eles podem ser transferidos por mecanismos de herança genética, gerando diversidade de espécies [42]. Eventos de HGTs são comuns em bactérias e *Archaeas* e são considerados de grande importância para a evolução desses reinos [44, 86, 86] [86] [42].

Considerada uma importante ferramenta na adaptação de procariotos a um nicho específico, eventos de HGTs possibilitam a aquisição de um conjunto gênico já preparado e melhorado capaz de aumentar a adaptabilidade de organismos [47]. Pode-se citar as ilhas genômicas, uma das mais fortes evidências da HGT: os mesmos genes em distintos organismos que apresentam a mesma função em diferentes contextos ecológicos. As ilhas genômicas são classificadas de acordo com as atividades desenvolvidas por um conjunto

gênico adquirido, sendo as mais conhecidas as ilhas de simbiose, as ilhas de resistência a antibióticos, as ilhas metabólicas e as ilhas de patogenicidade [34].

Em eucariotos, HGTs são considerados raros, entretanto, um número cada vez maior de estudos reportam a ocorrência deste tipo de evento entre esses tipo de organismos. Por isso, acredita-se que os HGTs também são mecanismos importantes no processo de evolução dos eucariotos [38]. Em [68], mais de 200 casos de HGTs em eucariotos foram documentados, uma revisão sobre os critérios para a detecção de transferência horizontal foi feita e exemplos recentes do fenômeno foram examinados. Nesse mesmo trabalho, os autores afirmam que eventos de HGTs em eucariotos tem potencial para moldar o conteúdo genômico de acordo com as interações ecológicas que as espécies apresentam, além de ter o poder de distorcer significativamente os padrões filogenéticos esperados do mecanismo de herança vertical.

Os mecanismos genéticos que atuam em eventos de HGTs como responsáveis pelos seus acontecimentos ainda não são bem conhecidos, bem como o fato de acontecerem entre espécies de filogenéticamente distante [38].

Alguns softwares estão disponíveis para auxiliar nos estudos de HGTs, os quais podem ser classificados quanto à estratégia de busca implementada: os baseados na análise filogenética, como HGT-Gen [36]; os baseados na combinação de comparação de sequências anotadas e seus registros de taxonomia, como HGTector [92]; e os baseados unicamente na composição da sequência onde os HGTs devem ser buscados, como Alien Hunter [81]. Esse último foi utilizado por este presente trabalho e está disponível para download sob licença GPL no endereço disponível em [4].

2.5 Regiões Anômalas e o software *Alien Hunter*

Alien Hunter [81] é uma ferramenta utilizada para detecção de regiões com características *anômalas* em uma sequência de DNA. Uma região é anômala em um genoma se esta apresenta características distintas das demais regiões deste genoma. Dentre essas características podemos regiões com alta frequência de CG [24, 73] e regiões variáveis ou repetitivas [32, 75]. Adicionalmente, uma região pode ser anômala, ou *alien* em um dado genoma se esta foi adquirida de outro genoma ao longo da evolução por eventos de HGTs [66, 44, 86, 42].

Utilizando o método *Interpolated Variable Order Motifs*, ou simplesmente IVOMs, a ferramenta explora distribuições de ordens variáveis de sequências *motifs*, para capturar a composição local de uma sequência. Objetivando melhorar os limites de cada região detectada, a ferramenta utiliza o modelo de *Hidden Markov Model* (HMM) [21] de dois estados.

Interpolated Variable Order Motifs (IVOMs)

Baseados em técnicas que utilizam mais de um índice de composição para identificação de regiões anômalas, como por exemplo [48], [41] e [40], a IVOMs utiliza índices de composição para definir o tamanho dos *motifs* que serão utilizados durante a pesquisa. Porém, o uso de indíces de composição de baixa ordem podem não fornecer informações suficientes para a composição atípica de uma região e, indíces de ordem muito superiores farão com que os *motifs* apresentem frequências muito baixas, ou seja, apresentam poucas informações para fornecer estimativas confiáveis. Assim, inicialmente a abordagem implementa variáveis ordens *k-mer* preferindo informações derivadas de *motifs* de ordem superior, porém quando apresentarem informações insuficientes, confia em *motifs* de ordens inferiores.

Seja B conjunto do alfabeto formado pelos *motifs*, ou seja, B = {A,C,G,T}, sendo que o número de todos os diferentes *motifs* possíveis aumenta exponencialmente com o tamanho do *motif*, que chamamos de k. Para $k-mers$ de tamanho k, há 4^k diferentes possíveis $k-mers$, já que o alfabeto definido apresenta 4 possíveis letras. Na IVOM todos os $k-mers$ com $1 \leq k \leq 8$ são explorados. Cada $k-mer$ pode ser visto como uma combinação linear dos componentes de ordem menor de *motifs*, incluindo ele mesmo. O primeiro passo é encontrar a frequência $P_m(S)$ de cada $k-mer$ m na sequência S, representado por:

$$P_m(S) = \frac{A_m(S)}{N-k+1} \tag{2.1}$$

onde $A_m(S)$ é o número de ocorrências de m na sequência S e N é o tamanho de S Normalmente, *motifs* de ordem mais alta ocorrem com menor frequência quando comparados aos de ordem mais baixa, pois o número de diferentes $k\text{-}mers$ possíveis aumenta exponencialmente de acordo com o tamanho dos *motifs*. Portanto, inicialmente calcula-se uma frequência para cada *k-mer*. Para o uso da combinação de diferentes ordens de *k-mer* é necessário analisar tanto o número de ocorrências quanto o número total de diferentes possíveis $k\text{-}mers$. Por isso a cada *k-mer* é atribuído um peso utilizando tais parâmetros, ou seja, cada *k-mer* possui um peso $W_m(S)$ calculada da seguinte maneira:

$$W_m(S) = \frac{A_m(S).|B|^k}{\sum_{j=1}^{8} A_j(S).|B|^j} \tag{2.2}$$

onde $|B|^k$ denota o número de possíveis *motifs* de tamanho k.

Após o cálculo dos pesos e das frequências dos $k\text{-}mers$, calcula-se a frequência IVOM, nome dado pelo autor, de cada *k-mer* m na sequência S. Calculada da seguinte maneira:

$$IVOM(S,m) = \begin{cases} W_m(S).P_m(S) + [1 - W_m(S)].IVOM(S, m_{2,|m|}) if |m| \geq 2 \\ W_m(S).P_m(S) if |m| = 1, \end{cases}$$

Onde $m_{2,|m|}$ denota a substring interpolada começando na posição 2 e terminando na posição $|m|$ no *k-mer* m.

Através do cálculo da frequência IVOM é possível observar todas as frequências de todas ordens dos *motifs*, assim, caso os *motifs* de ordem mais alta tenham estimativas confiáveis na contribuição da composição da sequência, os *motifs* de ordem mais baixas podem ser ignorados, e vice-versa.

Detecção de mudança de estado

Objetivando detectar regiões variáveis, aplica-se uma abordagem de janela deslizante na sequência genômica. Buscando encontrar o tamanho ótimo l de janela, foram feitas experiências com diferentes tamanhos de l, e os autores chegaram ao intervalo de descolamento, que chamamos de passo, da janela deslizante definido como 2,5kb. Deve-se observar que o aumento da ordem de $k\text{-}mers$ faz com que o tamanho da janela aumente também. Porém, utilizar um tamanho de passo muito grande pode gerar incertezas sobre os limites reais das regiões analisadas como atípicas.

Como dito, a escolha do passo para a abordagem de janela deslizante influencia na quantidade de cálculos necessários, assim, a diminuição do mesmo pode aumentar a precisão, porém, aumenta o tempo gosto com o processamento. Visando o aumento, há a redução da exatidão da localização das regiões de composição atípica. Por esse motivo, a implementação de *Hidden Markov Model* de dois estados é feita.

Utiliza-se então dois estados: o estado "nativo"que corresponde a regiões de composição típica e o estado "*alien*"que corresponde a regiões de composição atípica. O ponto de mudança corresponde à mudança de um estado para o outro, ou seja, definir os limites das regiões previstas, onde ocorre uma transição de estado. Este ponto de mudança vai representar o novo limite otimizado de cada previsão, oferecendo maior precisão da previsão em termos de localização de fronteira, a fim de detectar o ponto em que a transição do estado nativo para o *alien* ocorre e vice-versa.

Capítulo 3

Estratégia para análise da montagem de genomas guiada por mapeamento em referência

Usualmente o mapeamento, também chamado de remontagem guiada em uma referência, ocorre em duas grandes etapas, sendo a primeira o mapeamento dos *reads* em um genoma de referência e a segunda a obtenção dos *contigs* resultantes do mapeamento. Nestre trabalho, o mapeamento foi feito pelo *software* Bowtie [46], enquanto a obtenção dos *contigs* foi obtido através da utilização de ferramentas *Samtools* [51] e do *script pileup2fasta* [61].

Além das duas grandes etapas citadas no parágrafo anterior, neste trabalho propomos uma avaliação adicional do genoma de referência pela utilização do *software* Alien Hunter [81], a fim de detectar regiões que apresentem anomalias, as quais chamamos de regiões anômalas ou regiões *aliens*, em sua sequência e verificar a influência de tais regiões nos resultados obtidos do mapeamento de *reads*, para uma dada montagem guiada. Desta forma, dados sobre as regiões não mapeadas por algum conjunto de *reads* em uma dada referência e suas respectivas localizações, se em regiões *aliens* ou não, serão produzidos ao final desta estratégia de análise da montagem.

Adicionalmente, os *reads* mapeados e que não foram incluídos em nenhum *contig* devido à baixa cobertura, são recuperados e uma montagem *de novo* destes, com os demais *reads* não mapeados na primeira fase da montagem, será realizada. A comparação dos resultados desta montagem com a montagem realizada apenas com os *reads* não mapeados é realizada a fim de verificar os ganhos obtidos pela utilização dos *reads* recuperados.

A Figura 3.1 ilustra as etapas do processo de remontagem guiada pelo mapeamento em genoma de referência além das etapas de avaliação da referência para detecção de regiões *aliens*, identificação de regiões não mapeadas e remontagem *de novo* de *reads* não mapeados juntamente com os *reads* mapeados e não pertecentes à *contigs*.

Segue uma breve descrição de cada uma das subdivisões apresentadas na Figura 3.1.

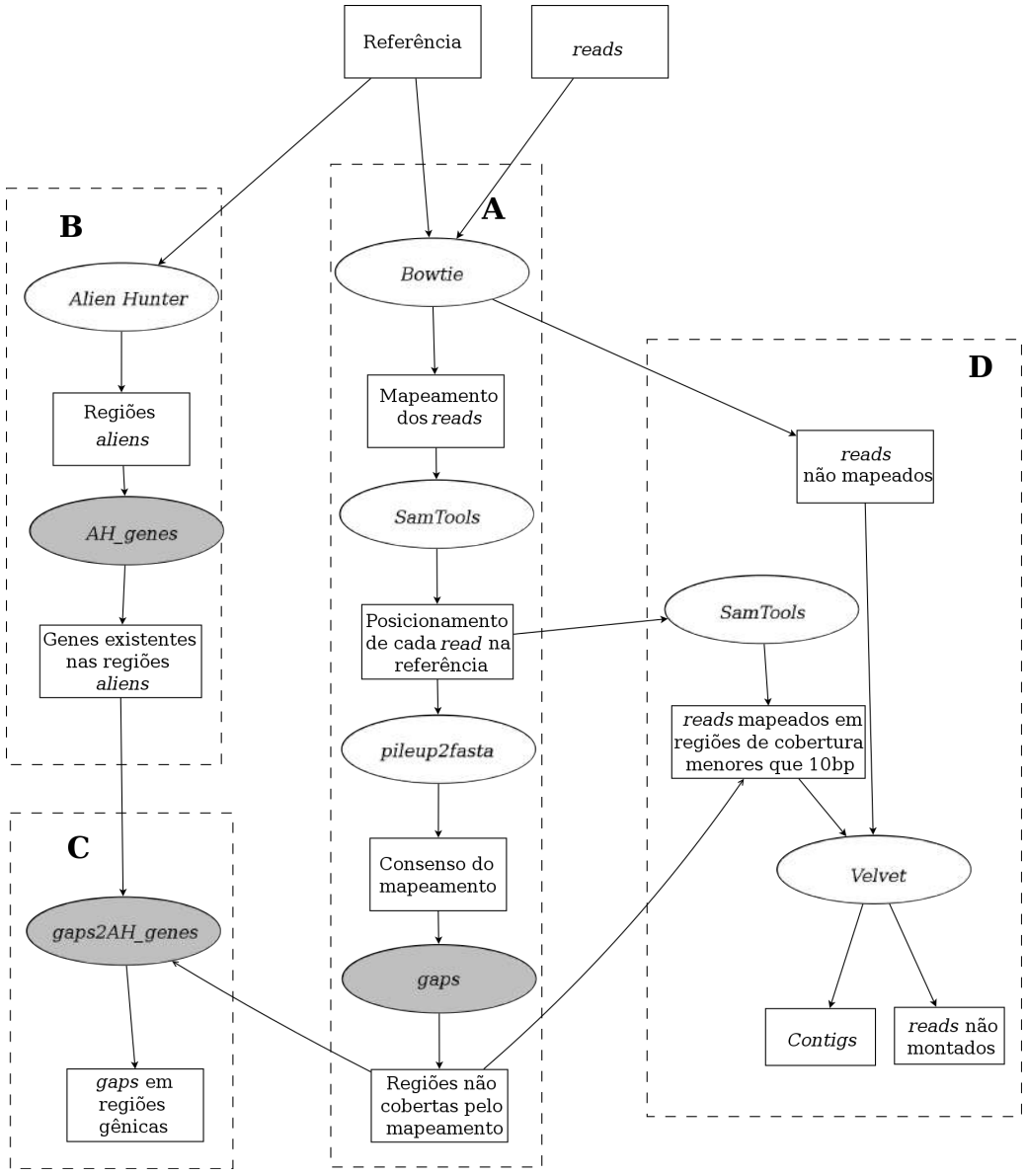


Figura 3.1: Etapas dos processos de mapeamento e montagem de *reads*.

A: Montagem guida pelo mapeamento executado pelo Bowtie, bem como a obtenção dos *contigs* através do Samtools e do *script pileup2fasta* e também a identificação das regiões de má cobertura, ditas *gaps*. A identificação de tais regiões foi feita por um programa desenvolvido em C, representado na figura pela elipse de texto *gaps*.

B: Identificação de regiões *aliens* na referência feita pelo *software* Alien Hunter seguida pela descoberta dos genes existentes em tais regiões. Esta última tarefa é realizada pelo programa desenvolvido em C, e indicado pela elipse de texto *AH_genes*.

C: Outro programa implementado, *gap2AH_genes*, realiza a tarefa de verificar, para cada *gap* do mapeamento, se este localiza-se dentro de em uma região correspondente

a um gene, apenas para os genes localizados em regiões *aliens* identificados em **A**.

D: Dadas as regiões de má cobertura encontrados em **A**, bem como o posicionamento de cada *read* no mapeamento, recupera-se os *reads* que não colaboraram em nenhum *contig*. Este conjunto de *reads*, juntamente com os *reads* não mapeados na referência são montados *de novo* pelo *software* Velvet.

Seguindo a estratégia descrita na Figura 3.1, apresentamos no Capítulo 4 a avaliação do genoma de referência AF2122/97 de *M. bovis* pelo Alien Hunter, o mapeamento de *reads* de 21 cepas de genomas do complexo *M. bovis* na referência AF2122/97 pelo software Bowtie e a análise das regiões mal mapeadas de cada cepa. No Capítulo 5 a montagem *de novo* dos *reads* é apresentada.

Capítulo 4

Determinação de regiões anômalas na referência e mapeamento dos *reads* do complexo *Mycobacterium bovis*

Mycobacterium bovis é o agente causador da tuberculose em bovinos, outros mamíferos, inclusive em seres humanos. O controle e a erradicação da tuberculose apresenta diversas dificuldades, o que contribui para a manutenção da infecção e de suas repercussões negativas para os animais [63]. A tuberculose animal é uma doença de relevância econômica no contexto da pecuária, pois afeta diretamente a produtividade dos animais e também influencia o comércio internacional de produtos de origem animal. Infecções por *M. bovis* também foram detectadas em animais selvagens, o que pode gerar graves consequências para o ecossistema. Além disso, animais com tuberculose carregam um potencial zoonótico, ou seja, é uma doença com transmissão entre animais vertebrados e seres humanos e é, portanto, uma preocupação para a saúde pública [56].

Além de uma ameaça para a saúde pública, a tuberculose bovina apresenta um impacto significativo sobre a agricultura, causando perdas anuais em todo o mundo de cerca de 3 bilhões de dólares americanos por consequência de despovoamento de rebanhos, restrições comerciais de rebanho e redução da produtividade agrícola, especialmente no mundo em desenvolvimento e em alguns países desenvolvidos [23].

O estudo da tuberculose objetiva observar seu diagnóstico pois constitui uma significativa causa de condenação de animais em matadouros, resultando no Brasil uma perda econômica estimada em 10% da produção leiteira e em 20% da produção da carne bovina [5], assim como infertilidade e condenação do animal já abatido [62].

Genomas do complexo *M. bovis* foram utilizados como estudo de caso para aplicação da estragégia de montagem proposta neste trabalho.

4.1 Regiões anômalas no genoma de referência

Considerando as características biológicas de transferência lateral de genes, alta frequência de conteúdo GC e regiões variantes, é possível analisar que elas expliquem, em parte, a falta de cobertura no mapeamento das cepas em algumas regiões da referência.

A fim de identificar possíveis regiões de transferência lateral, ou simplesmente regiões anômalas (*aliens*) no genoma da cepa AF2122/97, utilizado como referência nos mapeamentos aqui descritos, utilizou-se o *software* Alien Hunter. Como exposto na sessão 2.5, o *software* é baseado no método *Interpolated Variable Order Motifs*, ou simplesmente IVOMs, o qual explora distribuições de ordens variáveis de sequências (*motifs*), para capturar a composição local de uma sequência.

Como resultado da execução do Alien Hunter, um valor numérico é atribuído a cada uma das bases do genoma avaliado, o qual chamamos de *ah score*, ou simplesmente ah_{score}, bem como um valor limitante, dito *threshold* e representado por th, que serve como parâmetro para decidir se uma dada região é ou não *alien* no genoma. Bases para as quais o valor de ah_{score} é superior ao valor de th pertencem a uma região *alien*. Bases consecutivas de uma mesma região possuem o mesmo valor de ah_{score}.

Segundo a avaliação do Alien Hunter, 106 regiões são anômalas na referência *M. bovis* AF2122/97, sendo $th = 16$. O tamanho médio destas regiões é de 7.500 pares de bases (pb) e o valor médio de ah_{score} é de 24,35. A maior região identificada como anômala possui 37.500 pb e *score* igual a 40,41, enquanto as menores regiões possuem 5.000 pb e média de *score* igual a 19,4. A Figura 4.1 apresenta os valores de ah_{score} para cada uma das 106 regiões assinaladas como anômalas pelo Alien Hunter na referência.

Se o valor limitante utilizado para determinar a anormalidade de uma região for aumentado em duas vezes, isto é, para 32, apenas 17 regiões são assinaladas como *aliens*. Neste caso, as regiões obtidas apresentam tamanho e *score* médios de 16.470 pb e 42,1, respectivamente. A menor região, de tamanho 5.000 pb, possui *score* 45,9 e a maior região, com tamanho de 37.500 pb, possui *score* 40,4. A Figura 4.2 mostra a localização no genoma bem como os valores de ah_{score} destas 17 regiões.

As 17 regiões, que podem ser vistas na Figura 4.2, foram nomeadas como AR_1, AR_2, ..., AR_{17} e detalhes como ah_{score} atribuído pelo Alien Hunter, posição inicial e posição final da região no genoma, tamanho e número de genes anotados na região estão descritos na Tabela 4.1. Dadas as quantidades de genes em cada região, somam-se, segundo a análise do Alien Hunter, 198 genes nessas regiões anômalas, os quais, podem ter sido adquiridos ao longo da evolução por eventos de transferência lateral ou até mesmo pertencerem à regiões de alta frequência CG ou regiões repetitivas ou variáveis.

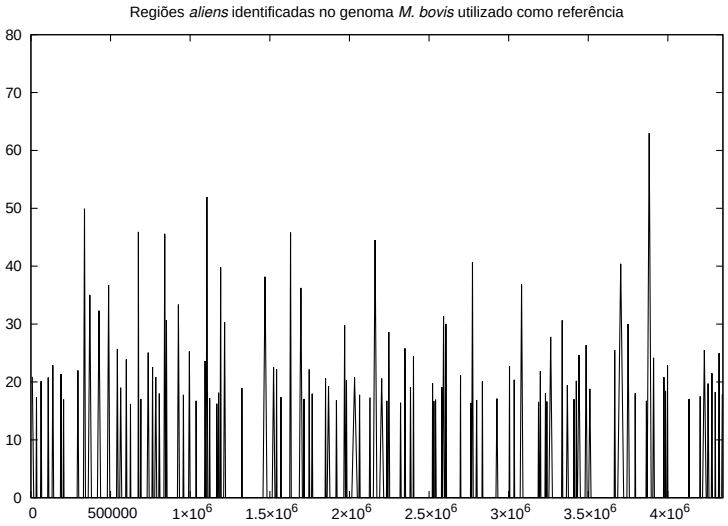


Figura 4.1: Valores de ah_{score} e posicionamento para as 106 regiões assinaladas como anômalas na referência *Mycobacterium bovis* AF2122/97.

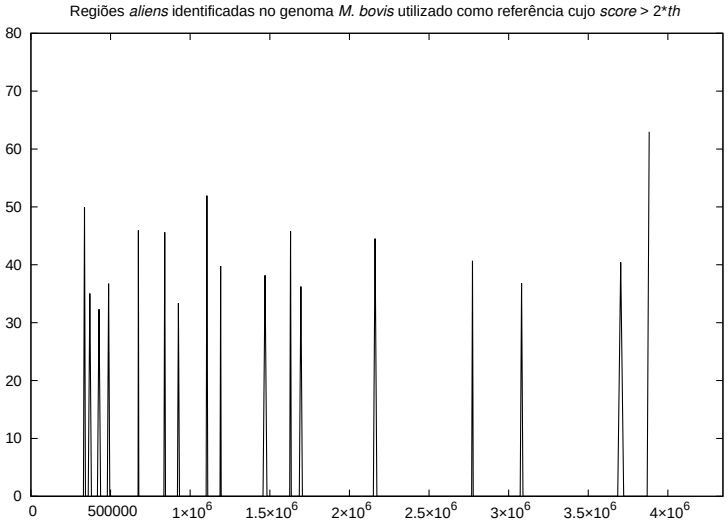


Figura 4.2: Valores de ah_{score} e posicionamento para as 17 regiões para as quais $ah_{score} > 32$ na referência *Mycobacterium bovis* AF2122/97.

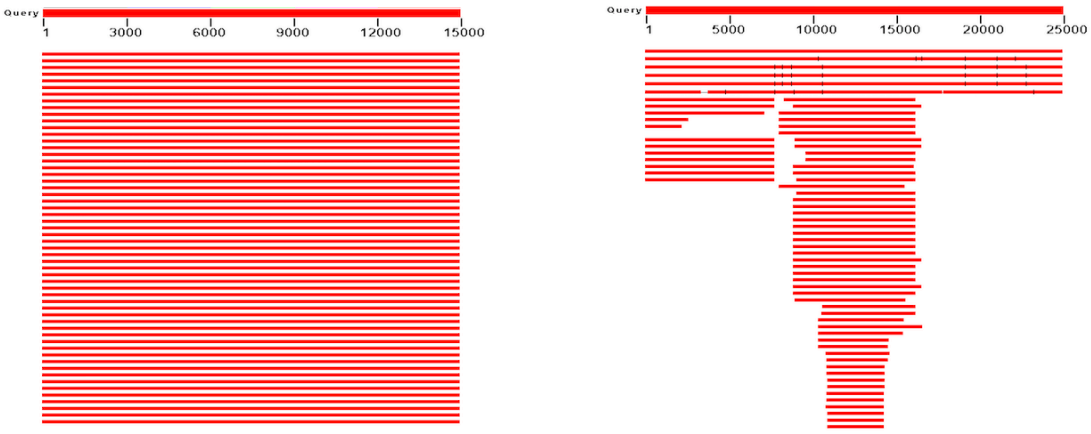
Os genes presentes em cada uma das regiões descritas na Tabela 4.1 são apresentados no Apêndice A.

Para cada uma das 17 regiões, uma comparação contra a base de dados de nucleotídeos

Tabela 4.1: Descrição das 17 regiões *aliens* para as quais $ah_{score} > 2 * th$.

Região *Alien*	ah_{score}	Posição inicial	Posição final	Tamanho (pb)	Nº de genes
AR_1	49,9	330.000	342.500	12.500	7
AR_2	35,0	360.000	380.000	20.000	16
AR_3	32,3	417.500	437.500	20.000	11
AR_4	36,7	480.000	495.000	15.000	12
AR_5	45,9	672.500	677.500	5.000	3
AR_6	45,5	835.000	845.000	10.000	11
AR_7	33,3	920.000	932.500	12.500	11
AR_8	51,9	1.100.000	1.110.000	10.000	12
AR_9	39,7	1.187.500	1.195.000	7.500	6
AR_{10}	38,1	1.457.500	1.482.500	25.000	22
AR_{11}	45,8	1.625.000	1.635.000	10.000	4
AR_{12}	36,2	1.685.000	1.705.000	20.000	20
AR_{13}	44,5	2.150.000	2.172.500	22.500	15
AR_{14}	40,6	2.767.500	2.777.500	10.000	10
AR_{15}	36,8	3.072.500	3.090.000	17.500	16
AR_{16}	40,4	3.685.000	3.722.500	37.500	12
AR_{17}	62,9	3.870.000	3.895.000	25.000	10

do NCBI foi realizada pela ferramenta Blastn [91]. A representação gráfica dos alinhamentos encontrados para as regiões AR_4 e AR_{17} são apresentadas na Figura 4.3. Note que a região AR_4 acontece por inteiro nos primeiros 56 hits encontrados na base de dados, o que pode indicar que tal região pode ser um resultado falso positivo do Alien Hunter ou, ainda, que tal região anômala foi adquirida por todas as cepas de *Mycobacterium bovis*, *M. africanum* e *M. tuberculosis* que compõem a lista de hits do Blastn. Já a região AR_{17} não acontece na íntegra em nenhuma das cepas que compõem a lista de hits. Os hits encontrados para todas as regiões *aliens* são apresentados no Apêndice A.



(a) Região AR_4 (b) Região AR_{17}

Figura 4.3: Alinhamentos encontrados pelo Blastn para as regiões *aliens* AR_4 e AR_{17}.

4.2 Mapeamento dos *reads* no genoma de referência

Vinte e uma cepas de bactérias do complexo *M. bovis* foram sequenciadas em um trabalho conjunto entre Brasil e Argentina, objetivando a caracterização, bem como a comparação entre as mesmas. Os sequenciadores utilizados para este trabalho foram Illumina [6] e IonTorrent[8]. Do total das 21 cepas sequenciadas, 17 cepas já foram montadas, anotadas e depositadas no GenBank [11] (veja o Bioproject número 214551, disponível em [2]). As demais 4 cepas de origem brasileira ainda necessitam ser montadas e anotadas para posterior submissão ao GenBank.

Algumas amostras foram sequenciadas em ambas as plataformas, enquanto outras somente pelo Illumina sendo, para estas últimas, realizadas duas rodadas de sequenciamento para cada amostra. Os *reads* gerados pelo Illumina são do tipo *paired-end* e possuem tamanhos entre 50 e 250 pares de bases. Os *reads* produzidos pelo IonTorrent possuem tamanhos entre 6 e 364 pares de bases e são do tipo *single-end*. As cepas de origem brasileira, identificadas como 08_B_45-08B, 09_B_18-08C, 10_B_07-08, 12_B_61-09, 13_B_32-08, 14_B_49-09, 16_B_08-08BF2, 17_B_07-08, 20_B_50, 26_B_128, 28_B_35, 30_B_268, 33_B_AN5 e 34_B_0822-11, não foram sequenciadas pelo IonTorrent. A Tabela 4.2 nomeia cada uma das 21 cepas de *M. bovis* e apresenta a quantidade de *reads* disponíveis para a remontagem de cada uma delas.

Até a data da escrita desse projeto, as cepas 10_B_07-08, 17_B_33, 26_B_128 e 30_B_268 ainda não passaram pelo processo de remontagem, anotação e deposito no GenBank. Apesar da montagem já realizada para 17 cepas, o conjunto das 21 cepas foi submetido ao processo de remontagem aqui descrito, utilizando o genoma AF2122/97 (número de acesso no GenBank: NC_002945) do complexo *M. bovis* que possui 4.345.492 pares de bases (pb) como referência. A escolha do genoma de referência utilizado foi devido ao alto nível de semelhança entre as cepas sequenciadas e o genoma AF2122/97. De fato, os organismos deste complexo possuem alto grau de similaridade entre seus genomas. A anotação da referência conta com 3.918 genes, dos quais 2.321 codificam para proteínas com função conhecida. Detalhes de sua anotação podem ser obtidos dos arquivos disponíveis no link [7].

Os *reads* foram mapeados no genoma de referência, utilizando o *software* Bowtie2 [45]. Como entrada o *software* recebe um arquivo contendo os nucleotídeos do genoma de referência no formato .fna, o(s) arquivo(s) contendo os *reads* no formato .fastq a serem mapeados, além dos parâmetros que irão controlar o mapeamento, como por exemplo número de *mismatches* permitidos no alinhamento e comprimento das sementes (*substrings*) a serem buscadas durante o processo de escolha da posição de mapeamento para um *read*. Valores menores para este último parâmetro torna a tarefa de busca por alinhamento entre trechos da referência e um *read* mais lenta, entretanto, produz resultados mais sensíveis.

Como resultado, os alinhamentos encontrados entre os *reads* e a referência são armazenados em um arquivo de saída cujo formato é denominado SAM (*Sequence Alignment/Mapping*). Os *reads* não mapeados são também armazenados em arquivo. Dentre outras informações, o arquivo SAM traz, para cada *read* mapeado, a posição na referência onde o *read*, iniciando da base mais a esquerda, se alinha na referência; um valor indicando a qualidade deste alinhamento; informação se o *read* tem um par, ou seja, se é *paired-end* e se o par foi mapeado ou não. Pode-se obter mais informações e detalhes sobre o formato SAM em [14].

A Tabela 4.2 compila as informações quanto à quantidade de *reads* mapeados e não mapeados para as 21 cepas de bactérias do complexo *M. bovis* mapeados na referência AF2122/97.

Apesar das baixas porcentagens de *reads* não mapeados mostrados na Tabela 4.2, observa-se que os valores absolutos são expressivos. Como exemplo temos as cepas 04_A_4303, 09_B_18-08C, 12_B_61-09, 17_B_33, 33_B_AN5 que apresentaram mais de 20 mil *reads* não mapeados. Tais *reads* não mapeados serão remontados pela estratégia *de novo*, descrita no Capítulo 5.

Após execução do Bowtie2, foi utilizado o *Samtools* [51], que corresponde a um conjunto de *softwares* para a manipulação de arquivos e alinhamentos armazenados no formato SAM e BAM (*Binary Alignment/Mapping*)[1], implementados em C [28]. Além de realizar conversões entre os formatos SAM e BAM, combinar diversos arquivos BAM em um novo arquivo e gerar relatório dos *reads* mapeados, é possível também, gerar um

[1]O arquivo no formato BAM contém as mesmas informações que o arquivo no formato SAM, exceto pelo fato dos dados serem armazenados em binário no arquivo BAM.

Tabela 4.2: *Reads* mapeados e não mapeados de cepas do complexo *M. bovis* na referência AF2122/97

Cepa	Total de *reads*	Nº de *reads* mapeados	Nº e % de *reads* não mapeados
01_A_91191	1.336.369	1.320.369	16.000 - 1,19%
02_A_05-567	1.728.660	1.711.753	16.907 - coemce0,97%
03_A_05-566	812.347	798.684	13.663 - 1,68%
04_A_4303	823.231	790.157	33.074 - 4,01%
05_A_534	1.248.015	1.234.672	13.343 - 1,06%
06_A_91193	910.835	897.244	13.591 - 1,49%
07_A_91192	2.111.465	2.095.595	15.870 - 0,75%
08_B_45-08B	1.113.377	1.104.076	9.301 - 0,83%
09_B_18-08C	1.384.047	1.361.305	22.742 - 1,64%
10_B_07-08	686.887	686.429	458 - 0,06%
12_B_61-09	1.517.443	1.482.309	35.134 - 2,31%
13_B_32-08	1.194.623	1.185.982	8.641 - 0,72%
14_B_49-09	1.671.231	1.657.103	14.128 - 0,84%
16_B_08-08BF2	586.095	575.540	10.555 - 1,8%
17_B_33	1.522.953	1.472.265	50.688 - 3,32%
20_B_50	964.794	956.334	8.460 - 0,87%
26_B_128	245.613	245.253	360 - 0,14%
28_B_35	473.231	472.491	740 - 0,15%
30_B_268	83.968	76.289	7.679 - 9,14%
33_B_AN5	2.269.762	2.229.737	40.025 - 1,76%
34_B_0822-11	85.8287	840.890	17.397 - 2,02%

arquivo com as informações do mapeamento ordenados pela posição na referência onde os *reads* foram mapeados. Tal ordenação é de suma importância para a realização das análises pós mapeamento, dentre elas a montagem do genoma mapeado.

Entre os arquivos gerados pelo *Samtools* está o de extensão .mpileup que traz, em cada uma de suas linhas, as seguintes informações para cada uma das bases do genoma de referência: nome do cromossomo, coordenada, nucleotídeo, quantidade de *reads* mapeados na base em questão (cobertura) e qualidade do mapeamento para tal base [9]. Assim sendo, em suma, este arquivo guarda as informações das coberturas, na referência, obtidas por um determinado conjunto de *reads* de uma determinada cepa mapeada.

Dado o arquivo .mpileup, que contém a posição de mapeamento de cada *reads*, os *contigs* resultantes do mapeamento podem ser obtidos pela execução de um *script* chamado *pileup2fasta*, escrito em *perl* e desenvolvido por *John Nash* [61]. Para tal, o *script* toma como parâmetro um inteiro que representa a cobertura mínima que cada base deve possuir para ser considerada coberta. Para as remontagens realizadas no contexto deste trabalho, foi considerada cobertura mínima 10, desta forma, bases que tiveram menos de 10 *reads* mapeados, não fazem parte de nenhum *contig* e são representadas no arquivo

.fasta resultante pelo caracter "*". Assim, alguns *reads*, apesar de mapeados, não colaboram com a montagem e, desta forma, podem ser utilizados para a montagem *de novo*, juntamente com os demais *reads* não mapeados.

Outro fator a ser considerado na análise do resultado do mapeamento e/ou do não mapeamento de *reads* é a existência de regiões distintas entre a cepa e a referência que podem ser explicadas por eventos naturais ocorridos em um dos organismos ou, em outras palavras, pela existência de regiões *aliens* em um dos genomas. A seção seguinte avalia a correlação das regiões de baixa cobertura obtidas do mapeamento das cepas com as regiões *aliens* descobertas na referência.

4.3 Regiões de baixa cobertura no mapeamento das cepas

Dadas as regiões anômalas na referência e a montagem guiada pelo mapeamento das cepas no genoma de referência, as regiões não cobertas foram avaliadas quanto ao seu posicionamento em relação à referência. Dada uma região de baixa cobertura na cepa, esta está posicionada em uma região anômala ou não? Dada uma região anômala, quais cepas não tiveram *reads* mapeados na mesma?

Responder aos questionamentos anteriores pode colaborar no entendimento das características específicas de cada cepa, bem como na descrição da ocorrência ou não de eventos representados por características biológicas no conjunto de *M. bovis* em estudo.

Após o mapeamento dos *reads* de cada cepa na referência, tem-se a informação, para cada uma das bases da referência, da cobertura obtida pelos *reads* da cepa em questão. Tais coberturas estão armazenadas em um arquivo contendo exatamente 4.345.492 valores inteiros, já que este é o tamanho do genoma utilizado como referência. O primeiro inteiro do arquivo indica a quantidade de *reads* que tiveram uma de suas bases mapeada na primeira posição do genoma de referência, o segundo inteiro indica a quantidade de *reads* que tiveram uma de suas bases mapeada na segunda posição do genoma de referência, e assim sucessivamente. Para uma base na referência que não possuiu nenhum *read* mapeado, o inteiro correspondente tem valor 0 (zero).

A média dos valores de cobertura de um determinado mapeamento é então calculada com a média dos valores armazenados. Adicionalmente, as informações de regiões de baixa cobertura podem ser obtidas do arquivo .fasta resultante da montagem, que contém o consenso. Nele os *gaps* são representados por asteriscos ("*"). Tais asteriscos são a representação das bases com valor de cobertura menor que 10.

A Figura 4.4 ilustra a visualização de um trecho do mapeamento da cepa 28_B_35 na referência AF2122/97 e foi gerada pela ferramenta Tablet [58], que é um visualizador gráfico capaz de representar as informações de mapeamentos armazenadas em arquivos .sam [58].

Os índices i e j destacados na Figura 4.4 indicam, respectivamente, as coordenadas 3.684.999 e 3.722.499 que delimitam a região AR_{16} na referência.

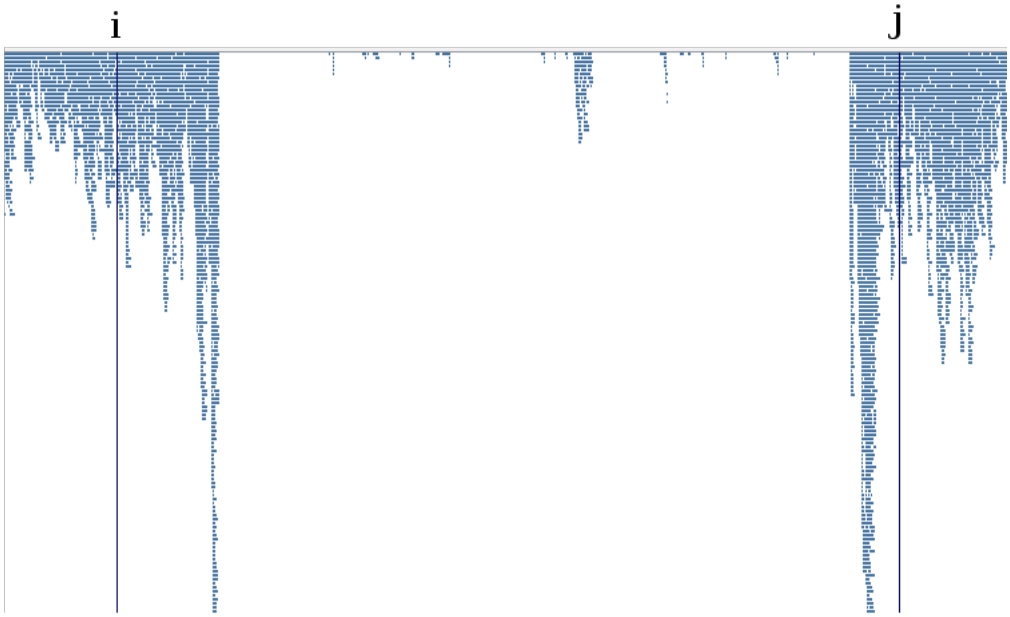


Figura 4.4: Trecho do mapeamento dos *reads* da cepa 28_B_35 na região AR_{16}.

Um programa nomeado *gaps* foi escrito em C para determinar, dado o arquivo .fasta correspondente à montagem de uma cepa, as regiões não cobertas, ou seja, os *gaps* em relação à referência. Apenas regiões 10 asteriscos consecutivos ou mais são consideradas *gaps*. A Tabela 4.3 compila as informações dos *gaps* resultantes de cada montagem apresentando o número e o tamanho médio dos *gaps*, além do tamanho do maior *gap* e da indicação da região *alien* correspondente à este maior *gap*. As células marcadas com * na Tabela 4.3 indicam que a maior região não coberta ocorreu em uma região *alien* cujo valor de $ah_{score} < 2 * th$. Tais regiões são as mesmas para as cepas 10_B_07-08, 13_B_32-08 e 33_B_AN5, cujo *score* é 18.

A Tabela 4.4 indica as porcentagens dos *gaps* que correspondem à alguma região anômala na referência. São calculadas, separadamente, os valores de tais porcentagem considerando todas as 106 regiões, bem como apenas as 17 regiões para as quais o *score* é ao menos 2**th*. A Figura 4.5 apresenta as posições relativas de todos os *gaps* encontrados na cepa 01_A_91191 em relação às 17 regiões *aliens*.

Exceto pelas cepas 26_B_128 e 30_B_268, mais de 69% dos *gaps* de todas as cepas estão posicionados em alguma região anômala na referência. Pode-se analisar os resultados da Tabela 4.4 juntamente com a quantidade de *reads* não mapeados de cada cepa (Tabela 4.2) na tentiva de explicar o porquê do não mapeamento dos mesmos. Tome como exemplo a cepa 17_B_33 que teve 50.688 de seus *reads* não mapeados e 80,99% de seus *gaps* ocorridos em regiões anômalas. Pode-se inferir que, neste caso, o não mapeamento se deu por causa de regiões de diferenças entre a cepa e a referência e que, possivelmente, tais diferenças sejam devidas à eventos representados por características biológicas ocorridos em apenas um dos organismos. Similarmente, as cepas 04_A_4303, 12_B_61-09, 17_B_33, 33_B_AN5 tiveram, respectivamente 33.074, 35.134, 50.688 e 40.025 *reads* não mapeados sendo ao menos 75% destes *gaps* ocorridos em regiões *aliens*.

Tabela 4.3: *Gaps* resultantes do mapeamento dos *reads* das cepas no genoma de referência.

Cepa	Nº de *gaps*	Tamanho médio dos *gaps* (pb)	Tamanho dos maiores *gaps* (pb)	Região *alien* correspondente
01_A_91191	110	69,58	477	
02_A_05-567	129	63,82	1.071	
03_A_05-566	205	69,38	576	
04_A_4303	228	102,81	1.007	AR_{17}
05_A_534	133	59,79	435	AR_{17}
06_A_91193	190	73,27	838	
07_A_91192	105	53,05	835	
08_B_45-08B	124	117,03	632	AR_{17}
09_B_18-08C	90	100,51	571	AR_{17}
10_B_07-08	131	158,56	4.502	*
12_B_61-09	82	81,24	468	AR_7
13_B_32-08	125	152,48	1.857	*
14_B_49-09	75	74,76	393	AR_7
16_B_08-08BF2	157	122,22	887	AR_{17}
17_B_33	121	247,63	5.988	*
20_B_50	117	134,6	632	AR_7
26_B_128	547	50,94	858	AR_{17}
28_B_35	132	448,65	5.281	AR_{16}
30_B_268	3.433	101,53	1.675	AR_6
33_B_AN5	75	198,42	4.248	*
34_B_0822-11	138	223,9	2.952	

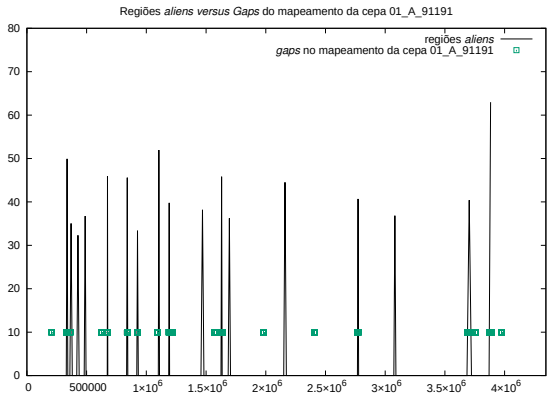


Figura 4.5: Posicionamento dos *gaps* da cepa 01_A_91191 em relação às 17 regiões *aliens* do genoma de referência.

Tabela 4.4: Porcentagem de *gaps* localizados em posições correspondentes à regiões *aliens* na referência

Cepa	Nº de *gaps*	Nº e porcentagem de *gaps* em alguma das 106 regiões *aliens*	Nº e porcentagem de *gaps* em alguma das 17 regiões *aliens*
01_A_91191	110	89 - 80,90%	71 - 64,54%
02_A_05-567	129	106 - 82,17%	77 - 59,68%
03_A_05-566	205	166 - 80,97%	130 - 63,41%
04_A_4303	228	179 - 78,50%	129 - 56,57%
05_A_534	133	111 - 83,45%	85 - 63,90%
06_A_91193	190	152 - 80,00%	116 - 61,05%
07_A_91192	105	86 - 81,90%	57 - 54,28%
08_B_45-08B	124	126 - 73,38%	87 - 45,96%
09_B_18-08C	90	87 - 74,44%	64 - 46,66%
10_B_07-08	131	95 - 72,51%	63 - 48,09%
12_B_61-09	82	62 - 75,60%	42 - 51,21%
13_B_32-08	125	66 - 73,60%	46 - 48,00%
14_B_49-09	75	56 - 74,66%	37 - 49,33%
16_B_08-08BF2	157	109 - 69,42%	69 - 43,94%
17_B_33	121	98 - 80,99%	60 - 49,58%
20_B_50	117	83 - 70,94%	52 - 44,44%
26_B_128	547	194 - 35,46%	69 - 12,61%
28_B_35	132	102 - 77,27%	69 - 52,27%
30_B_268	3433	836 - 24,35%	260 - 7,57%
33_B_AN5	75	60 - 80,00%	35 - 46,66%
34_B_0822-11	138	96 - 69,56%	52 - 37,68%

A comparação entre as cepas em análise, em relação à presença ou ausência de *gaps* nas 17 regiões *aliens* para as quais $ah_{score} > 2 * th$, pode ser feita pela análise dos dados mostrados na Tabela 4.5 que indica com um "*"a ocorrência de um ou mais *gaps* nas regiões *aliens* da referência.

As regiões AR_1, AR_6, AR_7, AR_{11}, AR_{14} e AR_{17} foram, de alguma forma, mal mapeadas em todas as cepas. A região AR_8 foi mal mapeada apenas na cepa 30_B_268. Note que, enquanto as regiões AR_1, AR_6, AR_7, AR_{11}, AR_{14} e AR_{17} podem apresentar alguma caraterística presente apenas no genoma de referência, pois não está presente por completo em nenhuma das cepas, cepa 30_B_268 pode se diferenciar das demais por algum aspecto presente na região AR_8, já que esta é a única cepa que apresentou *gap* nesta região.

Note, entretanto, que a existência de um "*"na Tabela 4.5, indica apenas que um ou mais *gaps* ocorreram no mapeamento de uma determinada cepa, em uma região contida em uma região *alien*. Assim, a informação da presença/ausência de *gaps* em regiões específicas de cada cepa foi refinada e agora tem-se, na Tabela 4.6, a substituição dos

Tabela 4.5: Ocorrência de *gaps* em regiões *aliens*

Cepa \ Região	AR_1	AR_2	AR_3	AR_4	AR_5	AR_6	AR_7	AR_8	AR_9	AR_{10}	AR_{11}	AR_{12}	AR_{13}	AR_{14}	AR_{15}	AR_{16}	AR_{17}
01_A_91191	*	*			*	*	*		*		*			*		*	*
02_A_05-567	*	*			*	*	*		*		*			*		*	*
03_A_05-566	*	*			*	*	*		*		*			*		*	*
04_A_4303	*	*			*	*	*		*		*			*		*	*
05_A_534	*	*			*	*	*		*		*			*		*	*
06_A_91193	*	*			*	*	*		*		*			*		*	*
07_A_91192	*	*				*	*		*		*			*		*	*
08_B_45-08B	*	*			*	*	*				*			*		*	*
09_B_18-08C	*	*			*	*	*				*			*		*	*
10_B_07-08	*	*			*	*	*		*		*			*	*	*	*
12_B_61-09	*				*	*	*		*		*			*		*	*
13_B_32-08	*	*			*	*	*		*		*			*		*	*
14_B_49-09	*	*			*	*	*				*			*		*	*
16_B_08-08BF2	*	*			*	*	*		*		*			*	*	*	*
17_B_33	*	*			*	*	*		*		*			*	*	*	*
20_B_50	*	*			*	*	*		*		*			*		*	*
26_B_128	*		*	*	*	*	*			*	*	*	*	*	*	*	*
28_B_35	*	*		*	*	*	*		*		*		*	*	*	*	*
30_B_268	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
33_B_AN5	*				*	*	*		*		*			*			*
34_B_0822-11	*	*			*	*	*		*		*			*		*	*

"*"por inteiros que contam a quantidade de *gaps* bem como pela somatória dos tamanhos destes *gaps* na região em questão.

Pelos valores apresentados na Tabela 4.6, verifica-se que o maior número *gaps* se concentram nas regiões AR_1, AR_6, AR_7 e AR_{17} que totalizam 190, 190, 158 e 371 *gaps* respectivamente. Se, ao invés da quantidade de *gaps* a somatória dos tamanhos dos *gaps* for considerada, as regiões AR_6, AR_7, AR_{16} e AR_{17} são as que apresentam maiores valores, sendo estes 24.458, 20.164, 33.872 e 42.403.

O maior número de *gaps* aconteceu na região AR_{17}, exceto pelas cepas 09_B_18-08C, 10_B_07-08, 28_B_35 e 30_B_268. Lembrando que a região AR_{17} é a região de maior valor de ah_{score} dentre todas, como apresentado na Tabela 4.1.

Dada a anotação do genoma de referência, é possível determinar ainda, quantos dos *gaps* descritos na Tabela 4.6 ocorrem em regiões gênicas. Esses números estão descritos na Tabela 4.7.

Tabela 4.6: Somatória da quantidade (linha superior da célula) e dos tamanhos (linha inferior da célula) dos *gaps* ocorridos em regiões *aliens* no mapeamento de cada cepa

Cepa \ Região	AR_1	AR_2	AR_3	AR_4	AR_5	AR_6	AR_7	AR_8	AR_9	AR_{10}	AR_{11}	AR_{12}	AR_{13}	AR_{14}	AR_{15}	AR_{16}	AR_{17}
01_A_91191	12 775	1 20			2 144	8 841	6 481		7 452		7 414			3 119		6 540	19 1082
02_A_05-567	15 467	1 22			4 347	14 785	7 685		7 429		5 359			3 303		3 65	18 1064
03_A_05-566	17 1071	2 86			4 329	13 1077	12 699		10 652		16 789			6 349		5 311	45 2655
04_A_4303	20 1789	5 139			4 475	12 1658	10 1093		12 1016		13 1447			4 413		14 612	35 4366
05_A_534	13 664	1 43			5 216	11 895	11 670		8 437		6 289			2 150		6 370	22 1612
06_A_91193	18 903	4 120			4 320	12 1191	11 729		9 615		12 580			4 271		8 675	34 2848
07_A_91192	10 519	2 31				7 501	8 474		4 133		3 64			2 84		3 514	18 617
08_B_45-08B	7 839	2 196			6 424	10 1649	4 1391				6 1090			9 1093		1 88	12 1821
09_B_18-08C	4 462	1 27			2 214	8 973	6 1066				5 762			7 794		2 52	7 1154
10_B_07-08	11 607	2 69			4 240	8 1061	8 901		1 17		6 572			7 518	3 104	3 193	10 2047
12_B_61-09	4 368				3 115	6 826	4 963		1 28		6 598			7 316		1 22	10 1044
13_B_32-08	7 944	1 93			4 304	8 1367	6 1292		1 65		6 747			7 916		4 238	16 1411
14_B_49-09	2 237	1 27			1 45	6 701	6 939				7 290			4 204		1 72	9 674
16_B_08-08BF2	6 1135	2 268			4 360	8 1993	5 1298		2 43		7 1041			8 1024	1 33	7 670	19 2649
17_B_33	7 664	1 30			4 333	9 733	5 616		1 29		6 292			5 139	7 243	2 19	13 4324
20_B_50	6 1075	2 149			5 536	8 1265	6 1271		1 39		5 1245			7 1145		3 174	9 1329
26_B_128	3 306		5 312	1 27	2 41	6 589	4 371			5 312	5 576	9 333	6 403	4 77	4 236	5 183	10 1378
28_B_35	3 350	1 30		1 38	1 63	6 562	6 501		1 74		6 136		1 22	4 118	1 254	23 25918	15 3745
30_B_268	16 1811	10 814	9 671	10 521	9 1528	14 3724	19 2797	10 897	7 850	18 1520	14 2334	12 716	12 726	14 3676	14 1627	38 3130	34 4417
33_B_AN5	3 281				4 163	7 582	6 523		1 12		5 141			2 68			7 348
34_B_0822-11	6 931	1 12			4 436	9 1485	8 1404		1 66		5 1343			8 1530		1 26	9 1818

Tabela 4.7: Quantidade de *gaps* (linha superior da célula) e dos tamanhos dos *gaps* (linha inferior da célula) com localização correspondente à de genes na referência, que estão presentes em regiões *aliens*

Cepa \ Região	AR_1	AR_2	AR_3	AR_4	AR_5	AR_6	AR_7	AR_8	AR_9	AR_{10}	AR_{11}	AR_{12}	AR_{13}	AR_{14}	AR_{15}	AR_{16}	AR_{17}
01_A_91191	12 775	1 20			2 144	8 841	6 481		7 452		7 414			3 119		1 17	19 1082
02_A_05-567	15 467	1 22			4 347	14 785	7 685		7 429		5 359			3 303		1 10	18 1064
03_A_05-566	17 1071	2 86			4 329	13 1077	12 699		10 652		16 789			6 349		1 19	45 2655
04_A_4303	20 1789	5 139			4 475	12 1658	10 1093		12 1016		13 1447			4 413		3 131	35 4366
05_A_534	13 664	1 43			5 216	11 895	11 670		8 437		6 289			2 150		1 19	22 1612
06_A_91193	18 903	4 120			4 320	12 1191	11 729		9 615		12 580			4 271		2 82	34 2848
07_A_91192	10 519	2 31				7 501	8 474		4 133		3 64			2 84			18 617
08_B_45-08B	7 839	2 196			6 424	10 1649	4 1391				6 1090			9 1093		1 88	12 1821
09_B_18-08C	4 462	1 27			2 214	8 973	6 1066				5 762			7 794		2 52	7 1154
10_B_07-08	11 607	2 69			4 240	8 1061	8 901		1 17		6 572			7 518		1 68	10 2047
12_B_61-09	4 368				3 115	6 826	4 963		1 28		6 598			7 316			10 1044
13_B_32-08	7 944	1 93			4 304	8 1367	6 1292		1 65		6 747			7 916		2 198	16 1411
14_B_49-09	2 237	1 27			1 45	6 701	6 939				7 290			4 204		1 72	9 674
16_B_08-08BF2	6 1135	2 268			4 360	8 1993	5 1298		2 43		7 1041			8 1024		1 286	19 2649
17_B_33	7 664	1 30			4 333	9 733	5 616		1 29		6 292			5 139		2 19	13 4324
20_B_50	6 1075	2 149			5 536	8 1265	6 1271		1 39		5 1245			7 1145		2 156	9 1329
26_B_128	3 306		5 312	1 27	2 41	6 589	4 371			5 312	5 576	9 333	5 394	4 77	1 46	2 52	10 1378
28_B_35	3 350	1 30		1 38	1 63	6 562	6 501		1 74		6 136		1 22	4 118		10 17282	15 3745
30_B_268	16 1811	9 743	9 671	10 521	9 1528	13 3632	16 2658	10 897	7 850	10 959	14 2334	12 716	11 709	14 3676	13 1547	21 1509	33 4316
33_B_AN5	3 281				4 163	7 582	6 523		1 12		5 141			2 68			7 348
34_B_0822-11	6 931	1 12			4 436	9 1485	8 1404		1 66		5 1343			8 1530		1 26	9 1818

Comparando as informações das tabelas 4.6 e 4.7 nota-se que poucos valores foram alterados (células em destaque na Tabela 4.7) o que revela que os *gaps* ocorridos em regiões *alien* ocorreram também, na sua grande maioria, em regiões gênicas da referência. Tal informação revela a importância da aplicação da estratégia para descoberta de genes posivelmente ausentes em cepas. A Figura 4.6 apresenta um trecho do mapeamento dos *reads* da cepa 28_B_35 na região AR_{16} bem como a indicação dos genes presentes nesta região da referência.

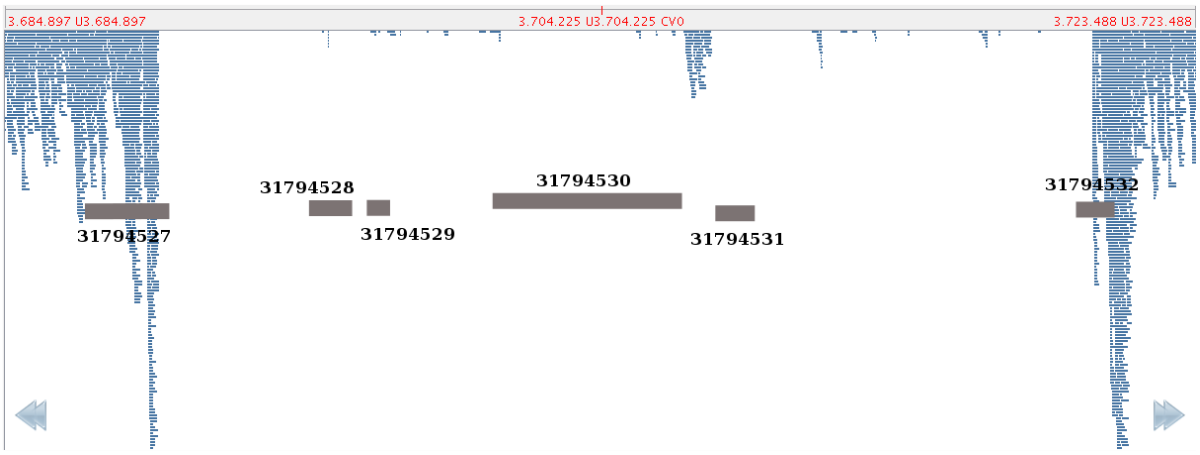


Figura 4.6: Trecho do mapeamento dos *reads* cepa 28_B_35 na região AR_{16} com indicação das posições relativas e PIDs dos genes anotados no genoma de referência.

A região apresentada na Figura 4.6 está compreendida entre as bases 3.648.897 e 3.7232.488. Pelo mapeamento apresentado, pode-se afirmar que os genes de PID 31794528, 31794529, 31794530 e 31794531 não possuem homólogos na cepa 28_B_35 ou que os *reads* correspondentes a estes homólogos não foram sequenciados. O Apêndice B apresenta mais algumas figuras de trechos de regiões *aliens* onde *gaps* ocorreram no mapeamento, bem como a indicação dos genes presentes na referência no trecho apresentado.

Capítulo 5

Reads não mapeados e montagem *de novo*

Regiões da referência onde dez ou menos *reads* foram mapeados são consideradas *gaps* (ver seção 4.2). Apesar de não colaborarem no processo de montagem guida por referência, tais *reads* podem ser utilizados para realização da montagem *de novo*, juntamente com os *reads* não mapeados. O processo de recuperação desses *reads* foi feito pelo *software* Samtools, já que este permite, através da sua função *view*, recuperar *reads* que estão mapeados em qualquer intervalo no genoma de referência. Assim, todos os *reads* localizados nos intervalos correspondentes à *gaps* foram coletados.

A Figura 5.1 apresenta um trecho do mapeamento da cepa 13_B_32-08 no intervalo entre as bases 1.759.265 a 1.778.560, onde *gaps* ocorreram e de onde *reads* podem ser recuperados. O trecho mostrado na imagem ocorre em umas das 106 regiões *aliens*, porém possui *score* menor que $th * 2$.

Figura 5.1: Trecho do mapeamento da cepa 13_B_32-08 no intervalo entre as bases 1.759.265 a 1.778.560, de onde *reads* são recuperados para montagem *de novo*.

As montagens *de novo* foram realizas pelo *software* Velvet [89], que implementa uma estratégia de montagem utilizando grafo de Bruijn [19]. Inicialmente realizou-se a montagem apenas dos *reads* não mapeados pelo Bowtie. A Tabela 5.1 apresenta os resultados desta montagem. Apenas os *contigs* maiores do que 300 pb são considerados.

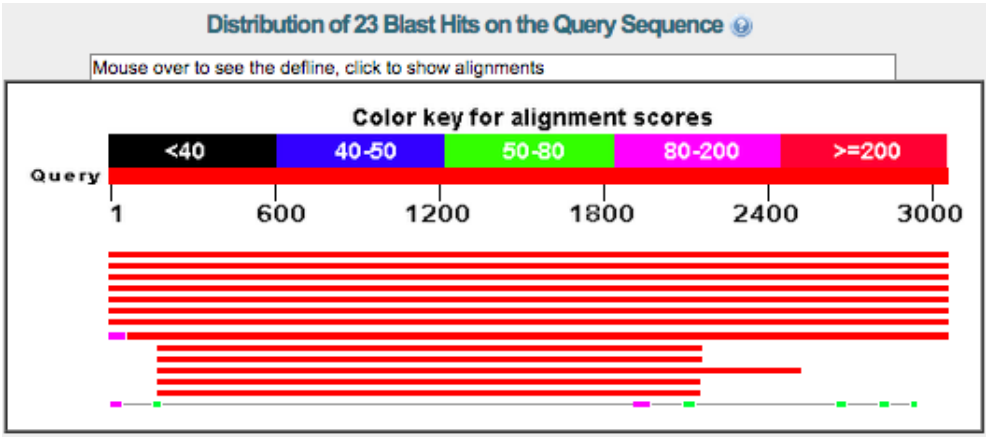
Tabela 5.1: Montagem *de novo* dos *reads* não mapeados

Cepa	Total de *reads*	Nº de *contigs*	L50	N50
01_A_91191	16000	1	1	306
02_A_05-567	16907	2	1	454
03_A_05-566	13663	2	1	540
04_A_4303	33074	140	61	356
05_A_534	13343	2	1	614
06_A_91193	13591	1	1	498
07_A_91192	15870	3	1	790
08_B_45-08B	9301	24	8	420
09_B_18-08C	22742	63	22	386
10_B_07-08	458	2	1	3077
12_B_61-09	35134	91	34	384
13_B_32-08	8641	17	6	562
14_B_49-09	14128	22	7	387
16_B_08-08BF2	10555	19	5	783
17_B_33	50688	2014	724	486
20_B_50	8460	20	8	414
26_B_128	360	3	2	627
28_B_35	740	5	2	795
30_B_268	7679	20	8	458
33_B_AN5	40025	1609	643	411
34_B_0822-11	17397	428	180	385

Os valores de N50 e L50 são medidas estatísticas auxiliares na avaliação da qualidade da montagem e são calculadas como segue. Ordena-se, de maneira decrescente, os *contigs* resultantes da montagem segundo o seu tamanho. Em seguida, soma-se os tamanhos dos *contigs* até que o valor desta somatória represente metade (50%) da soma total dos tamanhos dos *contigs* obtidos. O tamanho do último *contig* que colaborou para esta somatória será o valor de N50. O cálculo de L50 considera, para a mesma somatória e condição de parada definidas para N50, a quantidade de *contigs* que colaboraram para a somatória. Desta forma, valores maiores para N50 e menores para L50 são preferidos.

Os baixos valores de N50 apresentados na Tabela 5.1 apontam a má qualidade da montagem, o que pode ser justificada pela baixa quantidade de *reads* disponíveis além da baixa similaridade entre eles. A excessão se dá para a cepa 10_B_07-08, que possui um *contig* com 3.077 bases, para o qual o Blast, na busca por similaridade, retornou diversos *hits* de *Mycobacterium*, como mostra a Figura 5.2.

Após a recuperação dos *reads* de cada uma das cepas, estes, juntamente com o conjunto de *reads* não mapeados pelo Bowtie foram submetidos à montagem *de novo*. A Tabela 5.2 apresenta as quantidades de *reads* recuperados dos *gaps* e o total de *reads* disponíveis para a montagem além de apresentar o resultado da montagem dado pelo número de *contigs* resultantes além dos valores de N50 e L50.



(a) Melhores alinhamentos.

Sequences producing significant alignments:

Select: All None Selected:0

Alignments Download GenBank Graphics Distance tree of results

Description	Max score	Total score	Query cover	E value	Ident	Accession
Mycobacterium bovis strain 1595, complete genome	5683	5683	100%	0.0	100%	CP012095.1
Mycobacterium tuberculosis strain 96121, complete genome	5677	5677	100%	0.0	99%	CP009427.1
Mycobacterium africanum GM041182 complete genome	5672	5672	100%	0.0	99%	FR878060.1
Mycobacterium canettii CIPT 140070017 complete genome	5633	5633	100%	0.0	99%	FO203510.1
Mycobacterium canettii CIPT 140070008 complete genome	5627	5627	100%	0.0	99%	FO203508.1
Mycobacterium canettii CIPT 140060008 complete genome	5627	5627	100%	0.0	99%	FO203507.1
Mycobacterium canettii CIPT 140010059 complete genome	5627	5627	100%	0.0	99%	HE572590.1
Mycobacterium canettii CIPT 140070010 complete genome	5358	5459	100%	0.0	99%	FO203509.1
Mycobacterium kansasii 824, complete genome	1692	2960	64%	0.0	82%	CP009483.1
Mycobacterium kansasii ATCC 12478, complete genome	1688	2955	64%	0.0	82%	CP006835.1
Mycobacterium marinum E11 main chromosome genome	1587	1587	76%	0.0	79%	HG917972.2
Mycobacterium marinum M, complete genome	1587	1587	64%	0.0	81%	CP000854.1
Mycobacterium liflandii 128FXT, complete genome	1559	1559	64%	0.0	81%	CP003899.1
Mycobacterium tuberculosis EAI5/NITR206, complete genome	122	535	9%	3e-23	100%	CP005387.1

(b) Lista de hits dos melhores alinhamentos.

Figura 5.2: Resultado do Blast para o maior *contig* obtido na montagem dos *reads* não mapeados da cepa 10_B_07-08

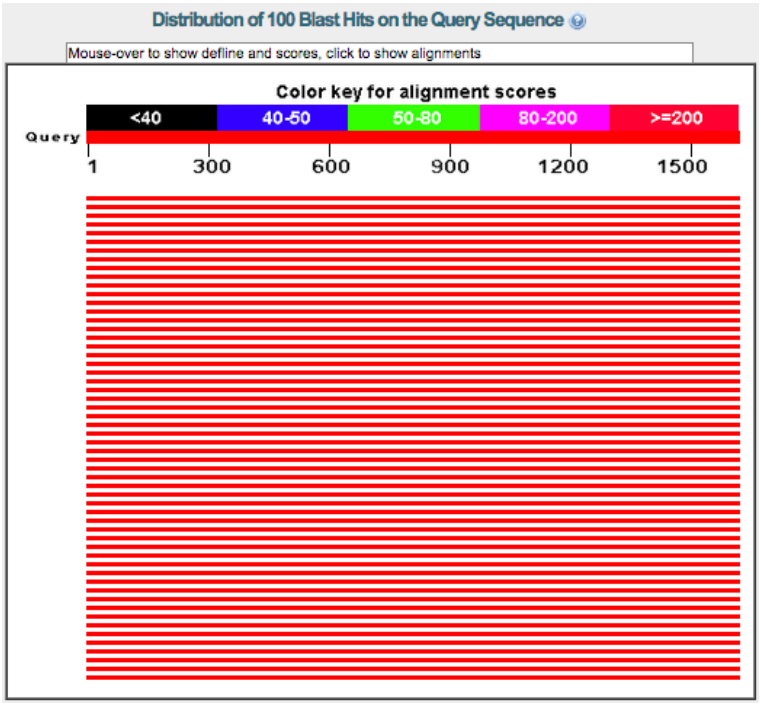
Como pode ser verificado na Tabela 5.2, exceto pela cepa 30_B_268, poucos *reads* foram recuperados e a influência destes na montagem *de novo* foi pouco significativa, visto que os valores de número total de *contigs*, L50 e N50 sofreram pouca ou nenhuma alteração. Conclui-se, assim, que os *reads* recuperados possuem pouca similaridade com os *reads* não mapeados e por isso, a utilização destes não colabora para esta etapa de montagem *de novo*.

Apesar dos valores da Tabela 5.2 não indicarem melhoras nas remontagens, uma busca sistemática nos *contigs* resultantes poderia revelar situações interessantes, como mostra a Figura 5.3, que apresenta o resultado da busca realizada pelo Blast para um *contig* de

Tabela 5.2: Montagem *de novo* dos *reads* não mapeados juntamente com os *reads* recuperados

Cepa	Nº de *reads* recuperados do mapeamento	Total de *reads*	Nº de *contigs*	L50	N50
01_A_91191	259	16.259	1	1	306
02_A_05-567	450	17357	2	1	454
03_A_05-566	304	13967	2	1	540
04_A_4303	340	33414	135	60	355
05_A_534	327	13670	2	1	614
06_A_91193	311	13902	1	1	454
07_A_91192	310	16180	3	2	360
08_B_45-08B	504	9805	25	8	420
09_B_18-08C	259	23001	63	22	380
10_B_07-08	467	925	3	1	2312
12_B_61-09	252	35386	90	33	393
13_B_32-08	471	9112	19	6	562
14_B_49-09	309	14437	23	9	387
16_B_08-08BF2	382	10397	19	6	579
17_B_33	780	51468	2015	722	487
20_B_50	293	8753	19	7	414
26_B_128	1352	1712	4	2	627
28_B_35	869	1609	9	4	491
30_B_268	11396	19075	864	335	430
33_B_AN5	666	40691	1616	643	413
34_B_0822-11	441	17838	421	177	385

tamanho 1.647 encontrado na remontagem *de novo* da cepa 13_B_32-08.



(a) Melhores alinhamentos.

Sequences producing significant alignments:

Select: All None Selected:0

Alignments Download GenBank Graphics Distance tree of results

Description	Max score	Total score	Query cover	E value	Ident	Accession
Influenza A virus (A/Singapore/DMS7/2009(H1N1)) segment 1 polymerase PB2 (PB2) gene, complete cds	3014	3014	100%	0.0	99%	KT180552.1
Influenza A virus (A/Singapore/DMS17/2009(H1N1)) segment 1 polymerase PB2 (PB2) gene, complete cds	3014	3014	100%	0.0	99%	KT180665.1
Influenza A virus (A/Singapore/DMS18/2009(H1N1)) segment 1 polymerase PB2 (PB2) gene, complete cds	3014	3014	100%	0.0	99%	KT180666.1
Influenza A virus (A/Singapore/DMS562/2009(H1N1)) segment 1 polymerase PB2 (PB2) gene, complete cds	3014	3014	100%	0.0	99%	KT180703.1
Influenza A virus (A/Singapore/DMS623/2009(H1N1)) segment 1 polymerase PB2 (PB2) gene, complete cds	3014	3014	100%	0.0	99%	KT180712.1
Influenza A virus (A/Singapore/DMS67/2010(H1N1)) segment 1 polymerase PB2 (PB2) gene, complete cds	3014	3014	100%	0.0	99%	KT180720.1
Influenza A virus (A/Jena/5258/2009(H1N1)) segment 1 polymerase PB2 (PB2) gene, complete cds	3014	3014	100%	0.0	99%	KJ549775.1
Influenza A virus (A/Santa Cruz/12541/2009(H1N1)) segment 1 polymerase PB2 (PB2) gene, complete cds	3014	3014	100%	0.0	99%	KF612037.1
Influenza A virus (A/Santa Cruz/46631/2009(H1N1)) segment 1 polymerase PB2 (PB2) gene, complete cds	3014	3014	100%	0.0	99%	KF612038.1
Influenza A virus (A/Santa Cruz/94841/2009(H1N1)) segment 1 polymerase PB2 (PB2) gene, complete cds	3014	3014	100%	0.0	99%	KF612039.1
Influenza A virus (A/Qingdao/1008/2009(H1N1)) segment 1 polymerase PB2 (PB2) gene, partial cds	3014	3014	100%	0.0	99%	KF411142.1
Influenza A virus (A/Uganda/MUWRP-066/2009(H1N1)) segment 1 polymerase PB2 (PB2) gene, complete cds	3014	3014	100%	0.0	99%	KJ690405.1
Influenza A virus (A/Netherlands/602-cHAP4/2009(H1N1)) polymerase PB2 (PB2) gene, complete cds	3014	3014	100%	0.0	99%	CY176941.1
Influenza A virus (A/Netherlands/602-cHAP3/2009(H1N1)) polymerase PB2 (PB2) gene, complete cds	3014	3014	100%	0.0	99%	CY176933.1
Influenza A virus (A/Netherlands/602-6F12LP5/2009(H1N1)) polymerase PB2 (PB2) gene, complete cds	3014	3014	100%	0.0	99%	CY176891.1

(b) Lista de hits dos melhores alinhamentos.

Figura 5.3: Resultado do Blast para o maior *contig* obtido na montagem dos *reads* não mapeados juntamente com os recuperados da cepa 13_B_32-08

Capítulo 6

Conclusões e Trabalhos Futuros

O mapeamento de *reads* em genomas de referência pode ser utilizado como um recurso para a montagem e anotação de organismos próximos. Entretanto, diferenças entre o organismo mapeado e o de referência podem existir e, assim, uma parcela dos *reads* pode não ser mapeada ou ainda, podem existir regiões com baixa cobertura de mapeamento. Tais diferenças podem ser explicadas, entre outros motivos, pela ocorrência de eventos de transferência lateral (HGTs),alta frequência de conteúdo GC e regiões variantes nestes organismos. Utilizando um conjunto de cepas do complexo *Mycobacterium bovis*, este trabalho propôs e testou uma estratégia de avaliação das regiões *aliens* presentes em genomas de referência e verificou forte correlação entre as regiões não mapeadas ou pouco mapeadas e as regiões apontadas como potenciais HGTs, ou simplesmente regiões *aliens*. As evidências encontradas comprovam que as regiões que apresetaram baixa cobertura localizam-se em regiões equivalentes às anômlas. Tal afirmação apresenta grande importância, pois pode nortear estudos sobre regiões de transferência lateral de genes, alta frequência de conteúdo GC e variantes, e colaborar no entendimento das diferenças genotípicas entre os organismos.

Muito pode-se fazer ainda dado o estudo inicial apresentado neste trabalho, dentre os quais destacamos: avaliação do genoma de referência por outras estratégias de determinação de regiões anômalas; análise e recuperação de *reads* mapeados em regiões bem cobertas que se localizam próximos ao início e fim de regiões pouco mapeadas, já que tais *reads* colaborariam para a determinação do posicionamento de *contigs* obtidos da montagem *de novo* em relação à montagem sugerida pelo mapeamento; realização de investigações em base de dados a fim de determinar a origem de *contigs* obtidos da montagem *de novo* dos *reads* não mapeados e, desta forma, esclarecer possíveis transferências laterais ocorridas entre os organismos em estudo e; estudo e implementação de uma ferramenta capaz de identificar a origem dos *reads* remontados em um *contig* para avaliação detalhada da influência dos *reads* recuperados de regiões pouco cobertas de um mapeamento.

Referências Bibliográficas

[1] ABI SOLiD. http://www.appliedbiosystems.com/absite/us/en/home/applications-technologies/solid-next-generation-sequencing.html. Accessado: 29-07-2014.

[2] Bioproject numero 214551. http://www.ncbi.nlm.nih.gov/bioproject/214551. Accessado: 04-02-2015.

[3] Bioproject numero PRJNA285833. http://www.ncbi.nlm.nih.gov/bioproject/PRJNA285833. Accessado: 10-08-2015.

[4] Download Alien Hunter. https://www.sanger.ac.uk/resources/software/alien_hunter/. Accessado: 13-09-2014.

[5] Exame microbiológico da tuberculose como subsídio à inspeção post-mortem de bovinos.

[6] Illumina/Solexa Genome Analyzer. http://www.illumina.com/. Accessado: 17-02-2015.

[7] Informações da anotação e dos genes utilizados na referência. ftp://ftp.ncbi.nlm.nih.gov/genomes/Bacteria/Mycobacterium_bovis_AF2122_97_uid57695/. Accessado: 24-05-2015.

[8] Ion Torrent^{TM}/Life Technology. http://www.lifetechnologies.com/br/en/home/brands/ion-torrent.html. Accessado: 29-07-2014.

[9] Manual Reference Pages - samtools (1). http://samtools.sourceforge.net/samtools.shtml#4. Accessado: 03-05-2015.

[10] National Human Genome Research Institute — DNA Sequencing Costs. http://www.genome.gov/sequencingcosts/. Accessado: 31-08-2015.

[11] NCBI—GenBang. http://www.ncbi.nlm.nih.gov/genbank/. Accessado: 17-02-2015.

[12] Novoalign. http://www.novocraft.com. Accessado: 26-08-2014.

[13] Roche GS-FLX 454 Genome Sequencer. http://454.com/. Accessado: 29-07-2014.

[14] Sequence Alignment/Map Format Specification. `http://samtools.github.io/hts-specs/SAMv1.pdf`. Accessado: 16-02-2015.

[15] C. Alkan, J. M. Kidd, T. Marques-Bonet, G. Aksay, F. Antonacci, F. Hormozdiari, J. O. Kitzman, C. Baker, M. Malig, O. Mutlu, S. C. Sahinalp, R. A. Gibbs, and E. E. Eichler. Personalized copy number and segmental duplication maps using next-generation sequencing. *Nature genetics*, pages 1061–1067, 2009.

[16] S. Batzoglou, D. B. Jaffe, K. Stanley, J. Butler, S. Gnerre, E. Mauceli, B. Berger, J. P. Mesirov, and E. S. Lander. Arachne: a whole-genome shotgun assembler. *Genome research*, 12(1), 2002.

[17] N. Biggs, E. K. Lloyd, and R. J. Wilson. *Graph Theory, 1736-1936*. Oxford University Press, 1976.

[18] M. Burrows and D. J. Wheeler. A block-sorting lossless data compression algorithm. Technical report, 1994.

[19] P. E. Compeau, P. A. Pevzner, and G. Tesler. How to apply de bruijn graphs to genome assembly. *Nature biotechnology*, 29(11), 2011.

[20] J. C. Dohm, C. Lottaz, T. Borodina, and H. Himmelbauer. Sharcgs, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. *Genome research*, 17(11), 2007.

[21] S. R. Eddy. Profile hidden markov models. *Bioinformatics*, 14(9), 1998.

[22] S. El-Metwally, T. Hamza, M. Zakaria, and M. Helmy. Next-generation sequence assembly: four stages of data processing and computational challenges. 2013.

[23] H. Esquivel-Solís, A. J. Vallecillo, A. Benítez-Guzmán, L. G. Adams, Y. López-Vidal, and J. A. Gutiérrez-Pabello. Nitric oxide not apoptosis mediates differential killing of mycobacterium bovis in bovine macrophages. *PloS one*, 8(5), 2013.

[24] R. Fenouil, P. Cauchy, F. Koch, N. Descostes, J. Z. Cabeza, C. Innocenti, P. Ferrier, S. Spicuglia, M. Gut, I. Gut, et al. Cpg islands and gc content dictate nucleosome depletion in a transcription-independent manner at mammalian promoters. *Genome research*, 22(12), 2012.

[25] P. Ferragina and G. Manzini. Opportunistic data structures with applications. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 390, Washington, DC, USA, 2000. IEEE Computer Society.

[26] J. S. Fraga. Algoritmos Genéticos e o Problema da Remontagem de Reads. 2014.

[27] P. M. Gontarz, J. Berger, and C. F. Wong. SRmapper: a fast and sensitive genome-hashing alignment tool. *Bioinformatics*, 2013.

[28] R. R. Gonzalez, R. Bonnal, M. Caccamo, and D. MacLean. Bio-samtools: Ruby bindings for SAMtools, a library for accessing BAM files containing high-throughput sequence alignments. *Source Code for Biology and Medicine*, 7(1):6, 2012.

[29] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, New York, USA, 1997.

[30] F. Hach, F. Hormozdiari, C. Alkan, F. Hormozdiari, I. Birol, E. E. Eichler, and S. C. Sahinalp. mrsFAST: a cache-oblivious algorithm for short-read mapping. *Nature methods*, 2010.

[31] J. Hacker and J. B. Kaper. Pathogenicity islands and the evolution of microbes. *Annual Reviews in Microbiology*, 54(1), 2000.

[32] G. H. Hardy. Mendelian proportions in a mixed population. *Science*, 28:49–50, 1908.

[33] A. Hatem and D. Bozda. Benchmarking short sequence mapping tools. *BMC*, 2013.

[34] U. Hentschel and J. Hacker. Pathogenicity islands: the tip of the iceberg. *Microbes and infection*, 3(7), 2001.

[35] N. Homer, B. Merriman, and S. F. Nelson. BFAST: An Alignment Tool for Large Scale Genome Resequencing. *PLoS ONE*, 2009.

[36] T. Horiike, D. Miyata, Y. Tateno, and R. Minai. Hgt-gen: a tool for generating a phylogenetic tree with horizontal gene transfer. *Bioinformation*, 7(5), 2011.

[37] X. Huang and A. Madan. Cap3: A dna sequence assembly program. *Genome research*, 9(9), 1999.

[38] V. D. A. Jaramillo, S. A. Sukno, and M. R. Thon. Identification of horizontally transferred genes in the genus colletotrichum reveals a steady tempo of bacterial to fungal gene transfer. *BMC Genomics*, 16(1), 2015.

[39] W. R. Jeck, J. A. Reinhardt, D. A. Baltrus, M. T. Hickenbotham, V¿ Magrini, E. R. Mardis, J. L. Dangl, and C. D. Jones. Extending assembly of short dna sequences to handle error. *Bioinformatics*, 23(21), 2007.

[40] S. Karlin. Detecting anomalous gene clusters and pathogenicity islands in diverse bacterial genomes. *Trends in microbiology*, 9(7), 2001.

[41] S. Karlin, J. Mrázek, and A. M. Campbell. Codon usages in different gene classes of the escherichia coli genome. *Molecular microbiology*, 29(6), 1998.

[42] K. Kitahara and K. Miyazaki. Revisiting bacterial phylogeny: Natural and experimental evidence for horizontal gene transfer of 16s rrna. *Mobile genetic elements*, 3(1), 2013.

[43] S. Knapp, J. Hacker, T. Jarchau, and W. Goebel. Large, unstable inserts in the chromosome affect virulence properties of uropathogenic escherichia coli o6 strain 536. *Journal of bacteriology*, 168(1), 1986.

[44] E. V. Koonin and Y. I. Wolf. Genomics of bacteria and archaea: the emerging dynamic view of the prokaryotic world. *Nucleic acids research*, 36(21), 2008.

[45] B. Langmead and S. L. Salzberg. Fast gapped-read alignment with bowtie 2. *Nat. Methods*, 9(4):357–359, April 2012.

[46] B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome biology*, 2009.

[47] J. G. Lawrence. Gene transfer in bacteria: speciation without species? *Theoretical population biology*, 61(4), 2002.

[48] J. G. Lawrence and H. Ochman. Amelioration of bacterial genomes: rates of change and exchange. *Journal of molecular evolution*, 44(4), 1997.

[49] C. E. Leiserson, C. Stein, R. L. Rivest, and T. H. Cormen. *Algoritmos: teoria e prática*. CAMPUS - RJ, 2002.

[50] A. M. Lesk. *Introdução à bioinformática*. Artmed, 2008.

[51] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, and R. Durbin. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, August 2009.

[52] H. Li, J. Ruan, and R. Durbin. Mapping short dna sequencing reads and calling variants using mapping quality scores. *Genome Res*, 2008.

[53] R. Li, C. Yu, Y. Li, T. Lam, S. Yiu, K. Kristiansen, and J. Wang. SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics*, 2009.

[54] A. Masoudi-Nejad, Z. Narimani, and N. Hosseinkhan. *Next Generation Sequencing and Sequence Assembly*. SpringerBriefs in Systems Biology. Springer New York, New York, NY, 2013.

[55] E. M. McCreight. A space-economical suffix tree construction algorithm. *J. ACM*, pages 262–272, 1976.

[56] A. L. Michel, B. Müller, and P. D. Van Helden. Mycobacterium bovis at the animal–human interface: A problem, or not? *Veterinary microbiology*, 140(3), 2010.

[57] J. R Miller, S. Koren, and G. Sutton. Assembly algorithms for next-generation sequencing data. *Genomics*, 2010.

[58] I. Milne, G. Stephen, M. Bayer, P. J. Cock, L. Pritchard, L. Cardle, P. D. Shaw, and D. Marshall. Using tablet for visual exploration of second-generation sequencing data. *Briefings in bioinformatics*, 2012.

[59] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, 1965.

[60] E. W. Myers. Toward simplifying and accurately formulating fragment assembly. *Journal of Computational Biology*, 2(2), 1995.

[61] J. Nash. Disponibilização do código da ferramenta utilizada — Public Health Agency of Canada. `https://nash-bioinformatics-codelets.googlecode.com/files/pileup2fasta.pl`. Accessado: 22-02-2015.

[62] I. A. S. Oliveira, H. P. C. Melo, A. Câmara, R. V. C. da Dias, and B. Soto-Blanco. Prevalência de tuberculose no rebanho bovino de mossoró, rio grande do norte. *Brazilian Journal of Veterinary Research and Animal Science*, 44(6), 2007.

[63] M. V. Palmer. Mycobacterium bovis: Characteristics of wildlife reservoir hosts. *Transboundary and Emerging Diseases*, 60:1–13, 2013.

[64] Vania R. M., Patrick D., Lluis Q. M., Roland B., Brigitte G., and Olivier N. Horizontal transfer of a virulence operon to the ancestor of mycobacterium tuberculosis. *Molecular biology and evolution*, 23(6), 2006.

[65] E. Rivals, L. Salmela, P. Kiiskinen, P. Kalsi, and J. Tarhio. Mpscan: fast localisation of multiple reads in genomes. 2009.

[66] U. L. Rosewich and H. C. Kistler. Role of horizontal gene transfer in the evolution of fungi 1. *Annual review of phytopathology*, 38(1), 2000.

[67] F. Sanger. Determination of nucleotide sequences in DNA. *Bioscience reports*, 1980.

[68] S. Schaack, C. Gilbert, and C. Feschotte. Promiscuous dna: horizontal transfer of transposable elements and why it matters for eukaryotic evolution. *Trends in ecology & evolution*, 25(9), 2010.

[69] S. Schbath, V. Martin, M. Zytnicki, J. Fayolle, V. Loux, and J. Gibrat. Mapping reads on a genomic sequence: an algorithmic overview and a practical comparative analysis. *Journal of computational biology : a journal of computational molecular cell biology*, pages 796–813, 2012.

[70] J.C. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology*. PWS-KENT Publishing Company, 1997.

[71] R. Shamir. Lecture 12: Algorithms for Next Generation Sequencing Data, 2011. Lecture note, Computational Genomics, Fall Semester, 2011.

[72] J. Simpson, K. Wong, S. Jackman, J. Schein, S. Jones, and I. Birol. Abyss: a parallel assembler for short read sequence data. *Genome Res*, 2009.

[73] Petr Šmarda, Petr Bureš, Lucie Horová, Ilia J Leitch, Ladislav Mucina, Ettore Pacini, Lubomír Tichỳ, Vít Grulich, and Olga Rotreklová. Ecological and evolutionary significance of genomic gc content diversity in monocots. *Proceedings of the National Academy of Sciences*, 111(39), 2014.

[74] Andrew D. Smith, Wen-Yu Chung, Emily Hodges, Jude Kendall, Greg Hannon, James Hicks, Zhenyu Xuan, and Michael Q. Zhang. Updates to the RMAP short-read mapping software. *Bioinformatics*, pages 2841–2842, 2009.

[75] C. Stern. The hardy-weinberg law. *Science*, 97(2510), 1943.

[76] G. G. Sutton, O. White, M. D. Adams, and A. R. Kerlavage. TIGR Assembler: A New Tool for Assembling Large Shotgun Sequencing Projects. *Genome Science and Technology*, 1995.

[77] J. L. Szwarcfiter. *Grafos e algoritimos computacionais*, volume 2. Campus, 1988.

[78] Guilherme P Telles, Nalvo F Almeida, and Fábio H Viduani Martinez. Algoritmos e heurısticas para comparaçoes exata e aproximada de seq uências. *XXIV Jornadas de Atualizaçao em Informática. A. Loureiro and M. Barcelos.(Org.)*, 2005.

[79] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, pages 249–260, 1995.

[80] L. Varuzza. Introdução à análise de dados de sequenciadores de nova geração. 2013.

[81] G. S. Vernikos and J. Parkhill. Interpolated variable order motifs for identification of horizontally acquired dna: revisiting the salmonella pathogenicity islands. *Bioinformatics*, 2006.

[82] M. A. M. Vieira. Ilhas de patogenicidade. *O Mundo da Saúde. São Paulo*, 33(4), 2009.

[83] A. Vignal, D. Milan, M. SanCristobal, and A. Eggen. A review on snp and other types of molecular markers and their use in animal genetics. *Genetics Selection Evolution*, 34(3), 2002.

[84] R. L. Warren, G. G. Sutton, S. J. M. Jones, and R. A. Holt. Assembling millions of short dna sequences using ssake. *Bioinformatics*, 23(4), 2007.

[85] P. Weiner. Linear pattern matching algorithms. In *Proceedings of the 14th Annual Symposium on Switching and Automata Theory (Swat 1973)*, SWAT '73, pages 1–11, Washington, DC, USA, 1973. IEEE Computer Society.

[86] Y. I. Wolf, K. S. Makarova, N. Yutin, and E. V. Koonin. Updated clusters of orthologous genes for archaea: a complex ancestor of the archaea and the byways of horizontal gene transfer. *Biol Direct*, 7, 2012.

[87] T. D. Wu and S. Nacu. Fast and snp-tolerant detection of complex variants and splicing in short reads. *Bioinformatics*, 2010.

[88] H. Xin, D. Lee, F. Hormozdiari, S. Yedkar, O. Mutlu, and C. Alkan. Accelerating read mapping with fasthash. *BMC Genomics*, page S13, 2013.

[89] D. R. Zerbino and E. Birney. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*, 2008.

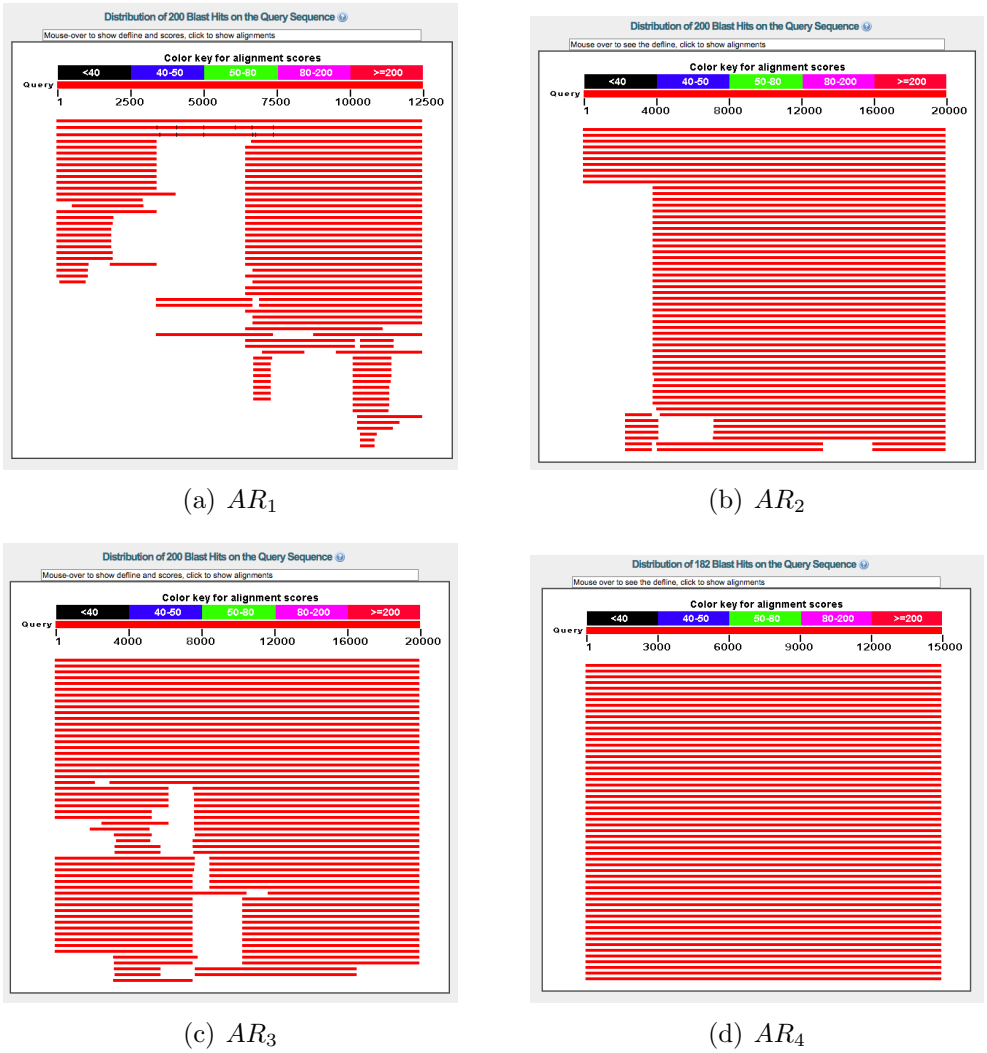
[90] J. Zhang, R. Chiodini, A. Badr, and G. Zhang. The impact of next-generation sequencing on genomics. *Journal of genetics and genomics*, 2011.

[91] Z. Zhang, S. Schwartz, L. Wagner, and W. Miller. A greedy algorithm for aligning dna sequences. *Journal of Computational biology*, 7(1-2), 2000.

[92] Q. Zhu, M. Kosoy, and K. Dittmar. Hgtector: An automated method facilitating genome-wide discovery of putative horizontal gene transfers. *BMC genomics*, 15(1), 2014.

Apêndice A

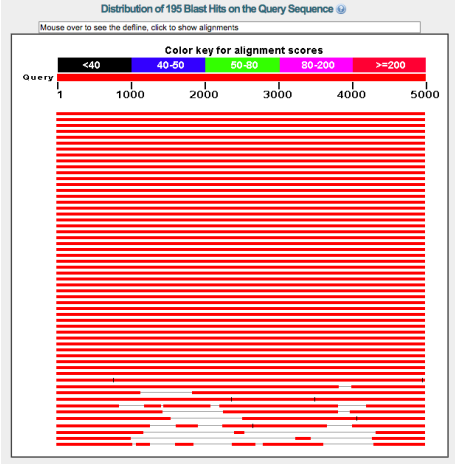
Blastn das Regiões *Aliens*



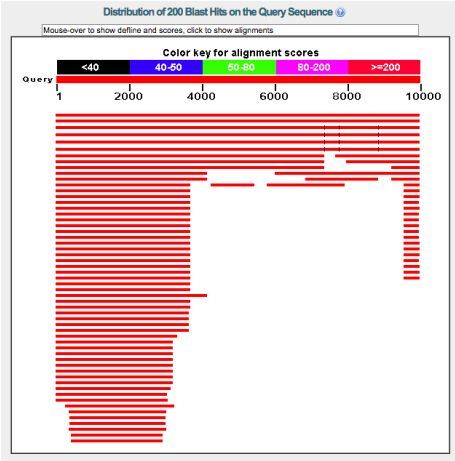
(a) AR_1 (b) AR_2

(c) AR_3 (d) AR_4

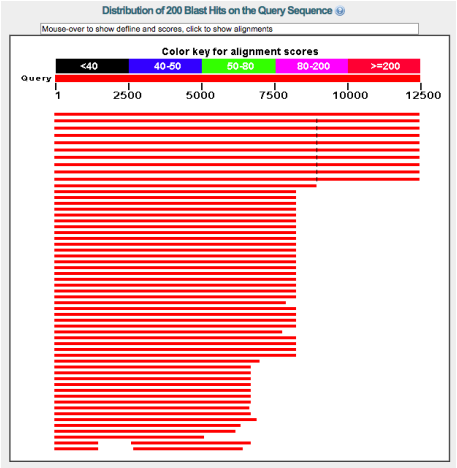
Figura A.1: *Overview* dos alinhamentos retornados pelo Blastn para as regiões *aliens* de AR_1 a AR_6.



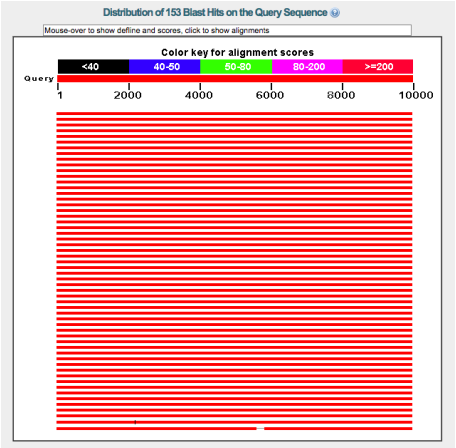
(a) AR_5



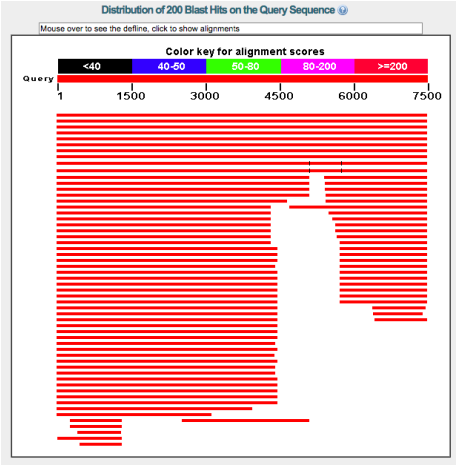
(b) AR_6



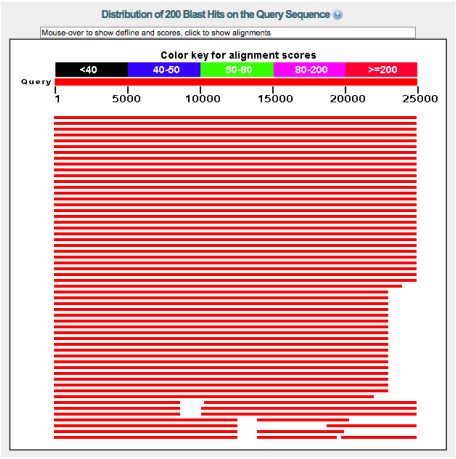
(c) AR_7



(d) AR_8

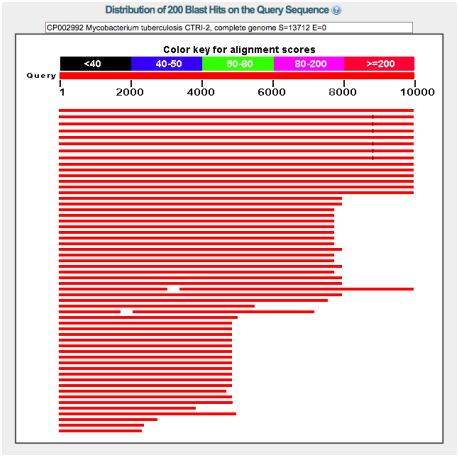


(e) AR_9

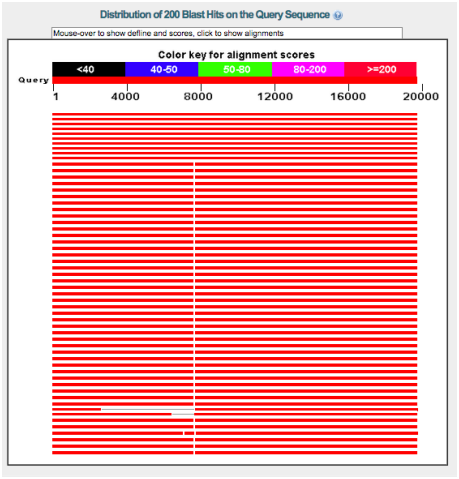


(f) AR_{10}

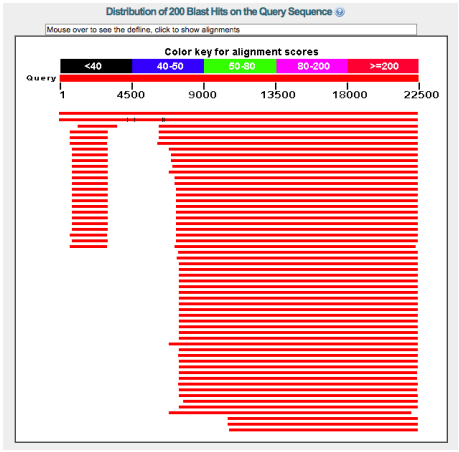
Figura A.2: *Overview* dos alinhamentos retornados pelo Blastn para as regiões *aliens* de AR_5 a AR_{10}.



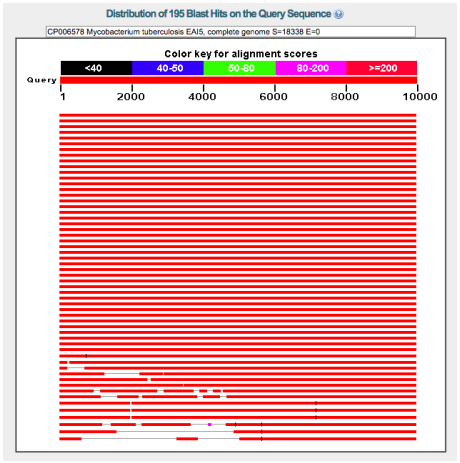
(a) AR_{11}



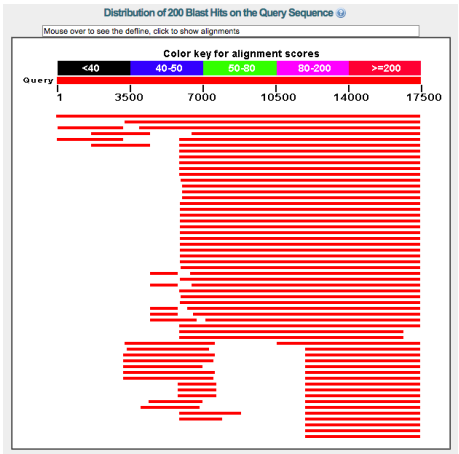
(b) AR_{12}



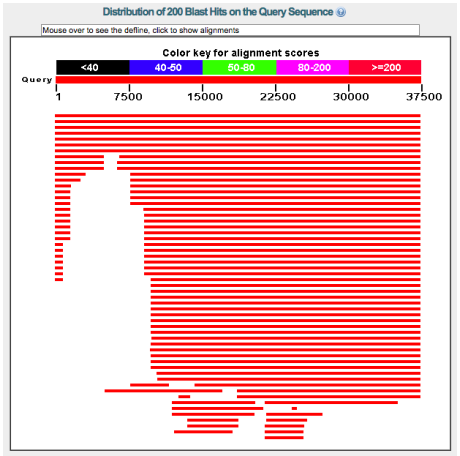
(c) AR_{13}



(d) AR_{14}

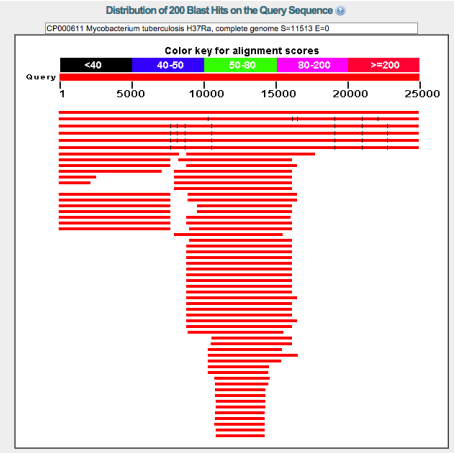


(e) AR_{15}



(f) AR_{16}

Figura A.3: *Overview* dos alinhamentos retornados pelo Blastn para as regiões *aliens* de AR_{11} a AR_{16}.



(a) AR_{17}

Figura A.4: *Overview* dos alinhamentos retornados pelo Blastn para a região *alien* AR_{17}

Tabela A.1: Descrição das 17 regiões *aliens* para as quais $ah_{score} > 2*th$ e seus respectivos genes.

Região *Alien*	ah_{score}	Posição inicial	Posição final	Descrição dos genes presentes na região
AR_1	49,9	330.000	342.500	hypothetical protein / PE-PGRS family protein / PPE family protein
AR_2	35,0	360.000	380.000	hypothetical protein / sulfatase / PE-PGRS family protein / TetR/ACRR family transcriptional regulator / dehydrogenase / reductase / PPE family protein/oxidoreductase
AR_3	32,3	417.500	437.500	hypothetical protein / transcriptional regulator / molecular chaperone DnaK / heat shock protein GrpE / molecular chaperone DnaJ / HEAT shock protein transcriptional regulator HspR / PPE family protein / adenylosuccinate synthetase
AR_4	36,7	480.000	495.000	acyl-CoA dehydrogenase / transmembrane protein / transmembrane transport protein MmpL1B / transmembrane transport protein MmpL1A / hypothetical protein / acyl-CoA synthetase / polyketide synthase / beta lactamase like protein / F420-dependent glucose-6-phosphate dehydrogenase / phosphate acetyltransferase / acetate kinase
AR_5	45,9	672.500	677.500	hypothetical protein / PE-PGRS family protein
AR_6	45,5	835.000	845.000	PE-PGRS family protein / hypothetical protein / transcriptional regulator / PE-PGRS family protein / 3-hydroxyisobutyrate dehydrogenase
AR_7	33,3	920.000	932.500	hypothetical protein / transcriptional regulator / deaminase / transposase / PE-PGRS family protein / lipoprotein LpqQ
AR_8	51,9	1.100.000	1.110.000	serine protease / pterin-4-alpha-carbinolamine dehydratase / large-conductance mechanosensitive channel / adhesion component transport ATP-binding protein ABC transporter / adhesion component transport transmembrane protein / hypothetical protein / polyprenyl synthetase / serine-rich protein / UTP-glucose-1-phosphate uridylyltransferase
AR_9	39,7	1.187.500	1.195.000	lipoprotein LPQV / hypothetical protein / PE-PGRS family protein
AR_{10}	38,1	1.457.500	1.482.500	undecapaprenyl-phosphate alpha-n-acetylglucosaminyltransferase rfe / hypothetical protein / ATP synthase F0F1 subunit A / ATP synthase F0F1 subunit B / ATP synthase F0F1 subunit C / ATP synthase F0F1 subunit delta / ATP synthase F0F1 subunit alpha / ATP synthase F0F1 subunit gamma / ATP synthase F0F1 subunit beta / ATP synthase F0F1 subunit epsilon / UDP-N-acetylglucosamine 1-carboxyvinyltransferase / methylated-DNA–protein-cysteine methyltransferase / adenylate cyclase
AR_{11}	45,8	1.625.000	1.635.000	transketolase / PE-PGRS family protein / protoheme IX farnesyltransferase / PE-PGRS family protein
AR_{12}	36,2	1.685.000	1.705.000	esterase / methyltransferase / hypothetical protein / glycosyltransferase / TDP-4-oxo-6-deoxy-D-glucose transaminase / sugar transferase / acyl-CoA synthetase / transmembrane transport protein MmpL12 / rhamnosyl transferase WbbL2
AR_{13}	44,5	2.150.000	2.172.500	hypothetical protein / isocitrate lyase / PPE family protein / lipoprotein LppF / lipoprotein / lipase LIPD / fatty-acid–CoA ligase / short chain dehydrogenase
AR_{14}	40,6	2.767.500	2.777.500	LuxR family transcriptional regulator / hypothetical protein / branched-chain alpha-keto acid dehydrogenase E2 subunit / pyruvate dehydrogenase E1 component subunit beta PdhB
AR_{15}	36,8	3.072.500	3.090.000	hypothetical protein / transposase /
AR_{16}	40,4	3.685.000	3.722.500	homoserine O-acetyltransferase / methyltransferase / PPE family protein / hypothetical protein / transposase / oxidoreductase
AR_{17}	62,9	3.870.000	3.895.000	acyl-CoA dehydrogenase / acyl-CoA synthetase / PE-PGRS family protein / hypothetical protein / PE-PGRS family protein / fatty-acid–CoA ligase /

Apêndice B

Regiões *aliens*, *gaps* e genes

As figuras a seguir apresentam trechos de regiões *aliens* onde *gaps* ocorreram no mapeamento de cepas de *M. bovis*. Em tais trechos, os genes existentes são também representados.

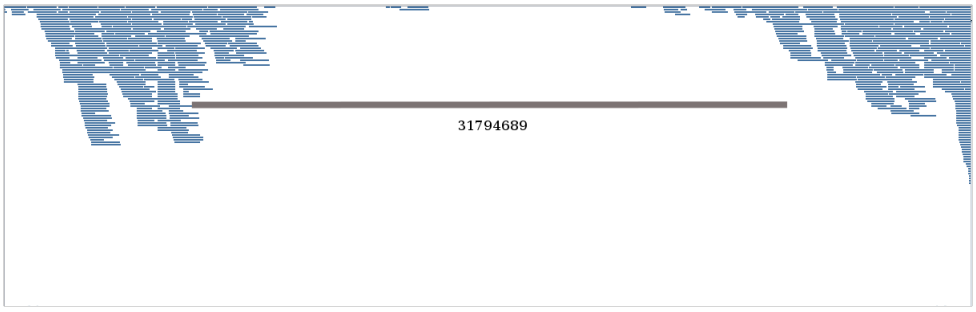


Figura B.1: Intervalo entre as bases 3.889.569 e 3.894.392 contindo na região *alien* AR_{17} referente ao mapeamento da cepa 04_A_4303. O gene em destaque possui as coordenadas de 3.890.501 a 3.893.479 na referência.

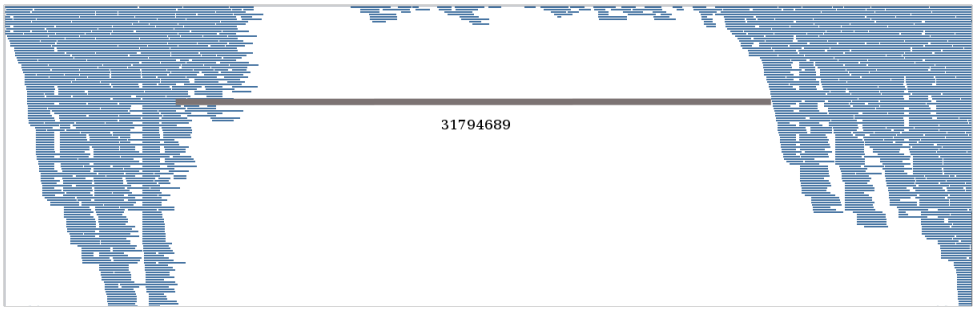


Figura B.2: Intervalo entre as bases 3.889.649 e 3.894.472 contindo na região *alien* AR_{17} referente ao mapeamento da cepa 05_A_534. O gene em destaque possui as coordenadas de 3.890.501 a 3.893.479 na referência.

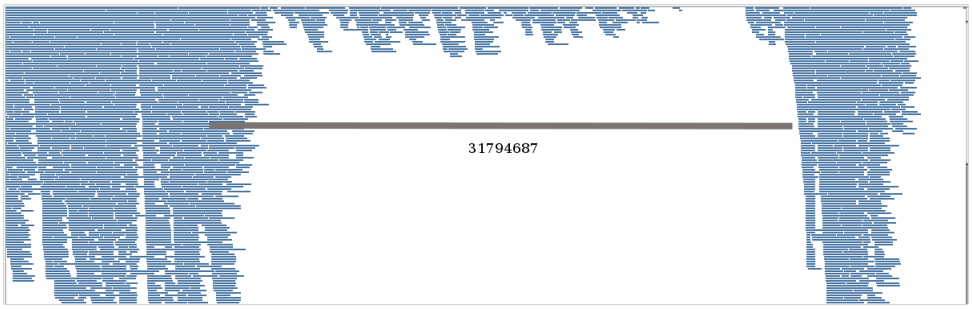


Figura B.3: Intervalo entre as bases 3.881.817 a 3.891.400 contido na região *alien* AR_{17} referente ao mapeamento da cepa 08_B_45-08B. O gene em destaque possui as coordenadas de 3.883.854 a 3.889.670 na referência.

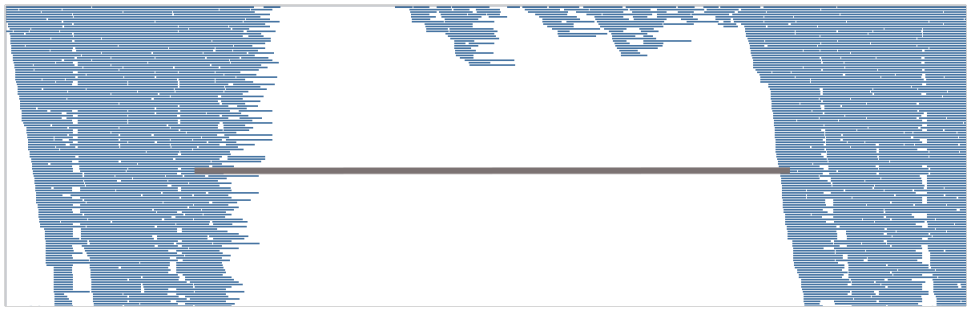


Figura B.4: Intervalo entre as bases 3.889.569 a 3.894.356 contido na região *alien* AR_{17} referente ao mapeamento da cepa 09_B_18-08C. O gene em destaque possui as coordenadas de 3.890.501 a 3.893.479 na referência.

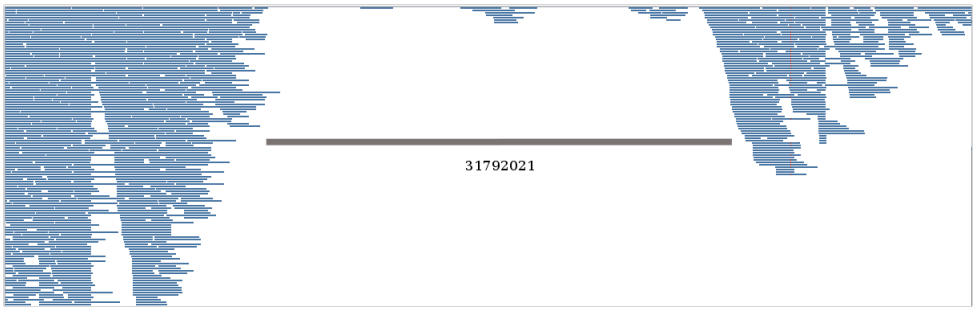


Figura B.5: Intervalo entre as bases 924.553 a 929.376 contido na região *alien* AR_7 referente ao mapeamento da cepa 13_B_32-08. O gene em destaque possui as coordenadas de 926.191 a 928.512 na referência.

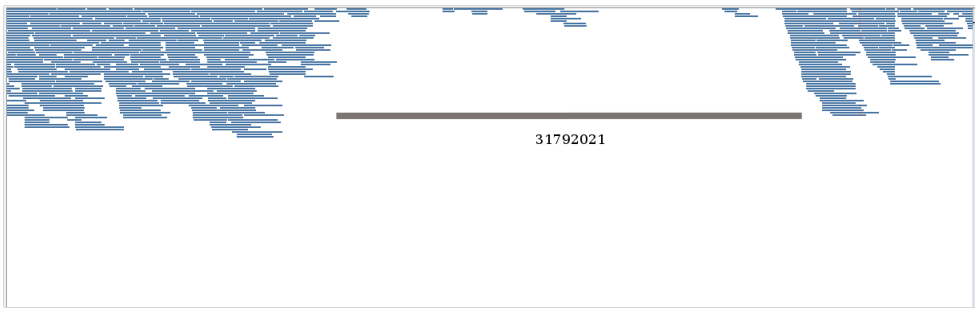


Figura B.6: Intervalo entre as bases 924.552 a 929.376 contido na região *alien* AR_7 referente ao mapeamento da cepa 16_B_08-08BF2. O gene em destaque possui as coordenadas de 926.191 a 928.512 na referência.

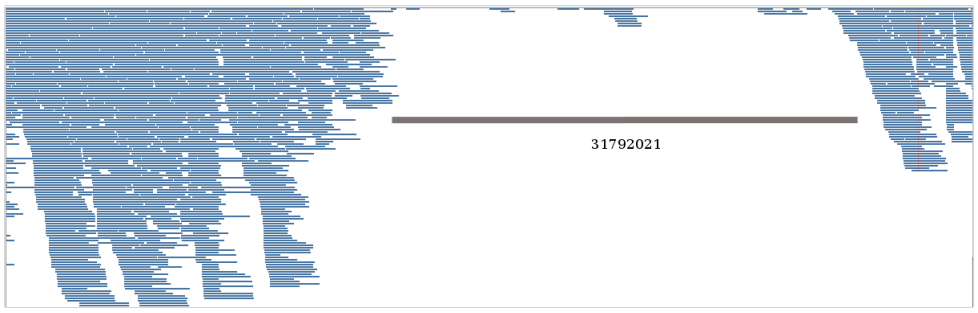


Figura B.7: Intervalo entre as bases 924.261 a 929.084 contido na região *alien* AR_7 referente ao mapeamento da cepa 34_B_0822-11. O gene em destaque possui as coordenadas de 926.191 a 928.512 na referência.