Inovações em técnicas de alinhamentos múltiplos e predições de genes

Vitor Ferreira Onuchic

Dissertação apresentada
ao
Programa Interunidades em Bioinformática
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa: Bioinformática

Orientador: Prof. Dr. Alan Mitchell Durham

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro da FAPESP

São Paulo, agosto de 2012

Inovações em técnicas de alinhamentos múltiplos e predições de genes

Esta versão da dissertação contém as correções e alterações sugeridas pela Comissão Julgadora durante a defesa da versão original do trabalho, realizada em 05/10/2012. Uma cópia da versão original está disponível no Instituto de Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

- Prof. Dr. Alan Mitchell Durham (orientador) - IME-USP
- Prof. Dr. Fernando Portella de Luna Marques - IB-USP
- Prof. Dr. João Carlos Setubal - IQ-USP

Resumo

ONUCHIC, V. F. **Inovações em técnicas de alinhamentos múltiplos e predições de genes**. 2012. Dissertação - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2010.

Neste trabalho foram abordados dois problemas: alinhamento de sequências, e predição de genes. Estes dois problemas, apesar de serem distintos, tem grande relação entre si. Assim, uma proposta deste trabalho foi uma técnica, baseada em transformações de consistência, que permite a integração entre os modelos pairHMM (utilizado para alinhamento entre pares de sequências) e GHMM (utilizado em predição de genes), de maneira que as predições de genes para um conjunto de sequências genômicas sejam consistentes com o alinhamento entre essas sequências. Além disso apresentamos dois novos algoritmos relacionados à predição de genes utilizando GHMMs. O primeiro deles permite o cálculo de probabilidades *a posteriori* de cada posição da sequência ter sido emitida por cada um dos estados do GHMM. Isso nos permite saber para uma determinada predição de genes, a probabilidade *a posteriori* da predição para cada posição estar correta segundo o modelo utilizado. Mostramos que esse valor pode ser utilizado como uma medida de confiabilidade das predições. O segundo algoritmo relacionado à predição de genes utiliza essas probabilidades *a posteriori* para construir a predição de genes consistente de máxima precisão esperada. Esse algoritmo visa maximiza o número esperado de posições corretamente preditas, e é essencial para nossa abordagem que combina alinhamentos e predição de genes. Em relação ao problema de alinhamento de múltiplas sequências, apresentamos duas modificações em técnicas de construção de alinhamentos múltiplos: *sequence annealing* e transformação de consistência. As modificações propostas para essas duas técnicas visam melhorar o posicionamento de gaps no alinhamento, fator que é muitas vezes negligenciado no momento da construção dos alinhamentos, assim como na avaliação da qualidade dos mesmos. Mostramos que essas modificações levam a melhoras estatisticamente significativas em relação às técnicas originais. Para tornar evidentes essas melhorias, utilizamos *benchmarks* amplamente utilizados na avaliação de ferramentas de alinhamentos múltiplos.

Palavras-chave: alinhamento múltiplo, predição de genes, consistência, máxima precisão esperada.

Sumário

Notações

- Letras minúsculas em negrito (*e.g.* $\mathbf{x}$): sequências de caracteres (bases nitrogenadas para DNA, e amino-ácidos para proteínas)
- Letras minúsculas com subscrito (*e.g.* x_i): posições das sequências (posição i da sequência x)
- $x_i \sim y_j$: posições alinhadas entre as sequências (posição i da sequência $\mathbf{x}$ alinhada com posição j da sequência $\mathbf{y}$)
- $x_i \sim y_-$: posição alinhada com um gap qualquer em outra sequência (x_i alinhada com um gap em alguma posição da sequência $\mathbf{y}$)
- $x_i \sim y_{-_j}$: posição alinhada com um gap específico em outra sequência (x_i alinhada com um gap na posição anterior a j da sequência $\mathbf{y}$)
- $\forall x_i \sim y_j \in A$: todas os pares de posições alinhadas que fazem parte do alinhamento A.
- $\forall x_i \sim y_- \in A$: todos os pares posição-gap que fazem parte do alinhamento A.
- $\forall x_i \sim y_j$: todos os pares de posições que podem estar alinhados.
- $\forall x_i \sim y_{-_j}$: todos os pares posição-gap que podem estar alinhados.
- $x_i \leftrightarrow y_j$: a posição i da sequência $\mathbf{x}$ é da mesma classe de estrutura gênica que a posição j da sequência $\mathbf{y}$.
- $P(\mathbf{x} <> \mathbf{y})$: probabilidade média dos pares alinhados no alinhamento ótimo entre duas sequências. (Definido na equação 2.10)

Capítulo 1

Introdução

Uma vez que podemos abordar a evolução a partir do mecanismo de modificação da sequência nucleotídica dos genomas, podemos modelar o processo evolutivo como um processo de edição, com inserções, alterações e remoções de sequências de ácidos nucleicos. Para estimar a distância evolutiva e extrair informações a respeito da conservação de diferentes posições de um conjunto de sequências homólogas devemos, portanto, partir de uma alinhamento de suas bases. Desta maneira, algoritmos de alinhamento de múltiplas sequências (alinhamento múltiplo) têm sido parte fundamental destes estudos (Kemena e Notredame , 2009). Esse tipo de algoritmo visa, em geral, parear caracteres homólogos, sendo eles nucleotídeos ou aminoácidos. Esse problema se torna mais complicado devido à mutações, inserções e deleções que acontecem nas sequências no curso da evolução de cada organismo.

O alinhamento de múltiplas sequências é essencial para o estudo de filogenias moleculares, anotação de genomas e estudos das pressões seletivas que agem sobre diferentes regiões dos genomas. Assim, a qualidade do alinhamento tem grande influência sobre os resultados e possíveis conclusões de estudos subsequentes (Wong *et al.* , 2008).

A construção de um alinhamento múltiplo que seja ótimo segundo um critério biológico de similaridade qualquer é um problema NP-completo. Este fato implica que a possibilidade de calcular esse alinhamento ótimo de maneira exata só exista para um conjunto de sequências tão pequeno que não seria útil para nenhuma aplicação real. Assim, alinhamentos de múltiplas sequências são sempre construídos utilizando algum tipo de heurística (Kemena e Notredame , 2009).

Em geral as técnicas conhecidas até o momento constroem o alinhamento múltiplo a partir do alinhamento entre os pares de sequências (Do *et al.* , 2005; Notredame e et al. , 2000; Paten *et al.* , 2009; Thompson *et al.* , 1994), sub-sequências (Blanchette *et al.* , 2004), ou caracteres (Bradley *et al.* , 2009; Sahraeian e Yoon , 2010) contidos no conjunto que desejamos alinhar. Isto é feito pois é possível calcular pontuações, ou valores de probabilidades, para esses pares de maneira eficiente (Durbin *et al.* , 1998).

Dentre estas técnicas de construção de alinhamentos múltiplos, daremos destaque neste trabalho para uma delas, a chamada *sequence annealing*. Faremos isso pois as ferramen-

tas que implementam esta técnica são as que têm apresentado os melhores resultados nos principais *benchmarks* de comparação de alinhamentos múltiplos (Bradley *et al.* , 2009; Sahraeian e Yoon , 2010).

Observamos que apesar destes bons resultados, apenas uma das implementações modela o posicionamento de gaps de maneira explícita. Outras ferramentas tratam esse posicionamento como se estivesse implícito no posicionamento correto dos elementos das sequências sendo alinhadas, o que pode levar a erros no alinhamento (Bradley *et al.* , 2009). Neste trabalho apresentamos uma nova técnica para tratar o posicionamento correto de gaps, quando utiliza-se a técnica de *sequence annealing*. Veremos que essa modificação de fato melhora a qualidade do alinhamento se levarmos em conta que os gaps devem estar corretamente posicionados.

A construção de alinhamentos múltiplos a partir de pares de sequências, apesar de ser essencial para a viabilidade computacional, apresenta alguns problemas. Em particular, esses algoritmos não fazem uso da informação completa contida no conjunto de sequências. Esse é um problema inerente da construção baseada nos pares de sequências, já que se estivéssemos tratando de todas as sequências simultaneamente, como seria o ideal, o alinhamento considerado ótimo provavelmente seria outro. Para contornar esse problema são utilizadas técnicas secundárias que levam a um refinamento dos alinhamentos finais.

Neste trabalho, apresentaremos algumas modificações na formulação original de uma dessas metodologias secundárias, chamada transformação de consistência, visando melhorar ainda mais a qualidade dos alinhamentos múltiplos. Esta modificação, assim como a alteração na construção por *sequence annealing*, envolve um tratamento mais adequado para o alinhamento entre os elementos das sequências e gaps. Veremos que essa nova formulação leva a melhorias estatisticamente significativas nos alinhamentos.

Como mencionamos anteriormente, definir quais são os caracteres homólogos em um conjunto de sequências pode ter diversas aplicações. Uma área que teve grandes benefícios advindos das inferências feitas pelos algoritmos de alinhamento foi a de anotação de genomas. Em particular, foi possível utilizar informações sobre homologia com sequências previamente anotadas para definir a anotação de novas sequências. No caso da predição da estrutura interna de genes, isso nem sempre é tão direto, existindo toda uma área, chamada predição de genes comparativa, dedicada à construção de algoritmos para integrar informações sobre homologia, e sinais estatísticos que indicam a presença de éxons, íntrons e outras estruturas gênicas (Axelson-Fisk , 2010).

Neste trabalho apresentaremos uma abordagem para predição de genes comparativa que pode ser utilizada quando não temos informação sobre a presença de genes em nenhuma das sequências genômicas homólogas, e nem sobre o alinhamento entre elas. Nesta abordagem utilizaremos um modelo para calcular as probabilidades de alinhamento entre cada par de posições do conjunto de sequências, assim como modelos de predição de genes *ab initio* para calcular a probabilidade de cada posição dessas sequências pertencer a cada uma das classes de estruturas gênicas. Essas informações são então combinadas utilizando uma nova versão

da técnica de transformação de consistência.

Para fazer uso dos indícios da presença de elementos de um gene em uma sequência genômica é necessário utilizar algum tipo de modelagem probabilística. Uma das abordagens mais comuns para isso é a utilização dos modelos ocultos de Markov (HMMs). Nesse modelo cada estado representa uma região do genoma. Essa opção de modelagem, apesar de conseguir resultados bastante satisfatórios, apresenta um problema quanto à predição do tamanho de éxons, já que a duração de cada estado nesse modelo é inerentemente descrita por uma distribuição geométrica, o que não representa a realidade nesse caso. Uma generalização do modelo foi então utilizada de forma que fosse possível modelar explicitamente a duração de cada estado. Esse tipo de modelo de Markov modificado é chamado de Modelo de Markov Oculto Generalizado (GHMM) (Rabiner , 1989).

Para que seja possível utilizar um GHMM em nossa abordagem combinada de alinhamento e predição de genes, é necessário saber a probabilidade *a posteriori* de cada posição das sequências em questão ter sido emitida por cada um dos estados do GHMM (equivalente à probabilidade de serem éxons, íntrons...). Este tipo de cálculo, comum para modelos ocultos de Markov tradicionais, se torna complicado para os modelos generalizados, devido ao tamanho variável das emissões de cada estado. Neste trabalho apresentamos um algoritmo que permite fazer esse cálculo. Mostramos que além do possível uso em nossa abordagem de predição de genes comparativa, essas probabilidades podem ser usadas como uma medida de confiabilidade de cada posição de uma predição de genes feita por aquele modelo.

Além disso, apresentamos um algoritmo que utiliza essas probabilidades *a posteriori* de predição para gerar uma predição de genes consistente, que maximiza o número esperado de posições corretamente preditas. Apresentaremos resultados da comparação deste com o algoritmo de Viterbi, tradicionalmente utilizado para fazer predições de gene utilizando um modelo oculto de Markov. Observamos que os resultados deste novo método são muito próximos àqueles obtidos pelo algoritmo de Viterbi.

Outro aspecto deste projeto que deve ser destacado é a maneira como implementamos e comparamos os métodos propostos neste trabalho e aqueles previamente conhecidos. A implementação de todos os modelos e algoritmos necessários para utilizá-los na análise de sequências foram implementados como extensões de um mesmo arcabouço, chamado ToPS (Kashiwabara e Durham , 2011). Neste arcabouço, inicialmente voltado para predição de genes *ab initio*, já estavam implementados diversos modelos, inclusive GHMM. A extensão feita neste trabalho foi a implementação dos novos algoritmos relacionados à predição de genes, além de toda a parte relacionada à alinhamento de sequências.

Como resultado deste esforço de extensão deste arcabouço foram implementados tanto os algoritmos originais, quanto os novos algoritmos propostos neste trabalho. Isso permitiu que fosse possível comparar as diversas técnicas mencionadas neste trabalho utilizando a mesma ferramenta. A vantagem disto é que não existem diferenças de performance advindas de diferenças na maneira como as metodologias foram implementadas. Assim, conseguimos uma comparação justa entre as técnicas em si, e não entre duas implementações destas

técnicas.

Outra vantagem da maneira como implementamos estes modelos é que é possível utilizar diferentes configurações de estados e parâmetros de qualquer um dos modelos implementados sem que seja necessária qualquer alteração no código do arcabouço. Estes parâmetros são definidos em arquivos de configuração. Além disso é possível fazer o treinamento desses modelos utilizando os diferentes métodos implementados. Devido a essa flexibilidade, o arcabouço ToPS se tornou uma excelente plataforma para desenvolvimento e para testes de metodologias tanto para alinhamento, predições de genes, entre outras análises de sequências.

Capítulo 2

Revisão bibliográfica

2.1 Predição de genes

Um gene eucariótico codificador de proteína é constituído por segmentos chamados éxons, partes que farão parte do RNA mensageiro maduro, e segmentos entre os éxons chamados íntrons, que serão removidos durante o processo de splicing. Além disso, existem nestas sequências alguns sinais (sequências conservadas) que estão envolvidos nos processos de transcrição, splicing e tradução. Éxons podem não ser completamente traduzidos. A porção traduzida da sequência de DNA (região codificadora), compreendida entre o códon inicial e o final, é a parte que os chamados preditores de genes identificam.

Um preditor de genes modela esses sinais conservados presentes na sequência, além de certas características evolutivas de éxons e íntrons para estabelecer um valor de probabilidade de uma dada sequência ter determinada estrutura gênica. Isto é feito utilizando diferentes modelos probabilísticos para reconhecer cada tipo de estrutura presente em um gene e um outro modelo para integrar todos eles. É possível com isso encontrar a estrutura gênica de maior probabilidade para uma sequência dados os parâmetros dos modelos probabilísticos.

Os componentes internos de um gene são em geral modelados por variações de modelos de Markov. Cadeias não-homogêneas de Markov de tamanho constante podem ser utilizadas para modelar sinais biológicos com sequências conservadas. O modelo para região codificadora pode ser também uma cadeia de Markov não-homogênea, entretanto, neste caso, o modelo pode representar sequências de qualquer tamanho. Regiões não codificadoras de um gene são em geral modeladas por cadeias homogêneas de Markov.

Modelos ocultos de Markov, por sua vez, permitem a integração das modelagens de diferentes características biológicas em um único arcabouço probabilístico, que é ao mesmo tempo flexível e matematicamente rigoroso. A idéia dos HMMs aplicados ao problema de predição de genes é que existe uma cadeia de Markov que gera uma sequência de estados. Estes estados tipicamente representam éxons, íntros e outras características biológicas. O termo oculto vem do fato de essa sequência de estados não ser aquela observada. A sequência observada é composta na realidade pelas emissões desses estados, que em predição de genes

seriam as bases do DNA (A, T, C e G). A distribuição de probabilidade de emissão dessas bases em cada um dos estados pode ser modelada de diferentes maneiras, um exemplo seria a utilização das cadeias de Markov mencionadas anteriormente.

Um modelo oculto generalizado de Markov (GHMM) generaliza HMMs no sentido que cada estado pode emitir mais do que uma única base. Em HMMs a cada transição de estado há uma única emissão, assim para um estado emitir uma sequência de N bases é necessário que aconteçam N transições daquele estado para ele mesmo. Como essas transições ocorrem com uma probabilidade fixa, o tamanho das sequências emitidas por um estado em um HMM segue intrinsecamente uma distribuição geométrica.

Em um GHMM existe uma distribuição de duração associada a cada estado. Um estado faz emissões primeiro escolhendo o número de bases que serão emitidas segundo a distribuição de duração, e posteriormente gerando aquele número de bases de acordo com uma distribuição de probabilidade de emissão. Isso permite que o tamanho das sequências emitidas em um estado siga uma distribuição diferente da geométrica, o que pode levar a melhoras significativas quando a característica biológica modelada nesse estado tem tamanhos que seguem uma distribuição muito diferente da geométrica.

Desta forma, podemos definir GHMMs como uma septupla $(S, O, p_{init}, p_{term}, d, e, t)$ em que:

- S é o conjunto de estados do GHMM.
- O é o conjunto de símbolos observáveis (A, T, C, e G em predição de genes).
- p_{init} é a função de probabilidade inicial, $\sum_{i \in S} p_{init}(i) = 1$.
- $p_{term} : S \to [0, 1]$ é a função de probabilidade final, $0 \leq p_{term}(i) \leq 1$.
- $t : S \times S \to [0, 1]$ é a função de probabilidade de transição entre os estados, $p_{term}(i) + \sum_{j \in S} t(i, j) = 1$ para todo $i \in S$.
- d é a função de probabilidade de duração, $\sum_{\ell \in N^*} d(\ell, j) = 1$ para todo $j \in S$ e N^* sendo o conjunto dos números naturais não nulos.
- $e : O^* \times S \to [0, 1]$ é a função de probabilidade de emissão, $\sum_{o \in O^\ell} e(o, j) = 1$ para todo $j \in S$, $\ell \in N^*$ e O^* é o conjunto de todas as palavras sobre o alfabeto finito O.

Tendo definidos todos os elementos desta septupla é possível encontrar para qualquer sequência de observações (sequência de DNA), a sequência de estados (sequência de elementos de um gene) com maior probabilidade de tê-la emitido. Isso é feito utilizado um algoritmo de programação dinâmica chamado algoritmo de Viterbi (Kashiwabara e Durham , 2011). A sequência de estados mais provável dada a sequência observada e o modelo é, em geral, o resultado gerado por ferramentas de predição de genes.

2.2 Alinhamento de sequências

Nesta seção será feita uma revisão de alguns métodos para alinhamento de sequências biológicas, tanto alinhamentos entre pares de sequências, quanto alinhamentos entre mais de duas sequências (alinhamentos múltiplos).

2.2.1 Alinhamento de pares de sequências

Existem diversas técnicas para solucionar o problema de alinhar um par de sequências. O algoritmo clássico (Needleman e Wunsch , 1970) utiliza uma pontuação para cada par de nucleotídeos ou aminoácidos alinhados, de forma que sequências mais similares e alinhamentos melhores obtêm uma pontuação maior. Em nosso trabalho, entretanto, assim como em diversos outros é utilizada uma abordagem probabilística para alinhamentos de pares de sequências Bradley *et al.* (2009); Do *et al.* (2005); Paten *et al.* (2009); Sahraeian e Yoon (2010). Para isso são utilizados, neste caso, modelos conhecidos como pairHMM, que são variações de modelos ocultos de Markov (HMM).

Esses modelos são especialmente úteis para encontrar alinhamentos entre pares de sequências e para avaliar a relevância dos alinhamentos encontrados. Diferente dos HMMs tradicionais, que geram apenas uma única sequência, os pairHMMs geram um par de sequências alinhadas. Esta abordagem apresenta bons resultados no alinhamento entre pares de sequências, além de ser bastante flexível e permitir o calculo de probabilidades a posteriori de alinhamento entre cada par de posições das sequências sendo alinhadas (Durbin *et al.* , 1998).

Consideremos por exemplo o pair-HMM representado na Figura 2.1. Nele, são geradas simultaneamente duas sequências alinhadas $\mathbf{x} = x_1x_2...x_{L_1}$ e $\mathbf{y} = y_1y_2...y_{L_2}$. O estado I emite um único símbolo não alinhado x_i na sequência $\mathbf{x}$. Da mesma maneira, o estado D também emite um símbolo y_j não alinhado, mas dessa vez na sequência $\mathbf{y}$. Por fim o estado M emite um par de símbolos x_i e y_j alinhados, tais que x_i é emitido na sequência $\mathbf{x}$ e y_j é emitido na sequência $\mathbf{y}$. A emissão em cada estado de um símbolo na sequência $\mathbf{x}$, um símbolo na sequência $\mathbf{y}$ ou um par de símbolos alinhados está associada a um valor de probabilidade. Além disso, a transição entre os estados tem também um valor de probabilidade, que irão determinar a probabilidade de abertura ($t(M, I)$ e $t(M, D)$) e extensão ($t(D, D)$ e $t(I, I)$) de gaps. Caminhos pelos estados desse modelo serão, portanto, alinhamentos entre pares de sequências, e terão valores de probabilidade associados a eles.

Um modelo pairHMM não tem necessariamente apenas esses três estados. Existem arquiteturas mais complexas, que permitem modelar, por exemplo, gaps grandes e gaps pequenos de maneiras diferentes, ou ainda representar regiões mais conservadas e menos conservadas, entre outras possibilidades. Essas modelagens mais detalhadas envolvem um aumento no número de estados do modelo.

Vamos agora definir pairHMMs mais formalmente. Sem perda de generalidade, vamos

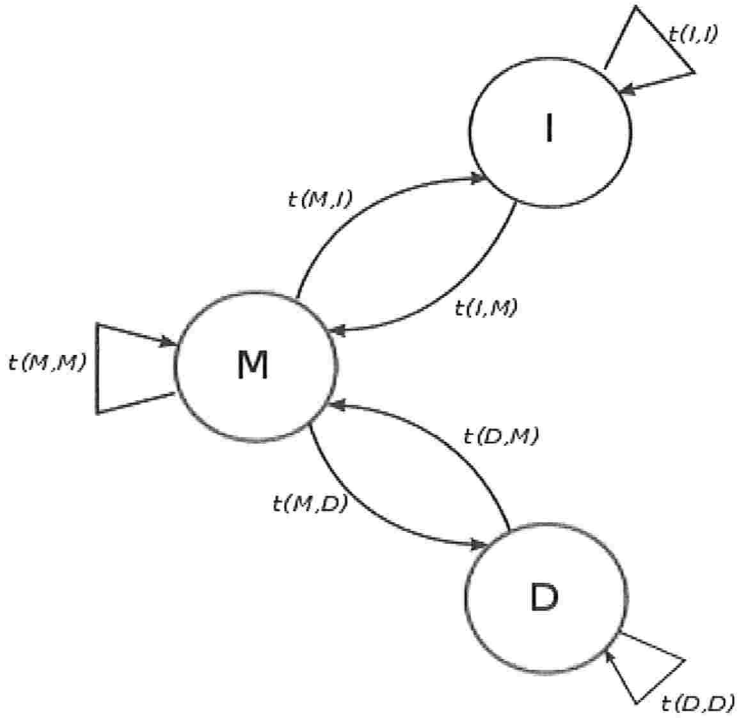

Figura 2.1: *Exemplo de modelo pairHMM*

chamar as sequências que estão sendo alinhadas de $\mathbf{x}$ e $\mathbf{y}$. PairHMM pode ser definido como uma sêxtupla $(S, O, p_{inic}, p_{term}, t, e)$ em que:

- S é o conjunto de estados do pairHMM. Existem três tipos de estados neste conjunto: os que emitem apenas na sequência $\mathbf{x}$, os que emitem apenas na sequência $\mathbf{y}$, e os que emitem um par de símbolos alinhados, sendo um na sequência $\mathbf{x}$ e outro na sequência $\mathbf{y}$.
- O é o conjunto de símbolos observáveis. Contém todos os símbolos presentes nas sequências $\mathbf{x}$ e $\mathbf{y}$, e mais um símbolo que representa gaps (não emissão).
- $p_{inic} : Q \to [0, 1]$ é a função de probabilidade inicial, $\sum_{i \in S} p_{inic}(i) = 1$
- $p_{term} : Q \to [0, 1]$ é a função de probabilidade final.
- $t : Q \times Q \to [0, 1]$ é a função de probabilidade de transição entre os estados do modelo, $p_{term}(i) + \sum_{j \in S} t(i, j) = 1$ para todo $i \in Q$.
- $e : \{O \times O\} \times Q \to [0, 1]$ é a função de probabilidade de emissão, $\sum_{o \in \{O \times O\}} e(o, i) = 1$ para todo $i \in S$. Estados que emitem em apenas uma sequência têm a probabilidade de emissão de qualquer $o \in \{O \times O\}$ não corresponda à emissão apenas nesta sequência igual a zero. Da mesma forma os estados que emitem em duas sequências, têm probabilidades de emissão de símbolos em apenas uma sequência iguais a zero.

Fazer um alinhamento entre duas sequências específicas utilizando este tipo de modelo, é equivalente a encontrar um caminho nos estados do modelo com a restrição de que as emissões que são feitas acabem gerando essas duas sequências inteiras. Cada caminho no modelo que emite essas duas sequências terá uma probabilidade associada a ele. Para um caminho $C = s_1 s_2 ... s_N$, com $s_i \in S$ gerando uma sequência de observações $A = o_1 o_2 ... o_N$, com $o_i \in \{O \times O\}$ esta probabilidade é dada por:

$$P(C, A) = p_{inic}(s_1) p_{term}(s_N) e(o_1, s_1) \prod_{i=1}^{N-1} t(s_i, s_{i+1}) e(o_{i+1}, s_{i+1}) \tag{2.1}$$

Entretanto, existem diversos caminhos em um modelo como este que podem emitir as sequências que se deseja alinhar. Assim, temos que ter algum critério que especifique qual destes caminhos é o correto. O número de alinhamentos possíveis entre duas sequências neste modelo (o número de caminhos possíveis) aumenta exponencialmente conforme aumentamos o tamanho destas sequências. Assim, é necessária a utilização de um algoritmo que consiga encontrar o alinhamento ótimo sem ter que calcular a probabilidade de cada alinhamento possível.

Um critério muito comumente utilizado ao fazer alinhamentos utilizando um pairHMM é selecionar o alinhamento mais provável. Este alinhamento pode ser encontrado de maneira eficiente utilizando um método de programação dinâmica chamado algoritmo de Viterbi (Durbin *et al.* , 1998). Utilizando este critério, o alinhamento obtido seria aquele que tem a maior probabilidade de ser idêntico ao alinhamento verdadeiro (desconhecido). Na prática, no entanto, não é possível definir com certeza qual é o alinhamento correto, assim, seria mais útil ter como objetivo encontrar um alinhamento que tivesse o maior número possível de similaridades com o alinhamento real (Sahraeian e Yoon , 2010).

Desta forma, uma outra definição possível do melhor alinhamento entre duas sequências segundo um pairHMM é o alinhamento que tenha o maior número esperado de bases alinhadas corretamente. Este critério, definido em Durbin et al. (Durbin *et al.* , 1998), foi utilizado pela primeira vez em uma ferramenta para alinhamentos múltiplos no trabalho de Do et al. (Do *et al.* , 2005) e é chamado de alinhamento de máxima precisão esperada (MEA, Maximum Expected Accuracy). Posteriormente o mesmo critério foi utilizado em diversas outras ferramentas de alinhamentos múltiplos (Sahraeian e Yoon , 2010), e mostrou-se que esta abordagem pode de fato aumentar a precisão média dos alinhamentos gerados. A precisão esperada de um alinhamento A pode assim ser definida como:

$$E[prec(A)] = \sum_{\forall x_i \sim y_j \in A} P(x_i \sim y_j | \mathbf{x}, \mathbf{y}) \tag{2.2}$$

Esta definição serve tanto para alinhamentos de pares de sequências quanto para alinhamentos múltiplos, já que se A é um alinhamento múltiplo basta considerar todos os pares de posições alinhadas presentes nele.

Assim, para encontrar o alinhamento de máxima precisão esperada entre duas sequências

$\mathbf{x}$ e $\mathbf{y}$, é necessário saber a probabilidade $P(x_i \sim y_j|\mathbf{x},\mathbf{y})$ de cada par de bases x_i e y_j estar corretamente alinhado. Esta probabilidade pode ser encontrada de maneira eficiente em um pairHMM através de um algoritmo similar ao algoritmo de Viterbi, chamado forward-backward (Durbin *et al.* , 1998). A idéia deste algoritmo é calcular a probabilidade conjunta de todos os alinhamentos entre aquelas sequências que tenham as posições x_i e y_j alinhadas e depois dividir este valor pela probabilidade conjunta de todos os alinhamentos possíveis entre estas mesmas sequências. Formalmente temos:

$$\begin{aligned} P(x_i \sim y_j|\mathbf{x},\mathbf{y}) =& \frac{P(\mathbf{x},\mathbf{y},x_i \sim y_j)}{P(\mathbf{x},\mathbf{y})} = \\ =& \frac{1}{P(\mathbf{x},\mathbf{y})} P(x_{1\ldots i-1}, y_{1\ldots j-i}, x_i \sim y_j) P(x_{i+1\ldots L_x}, y_{j+1\ldots L_y}|x_{1\ldots i-1}, y_{1\ldots j-i}, x_i \sim y_j) = \\ =& \frac{1}{P(\mathbf{x},\mathbf{y})} P(x_{1\ldots i-1}, y_{1\ldots j-i}, x_i \sim y_j) P(x_{i+1\ldots L_x}, y_{j+1\ldots L_y}|x_i \sim y_j) \end{aligned} \tag{2.3}$$

estamos usando no primeiro passo a definição de probabilidade condicional. No segundo passo dizemos que a probabilidade de o modelo emitir a sequência $\mathbf{x}$ e $\mathbf{y}$ com as posições x_i e y_j alinhadas é igual a probabilidade do mesmo emitir as sub-sequências $x_{1\ldots i-1}$, $y_{1\ldots j-1}$ e as bases x_i e y_j alinhadas, vezes a probabilidade de emitir o resto das sequências $\mathbf{x}$ e $\mathbf{y}$ dado que essa primeira parte foi emitida. O úlimo passo usa a definição de cadeia de Markov de ordem 1.

O algoritmo é dividido em duas etapas. Na primeira delas, chamada forward, calculamos $P(x_{1\ldots i-1}, y_{1\ldots j-i}, x_i \sim y_j)$ para todo $1 \leq i \leq L_x$ e $1 \leq j \leq L_y$, assim como $P(\mathbf{x},\mathbf{y})$. Para isso, definimos uma função $f^k(i,j)$ que representa a probabilidade combinada de todos os alinhamentos entre as sub-sequências $x_{1\ldots i}$ e $y_{1\ldots j}$ acabando no estado k do pairHMM. Quando estamos utilizando um pairHMM de três estados como aquele apresentado na Figura 2.1, $f^M(i,j)$ é equivalente a $P(x_{1\ldots i-1}, y_{1\ldots j-i}, x_i \sim y_j)$. Assim, o algoritmo forward consiste em calcular o valor dessa função para todo para todo $1 \leq i \leq L_x$, $1 \leq j \leq L_y$ e todo $k \in S$.

Abaixo apresentaremos o algoritmo forward para um pairHMM. Este tipo de modelo pode ter mais de um estado de match, inserção na primeira sequência e inserção na segunda sequência. Assim, consideraremos aqui que M representa conjunto de todos os estados que emitem símbolos alinhados, X representa o conjunto dos estados que fazem uma inserção na sequência $\mathbf{x}$ e Y o conjunto dos estados que fazem uma inserção na sequência $\mathbf{y}$. Além disso, a emissão de um gap será representada por um hífen.

Algoritmo forward para pairHMM

Inicialização:

$f^k(0,0) = p_{inic}(k) \quad \forall k \in S$

$f^k(i,-1) = 0 \quad \forall k \in S \text{ e } 0 \leq i \leq L_x$

$f^k(-1,j) = 0 \quad \forall k \in S \text{ e } 0 \leq j \leq L_y$

Recursão: $i = 0...L_x, j = 0...L_y$, exceto $(0,0)$

$$
\begin{aligned}
f^{k\in M}(i,j) &= e(x_i \sim y_j, k) \sum_{\ell \in S} t(\ell,k) f^{\ell}(i-1, j-1) \\
f^{k\in X}(i,j) &= e(x_i \sim -, k) \sum_{\ell \in S} t(\ell,k) f^{\ell}(i-1, j) \\
f^{k\in Y}(i,j) &= e(- \sim y_j, k) \sum_{\ell \in S} t(\ell,k) f^{\ell}(i, j-1)
\end{aligned}
\tag{2.4}
$$

Terminação:

$$P(\mathbf{x}, \mathbf{y}) = \sum_{k \in S} p_{term}(k) f^k(L_x, L_y)$$

Podemos ver que, pela definição da função forward, $P(x_{1...i-1}, y_{1...j-i}, x_i \sim y_j)$ é equivalente a $\sum_{k\in M} f^k(i,j)$. Além disso, é fácil ver que $\sum_{k\in S} p_{term}(k) f^k(L_x, L_y) = P(\mathbf{x}, \mathbf{y})$, já que essa soma representa a probabilidade de emitirmos a sequência toda acabando em um determinado estado, somado para todos os estados, o que nos dá a probabilidade de emitirmos essas duas sequências nesse modelo. A segunda parte do algoritmo forward-backward é chamada backward, e tem como objetivo calcular para todo $1 \leq i \leq L_x$, $1 \leq j \leq L_y$ e todo $k \in S$, o valor de $P(x_{i+1...L_x}, y_{j+1...L_y} | x_i \sim y_j)$. Para isso, de maneira similar ao algoritmo forward, definimos uma função $b^k(i,j)$ que representa a probabilidade que desejamos calcular. Abaixo apresentaremos o algoritmo backward.

Algoritmo backward para pairHMM

$$
\begin{aligned}
&\text{Inicialização:}\\
&b^k(L_x, L_y) = p_{term}(k) \quad \forall k \in S\\
&b^k(i, L_y + 1) = 0 \quad \forall k \in S \text{ e } 0 \leq i \leq L_x\\
&b^k(L_x + 1, j) = 0 \quad \forall k \in S \text{ e } 0 \leq j \leq L_y
\end{aligned}
\tag{2.5}
$$

$$
\begin{aligned}
&\text{Recursão:} i = L_x...1,\ j = L_y...1, \text{ exceto } (L_x, L_y)\\
&b^k(i,j) = \sum_{\ell \in M} t(k,\ell) e(x_{i+1} \sim y_{j+1}, \ell) b^\ell(i+1, j+1) +\\
&\qquad + \sum_{\ell \in X} t(k,\ell) e(x_{i+1} \sim -, \ell) b^\ell(i+1, j) +\\
&\qquad + \sum_{\ell \in Y} t(k,\ell) e(- \sim y_{j+1}, \ell) b^\ell(i, j+1)
\end{aligned}
$$

Feitos os cálculos das funções forward e backward para todas as posições das sequências e todos os estados do modelo, podemos então calcular o valor das probabilidades a posteriori de alinhamento entre cada par de bases possível para essas duas sequências, finalizando o algoritmo forward-backward. Para isso, utilizamos a seguinte fórmula derivada da equação 2.3:

$$
P(x_i \sim y_j | \mathbf{x}, \mathbf{y}) = \frac{\sum_{k \in M} f^k(i,j) b^k(i,j)}{P(\mathbf{x}, \mathbf{y})} \tag{2.6}
$$

Utilizando essas probabilidades de alinhamento, podemos agora encontrar o alinhamento de máxima precisão esperada entre esse par de sequências através de um algoritmo tradicional de programação dinâmica, mas que usa essas probabilidades como o seu sistema de pontuação. As equações de recursão são:

$$
A(i,j) = max \begin{cases} A(i-1, j-1) + P(x_i \sim y_j | \mathbf{x}, \mathbf{y}) \\ A(i-1, j) \\ A(j, i-1) \end{cases} \tag{2.7}
$$

Um procedimento padrão de traceback sobre essa matriz fornecerá o alinhamento que maximiza a função de precisão esperada definida na equação 2.2.

2.2.2 Construção de alinhamentos múltiplos

Denominamos alinhamento múltiplo, o problema de alinhar mais de duas sequências de caracteres. Assim como em alinhamentos de pares de sequências, existe algum critério de

pontuação para alinhamentos múltiplos que deve ser maximizado. A construção de um alinhamento múltiplo que seja ótimo segundo um critério biológico de similaridade qualquer é, entretanto, um problema NP-completo (Kemena e Notredame , 2009). Este fato implica que a possibilidade de calcular esse alinhamento ótimo de maneira exata só existe para um conjunto de sequências tão pequeno que não seria útil para a maioria de suas aplicações. Assim, alinhamentos de múltiplas sequências são em geral calculados utilizando algum tipo de heurística (Kemena e Notredame , 2009). Diversos tipos de aproximações já foram utilizados na tentativa de solucionar este problema de maneira adequada. Nesta sessão descreveremos algumas delas.

Alinhamento progressivo

O método de construção de alinhamentos múltiplos mais utilizado atualmente é o chamado "alinhamento progressivo"(Hogeweg and Hesper, 1984). Este é o método utilizado para por ferramentas de alinhamento múltiplo amplamente utilizadas como: CLUSTALW (Thompson *et al.* , 1994), T-Coffee (Notredame e et al. , 2000), ProbCons (Do *et al.* , 2005), MUSCLE (Edgar , 2004), entre outras.

Este método consiste em um processo aglomerativo guloso. Ele se baseia no fato de a chance de um alinhamento entre duas sequências estar correto aumentar conforme aumenta a similaridade entre as sequências. Nesta técnica, as sequências são primeiramente comparadas duas a duas para preencher uma matriz de distâncias, baseada na similaridade entre elas. Algum algoritmo de clusterização, que pode ser diferente para cada ferramenta, é aplicado nessa matriz para gerar uma árvore binária enraizada (árvore guia). O algoritmo segue então esta árvore das folhas até a raiz (sequências mais similares até as menos similares) alinhando duas a duas as sequências (ou alinhamentos) associados aos nós da árvore. Esse algoritmo funciona com qualquer método capaz de alinhar pares de sequências e de alinhamentos.

A maneira como alinhamentos entre alinhamentos são feitos, varia de ferramenta para ferramenta. Para exemplificar uma possível maneira de solucionar este problema, descreveremos aqui o método utilizado na ferramenta ProbCons (Do *et al.* , 2005). Nesta ferramenta utiliza-se o critério de máxima precisão esperada para alinhar um par de sequências. Isto quer dizer que a pontuação para alinhar um par de caracteres de duas sequências é igual a probabilidade *a posteriori* de alinhar aqueles dois caracteres. Assim, quando esta ferramenta alinha duas colunas de dois alinhamentos múltiplos, a pontuação associada a este alinhamento é dada por uma soma dos valores de probabilidade *a posteriori* de todos os pares de caracteres envolvidos no alinhamento dessas duas colunas dos alinhamentos. Com esse esquema de pontuação, é possível alinhar dois alinhamentos a mesma maneira que alinha-se um par de sequências. As equações de recursão deste algoritmo estão apresentadas abaixo.

$$A(col_i, col_j) = max \begin{cases} A(col_{i-1}, col_{j-1}) + \sum_{m \in col_i} \sum_{n \in col_j} P(m \sim n) \\ A(col_{i-1}, col_j) \\ A(col_i, col_{j-1}) \end{cases} \quad (2.8)$$

Uma grande desvantagem deste método é o fato de ele ser guloso e portanto se comprometer com os alinhamentos entre pares de sequências realizados nas etapas iniciais do algoritmo, que ignoram a maior parte da informação contida no conjunto de dados. Erros nesta etapa do algoritmo serão propagados até o fim, podendo gerar erros ainda maiores. Existem algumas técnicas para tentar contornar este problema (Kemena e Notredame , 2009), que serão apresentadas mais à frente, já que são também utilizadas para outras abordagens de alinhamentos múltiplos.

Alinhamento múltiplo por *sequence annealing*

Outra técnica que tem apresentado bons resultados para realizar a tarefa de alinhar múltiplas sequências é chamada de *sequence annealing*. Ela foi implementada em três ferramentas diferentes: PicXAA (Sahraeian e Yoon , 2010), FSA (Bradley *et al.* , 2009) e AMAP (Bradley *et al.* , 2009).

Este método envolve também um algoritmo guloso, mas que em cada passo do algoritmo é alinhado apenas um par de posições, ao invés de uma sequência inteira. O próximo par de posições a ser alinhado é sempre aquele com o maior peso, e que seja consistente com as outras posições previamente alinhadas. A função de peso é diferente em diferentes ferramentas que utilizam esta abordagem Bradley *et al.* (2009); Sahraeian e Yoon (2010), mas está sempre relacionada com as probabilidades *a posteriori* de alinhamento entre pares de posições, calculadas a partir de um modelo de alinhamento de pares de sequências (pairHMM por exemplo).

Em algumas implementações o peso de um determinado par de bases alinhadas depende das bases previamente inseridas no alinhamento e deve ser recalculado antes que o próximo par de bases seja inserido (Bradley *et al.* , 2009). Em outras implementações (PicXAA), entretanto, a utilização de transformações de consistência, que serão definidas mais à frente neste documento, antes da etapa de construção do alinhamento permite que não haja uma diminuição na qualidade do alinhamento mesmo sem que o peso seja recalculado em cada passo (Sahraeian e Yoon , 2010).

Para que seja possível verificar a consistência entre os novos pares de posições alinhados com aqueles que já estavam presentes no alinhamento, representa-se, nesta técnica, o alinhamento como um grafo acíclico direcionado (DAG) (Lee *et al.* , 2002). Cada nó deste grafo representa uma coluna do alinhamento, e as arestas direcionadas representam a ordem das colunas neste alinhamento, que está diretamente relacionada com a ordem dos caracteres das sequências sendo alinhadas. Um exemplo de um grafo representado alinhamentos da maneira descrita neste parágrafo pode ser observado na Figura 2.2.

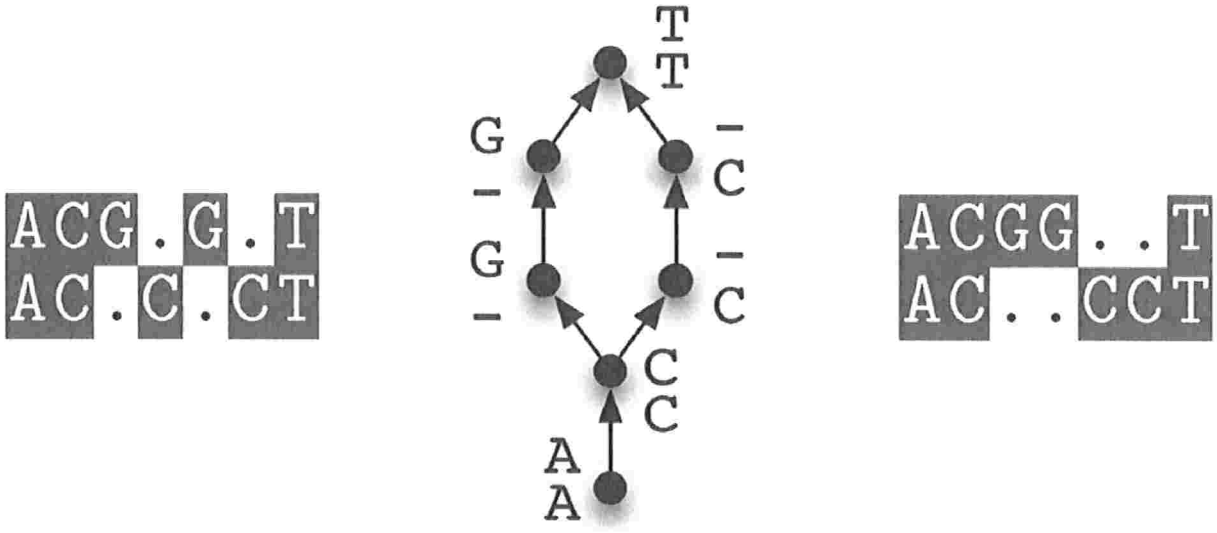

Figura 2.2: *Representação de alinhamentos como um grafo direcionado acíclico. O mesmo grafo (centro) representa os dois alinhamentos apresentados na figura. Figura extraída de (Bradley et al. , 2009).*

Nesta representação, a inserção de um par de posições inconsistente com as posições previamente alinhadas leva ao aparecimento de um ciclo no grafo. Existem algoritmos bastante eficientes para verificar o aparecimento de ciclos neste tipo de grafo quando são inseridas novas arestas (Pearce e Kelly , 2006), assim, é possível verificar essa consistência também de maneira eficiente.

A representação do alinhamento como um DAG não necessariamente especifica uma ordenação única entre as colunas do alinhamento, como evidenciado na Figura 2.2. Assim, é preciso selecionar uma ordenação topológica dos nós do grafo como a correta. Nas implementações desta técnica feitas até o momento a ordem escolhida é aquela que minimiza o número de aberturas de gaps. Encontrar essa ordem pode ser feito de maneira linear no número de nós e arestas do grafo (Bradley *et al.* , 2009).

A vantagem desta técnica quando comparada com "alinhamentos progressivos"é que ao invés de nos comprometermos com o alinhamento completo entre duas sequências em cada etapa, nos comprometemos apenas com o alinhamento entre um par de posições. Isso permite que sejam construídas primeiramente as regiões do alinhamento múltiplo que temos mais confiança de estarem corretas, para depois inserirmos as posições que estamos menos certos de estarem corretas, de forma que elas sejam consistentes com o alinhamento das regiões com alta similaridade local entre as sequências.

Esta técnica apresenta o problema de não utilizar toda a informação do conjunto de dados em cada etapa, já que o peso é calculado baseado apenas na probabilidade de alinhamento entre pares de posições dado um par de sequências.

Devido às vantagens mencionadas acima, e aos bons resultados apresentados pelas ferramentas que utilizam este método em diversos *benchmarks* Bradley *et al.* (2009); Sahraeian e Yoon (2010), esta foi a técnica de construção de alinhamentos múltiplos implementada em nossa

ferramenta.

2.2.3 Técnicas para melhorar a qualidade de AMs

Como mencionado na sessão anterior, existem basicamente dois problemas principais que são comuns às técnicas de construção de alinhamentos múltiplos.

O primeiro deles é o fato de esses algoritmos serem gulosos, ou seja, a partir do momento em que o algoritmo se compromete com o alinhamento, de um par de sequências ou posições, não é possível voltar atrás. Assim, se em uma etapa inicial dos algoritmos de construção for feito um alinhamento incorreto, este erro será propagado para o alinhamento final, e pode acarretar no alinhamento incorreto de outras posições.

O segundo é o fato de esses algoritmos serem baseados no alinhamento entre pares de sequências, o que implica na não utilização de toda a informação contida no conjunto. Para ilustrar este segundo problema, podemos pensar que a probabilidade de duas posições estarem alinhadas pode não ser a mesma quando consideramos o alinhamento de duas sequências, ou quando consideramos que estamos alinhado três ou mais sequências. Este fato pode levar ao aparecimento de erros no alinhamento múltiplo, já que as técnicas mencionadas acima sempre calculam essas probabilidades através da análise de apenas duas sequências de cada vez.

Existem algumas técnicas que foram criadas para tentar contornar estes problemas. Aqui apresentaremos duas delas. A primeira, que visa solucionar o primeiro problema é chamada de refinamento iterativo, e a segunda, que visa contornar o segundo problema é chamada transformação de consistência. Estas duas técnicas são bastante populares, tendo sido implementadas em diversas ferramentas de alinhamentos múltiplos (Kemena e Notredame , 2009).

Refinamento iterativo

Uma heurística comumente utilizada para contornar o aspecto guloso dos algoritmos tradicionais de alinhamentos múltiplos é chamada refinamento iterativo. Esta técnica é em geral aplicada após o alinhamento entre as sequências estar completo. Ela consiste em realinhar, ou ajustar partes do alinhamento um número determinado de vezes, ou até que a pontuação do alinhamento convirja.

O método mais comumente utilizado para este fim, consistem em dividir as sequências alinhadas em dois grupos, de forma que as sequências pertencentes a um determinado grupo estejam alinhadas da maneira especificada pelo alinhamento inicial. Em seguida, é feito um novo alinhamento entre os alinhamentos de cada um desses grupos, da mesma maneira apresentada para a técnica de "alinhamento progressivo". A maneira com que os grupos de sequências são divididos pode ser diferente para diferentes ferramentas de alinhamento múltiplo Do *et al.* (2005); Sahraeian e Yoon (2010).

Alguma variação deste método é utilizada em praticamente todas as ferramentas de alinhamento múltiplo atuais. Até o momento esta técnica ainda não foi implementada em nossa ferramenta, já que por enquanto estamos apenas tentando validar as modificações nas técnicas de alinhamento propostas neste trabalho. Entretanto, isto definitivamente terá que ser feito se quisermos que nossa ferramenta apresente resultados competitivos com as melhores ferramentas conhecidas.

Transformações de consistência

Como mencionado anteriormente, um problema comum para as técnica de alinhamento múltiplo é o fato de não ser possível utilizar toda a informação contida no conjunto de dados ao mesmo tempo de maneira eficiente, ou seja, as colunas do alinhamento múltiplo são construídas a partir das probabilidades de alinhamento entre pares de caracteres, ao invés de calculadas diretamente a partir de um modelo de alinhamento múltiplo.

Uma metodologia utilizada para tentar contornar este problema são as chamadas transformações de consistência. Esta técnica, aplicada antes de o alinhamento múltiplo ser construído, consiste em atualizar as pontuações de alinhamento entre pares de caracteres de duas sequências baseando-se na pontuação de alinhamento desses dois caracteres com os caracteres de uma terceira sequência. A ideia é que a aplicação desta técnica permita que sejam obtidos alinhamentos entre os pares de sequências que sejam consistentes uns com os outros.

No artigo de Do *et al.* (Do *et al.* , 2005), em que é apresentada a ferramenta ProbCons, é descrita uma abordagem probabilística para essa técnica, em que ao invés de serem atualizadas as pontuações de alinhamento, são atualizadas as probabilidades *a posteriori* de alinhamento entre as posições das sequências, calculadas através de um pairHMM. Nesta sessão apresentaremos a abordagem probabilística, já que é aquela utilizada neste trabalho e é a versão que dá origem a algumas das propostas originas deste projeto.

Dadas as probabilidades de alinhamento entre os pares de caracteres de duas sequências, calculadas a partir de um pairHMM, por exemplo, a transformação de consistência probabilística visa transformar estas probabilidades utilizando informações das outras sequências de forma que os alinhamentos entre os pares de sequências sejam consistentes uns com os outros. Os alinhamentos entre os pares de sequências serem consistentes significa, neste contexto, que se um resíduo x_i da sequência $\mathbf{x}$ está alinhado com um resíduo y_j da sequência $\mathbf{y}$, no alinhamento entre $\mathbf{x}$ e $\mathbf{y}$, e se y_j alinha com o resíduo z_k da sequência $\mathbf{z}$ no alinhamento entre $\mathbf{y}$ e $\mathbf{z}$, então x_i tem que também estar alinhado com z_k no alinhamento entre $\mathbf{x}$ e $\mathbf{z}$.

Com esse intuito, foram introduzidas as chamadas transformações de consistência probabilísticas, que atualizam a probabilidade de alinhamento entre x_i e y_j utilizando as probabilidades de alinhamento entre x_i e z_k e entre z_k e y_j, com z_k sendo qualquer posição de qualquer sequência pertencente ao conjunto S de sequências sendo alinhadas. Essa transformação é dada por:

$$P^*(x_i \sim y_j) = \frac{1}{|S|} \sum_{\mathbf{z} \in S} \sum_{k=1}^{|\mathbf{z}|} P(x_i \sim z_k) P(z_k \sim y_j) \quad \forall x_i \sim y_j \tag{2.9}$$

Nesta fórmula, consideramos que $P(x_i \sim x_j) = 1$ se $i = j$ e $P(x_i \sim x_j) = 0$ se $i \neq j$, além disso, $|S|$ e $|\mathbf{z}|$ significam respectivamente o tamanho do conjunto S (número de sequências sendo alinhadas) e o tamanho da sequência $\mathbf{z}$. Essa transformação pode ser aplicada de maneira iterativa, aplicando a transformação às probabilidades transformadas na última iteração.

Apresentaremos ainda uma outra versão desta técnica, utilizada na ferramenta PicXAA (Sahraeian e Yoon , 2010), que parece apresentar melhores resultados do que a formulação apresentada na equação 2.9. Nesta versão são utilizados pesos diferentes para cada um dos alinhamentos utilizados para transformar as probabilidades *a posteriori*.

Vamos definir A como o alinhamento de precisão máxima entre um par de sequências quaisquer, $\mathbf{x}$ e $\mathbf{y}$, e $|A|$ como o número de pares de caracteres alinhados em A. O peso que será utilizado na nova transformação de consistência é dado por:

$$P(\mathbf{x} <> \mathbf{y}) = \frac{1}{|A|} \sum_{x_i \sim y_j \in A} P(x_i \sim y_j) \tag{2.10}$$

Podemos ver que este peso é a probabilidade média dos pares alinhados no alinhamento ótimo entre duas sequências. Consideramos que $P(\mathbf{x} <> \mathbf{x}) = 1 \quad \forall \mathbf{x} \in S$. Assim, este peso pode ser interpretado como a confiança que temos no alinhamento entre estas duas sequências. Com isso, a nova transformação de consistência é dada por:

$$P^*(x_i \sim y_j) = \frac{\sum_{\mathbf{z} \in S} P(\mathbf{x} <> \mathbf{z}) P(\mathbf{y} <> \mathbf{z}) \sum_{k=1}^{|\mathbf{z}|} P(x_i \sim z_k) P(z_k \sim y_j)}{\sum_{\mathbf{z} \in S} P(\mathbf{x} <> \mathbf{z}) P(\mathbf{y} <> \mathbf{z})} \tag{2.11}$$

Podemos ver, portanto, que alinhamentos em que temos menos confiança terão um peso menor na transformação das outras probabilidades. Essa nova transformação é especialmente importante quando o conjunto S contém algumas sequências bastante similares e outras muito pouco similares. Isso pois aplicar a transformação dando o mesmo peso para as sequências bastante divergentes que para aquelas bastante próximas, pode levar a uma deterioração na qualidade do alinhamento.

2.2.4 Avaliação de alinhamentos múltiplos

Como não se dispõe de informações diretas sobre as sequências macromoleculares de organismos ancestrais, as verdadeiras posições de mutações, inserções e deleções que ocorreram durante a evolução são evidentemente desconhecidas. Tal informação seria necessária para que se pudesse avaliar com precisão absoluta o desempenho dos programas de alinhamento múltiplo. Entretanto, as informações disponíveis a respeito da teoria da evolução

molecular, bem como a partir de dados empíricos de comparação de sequências macromoleculares, podem prover abordagens alternativas. Uma delas é a construção de conjuntos de dados simulados através de parâmetros evolutivos estimados Blanchette *et al.* (2004); Varadarajan *et al.* (2008). Outra possibilidade é realizar medidas indiretas, como a periodicidade observada no alinhamento em regiões codificadoras por exemplo (Margulies e et al. , 2007), ou informações estruturais no caso de alinhamentos de proteínas (Thompson *et al.* , 2005).

Apesar de nenhuma dessas abordagens ser a ideal, todas as ferramentas construídas até o momento foram testadas utilizando este tipo de dado (Kemena e Notredame , 2009). Isso pode potencialmente levar a um erro coletivo, entretanto sugerir novas maneiras de testar ferramentas de alinhamento múltiplo não é tarefa trivial. Portanto, neste projeto pretendemos testar nossa ferramenta com os mesmos conjuntos de teste utilizados por outras ferramentas que têm o mesmo intuito.

Considerando que temos um alinhamento referência correto, podemos utilizar diferentes critérios de qualidade de um alinhamento baseando-nos na similaridade com o alinhamento referência. Dois critérios são os mais usualmente utilizados: porcentagem dos pares presentes no alinhamento referência que estão também presentes no alinhamento que desejamos avaliar (SP), e porcentagem de colunas inteiras que foram corretamente alinhadas (CS).

É interessante notar que nenhum desses dois critérios leva em conta o posicionamento correto de gaps no alinhamento. Em nossos testes apresentaremos duas medidas similares a estas, mas que levam em conta a posição dos gaps.

BaliBase

Alinhamentos de proteínas são mais facilmente avaliados do que alinhamentos entre sequências de DNA. Proteínas devem manter determinadas estruturas para que possam se manter funcionais ao longo da evolução, assim, a utilização de informações estruturais pode ajudar a determinar o alinhamento correto entre algumas proteínas. Estes alinhamentos curados são amplamente utilizados para avaliar ferramentas de alinhamento múltiplo desse tipo de sequência.

O *benchmark* BaliBase (Thompson *et al.* , 2005) é um exemplo de conjunto de teste para ferramentas de alinhamento múltiplo de proteínas. Este conjunto foi construído a partir de alinhamentos estruturais de proteínas homólogas. Neste tipo de alinhamento, são levadas em conta as estruturas tridimensionais das proteínas que estão sendo alinhadas, gerando um alinhamento inicial bastante preciso. Feito esse alinhamento inicial, é feita ainda uma edição manual cuidadosa, para garantir que os alinhamentos presentes neste conjunto, que serão usados como referência, estejam de acordo com o que se conhece a respeito das modificações que ocorrem ao longo da evolução de sequências protéicas.

Após este processo de construção dos alinhamentos, é utilizada ainda uma metodologia para identificar regiões dentro destes alinhamentos onde há uma grande certeza de os alinha-

mentos estarem corretos. Essa regiões, chamadas *core blocks*, são identificadas com base na presença de informações estruturais e conservação da sequência primária. Para este projeto, é importante notar que um dos critérios para a determinação dos *core blocks* é que não pode haver gaps nestas regiões.

Em muitos casos em que é utilizado este *benchmark*, o alinhamento é avaliado apenas nos *core blocks*, entretanto, o alinhamento nas regiões não classificadas como *core blocks* foi também cuidadosamente construído e manualmente editado. Assim, apesar de termos menos certeza da precisão dos alinhamentos fora dos *core blocks*, faz sentido considerar as regiões como referência.

A versão 3.0 deste *benchmark*, que utilizaremos neste trabalho, contém 218 alinhamentos e é dividida em seis categorias. Cada uma dessas categorias representa um problema diferente para a construção de alinhamentos múltiplos. Essa divisão permite que as diferentes ferramentas tenham sua performance avaliada em cada uma dessas situações. Abaixo estão descritos estes conjuntos de alinhamentos:

- **11:** Contém alinhamentos em que as sequências são equidistantes, e têm menos de 20% de identidade.
- **12:** Contém alinhamentos em que as sequências são equidistantes, e têm entre 20% e 40% de identidade.
- **20:** Contém alinhamentos em que a maior parte das sequências têm mais de 40% de identidade, mas com algumas sequências órfãs (com menos de 20% de identidade com todas as outras).
- **30:** Contém alinhamentos em que estão presentes sub-famílias de sequências. As sequências dentro de um sub-família têm mais de 40% de identidade, mas sequências de sub-famílias diferentes têm menos de 20% de identidade.
- **40:** As sequências presentes nestes alinhamentos têm mais de 20% de identidade com pelo menos uma outra sequência, mas há sequências com longas inserções nas extremidades N/C-terminal.
- **50:** As sequências presentes nestes alinhamentos têm mais de 20% de identidade com pelo menos uma outra sequência, mas há sequências com longas inserções internas.

IRMBASE

Outro conjunto de testes utilizado na avaliação de diversas ferramentas de alinhamento múltiplo é o IRMBASE (Subramanian *et al.* , 2008). Este é um *benchmark* é simulado, desenvolvido para avaliar a capacidade das diferentes ferramentas de alinhamento múltiplo em capturar similaridades locais entre as sequências.

A maioria das ferramentas populares de alinhamentos múltiplos apresenta uma performance ruim neste conjunto de teste. Em especial, aquelas que fazem uso da metodologia de alinhamento progressivo apresentam grande dificuldade em alinhar este conjunto corretamente.

Este *benchmark* foi construído através da inserção de segmentos altamente conservados em longas sequências aleatórias. Os segmentos conservados foram gerados através da ferramenta de simulação de evolução ROSE (Stoye *et al.* , 1998), e têm portanto um alinhamento conhecido entre eles. O IRMBASE 2.0 é dividido em quatro conjuntos de alinhamentos, em que varia-se a quantidade de sequência aleatória, assim como o comprimento dos segmentos conservados. No total contém 192 alinhamentos.

É importante notar que a avaliação da performance das ferramentas neste conjunto é feita apenas nas regiões que são de fato conservadas, já que não faz sentido alinhar sequências aleatórias.

Teste Wilcoxon signed-rank

Quando comparamos duas ferramentas de alinhamento múltiplo em um mesmo *benchmark*, fazemos vários alinhamentos. A performance das ferramentas em cada teste varia bastante, já que existem casos mais complicados, e outros mais simples. Assim, uma comparação apenas entre as médias dos resultados não traria muita informação, já que a variância é em geral muito grande. Por isso, utilizamos o teste de hipótese Wilcoxon signed-rank (Wilcoxon , 1945) que é aplicado em um conjunto de amostras pareadas (resultados de duas ferramentas em um mesmo teste), para verificar se a diferença entre a pontuação das ferramentas nos testes é significativamente diferente de zero ou não.

Seja N o número de testes em um *benchmark*. Para $i = 1...N$, sejam $x_{1,i}$ e $x_{2,i}$ as medidas de qualidade de duas ferramentas no teste i. Assim, temos as seguintes hipóteses:

$$H_0 : \text{ a diferença mediana entre os pares } x_{1,i} \text{ e } x_{2,i} \text{ é zero.} \qquad H_1 : \text{a diferença mediana não é zero} \tag{2.12}$$

O teste procede da seguinte maneira:

1. Para $i = 1, ..., N$, calcule $|x_{2,i} - x_{1,i}|$ e a função sinal (+1, 0, ou -1, para números positivos, zero, ou negativos respectivamente) desta diferença.
2. Exclua os pares em que a diferença é zero. Seja N_r a amostra reduzida.
3. Ordene os pares restantes do menor módulo da diferença para o maior módulo da diferença.
4. Ranqueie os pares começando com 1 para o par com menor módulo da diferença. Quando há empate, o ranqueamento é igual a média dos ranqueamentos envolvidos. Seja R_i o ranqueamento.

5. Calcule a estatística W do teste. $W = |\sum_1^{N_r}[sinal(x_{2,i} - x_{1,i})R_i]|$.

6. Conforme N_r aumenta, a distribuição de W converge para uma normal. Assim, para $N_r > 10$ podemos calcular um valor $z = \frac{W-0.5}{\sigma_w}$, $\sigma_w = \sqrt{\frac{N_r(N_r+1)(2N_r+1)}{6}}$. Se $z > z_{crtico}$, ou $z < -z_{crtico}$ rejeitamos H_0.

Utilizaremos este teste para verificar se uma técnica proposta aqui é significativamente melhor do que outra. Utilizaremos nível de significância de 1% para dizer que a diferença de performance entre uma técnica e outra é significativamente diferente de zero.

Capítulo 3

Modificações na construção de AMs

Neste capítulo descreveremos as propostas feitas neste projeto que estão relacionadas puramente com alinhamentos múltiplos. A primeira delas é uma nova formulação para transformações de consistência levando gaps em consideração. A segunda são modificações nas técnicas de construção do alinhamento múltiplo para maximizar a precisão esperada penalizando o alinhamento incorreto de posições que não têm homólogos. Aquelas propostas que têm relação com predição de genes e com a combinação de alinhamentos múltiplos e predição de genes serão descritas nos próximos capítulos.

Como mencionado anteriormente, podemos fazer alinhamentos utilizando diferentes funções de pontuação, que em geral tenta-se maximizar. Uma destas funções, supracitada neste texto, é a chamada precisão máxima esperada. Na maioria dos trabalhos sobre alinhamentos conhecidos essa função é definida como a soma das probabilidades de cada par de posições alinhadas encontrado no alinhamento Do *et al.* (2005); Sahraeian e Yoon (2010), como definido na equação 2.2. Com base nesta definição, a precisão se um alinhamento é dada pelo número de pares de posições das sequências que estão corretamente alinhados.

Apesar deste critério aparentemente fazer sentido, e de fato levar à construção de alinhamentos de alta qualidade, em 2008 Bradley e seus colaboradores (Bradley *et al.* , 2009) argumentaram que esta medida é na verdade uma medida de sensibilidade de um alinhamento, já que não penaliza o alinhamento entre caracteres que na realidade não são homólogos a nenhum outro. Neste mesmo artigo este autor mostra que a utilização de uma função que de fato mede a precisão dos alinhamentos, levando em conta a probabilidade de um caractere de uma sequência não ser homólogo a nenhum outro em uma segunda sequência, leva a um aumento da robustez do alinhamento em regiões em que não há homologia entre as sequências. A função de precisão sugerida por Bradley é dada por:

$$E[prec(A)] = \sum_{\forall x_i \sim y_j \in A} P(x_i \sim y_j) + \sum_{\forall x_i \sim y_- \in A} P(x_i \sim y_-) \tag{3.1}$$

Podemos ver que nesta definição todos os pares de posições do alinhamento são avaliados, não apenas aqueles que representam o alinhamento entre duas posições das sequências,

mas também aqueles que representam o alinhamento entre uma posição de uma sequência em um gap. A probabilidade de alinhamento entre uma posição em uma sequência $\mathbf{x}$ com um gap (nenhuma posição) em uma sequência $\mathbf{y}$ pode ser facilmente calculada através um pairHMM, utilizando o algoritmo forward-backward, da mesma maneira que as probabilidades de alinhamento entre dois resíduos.

No artigo em que Bradley apresenta essa nova medida de precisão para um alinhamento, a abordagem que ele utiliza para maximizar esta função, implementada na ferramenta FSA, é a construção de um alinhamento usando a técnica de *sequence annealing* utilizando uma função peso que leva em conta essas probabilidades de gap (Bradley *et al.* , 2009). Nessa implementação, o peso de cada par de resíduos inseridos no alinhamento é recalculado a cada passo, de maneira que o peso do novo par inserido seja consistente com aqueles previamente inseridos no alinhamento. Essa recálculo dos pesos faz com que não seja necessária a utilização de transformações de consistência nesta ferramenta.

Diversas outras ferramentas, incluindo tanto as que fazem alinhamento progressivo quanto as que utilizam *sequence annealing* com pesos que são diretamente as probabilidades a posteriori de alinhamento, poderiam se beneficiar da utilização desta outra definição de precisão esperada (Bradley *et al.* , 2009). Entretanto, se transformações de consistência fossem utilizadas, seria necessário definir uma transformação de consistência que deixe consistentes as probabilidades de alinhamentos com gaps, e que deixe as probabilidades de alinhamento entre os resíduos consistentes com as probabilidades de alinhamento com gap. Além disso, teria que ser feita uma modificação no algoritmo de construção de alinhamento que levasse em conta essas probabilidades.

3.1 Transformações de consistência com gaps

Dadas as probabilidades de alinhamento entre os pares de caracteres de duas sequências, a transformação de consistência probabilística original visa transformar estas probabilidades utilizando informações das outras sequências de forma que os alinhamentos entre os pares de sequências sejam consistentes uns com os outros. Os alinhamentos entre os pares de sequências serem consistentes significa, neste contexto, que se um resíduo x_i da sequência $\mathbf{x}$ está alinhado com um resíduo y_j da sequência $\mathbf{y}$, no alinhamento entre $\mathbf{x}$ e $\mathbf{y}$, e se y_j alinha com o resíduo z_k da sequência $\mathbf{z}$ no alinhamento entre $\mathbf{y}$ e $\mathbf{z}$, então x_i tem que também estar alinhado com z_k no alinhamento entre $\mathbf{x}$ e $\mathbf{z}$. Essa formulação de transformações de consistência foi definida na equação 2.9.

Abaixo apresentaremos uma proposta para uma nova transformação de consistência que leva em conta as probabilidades de alinhamento com gaps. Essa transformação pode ser utilizada em qualquer ferramenta de alinhamento múltiplo que utilize transformações de consistência. A nova fórmula pode beneficiar as ferramentas mesmo quando a definição da função de precisão esperada não leva em conta os gaps, já que na transformação de

consistência original a possibilidade de x_i estar alinhada com y_j no alinhamento entre $\mathbf{x}$ e $\mathbf{y}$ e estas mesmas duas posições não estarem alinhadas com nenhuma posição na sequência $\mathbf{z}$ (um gap na sequência $\mathbf{z}$) é negligenciada. Sendo S o conjunto de todas as sequências que estão sendo alinhadas, $|S|$ sendo o tamanho deste conjunto e $|\mathbf{z}|$ o tamanho da sequência $\mathbf{z}$, a transformação que sugerimos é dada por:

$$P^*(x_i \sim y_j) = \frac{\sum_{\mathbf{z}\in S}[\sum_{k=1}^{|z|} P(x_i \sim z_k)P(z_k \sim y_j) + \sum_{k=1}^{|z|+1} P(x_i \sim z_{-_k})P(z_{-_k} \sim y_j)]}{|S|} \quad \forall x_i \sim y_j$$

$$P^*(x_i \sim y_{-_j}) = \frac{\sum_{\mathbf{z}\in S} \sum_{k=1}^{|\mathbf{z}|} P(x_i \sim z_k)P(z_k \sim y_{-_j})}{|S|} \quad \forall x_i \sim y_{-_j} \tag{3.2}$$

As partes destacadas com cores são as inovações desta fórmula em relação à formulação original de consistência. Podemos ver que utilizamos as probabilidades de alinhamento entre posições e gaps localizado entre posições específicas para alterar a probabilidade de alinhamento entre duas posições (vermelho). Além disso, tornamos as probabilidades de alinhamento entre uma posição e um gap em um local determinado consistentes com os as outras probabilidades de alinhamento (azul).

Da mesma maneira que foi feito para a transformação de consistência original, podemos adicionar a esta fórmula os pesos para cada alinhamento, definidos na equação 2.10. Utilizando esses pesos temos:

$$P^*(x_i \sim y_j) = \frac{1}{\sum_{\mathbf{z}\in S} P(\mathbf{x} <> \mathbf{z})P(\mathbf{z} <> \mathbf{y})} \sum_{\mathbf{z}\in S}[P(\mathbf{x} <> \mathbf{z})P(\mathbf{z} <> \mathbf{y})\times$$
$$\times (\sum_{k=1}^{|z|} P(x_i \sim z_k)P(z_k \sim y_j) + \sum_{k=1}^{|z|+1} P(x_i \sim z_{-_k})P(z_{-_k} \sim y_j))] \quad \forall x_i \sim y_j$$

$$P^*(x_i \sim y_{-_j}) = \frac{\sum_{\mathbf{z}\in S} P(\mathbf{x} <> \mathbf{z})P(\mathbf{z} <> \mathbf{y}) \sum_{k=1}^{|\mathbf{z}|} P(x_i \sim z_k)P(z_k \sim y_{-_j})}{\sum_{\mathbf{z}\in S} P(\mathbf{x} <> \mathbf{z})P(\mathbf{z} <> \mathbf{y})} \quad \forall x_i \sim y_{-_j} \tag{3.3}$$

As diferenças entre essa equação e a 3.2 estão destacadas em vermelho. Podemos ver que a única diferença é a utilização dos pesos. Nas equações acima (3.2 e 3.3) consideramos que $P(x_i \sim x_{-_j}) = 0$, $P(x_i \sim x_j) = 1$ se $i = j$, e $P(x_i \sim x_j) = 0$ se $i \neq j$. Consideramos ainda que $P(\mathbf{x} <> \mathbf{x}) = 1$.

Como mencionamos anteriormente, um problema com as transformações de consistência originais é que as probabilidades de alinhamento com gaps são negligenciadas. Propusemos então as duas novas transformações acima. Nelas consideramos o alinhamento entre uma po-

sição de uma sequência e um gap em uma posição específica de outra sequência. Entretanto, podemos formular essas transformações de consistência de maneira que sejam considerados alinhamentos entre uma posição e um gap qualquer em outra sequência. A probabilidade de uma posição estar alinhada com algum gap em outra sequência é a soma das probabilidades de ela estar alinhada com cada um dos possíveis gaps. Essa será a probabilidade utilizada nesta versão de consistência. Essa nova versão está apresentada abaixo:

$$P^*(x_i \sim y_j) = \frac{\sum_{\mathbf{z}\in S}[P(x_i \sim z_-)P(z_- \sim y_j)P(x_i \sim y_j) + \sum_{k=1}^{|z|} P(x_i \sim z_k)P(z_k \sim y_j)]}{|S|} \quad \forall x_i \sim y_j$$

$$P^*(x_i \sim y_-) = \frac{\sum_{\mathbf{z}\in S} P(x_i \sim z_-)P(x_i \sim y_-) + \sum_{k=1}^{|z|} P(x_i \sim z_k)P(z_k \sim y_-)}{|S|} \quad \forall x_i \sim y_- \tag{3.4}$$

Neste caso podemos também utilizar os pesos. Com eles a fórmula ficaria da seguinte maneira:

$$P^*(x_i \sim y_j) = \frac{1}{\sum_{\mathbf{z}\in S} P(\mathbf{x} <> \mathbf{z})P(\mathbf{z} <> \mathbf{y})} \sum_{\mathbf{z}\in S}[P(\mathbf{x} <> \mathbf{z})P(\mathbf{z} <> \mathbf{y}) \times$$
$$\times P(x_i \sim z_-)P(z_- \sim y_j)P(x_i \sim y_j) + \sum_{k=1}^{|z|} P(x_i \sim z_k)P(z_k \sim y_j)] \quad \forall x_i \sim y_j$$

$$P^*(x_i \sim y_-) = \frac{1}{\sum_{\mathbf{z}\in S} P(\mathbf{x} <> \mathbf{z})P(\mathbf{z} <> \mathbf{y})} \sum_{\mathbf{z}\in S}[P(\mathbf{x} <> \mathbf{z})P(\mathbf{z} <> \mathbf{y})P(x_i \sim z_-)P(x_i \sim y_-) +$$
$$+ \sum_{k=1}^{|z|} P(x_i \sim z_k)P(z_k \sim y_-)] \quad \forall x_i \sim y_- \tag{3.5}$$

Podemos notar que neste caso consideramos que se x_i, ou y_j têm uma probabilidade grande de não estarem alinhados com nenhuma posição de $\mathbf{z}$, mantemos as probabilidades de alinhamento originais entre x_i e y_j (em vermelho nas equações 3.4 e 3.5). Vemos ainda que ao invés de alterar as probabilidades de alinhamento com gaps específicos, tornamos consistentes as probabilidades de alinhamento com gaps quaisquer (em azul nas equações 3.4 e 3.5).

3.2 *Sequence annealing* modificado

Apresentamos nesta seção uma versão modificada para o algoritmo de construção de alinhamento múltiplo por *sequence annealing*. Esta modificação visa fazer com que o algoritmo maximize a função de precisão esperada que leva em conta o alinhamento com gaps, definida na equação 3.1.

Já existe uma versão deste algoritmo que visa maximizar esta função de precisão esperada. Entretanto, nesta versão, apresentada no artigo de Bradley *et al.* (Bradley *et al.* , 2009), não são utilizadas transformações de consistência, e os pesos têm que ser recalculados em cada passo do algoritmo.

Em nossa versão modificada, ao invés de utilizarmos pesos que levam em conta as probabilidades de alinhamento com gaps, como foi feito em Bradley *et al.*, inserimos pares posição-gap diretamente no alinhamento. Para isso, ao ordenar os pares de posições alinhados de acordo com suas probabilidades *a posteriori*, devem ser incluídos os pares posição-gap. Esses pares posição-gap devem ser tratados como um par posição-posição, sendo portanto inseridos no alinhamento quando forem o par com maior probabilidade *a posteriori* ainda não inserido no alinhamento, desde que seja consistente com os pares previamente inseridos.

A verificação de consistência com os pares previamente inseridos é feita da mesma maneira que no algoritmo original, já que a inserção de um par posição-gap inconsistente com as posições já presentes no alinhamento também acarreta no aparecimento de um ciclo no grafo de alinhamento, o que pode ser verificado de maneira eficiente.

3.3 Validação

A maneira como implementamos a nossa ferramenta permite fazermos alinhamentos utilizando as transformações de consistência modificadas ou as clássicas, assim como podemos utilizar os método de construção de alinhamentos múltiplos por *sequence annealing* sugerido neste capítulo, ou a formulação original deste método. Isso permite comparar os alinhamentos gerados utilizando-se cada uma dessas técnicas. Além disso, como utilizamos um mesmo *framework* e modelo para gerar os alinhamentos, podemos garantir que as diferenças na qualidade dos alinhamentos sejam advindas apenas das diferentes técnicas utilizadas, o que pode não ocorrer quando comparamos os resultados gerados por ferramentas diferentes.

Apresentamos neste documento 6 versões diferentes de transformações de consistência, sendo duas delas as versões implementadas por outras ferramentas, e as outras quatro as versões propostas neste trabalho. Além disso, apresentamos duas versões do algoritmo de *sequence annealing*. Portanto, avaliamos a performance de 14 versões diferentes de nossa ferramenta, essas versões estão apresentadas na Tabela 3.1.

	Sequence annealing		Transformação de consistência					
Abreviação	Original	Modificado	Nenhuma	Sem pesos	Com pesos	Gaps específicos	Gaps genéricos	Equação
O - N	X		X					
O - SP	X			X				2.9
O - CP	X				X			2.11
O - SPGE	X			X		X		3.2
O - CPGE	X				X	X		3.3
O - SPGG	X			X			X	3.4
O - CPGG	X				X		X	3.5
M - N		X	X					
M - SP		X		X				2.9
M - CP		X			X			2.11
M - SPGE		X		X		X		3.2
M - CPGE		X			X	X		3.3
M - SPGG		X		X			X	3.4
M - CPGG		X			X		X	3.5

Tabela 3.1: *Descrição das diferentes versões de nossa ferramenta de construção de alinhamentos múltiplos que testaremos. Elas são combinações de um dos métodos de construção por sequence annealing e uma versão de transformações de consistência. Há também duas versões que não têm transformações de consistência.*

Todos os testes descritos nesta seção foram feitos utilizando um modelo pairHMM de 5 estados (dois conjuntos de estados de inserção e deleção), com parâmetros estimados a partir da matriz BLOSSUM62 e treinados sobre o conjunto de alinhamentos presentes no BaliBase através de um algoritmo de maximização da esperança.

As propostas de modificações na formulação de transformações de consistência, e na maneira como a técnica de *sequence annealing* constrói alinhamentos múltiplos, podem ser aplicadas tanto em alinhamentos de proteínas, quanto de sequências genômicas. Entretanto, como mencionado na seção 2.2.4, alinhamentos de proteínas são mais facilmente avaliados do que alinhamentos entre sequências genômicas. Assim, utilizamos os *benchmarks* BaliBase 3.0 (Thompson *et al.* , 2005), e IRMBASE 2.0 (Subramanian *et al.* , 2008), descritos na seção 2.2.4 para avaliar se as técnicas propostas neste capítulo de fato trazem benefícios para os alinhamentos. No caso dos testes com o BaliBase 3.0, faremos medidas de qualidade utilizando apenas os *core blocks* assim como utilizando todas as colunas do alinhamento referência.

Como mencionado na seção 2.2.4, existem duas medidas de qualidade de alinhamentos múltiplos que são usualmente utilizadas: porcentagem de pares de posições corretamente alinhados (SP), e porcentagem de colunas inteiras que foram corretamente alinhadas (CS). É interessante notar que no critério SP, não são levados em conta os pares posição-gap. Além disso, dizer que uma coluna foi corretamente alinhada significa que todas as posições das sequências que deveriam estar alinhadas naquela coluna, de fato estavam. Assim, se houver um gap em uma coluna no alinhamento referência, mas colocarmos uma outra posição no lugar deste gap no alinhamento que está sendo avaliado, esta coluna continuará sendo

considerada como corretamente alinhada, já que contém todas as posições que deveriam estar alinhadas segundo a referência.

Fica claro, portanto, que o alinhamento entre duas posições que deveriam na realidade estar alinhadas com gaps, não será penalizado em nenhuma dessas duas medidas de qualidade. Por essa razão, sugerimos aqui duas outras medidas, similares a estas, mas que levam em conta o alinhamento correto de gaps. Chamaremos essas medidas de SPG e CSG. Definimos SPG como a porcentagem de pares posição-posição, ou posição-gap corretamente alinhados. CSG é a porcentagem de colunas corretamente alinhadas, entretanto, só consideraremos que uma coluna está alinhada corretamente se ela for idêntica a uma coluna presente na referência, inclusive com os gaps nas sequências corretas.

Os testes que fizemos consistem, portanto, em 4 etapas:

1. Alinhar as sequências contidas em cada teste do BaliBase 3.0 e do IRMBASE 2.0 utilizando cada uma das versões da ferramenta.

2. Avaliar cada um desses alinhamentos de acordo com as medidas SP, CS, SPG e CSG. No caso do BaliBase 3.0, calculamos esses valores utilizando tanto apenas os *core blocks*, quanto todas as colunas do alinhamento. Temos que lembrar que não há gaps nos *core blocks*, logo não precisamos calcular SPG e CSG nesse caso. Geramos portanto 10 conjuntos de resultados para cada versão da ferramenta: SP-BaliBaseCB, CS-BaliBaseCB, SP-BaliBaseTC, CS-BaliBaseTC, SPG-BaliBaseTC, CSG-BaliBaseTC, SP-IRMBASE, CS-IRMBASE, SPG-IRMBASE, e CSG-IRMBASE.

3. Encontrar a média, mediana e desvio padrão para cada versão da ferramenta e cada um dos conjuntos de resultados.

4. Para cada par de versões de nossa ferramenta e cada conjunto de resultados, fazer o teste de hipótese *Wilcoxon Signed-Rank* (Wilcoxon , 1945), para verificar se alguma delas é significativamente melhor do que a outra. Este teste está descrito na seção 2.2.4.

A última etapa dos testes envolve realizar o teste de hipótese *Wilcoxon Signed-Rank*. Para exemplificar a aplicação deste teste, pensemos no conjunto de resultados SP-BaliBaseTC. Neste conjunto de resultados temos para cada alinhamento do *benchmark* BaliBase 3.0, um valor de SP para cada versão de nossa ferramenta. No teste *Wilcoxon Signed-Rank* temos sempre que comparar apenas duas versões da ferramenta, M-CPGG e O-CPGG por exemplo. Para este conjunto de resultados e essas duas versões da ferramenta, o que este teste visa verificar é se a mediana das diferenças entre os valores de SP para cada um dos alinhamentos do conjunto BaliBase 3.0 é igual ou diferente de zero.

Realizamos esse teste para todos os pares de versões da ferramenta, e também todas as medidas de qualidade apresentadas aqui. Devido à grande quantidade de conjuntos de resultados e também de versões de nossa ferramenta que foram testadas, apresentamos para cada conjunto de medidas, apenas os resultados das versões da ferramenta que não foram

significativamente piores (com nível de significância de 1%) do que nenhuma outra versão. Esses resultados podem ser observados na Tabela 3.2.

A primeira observação que fizemos a partir dos resultados foi que quando utilizamos uma medida de qualidade do alinhamento que não envolve o alinhamento correto de gaps, é mais adequado utilizar a versão original do algoritmo de *sequence annealing*. Observamos ainda que a versão modificada do *sequence annealing* é mais adequada quando queremos maximizar um critério de qualidade que envolve o posicionamento correto de gaps. Esta observação é consistente em todos os conjuntos de teste. Este resultado era esperado, já que o *sequence annealing* original visa maximizar a precisão esperada sem gaps do alinhamento múltiplo (SP), enquanto o *sequence annealing* proposto neste projeto visa maximizar a precisão esperada com gaps (SPG).

Pudemos observar que no conjunto de resultados CSG-BaliBaseTC, acontece um empate técnico entre uma versão da ferramenta com o *sequence annealing* modificado, e outras duas que utilizam o *sequence annealing* original. Entretanto, como nos SPG-IRMBASE, CSG-IRMBASE e SPG-BaliBase as versões com o *sequence annealing* modificado tem performances significativamente melhores, a conclusão de que é melhor utilizar esta modificação quando deseja-se maximizar os critérios com gaps ainda é válida.

Em termos de transformações de consistência, pudemos concluir que a utilização de gaps genéricos, como apresentado na equação 3.5 é a melhor opção. As versões de nossa ferramenta que utilizam esta transformação de consistência (M-CPGG, ou O-CPGG) obtêm uma das melhores performances (nenhuma outra versão é significativamente melhor que ela a 1% de significância), se não a melhor (significativamente melhor que todas as outras a 1% de significância), em todos os conjuntos de teste que fizemos e utilizando qualquer critério de avaliação. É importante lembrar que para critérios com gaps é mais adequado utilizar M-CPGG, enquanto para critérios sem gap é mais adequado utilizar O-CPGG.

É importante notar que quando utilizamos transformações de consistência, se não fizermos as consistências com gaps, como proposto neste trabalho, o *sequence annealing* modificado tem uma performance significativamente pior do que todas as outras versões da ferramenta em qualquer conjunto de teste. Acreditamos que isso ocorra pois as transformações de consistência sem gaps acabam diminuindo o valor das probabilidades de alinhamento entre as posições em cada iteração, enquanto as probabilidades de alinhamento com gaps não são modificadas. Assim, quando construímos os alinhamentos levando em conta as probabilidades de alinhamento com gaps, o alinhamento fica deteriorado.

Conjunto de medidas	Ferramenta	Mediana	Média	Desvio padrão
SP-BaliBaseCB	O-CPGE	95,50	85,14	17,60
	O-CP	95,50	86,72	14,56
	O-CPGG	95,60	86,13	15,61
CS-BaliBaseCB	O-CPGE	75,00	55,45	30,36
	O-CP	75,00	55,61	30,23
	O-CPGG	75,00	54,36	31,35
SP-BaliBaseTC	O-CPGG	70,50	69,71	19,40
CS-BaliBaseTC	O-CPGE	40,00	37,82	25,33
	O-CPGG	40,00	37,34	25,88
SPG-BaliBaseTC	M-CPGG	77,64	77,36	12,45
CSG-BaliBaseTC	O-CP	33,79	36,96	22,51
	M-CPGG	28,34	37,15	22,39
	O-SP	33,31	35,89	22,27
SP-IRMBASE	O-CPGE	97,84	88,34	22,00
	O-SPGE	97,66	90,62	18,56
	O-CP	97,66	88,36	21,99
	O-SP	97,66	90,66	18,55
	O-CPGG	97,51	88,91	21,03
	O-SPGG	97,84	89,12	19,39
CS-IRMBASE	O-CPGE	92,19	77,15	26,32
	O-SPGE	92,19	79,90	24,50
	O-CP	92,19	77,2	26,26
	O-SP	92,19	79,97	24,49
	O-CPGG	92,19	77,74	26,23
	O-SPGG	92,19	77,31	26,61
SPG-IRMBASE	M-N	96,22	87,59	12,47
	M-CPGG	95,64	86,25	15,32
CSG-IRMBASE	M-N	92,75	64,77	26,14
	M-SPGG	91,30	62,36	28,15
	M-CPGG	90,58	65,05	25,91

Tabela 3.2: *Resultados obtidos pelas versões da ferramenta mais bem sucedidas. Apenas aquelas versões que não foram consideradas piores que nenhuma outra em cada um dos conjuntos de medidas, com nível de significância de 1%, têm seus resultados apresentados aqui. Vemos que as versões da ferramenta com consistência CPGG (3.5) não foram superadas por nenhuma outra variação de transformação de consistência em nenhum dos testes. Vemos que a versão do algoritmo sequence annealing original (O) é a melhor quando utilizamos medidas de qualidade que não levam em conta o posicionamento correto de gaps (SP e CS). Entretanto, quando o posicionamento dos gaps é considerado (SPG e CSG), a versão modificada do sequence annealing (M) é mais adequada.*

Capítulo 4

Novos algoritmos para GHMM

O cálculo de probabilidades *a posteriori* de caracteres serem emitidos por determinados estados de um modelo oculto de Markov envolve o algoritmo *forward-backward* em HMMs tradicionais, assim como em pairHMMs. Entretanto, quando tratamos de modelos GHMM, esse mesmo algoritmo não é adequado devido à presença de estados que podem emitir um número variável de caracteres. Neste capítulo apresentaremos um algoritmo para realizar esta tarefa em modelos GHMM, que até onde pudemos investigar, é inédito.

Apresentaremos ainda duas possíveis utilizações destas probabilidades. A primeira delas é como uma medida de qualidade da predição de genes para posições individuais de uma sequência. Mostraremos que essas probabilidades podem de fato dar indícios sobre a qualidade da predição em cada ponto.

A segunda aplicação desses valores de probabilidade é na construção de predições de genes de máxima precisão esperada, ou seja, aquela que maximiza o número esperado de posições corretamente preditas. Para isso utilizamos um algoritmo de programação dinâmica também original deste trabalho. Neste capítulo apresentaremos esse algoritmo, assim como a comparação dos resultados obtidos por ele e aqueles obtidos utilizando o algoritmo de Viterbi.

4.1 Cálculo de probabilidades *a posteriori* com GHMM

O cálculo de probabilidades *a posteriori* de cada posição da sequência ter sido emitida por cada um dos estados do modelo pode ser realizado através do algoritmo *forward-backward* em HMMs tradicionais (Durbin *et al.* , 1998). Entretanto, o fato de os preditores de genes tradicionalmente utilizarem GHMMs faz com que seja necessário introduzir uma modificação neste algoritmo, de maneira que seja levado em conta o fato de alguns estados neste modelo poderem emitir um número arbitrário de caracteres, sorteado segundo uma distribuição especificada no modelo.

Consideremos que $\mathbf{x} = x_1...x_L$ é uma sequência de nucleotídeos de tamanho L para a qual queremos calcular as probabilidades a posteriori de cada uma de suas posições terem sido

emitidas pelos diferentes estados do GHMM em questão. Seja $\phi_i = (s_1, \mathbf{x_1}), (s_2, \mathbf{x_2}), ..., (s_n, \mathbf{x_n})$ uma sequência de estados do GHMM e sub-sequências de $\mathbf{x}$, definida de maneira que a concatenação dos segmentos $\mathbf{x_1 x_2 ... x_n} = x_1...x_i$. Definimos então a função *forward* para um GHMM como:

$$f(i, s) = \sum_{\forall \phi_i} P(\phi_i, s_n = s) \tag{4.1}$$

Ou seja, $f(i, s)$ é a somatória das probabilidades de todas as maneiras de emitir a sub-sequência $x_1...x_i$ pelo modelo, de forma que o último estado a fazer uma emissão seja o estado s.

Seja $\beta_i = (s_1, \mathbf{x_1}), (s_2, \mathbf{x_2}), ..., (s_n, \mathbf{x_n})$ uma sequência qualquer de estados e sub-sequências de $\mathbf{x}$, definida de maneira que a concatenação dos segmentos $\mathbf{x_1 ... x_n} = x_{i+1}...x_L$. Definimos a função *backward* para um GHMM como:

$$b(i, s) = \sum_{\forall \beta_i} P(\beta_i | s_0 = s) \tag{4.2}$$

Ou seja, $b(i, s)$ é a somatória das probabilidades de todas as maneiras de o modelo emitir a sequência $x_{i+1}...x_L$, dado que o modelo estava no estado s antes de entrar no caminho β_i.

Tendo as funções *forward* e *backward* definidas desta maneira, podemos, assim como em um HMM tradicional, calcular a probabilidade da sequência x ser emitida por este modelo. Pela definição das funções *forward* e *backard* temos que:

$$P(\mathbf{x}) = \sum_{\forall s} b(0, s) = \sum_{\forall s} f(L, s) \tag{4.3}$$

Os valores das funções *forward* e *backward* podem ser calculados de maneira eficiente através de algoritmos de programação dinâmica, já que:

$$\begin{aligned}
f(i, s) &= \sum_{j=0}^{j=i-1} \sum_{\ell \in S} f(j, \ell) t(\ell, s) e(x_{j+1}...x_i, s) d(i - j, s) \quad \text{se } i \geq 1 \\
f(0, s) &= p_{init}(s) \\
b(i, s) &= \sum_{j=i+1}^{j=L} \sum_{\ell \in S} b(j, \ell) t(s, \ell) e(x_{i+1}...x_j, \ell) d(j - i, \ell) \quad \text{se } i < L \\
b(L, s) &= p_{term}(s)
\end{aligned} \tag{4.4}$$

de acordo com a definição de GHMM apresentada no capítulo 2.1, S é o conjunto de estados do GHMM, $t(a, b)$ é a probabilidade de transição do estado a para o estado b, $e(x_i...x_j, s)$ é a probabilidade de emissão da sub-sequência $x_i...x_j$ no estado s, $d(a, s)$ é a probabilidade de o estado s emitir a símbolos, $p_{init}(s)$ é a probabilidade de começar a emissão da sequência no estado s e $p_{term}(s)$ é a probabilidade de acabar a emissão da sequência no estado s.

Utilizando os valores das funções *forward* e *backward*, e a probabilidade da sequência $\mathbf{x}$ ser emitida pelo modelo, pode-se calcular a probabilidade *a posteriori* de um determinado segmento de sequência ser emitido por um estado específico do modelo, dada a sequência $\mathbf{x}$. Isto é feito da seguinte maneira:

$$\begin{aligned} P((s, x_i...x_j)|\mathbf{x}) =& \frac{P((s, x_i...x_j), \mathbf{x})}{P(\mathbf{x})} = \frac{P(x_1...x_{i-1}, (s, x_i...x_j), x_{j+1}...x_L)}{P(\mathbf{x})} \\ =& \frac{e(x_i...x_j, s)d(j-i+1, s)b(j, s)\sum_{\ell \in S} f(i-1, \ell)t(\ell, s)}{P(\mathbf{x})} \end{aligned} \tag{4.5}$$

onde $(s, x_i...x_j)$ é equivalente a dizer que o segmento $x_i...x_j$ é emitido pelo estado s.

Com base nas definições acima, propomos um algoritmo para calcular a probabilidade *a posteriori* de cada base da sequência $\mathbf{x}$ ter sido emitida por cada estado s, ao invés de calcular a probabilidade a posteriori de um segmento inteiro da sequência ser emitido como mencionado acima.

Como podemos ver no pseudo-código 4.6, definimos para cada classe s e posição x_i da sequência uma variável p que guardará a probabilidade *a posteriori* de x_i ter sido emitida pelo estado s. Iniciamos o algoritmo dando o valor zero para todas as variáveis p. Calcula-se então todos os valores da função *backward* para a sequência $\mathbf{x}$ da maneira tradicional mencionada acima.

Em seguida calculamos o valor da função *forward*. Entretanto, no momento em que fazemos a soma para todas as possíveis emissões de um estado acabando em uma determinada posição da sequência, necessária para calcular a função forward, calculamos também a probabilidade *a posteriori* de cada um desses segmentos ser emitido por aquele estado. Para isso, necessita-se apenas dos valores de *forward* e *backward* previamente calculados, dos parâmetros do modelo, e da probabilidade total da sequência neste modelo, calculada no algoritmo *backward*. Calculada essa probabilidade, adicionamos a mesma às variáveis p correspondentes àquele estado e à todas as posições da sequência contidas no segmento em questão. Esta etapa do algoritmo, original deste trabalho, está destacada em vermelho no pseudo-código 4.6.

Após finalizado o algoritmo teremos somado para cada posição e cada estado a probabilidade de todos os segmentos que envolvem aquela posição que podem ser emitidos por cada um dos estados, dando portanto a probabilidade a posteriori da posição ter sido emitida por aquele estado dada a sequência de observações.

Algoritmo de cálculo de probabilidades *a posteriori* em GHMM

Inicialização:

$p(i,s) = 0 \quad \forall s \in S \text{ e } 1 \leq i \leq L$

Calcular $b(i,s) \quad \forall s \in S \text{ e } 0 \leq i \leq L$

$P(\mathbf{x}) = \sum_{\forall s} b(0,s)$

$f(0,s) = p_{init}(s) \quad \forall s$

Recursão: (4.6)

Loop 1: Para $i = 1$ até $i = L$

Loop 2: $\forall s \in S$

$f(i,s) = 0$

Loop 3: Para $j = 0$ até $j = i - 1$

$temp = \sum_{\forall \ell \in S} f(j,\ell) t(\ell,s) e(x_{j+1}...x_i, s) d(i-j,s)$

$f(i,s) = f(i,s) + temp$

Loop 4: $k = j + 1$ até $k = i$

$p(k,s) = p(k,s) + \frac{temp \times b(i,s)}{P(\mathbf{x})}$

Um problema com esta abordagem seria que o número de segmentos que podem ser emitidos pelos estados pode chegar a ser excessivamente grande (Loop 3), já que eles podem ter qualquer tamanho. Felizmente, os GHMMs utilizados para predição de genes têm algumas características particulares do espaço de estados que facilitam essa tarefa. Nesses modelo nem todos os estados têm duração arbitrária como mencionado anteriormente. Na realidade apenas os estados que modelam éxons têm essa característica (Kashiwabara e Durham , 2011). Sinais como códons de inicial, códon de parada, sítio aceitador de splicing e sítio doador de splicing, são modelados como cadeias de Markov de tamanho constante e íntrons são estados de duração geométrica, emitindo apenas um caractere por vez.

Algum par de sinais deve sempre aparecer ao redor dos exons em um gene, assim, um éxon só tem probabilidade maior que zero de começar e terminar entre regiões da sequência que tenham probabilidade maior do que zero de apresentarem os sinais adequados. Isso faz com que seja possível manter uma lista com os possíveis pontos em que os estados de duração explícita, como os que representam éxons, podem começar e terminar, aumentando muito a eficiência dos algoritmos *forward*, *backward* e consequentemente do algoritmo proposto neste trabalho. Além disso, quando calculamos a variável *temp*, não é necessário fazer essa soma

para todos os estados ℓ, mas apenas para aqueles que têm transição para s, o que aumenta ainda mais a eficiência deste algoritmo.

4.2 Algoritmo para encontrar a predição de MPE

Após termos calculado as probabilidades de cada posição da sequência ter sido emitida por cada estado do modelo, podemos utilizá-las para encontrar a predição de máxima precisão esperada. Para fazer isso, utilizaremos um algoritmo de programação dinâmica, similar ao posterior decoding para HMMs tradicionais. Entretanto, é importante notar que a predição que tem o maior número esperado de posições corretas, aquela em que escolhemos para cada posição o estado mais provável de tê-la emitido, pode não corresponder a um caminho que faça sentido no modelo, ou seja, uma predição inconsistente. O que desejamos, portanto, é a predição de genes consistente de máxima precisão esperada. Para isso, primeiramente, é necessário definir a seguinte função para cada par de estados ℓ e s pertencentes ao conjunto de estados do GHMM S:

$$I(\ell, s) = \begin{cases} 1 & \text{se } t(\ell, s) > 0 \\ 1 & \text{se } \ell = s \\ 0 & \text{se } t(\ell, s) = 0 \text{ e } \ell \neq s \end{cases} \tag{4.7}$$

Vamos convencionar aqui que $c_1, c_2, ..., c_L$ são as posições $1, 2, ..., L$ de um caminho pelo GHMM que emite a sequência $\mathbf{x} = x_1...x_L$, convencionemos ainda que $C^* = c_1^*...c_L^*$ é o caminho no modelo GHMM que corresponde à predição de genes consistente de máxima precisão esperada, que emite a sequência $\mathbf{x}$. Ser consistente, neste contexto, significa não ter transições que violem o caminho definido pelo GHMM em questão.

O caminho C^* pode ser encontrado recursivamente. Assuma que a precisão esperada da predição consistente de máxima precisão esperada terminando no estado s com a observação x_i, $pe(i, s)$, seja conhecida para todo estado $s \in S$. Logo, é possível calcular o valor de $pe(i+1, s)$ como:

$$pe(i+1, s) = max_\ell(pe(i, \ell)I(\ell, s)) + P(c_{i+1} = s|\mathbf{x}) \tag{4.8}$$

A precisão esperada do caminho consistente de máxima precisão esperada terminando no estado s com a observação x_1, será sempre $P(c_1 = s|\mathbf{x})$, ou seja, a probabilidade a posteriori do estado s na posição 1. Assim, a condição inicial é que $pe(1, s) = P(c_1 = s|\mathbf{x}) \quad \forall s \in S$. Guardando ponteiros para o estado anterior, a sequência de estados C^* pode ser encontrada através de um procedimento de *traceback*. O algoritmo completo é portanto dado por:

Predição de precisão máxima esperada

Inicialização:

$$pe(1, s) = P(c_1 = s|\mathbf{x}) \quad \forall s \in S$$

Recursão: $i = 2...L$

$$pe(i, s) = max_\ell(pe(i-1, \ell)I(\ell, s)) + P(c_i = s|\mathbf{x}) \quad \forall s \in S$$
$$ptr(i, s) = argmax_\ell(pe(i-1, \ell)I(\ell, s)) \quad \forall s \in S \tag{4.9}$$

Terminação:

$$MPE = max_s(pe(L, s))$$
$$c_L^* = argmax_s(pe(L, s))$$

Traceback: $i = L - 1...1$

$$c_i^* = ptr(i, c_{i+1}^*)$$

onde MPE é a precisão esperada do caminho C^*. Se dividirmos MPE pelo tamanho L da sequência, obteremos um valor de precisão média das predições feitas para cada posição da sequência, que pode ser utilizado como uma medida de confiabilidade, segundo o modelo, daquela predição.

4.3 Validação

Nesta seção apresentaremos os testes que realizamos para validar os algoritmos apresentados neste capítulo. Fizemos dois testes principais. O primeiro deles visava validar a técnica de alinhamento de máxima precisão esperada. Neste teste comparamos os resultados obtidos por esta nova técnica com aqueles encontrados utilizando o algoritmo de Viterbi. O segundo teste visa verificar se o valor de probabilidade *a posteriori* para a predição em uma determinada posição da sequência pode servir como uma medida de confiabilidade para a predição feita para aquela posição específica.

4.3.1 Comparação entre algoritmo MPE com o de Viterbi

O primeiro teste que fizemos para validar os algoritmos apresentados neste capítulo foi fazer predições de gene e compará-las com as predições geradas pelo algoritmo de Viterbi. É importante lembrar que a implementação dos algoritmos apresentados neste capítulo foi feita

sobre o arcabouço de predição de genes ToPS (Kashiwabara e Durham , 2011), portanto, métodos de treinamento de preditores, assim como o algoritmo de Viterbi (Durbin *et al.* , 1998) para encontrar a predição mais provável para uma sequência já haviam sido implementados. Assim, foi possível fazer a comparação destes dois algoritmos de predição em cima de um mesmo arcabouço, utilizando os mesmos modelos treinados. A vantagem deste tipo de comparação é que por estarmos utilizando o mesmo modelo e o mesmo arcabouço, não há viéses advindos de implementação ou modelagem diferentes.

No trabalho de Kashiwabara *et al.*, foi evidenciado que a maioria das comparações entre preditores de genes não são feitas de maneira justa. Isso ocorre pois diversos preditores de genes utilizam modelo pré-treinados para fazer predições, e o retreinamento destes modelos pode não ser tarefa trivial. Sendo assim, muitas vezes os genes em que o teste foi realizado podem ter feito parte do conjunto de treinamento do modelo, o modelo pode ter sido treinado em genes de outro organismo, entre outros fatores que podem comprometer a comparação. Também no trabalho de Kashiwabara *et al.* foi descrito um protocolo mais adequado de comparação entre preditores, que será utilizado neste trabalho para comparar os dois algoritmos de predição (Kashiwabara e Durham , 2011). Este protocolo consiste em:

- Obter conjuntos de genes bem anotados. A seleção destes conjuntos de genes já havia sido feita para a validação da plataforma ToPS (Kashiwabara e Durham , 2011). Nesta seleção só eram escolhidos aqueles genes com evidência de expressão e sem erro aparente de anotação. Genes com sítios de splicing não canônicos também foram removidos, já que os preditores avaliados não modelam esses tipos de sítio. O teste que realizamos foi sobre os genes de camundongo, já que existe uma grande quantidade de genes (16385 genes) deste organismo que se encaixam no critério de boa anotação.

- Selecionar de maneira aleatória uma série de subconjuntos $G_1 \subset G_2 \subset G_3 \subset ... \subset G_15$ dos genes escolhidos na primeira etapa, em que o primeiro subconjunto está contido no segundo, que está contido no terceiro, etc. Para construir estes conjuntos são selecionados 125 genes aleatóriamente para o primeiro conjunto, estes mesmo 125 genes e mais 125 outros genes para o segundo, e assim por diante até construirmos o último conjunto.

- Preparar 15 testes de validação cruzada de tamanho crescente, particionando cada conjunto G_k em 5 partes iguais, ou seja, partes de tamanho $k \times 25$. Cada parte foi indexada com um número de 1 até 5. Para cada índice $i \in \{1, 2, 3, 4, 5\}$, os genes da parte i foram utilizados como amostra de teste e todos os outros elementos das partes indexados por $j \neq i$ como amostras de treinamento. Assim, para cada conjunto G_k executamos 5 experimentos utilizando $k \times 100$ elementos para o treinamento e $k \times 25$ elementos para teste. No total, teremos 75 resultados com diversas medidas de exatidão calculadas pelo programa de comparação de anotações, SGEval (Kashiwabara e Durham , 2011).

É interessante notar que este protocolo de teste deixa claro não somente o resultado da

utilização de diferentes métodos de predição utilizando um modelo treinado específico, mas também evidencia a curva de aprendizado de cada um dos métodos de predição à medida que aumenta o conjunto de treinamento.

Em nosso trabalho, os testes mencionados na terceira etapa do protocolo foram feitos utilizando tanto o algoritmo de Viterbi, quanto o de máxima precisão esperada (MPE). O treinamento em cada um dos testes era feito apenas uma vez, e o mesmo modelo treinado foi utilizado para fazer as predições utilizando cada um dos algoritmos.

O programa SGEval, mencionado acima, realiza uma comparação das estruturas de genes preditas com a anotação referência para estes mesmos genes. Como resultado da comparação são geradas três medidas de exatidão: *positive predictive value*, *sensitivity*, e *F-score*. Estes valores de exatidão são medidos em três pontos de vista: sítios codificadores preditos, éxons preditos, e variantes de estruturas de genes preditas (Kashiwabara e Durham , 2011).

Na Figura 4.1, apresentamos os resultados obtidos pelo algoritmo de Viterbi e MPE nos três contextos mencionados acima, e em cada uma das medidas de qualidade da predição. Cada ponto nos gráficos desta figura representa a média das medidas de exatidão para os 5 experimentos feitos utilizando cada um dos subconjuntos de genes.

Podemos observar nesta figura que em termos de nucleotídeos corretamente preditos, o algoritmo MPE obtem resultados muito próximos aos do algoritmo de Viterbi, sendo em média 0,8% pior, em relação ao F-score. Quando analisamos estes mesmos resultados no contexto de éxons completos corretamente preditos, verificamos que a sensibilidade do algoritmo MPE, tem diferença de em média 0,1% em quando comparada com o algoritmo de Viterbi. Já o PPV para este mesmo contexto, é em média 6% pior para o algoritmo MPE do que para o Viterbi, o que resulta em uma desvantagem também em termos de F-score. Na análise em termos de genes completos, podemos observar que o algoritmo MPE é em média 1,4% pior que o Viterbi em todas as medidas.

Apesar de essas medidas de exatidão fornecerem um bom resumo da qualidade dos preditores, sua mera utilização pode não ser suficiente para decidir qual preditor é o melhor quando observamos resultados muito similares, já que as medidas escondem muitos detalhes. Nestes casos, diagramas de Venn, também gerados pela ferramenta SGEval, nos mostram se os diferentes preditores fornecem anotações complementares, que não deixam de ser evidências válidas de genes codificadores de proteínas.

Através desses diagramas, pudemos observar que não há um alto grau de complementariedade entre os algoritmos de Viterbi e MPE. Isso era de fato esperado, já que, diferente de quando comparamos duas ferramentas de predição, estamos neste caso utilizando exatamente os mesmos modelos. O diagrama de Venn está representado na Tabela 4.1. Podemos observar nesta tabela que o algoritmo há muito poucos éxons encontrados por um dos algoritmos e não pelo outro, evidenciando este baixo grau de complementariedade entre as predições feitas utilizando cada um desses algoritmos. Fica evidente ainda a quantidade elevada de éxons falsos positivos preditos pelo algoritmo MPE, consistente com o PPV reduzido neste quesito.

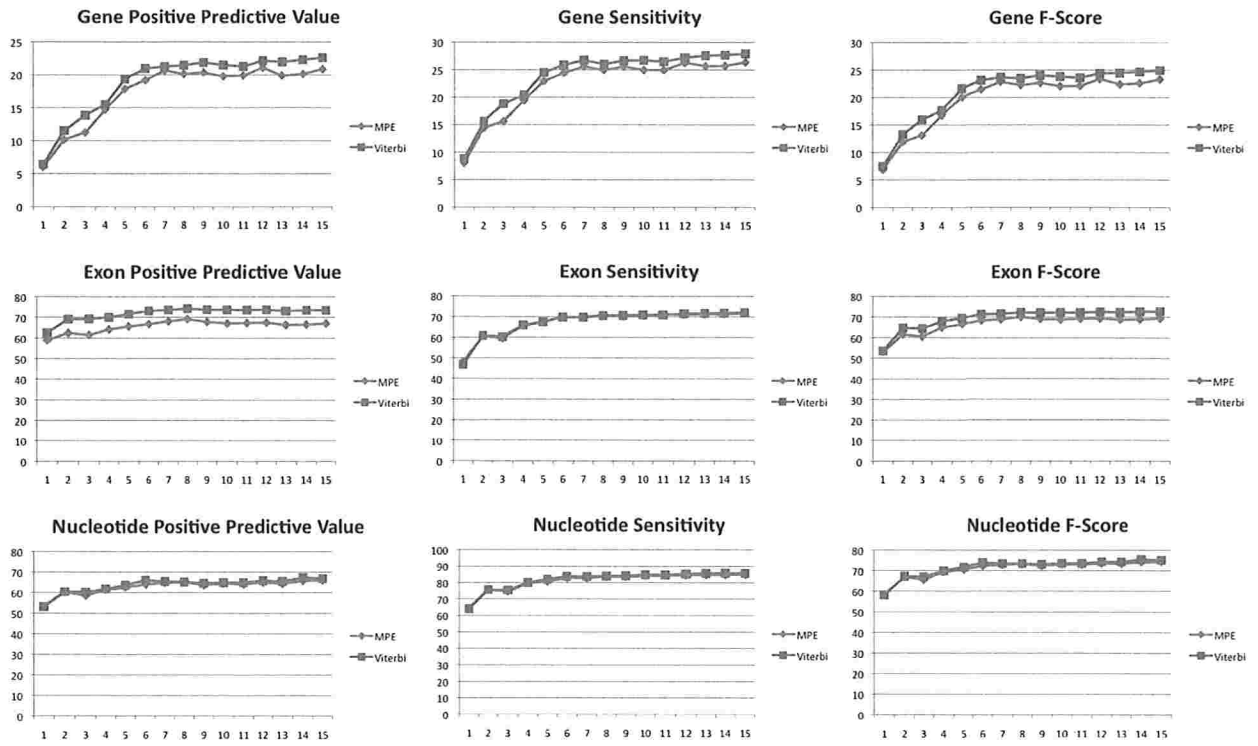

Figura 4.1: *Resultados obtidos pelos algoritmos de predição de genes MPE e Viterbi no protocolo de teste aplicado em genes de camundongo. São utilizadas três medidas de exatidão: PPV, sensibilidade e F-score. Essas são medidas em três contextos: genes completos corretamente preditos, éxons completos corretamente preditos, e nucleotídeos codificadores corretamente preditos. Cada ponto dos gráficos representa a média daquela medida para os 5 experimentos realizados sobre os grupos de genes $G_1...G_{15}$, descritos no protocolo de teste.*

Conjunto	Quantidade
Verdadeiros positivos(Viterbi e MPE)	11806
Verdadeiros positivos(apenas MPE)	252
Verdadeiros positivos(apenas Viterbi)	313
Falsos positivos(MPE e Viterbi)	3036
Falsos positivos(apenas MPE)	2477
Falsos positivos(apenas Viterbi)	914
Falsos negativos(MPE e Viterbi)	4452

Tabela 4.1: *Representação tabular do diagrama de Venn gerado como resultado do teste de validação cruzada do maior subconjunto de genes (G_{15}) em termos de éxons.*

4.3.2 Probabilidades a posteriori como medida de confiabilidade das predições

Utilizando o algoritmo apresentado na seção 4.1 é possível calcular a probabilidade *a posteriori* de cada posição de uma determinada sequência ter sido emitida por cada um dos estados de um modelo GHMM. Utilizando esta matriz de probabilidades, é possível saber o valor de probabilidade *a posteriori* de cada posição de uma dada predição feita para aquela sequência. Aliando a isso a anotação de referência desta sequência, é possível analisar os valores de probabilidade *a posteriori* das posições preditas corretamente, daquelas que a predição feita estava errada e também a probabilidade atribuida pelo modelo às anotações de referência.

Para este experimento, utilizamos o mesmo conjunto de teste apresentado na seção anterior. Cada conjunto de genes de camundongo era dividido em 5 partições. Quatro delas foram utilizadas para treinamento e para a quinta partição era calculada a predição feita pelo algoritmo de Viterbi. Foram também calculadas as probabilidades *a posteriori* de cada posição das sequências terem sido emitidas por cada um dos estados do modelo. Isso era feito utilizando cada uma das partições como teste e as outras 4 como conjunto de treinamento.

Utilizando o programa SGEval (Kashiwabara e Durham , 2011), foi possível comparar as predições feitas com suas anotações de referência. Este programa devolve além de valores de qualidade da predição, arquivos descrevendo as posições (início - fim) nas sequências em que são encontrados éxons corretamente preditos, éxons preditos incorretamente, e éxons que não foram preditos.

Utilizando essas informações, escrevemos um script na linguagem Perl que nos fornecia para cada posição predita como um éxon, ou que era de fato um éxon na referência (predito ou não), a probabilidade *a posteriori* de essa posição ser um éxon segundo o modelo utilizado na predição. Este script dividia esses valores de probabilidade em três conjuntos: os que foram preditos corretamente (verdadeiros positivos), os que foram preditos incorretamente (falsos positivos), e aqueles que não foram preditos (falsos negativos).

Com esses dados foi possível gerar histogramas que evidenciavam a distribuição dos valores de probabilidade para cada um desses conjuntos. Na figura 4.2 estão apresentados estes histogramas para o último conjunto de validação cruzada. Neste histograma estão incluídos os valores de probabilidades encontrados nos cinco conjuntos de teste deste experimento de validação cruzada.

A primeira coisa que devemos notar nesta figura é que há muito mais posições corretamente preditas do que incorretamente preditas, ou que deixaram de ser preditas. Isso é condizente com os bons resultados encontrados pela predição utilizando o algoritmo de Viterbi. Podemos observar que para os três conjuntos de posições há muito poucos valores de probabilidade intermediários, ou seja os picos estão principalmente nas regiões acima de 0,90 e abaixo de 0,05.

No caso do conjunto de posições corretamente preditas, aproximadamente 82% das predi-

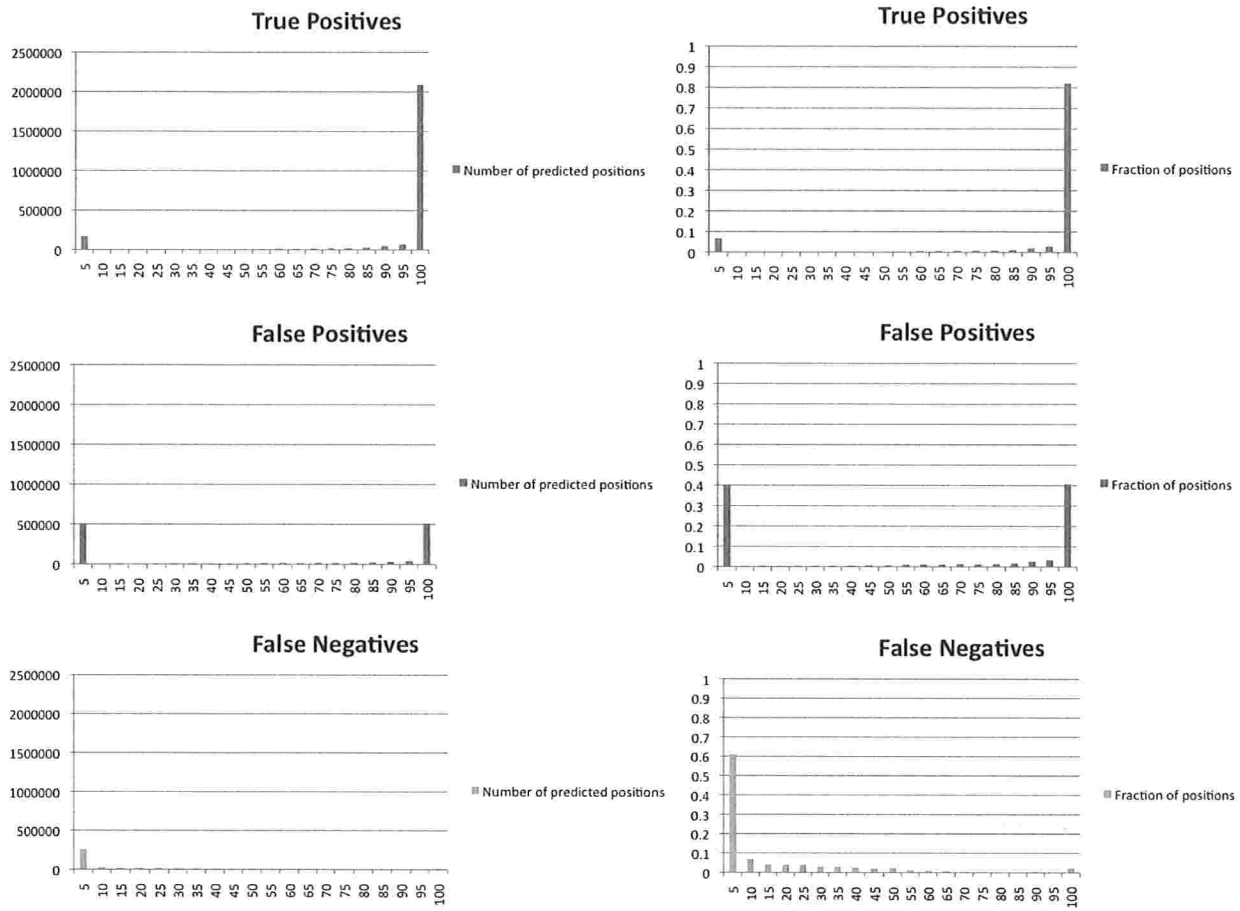

Figura 4.2: *Há três conjuntos de dados representados nesta figura. O primeiro é o conjunto de posições corretamente preditas como éxons (azul), o segundo é o conjunto de posições preditas incorretamente como éxons (vermelho), e o terceiro é o conjunto de posições que eram éxons, mas não foram preditas como tal (verde). Os gráficos na primeira linha representam o número absoluto de posições que o modelo encaixa em cada nível de probabilidade a posteriori. Os gráficos na segunda linha representam a fração de posições de cada conjunto, que se encaixam em cada nível de probabilidade de serem éxons segundo o modelo utilizado na predição.*

ções são feitas com probabilidade *a posteriori* entre 0,95 e 1. A porcentagem dessas predições que têm probabilidade abaixo de 0,05 é de apenas 6%. Já no caso dos falsos positivos, podemos observar que a porcentagem de predições feitas com valores de probabilidade entre 0,95 e 1 é de 40%, assim como a porcentagem de predições feitas com probabilidade abaixo de 0,05. No caso das posições que eram éxons, mas não foram preditas como tal, verificamos que a maioria delas têm probabilidade de ser éxon, segundo o modelo, abaixo de 0,05, e apenas uma quantidade ínfima tem probabilidade entre 0,95 e 1.

Baseados nos dados destes histogramas, fizemos uma análize dos valores de PPV (porcentagem das posições preditas como éxons que de fato eram éxons), sensibilidade (porcentagem das predições que eram éxons que foram preditas como éxons), e F-score (média harmônica do PPV e sensibilidade) das predições se utilizássemos todas as posições preditas, apenas aquelas que tivessem probabilidade acima de 0.05, ou apenas as com probabilidade acima de 0.90. Esses valores são baseados na medida de nucleotídeos preditos, já que toda a análise é feita sobre predições em posições individuais.

Na figura 4.3 podemos observar os resultados dessa análise. Podemos observar que aumentar o limiar de corte de probabilidade leva a um grande aumento no PPV (9.9% em média para o corte em 0.05, e 13.7% em média para o corte em 0,90). Isso indica que de fato podemos confiar mais nas predições feitas com probabilidade *a posteriori* mais próximas de 100%. Podemos observar que há também uma diminuição na sensibilidade da predição devido à utilização desses cortes, entretanto, como podemos observar pelas medidas de F-score, essa diminuição na sensibilidade não é tão intensa quanto o aumento no PPV. Pode-se notar ainda que as diferenças na sensibilidade, PPV, e F-score advindas da utilização desses cortes de probabilidade se mantém conforme aumentamos os conjuntos de teste e treinamento.

4.3.3 Discussão

Pudemos observar pelos resultados apresentados neste capítulo que o cálculo de probabilidades *a posteriori* em um GHMM pode ser bastante útil como uma medida de confiabilidade nas predições feitas para cada uma das posições de uma sequência.

Vimos que considerar corretas apenas aquelas predições feitas com mais de 0,05 de probabilidade segundo o modelo leva a um grande aumento (aproximadamente 10%) na chance de as posições que foram preditas como éxons realmente serem. No caso do corte em 0,05 esse aumento no PPV é aliado a uma diminuição não tão intensa na sensibilidade (5,3%), indicando que o preço em termos de verdadeiros positivos a ser pago para conseguir esse aumento no PPV é compensador, o que podemos enxergar pelo valor de F-score.

No caso da utilização do nível de corte em 0,90, o aumento do PPV é ainda maior em relação ao PPV da predição completa (13,7%). Entretanto, a diminuição da sensibilidade tem aproximadamente a mesma intensidade (12,7%). Esse fato faz com que o F-score seja praticamente o mesmo para a predição sem cortes, e para quando usamos o corte em 0,90.

É interessante notar que esses resultados envolvem descartar determinadas posições pre-

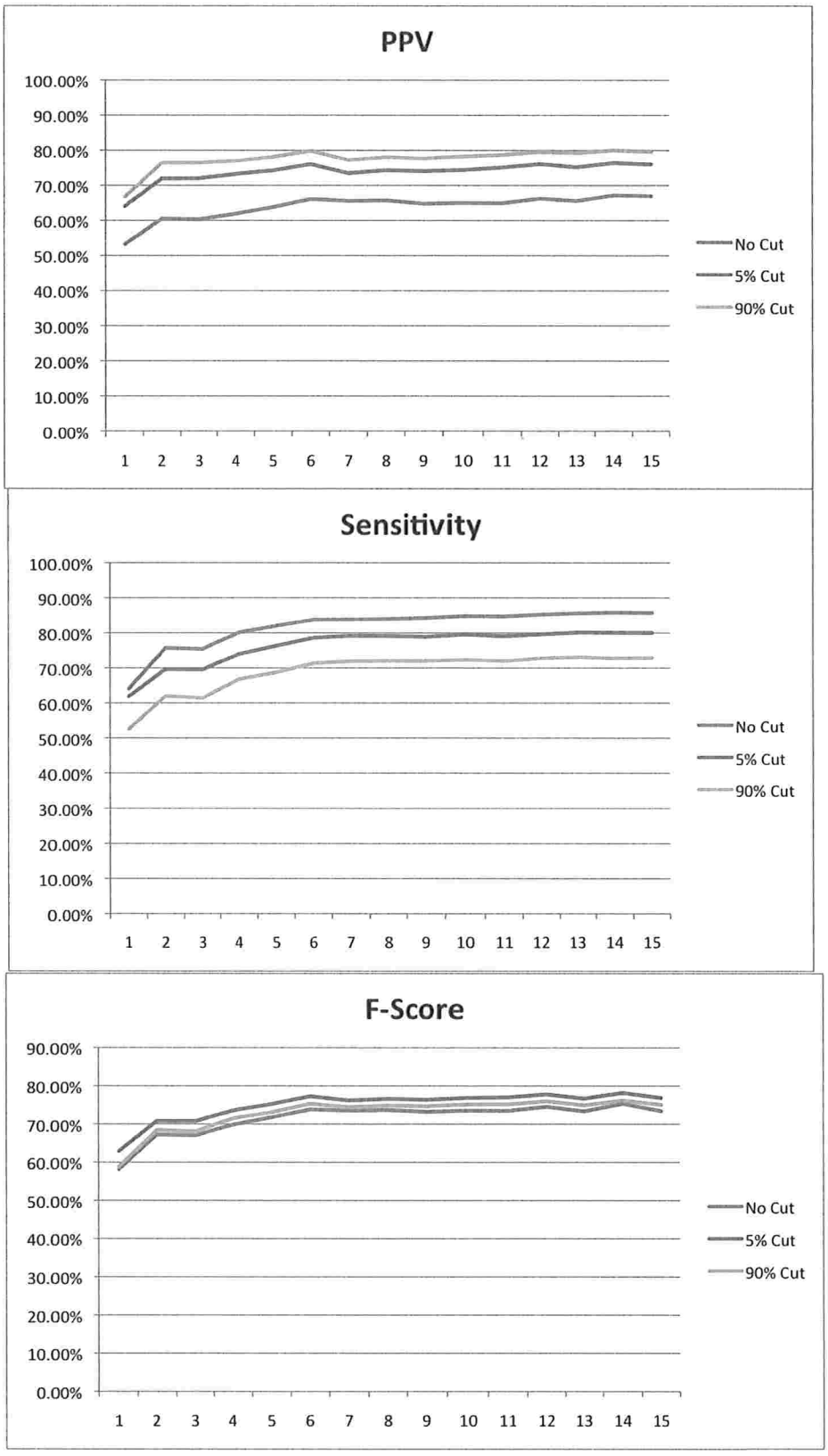

Figura 4.3: *Cada ponto nesses gráficos corresponde à média dos valores de PPV, sensibilidade e F-score para os 5 testes de cada uma dos conjuntos de validação cruzada utilizando genes de camundongo, apresentados na seção anterior. Apresentamos essas medidas utilizando três limiares de corte de probabilidades a posteriori, sem corte, maiores do que 5%, e maiores do que 90%.*

ditas (não vai haver predição alternativa para aquelas posições), o que não é o mesmo que fazer uma predição completa mais precisa. Entretanto, em muitos casos é mais importante saber que determinadas posições de uma predição estão corretas, do que saber a predição completa para um gene. Para esses casos a utilização dessas probabilidades *a posteriori* seria bastante adequada. Além disso, apesar de a utilização do corte em 0,90 levar a uma diminuição na sensibilidade, descartando posições que foram preditas corretamente, o grande aumento no PPV pode ser desejável em casos onde a precisão das predições é mais importante do que a sua completude.

A utilização das probabilidades *a posteriori* para construir a predição de genes consistente de máxima precisão esperada mostrou-se não tão preciso quanto o algoritmo de Viterbi. O principal problema com essa abordagem foi a quantidade de éxons incorretamente preditos. No caso de nucleotídeos as duas predições tiveram resultados muito próximos.

Já era esperado que o contexto de nucleotídeos seria onde o algoritmo MPE obteria sua melhor performance em relação ao algoritmo de Viterbi, já que este algoritmo visa maximizar a quantidade de posições corretamente preditas.

Capítulo 5

Combinando alinhamento e predição de genes

Encontrar o alinhamento e fazer a predição de genes para múltiplas sequências genômicas são problemas recorrentes na área de biologia molecular computacional. Apesar de na maioria das vezes serem tratados separadamente, esses problemas são intimamente relacionados.

Essa relação entre esses problemas vem do fato de a estrutura de um gene muitas vezes se manter bastante conservada ao longo da evolução (Alexanderson *et al.* , 2003). Isso implica que caracteres homólogos, em geral, fazem parte de um mesmo elemento de um gene (são todos éxons, por exemplo). Esse fato já foi utilizado com sucesso em algumas ferramentas de predição de genes (Alexanderson *et al.* , 2003).

Apresentamos neste trabalho algumas abordagens para os problemas de alinhamento múltiplo e predição de genes. Vimos que é possível, através de modelo de Markov, inferir valores de probabilidade de homologia entre pares de caracteres presentes no conjunto de sequências que desejamos alinhar, assim como é possível inferir a probabilidade de cada posição dessas sequências pertencer a cada uma das classes de estruturas de um gene. Além disso, através do algoritmo de predição de genes de máxima precisão esperada, descrito na seção 4.2, é possível construir predições de genes consistentes com a arquitetura do modelo a partir destas probabilidades.

Neste capítulo apresentaremos uma abordagem que visa fazer predições de genes para duas ou mais sequências homólogas, levando em conta a informação de homologia entre elas. Para isso, calcularemos as probabilidades *a posteriori* de alinhamento entre pares de caracteres, assim como as probabilidades *a posteriori* de cada posição dessas sequências pertencer a uma classe de estrutura gênica. Essas probabilidades serão combinas através de transformações de consistência e as predições para cada uma das sequências serão feitas utilizando essas probabilidades modificadas e o método de construção de máxima precisão esperada.

5.1 Tornando as predições consistentes

Nesta seção será apresentada uma transformação de consistência que tem como objetivo tornar as predições de genes feitas para caracteres homólogos de duas ou mais sequências consistentes umas com as outras. A idéia desta transformação é que se duas bases são homólogas, elas têm maior probabilidade de pertencer à mesma classe de estrutura gênica. Para determinar quais as bases homólogas utilizaremos as probabilidades de alinhamento entre pares de bases, calculadas através de um pairHMM da maneira descrita na seção 2.2.1.

Vamos assumir que S é o conjunto de sequências homólogas, para as quais queremos fazer a predição de genes. $P(c_i^{\mathbf{x}} = c)$ é a probabilidade *a posteriori* de a posição i da sequência $\mathbf{x}$ ser emitida pelo estado c, que pode ser vista como a probabilidade de x_i pertencer à classe de estruturas gênicas c. Assim como nos capítulos anteriores, $P(x_i \sim y_j)$ é a probabilidade *a posteriori* da posição i da sequência $\mathbf{x}$ estar alinhada à posição j da sequência $\mathbf{y}$. As probabilidades a posteriori, formalmente deveriam ser escritas como $P(c_i^x = c|\mathbf{x})$ e $P(x_i \sim y_j|\mathbf{x}, \mathbf{y})$, entretanto, por questão de clareza das fórmulas que serão apresentadas a seguir, estamos omitindo as sequências dadas da notação.

A transformação de consistência entre predições é, portanto, dada por:

$$P^*(c_i^{\mathbf{x}} = c) = \frac{\sum_{\mathbf{y} \in S}[P(x_i \sim y_-)P(c_i^{\mathbf{x}} = c) + \sum_{j=1}^{|\mathbf{y}|} P(x_i \sim y_j)P(c_j^{\mathbf{y}} = c)]}{|S|} \tag{5.1}$$

com $|S|$ sendo o número de sequências em S, $|\mathbf{y}|$ sendo o tamanho da sequência $\mathbf{y}$, $P(x_i \sim x_i) = 1$ e $P(x_i \sim x_j) = 0$ se $i \neq j$ ou se $j = -$.

Podemos observar que ao aplicar esta transformação, a probabilidade de uma determinada base pertencer a uma classe de estrutura de um gene aumenta quando as posições com as quais ela tem grande probabilidade de estar alinhada também têm grande probabilidade de pertencer a esta mesma classe. Por outro lado, se as bases inferidas como homólogas a ela não tiverem grande chance de serem desta classe, esta probabilidade será reduzida.

É interessante notar que quando uma base tem grande probabilidade de estar alinhada com um gap em outra sequência, ou seja, não ter posições homólogas a ela, optamos por manter a probabilidade de predição originalmente calculada para esta base. Isto é feito através da adição do termo $P(x_i \sim y_-)P(c_i^{\mathbf{x}} = c)$.

Assim como nas transfomações apresentadas na seção 2.2.3, é possível utilizar um peso diferente para cada uma das sequências na transformação de consistência. O peso apresentado naquela seção era a probabilidade média com que os pares de posições estavam alinhados no alinhamento de precisão máxima entre duas sequências, o que era representado por $P(\mathbf{x} <> \mathbf{y})$. Este peso faz sentido se considerarmos que quanto mais próximas evolutivamente são duas sequências, maior será este peso, e maior também é a chance de a estrutura do gene presente em cada uma delas ser a mesma. Assim, a transformação utilizando este peso fica:

$$P^*(c_i^{\mathbf{x}} = c) = \frac{\sum_{\mathbf{y} \in S}[P(\mathbf{x} <> \mathbf{y})(P(x_i \sim y_-)P(c_i^{\mathbf{x}} = c) + \sum_{j=1}^{|\mathbf{y}|} P(x_i \sim y_j)P(c_j^{\mathbf{y}} = c))]}{\sum_{\mathbf{y} \in S} P(\mathbf{x} <> \mathbf{y})} \tag{5.2}$$

assumindo que $P(\mathbf{x} <> \mathbf{x})$ é igual a 1.

Podemos ainda utilizar um peso relacionado à qualidade das predições para cada uma das sequências. Este peso é a média das probabilidades das predições feitas para cada posição de uma sequência na predição de máxima precisão esperada. Sendo $\mathbf{X} = c_1^{\mathbf{X}}...c_L^{\mathbf{X}}$ a predição consistente de máxima precisão esperada para uma sequência $\mathbf{x}$, definimos este peso como:

$$P(\mathbf{X}) = \frac{\sum_{i=1}^{L} P(c_i^{\mathbf{x}} = c_i^{\mathbf{X}})}{L} \tag{5.3}$$

É razoável assumir que quando uma sequência tem um valor muito baixo para $P(\mathbf{X})$, o nível de certeza com que a predição e o cálculo das probabilidades *a posteriori* é feito é menor. Assim, se temos algumas sequências no conjunto S para as quais temos um nível maior de certeza a respeito das predições feitas para elas, desejamos dar um peso maior para a informação vinda dessas predições. A transformação utilizando esse tipo de peso ficaria:

$$P^*(c_i^{\mathbf{x}} = c) = \frac{\sum_{\mathbf{y} \in S}[P(\mathbf{X})P(\mathbf{Y})(P(x_i \sim y_-)P(c_i^{\mathbf{x}} = c) + \sum_{j=1}^{|\mathbf{y}|} P(x_i \sim y_j)P(c_j^{\mathbf{y}} = c))]}{\sum_{\mathbf{y} \in S} P(\mathbf{X})P(\mathbf{Y})} \tag{5.4}$$

É claro que podemos ainda utilizar tanto os pesos relacionados à qualidade dos alinhamentos, quanto os pesos relacionados à qualidade de predição simultaneamente. Esta versão da transformação é dada por:

$$P^*(c_i^{\mathbf{x}} = c) = \frac{\sum_{\mathbf{y} \in S}[P(\mathbf{x} <> \mathbf{y})P(\mathbf{X})P(\mathbf{Y})(P(x_i \sim y_-)P(c_i^{\mathbf{x}} = c) + \sum_{j=1}^{|\mathbf{y}|} P(x_i \sim y_j)P(c_j^{\mathbf{y}} = c))]}{\sum_{\mathbf{y} \in S} P(\mathbf{x} <> \mathbf{y})P(\mathbf{X})P(\mathbf{Y})} \tag{5.5}$$

Cada uma dessas versões da transformação será testada neste projeto. Os testes e resultados serão descritos nas últimas duas sessões deste capítulo.

5.2 Tornando os alinhamentos consistentes com as predições

Apresentaremos nesta seção uma transformação de consistência que é aplicada sobre as probabilidades de alinhamento entre os pares de posições, com o intuito de torná-las consistentes com as predições de genes feitas para cada uma das sequências sendo alinhadas. Analogamente à transformação anterior, esta se baseia no fato de que se duas bases são

da mesma classe de estrutura gênica, elas devem ter uma probabilidade maior de serem homólogas do que aquelas que são de classes diferentes. As transformações apresentadas nesta seção podem ser utilizadas em conjunto com as da seção anterior, visando melhorar ainda mais a qualidade da predição.

Antes de apresentar a fórmula para essa transformação de consistência, é necessário definir a probabilidade de três posições serem da mesma classe. Vamos introduzir a notação $x_i \leftrightarrow y_j$ para dizer que as posições x_i e y_j pertencem à mesma classe de estrutura gênica. Para calcular a probabilidade deste evento, vamos assumir que se temos duas sequências diferentes $\mathbf{x}$ e $\mathbf{y}$, as probabilidade de uma posição de $\mathbf{x}$ e uma posição de $\mathbf{y}$ pertencerem a uma determinada classe de estrutura de um gene são independentes. Assim, esta probabilidade é dada por:

$$\begin{aligned} P(x_i \leftrightarrow y_j \leftrightarrow z_k) = & \\ &= \sum_{\forall c} P(c_i^{\mathbf{x}} = c, c_j^{\mathbf{y}} = c, c_k^{\mathbf{z}} = c) = \\ &= \begin{cases} \sum_{\forall c} P(c_i^{\mathbf{x}} = c) P(c_j^{\mathbf{y}} = c) P(c_k^{\mathbf{z}} = c) & \text{se } x_i \neq y_j \neq z_k \\ P(x_i \leftrightarrow y_j) = \sum_{\forall c} P(c_i^{\mathbf{x}} = c) P(c_j^{\mathbf{y}} = c) & \text{se } x_i \neq y_j = z_k \end{cases} \end{aligned} \tag{5.6}$$

Com isso, podemos escrever essa nova transformação de consistência como:

$$\begin{aligned} P^*(x_i \sim y_j) = & \frac{1}{|S|} \sum_{\mathbf{z} \in S} [P(x_i \sim z_-) P(y_j \sim z_-) P(x_i \sim y_j) P(x_i \leftrightarrow y_j) + \\ & + \sum_{k=1}^{|\mathbf{z}|} P(x_i \sim z_k) P(z_k \sim y_j) P(x_i \leftrightarrow z_k \leftrightarrow y_j)] \end{aligned} \tag{5.7}$$

Assim como na sessão anterior, podemos utilizar os pesos tanto de alinhamentos, quanto de predições, gerando outras três versões desta transformação de consistência. Os pesos que são utilizados nas versões com os dois pesos, com o peso apenas do alinhamento e apenas com o peso da predição seriam respectivamente: $P(\mathbf{x} <> \mathbf{z}) P(\mathbf{y} <> \mathbf{z}) P(\mathbf{X}) P(\mathbf{Y}) P(\mathbf{Z})$, $P(\mathbf{x} <> \mathbf{z}) P(\mathbf{y} <> \mathbf{z})$, e $P(\mathbf{X}) P(\mathbf{Y}) P(\mathbf{Z})$. Não vamos apresentar aqui a fórmula completa de todas essa versões, já que esses pesos são aplicados à esta transformação de consistência de maneira análoga aos da seção anterior.

5.3 Validação

Nesta seção será descrito o experimento que realizamos para verificar se a metodologia de predição de genes comparativa apresentada neste capítulo acarreta em melhorias em relação às predições não comparativas. Será feita ainda uma discussão dos resultados obtidos nesta avaliação.

5.3.1 Testes e resultados

O primeiro passo para que fosse possível testar nossa abordagem era selecionar um conjunto de genes homólogos de dois ou mais organismos. Devido à grande quantidade de genes anotados, e à qualidade dessas anotações, decidimos realizar os testes sobre genes de três organismos: humanos, camundongos, e ratos. Desses três organismos selecionamos genes com evidência de expressão e sem erro aparente de anotação. Genes com sítios de splicing não canônicos também foram removidos, já que o preditor utilizado não modela esses tipos de sítio.

Dos genes que foram considerados bem anotados, selecionamos apenas aqueles que tínham menos de 8000 nucleotídeos. Isso foi feito devido à complexidade do algoritmo de cálculo de probabilidades *a posteriori* de alinhamento entre os pares de base. Para um par de sequências este algoritmo é $O(N^2)$ onde N é o tamanho das sequências, tanto em termos de tempo, quanto de memória. Isso faz com que os requisitos de memória desta técnica cresçam quadraticamente conforme aumentamos os tamanhos das sequências. Existem heurísticas que permitem fazer esse cálculo de maneira mais eficiente (Paten *et al.* , 2009), entretanto decidimos primeiramente avaliar a validade da metodologia em sequências menores ao invés de partirmos diretamente para a implementação de tais heurísticas.

Com os conjuntos de genes bem anotados e menores que 8000 nucleotídeos para esses três organismos, o próximo passo foi selecionar aqueles que eram homólogos. Para isso, utilizamos o banco de dados de homologia HomoloGene (Geer *et al.* , 2010). Construímos um script na linguagem Perl, que com consultas sobre esse banco, extraía das listas de genes de cada organismo, apenas aqueles que tínham homólogos nos outros dois organismos. Assim, conseguimos um conjunto de 1728 genes de cada organismo, que tínham homólogos nos outros dois organismos, estavam bem anotados, e tinham tamanho menor que 8000 nucleotídeos.

Como mencionado anteriormente, implementamos em nossa ferramenta as várias versões das transformações de consistência apresentadas neste capítulo. Assim, foi possível testar cada uma dessas versões neste mesmo conjunto de dados. Além disso era possível fazer as predições de genes para este mesmo conjunto sem levar em conta as probabilidades de alinhamento, através do algoritmo de Viterbi, ou de MPE. Isso tudo pôde ser feito no mesmo arcabouço, e com os mesmos modelos tanto de alinhamento quanto de predição. Além das diferentes versões das transformações de consistência, testamos ainda diferentes números de iterações de transformações de consistência. Testamos também a possibilidade de aplicar a consistência de alinhamentos com as predições e as de predições com alinhamentos alternadamente.

É uma prática bastante comum aplicar modelos de predição de genes treinados com genes de um organismo para tentar encontrar genes em um outro organismo próximo (Kashiwabara e Durham , 2011). Isso ocorre, por exemplo, durante o processo de anotação do genoma de algum organismo novo, em que os seus genes ainda não foram anotados, sendo assim impossível gerar

um bom conjunto de treinamento. Para representar esse tipo de situação, decidimos utilizar um mesmo modelo de predição de genes para calcular as probabilidades *a posteriori* de predição para os genes dos três organismos em questão neste teste. Com isso desejamos verificar se a inserção de informação de homologia poderia melhorar a qualidade das predições nos organismos para os quais o modelo não foi treinado especificamente. Assim, nestes testes fizemos o treinamento do modelo GHMM apenas em genes de camundongo. O pairHMM utilizado neste teste foi o modelo Tamura-Nei (Tamura e Nei , 1993) com os parâmetros padrão utilizados na ferramenta PicXAA (Sahraeian e Yoon , 2010) para alinhar sequências nucleotídicas.

Para avaliar a qualidade das predições para todos esses genes utilizamos o programa SGEVal, mencionado no capítulo 4. Assim, foi possível chegar, para cada versão da ferramenta, a valores médios de PPV, sensibilidade, e F-score para todas as predições feitas para esse conjunto de genes. Pudemos observar que a versão de consistência que obteve melhor performance foi aquela com pesos tanto de alinhamento e de predição (equação 5.5, e sem a utilização de nenhuma consistência transformando as probabilidades de alinhamento. Como nenhuma outra versão dessas consistências atinge melhor performance que esta, apresentamos na Tabela 5.1 apenas os resultados obtidos por esta versão, que chamamos de Predalign, e pelas predições de Viterbi e MPE sem consistências.

Podemos observar pelos resultados da Tabela 5.1 que a utilização da transformações de consistência sugeridas neste capítulo não traz uma melhora significativa para as predições de genes. Na maioria dos casos, inclusive, há uma diminuição na qualidade das predições, principalmente quando estamos olhando no contexto de genes completos preditos.

Além disso, o fato de termos que gerar as probabilidades *a posteriori* de alinhamento entre os pares de base faz com que a eficiência computacional da ferramenta que integra os dois problemas seja muito pior, já que o algoritmo de cálculo dessas probabilidades tem complexidade quadrática no tamanho das sequências, enquanto os de predição de genes sem alinhamento têm complexidade linear. Essa perda de eficiência poderia valer a pena se houvesse uma melhora bastante significativa, e existe ainda a possibilidade de utilizar certas heurísticas para diminuir esta diferença de performance.

Uma das vantagens de utilizar a técnica apresentada neste capítulo é que assim é possível gerar também o alinhamento múltiplo entre as sequências sobre as quais estamos fazendo as predições. É possível ainda que exista uma melhoria na qualidade dos alinhamentos múltiplos devido à utilização das consistências com as predições de genes, entretanto esta alteração da performance não foi medida neste trabalho, estando ainda em aberto.

A principal razão para não termos feito esta análise foi o fato de não haver uma maneira de conhecer o alinhamento verdadeiro entre essas sequências genômicas. A maior parte das ferramentas que fazem alinhamento múltiplo de sequências genômicas têm suas performances avaliadas utilizando algum conjunto de testes simulado. Entretanto, no caso da combinação de alinhamento com predição de genes seria necessário um conjunto que simulasse a evolução de um gene levando em conta como se dá a evolução de cada um dos sinais (códon inicial,

Organismo	Algoritmo	Contexto	PPV	Sensibilidade	F-Score
Humano	Viterbi	Genes	50,82	46,78	48,72
		Éxons	74,07	65,57	69,56
		Nucleotídeos	87,10	83,92	85,48
	MPE	Genes	48,68	45,49	47,03
		Éxons	72,01	65,03	68,34
		Nucleotídeos	87,25	83,15	85,14
	Predalign	Genes	44,56	44,61	44,58
		Éxons	68,56	66,90	67,72
		Nucleotídeos	88,48	83,10	85,71
Rato	Viterbi	Genes	58,52	53,73	56,03
		Éxons	80,51	71,88	75,95
		Nucleotídeos	92,30	86,69	89,41
	MPE	Genes	56,10	51,52	53,71
		Éxons	79,19	70,61	74,65
		Nucleotídeos	92,22	85,47	88,71
	Predalign	Genes	52,40	50,99	51,68
		Éxons	76,43	72,25	74,28
		Nucleotídeos	93,17	85,25	89,03
Camundongo	Viterbi	Genes	57,26	52,14	54,58
		Éxons	79,75	69,52	74,28
		Nucleotídeos	90,53	83,23	86,73
	MPE	Genes	55,14	50,79	52,88
		Éxons	78,45	68,88	73,35
		Nucleotídeos	90,59	82,46	86,34
	Predalign	Genes	53,01	51,20	52,09
		Éxons	75,97	70,57	73,17
		Nucleotídeos	91,49	82,41	86,71

Tabela 5.1: *Resultados para o conjunto de genes de cada um dos organismos em termos de genes completos coretamente preditos, éxons corretamente preditos, e nucleotídeos codificadores corretamente preditos. Viterbi e MPE correspondem às predições de genes utilizando cada um desses algoritmos, e Predalign corresponde à versão em que as probabilidades de predição e alinhamento são combinadas utilizando a transformação de consistência descrita na equação 5.5.*

sítios de splicing, e códon final) utilizados em predição de genes.

Neste teste é interessante notar ainda que a diferença de performance entre as predições feitas com o algoritmo de Viterbi e de MPE estão de acordo com os resultados apresentados na seção 4.3.

5.3.2 Discussão

Como mencionado acima, os resultados obtidos através da utlização desta técnica de combinação de alinhamentos múltiplos e predição de genes não indicam que esta técnica traz grandes benefícios para a predição de genes. Entretanto, não podemos chegar a uma conclusão definitiva a respeito da utilidade desta técnica. Isso pois os testes que foram feitos neste trabalho utilizaram apenas genes menores do que 8000nt. Neste conjunto de genes, a dificuldade de predição destes é baixa, o que é evidenciado pelos valores de F-score muito maiores utilizando o algoritmo de Viterbi neste teste quando comparados com os resultados apresentados na figura 4.1, em que o conjunto de testes não era limitado a genes menores que 8000.

É possível que se aplicada a sequências com genes longos, para os quais os preditores de genes *ab initio* apresentam mais dificuldade de determinar a anotação, a utilização destas transformações de consistência trouxesse alguma vantagem. Até o momento, no entanto, esse tipo de teste não pode ser feito, já que o consumo de memória do algoritmo necessário para calcular probabilidades *a posteriori* de alinhamentos é demasiadamente alto (complexidade $O(N^2)$ em memória), se quisermos alinhar os maiores genes de organismos eucariotos.

Esse custo em termos de memória, assim como de velocidade do algoritmo poderia ser reduzido drasticamente através da utilização da heurística de alinhamentos ancorados (Paten *et al.* , 2009). Este seria um possível desenvolvimento futuro de nossa ferramenta.

Capítulo 6

Implementação

As idéias apresentadas neste projeto envolveram a utilização de uma grande diversidade de modelos probabilísticos. A tarefa de predição de genes, por exemplo, já envolve um número de modelos, que devem então ser combinados por um modelo unificador como o GHMM. Além disso, apresentamos abordagens que envolviam a combinação desses modelos de predição de genes com modelos de alinhamento de sequências. Assim, para que fosse possível utilizar todos esses modelos conjuntamente, todos eles foram implementados em um mesmo arcabouço.

Na área de predição de genes este tipo de arcabouço é bastante comum, já que uma grande variedade de modelos têm que ser combinados para que seja possível fazer uma predição de qualidade, entretanto, a maioria desses arcabouços são voltados somente para predição de genes, e não permitem a utilização de cada sub-modelo implementado para outras análises. Além disso, esses arcabouços em geral têm um conjunto de modelos fixos que serão combinados para gerar as predições de genes, não dando espaço para experimentação da combinação de outros tipos de modelos nas análises de sequências. Um outro problema com a utilização da maioria dos arcabouços de predição de genes é que o treinamento e especificação de arquiteturas diferentes de modelos probabilísticos são muitas vezes bastante trabalhosos, e algumas vezes impossíveis sem fazer alterações no código em si.

Um arcabouço chamado ToPS (Kashiwabara e Durham , 2011) soluciona a maioria desses problemas. Ele foi desenvolvido com a finalidade de fazer predições de genes, entretanto permite também fazer outras análises de sequências, já que é possível utilizar cada um dos modelos implementados por ele. Por ter como objetivo a predição de genes, tem diversos modelos implementados, e para qualquer um destes é possível especificar uma arquitetura e um conjunto de parâmetros através de um arquivo de configuração, de forma que não seja necessário nenhuma alteração no código. Modelos como GHMM, que utilizam uma combinação de outros modelos, podem também ser especificados em arquivos de configuração, e podem ter seus componentes alterados sem grande dificuldade. Isso torna a experimentação com diferentes tipos de modelos uma tarefa relativamente simples.

Por essas razões, decidimos que seria adequado utilizar este arcabouço para nossos ex-

perimentos. Entretanto, para que fosse possível testar todas as idéias apresentadas neste trabalho, tivemos que estender este arcabouço. As extensões que tiveram que ser feitas foram:

- Implementação de pairHMMs e todos os algoritmos relacionados a este tipo de modelo. Entre estes algoritmos estavam: alinhamento com algoritmo de Viterbi, alinhamento de máxima precisão esperada, *forward*, *backward*, cálculo de probabilidades *a posteriori* de alinhamento entre pares de posições e treinamento Baum-Welch.
- Implementação de algoritmos de cálculo de probabilidades *a posteriori* em GHMMs, e predição de genes de máxima precisão esperada.
- Implementação dos métodos de construção de alinhamentos múltiplos por *sequence annealing*.
- Implementação de todas as transformações de consistência apresentadas neste trabalho.
- Implementação de matrizes esparsas para tornar a aplicação de transformações de consistência mais eficiente.

O ToPS foi implementado em linguagem C++ que possibilita o uso em diferentes plataformas e também possibilita a criação dos chamados wrappers para acessar os algoritmos do ToPS em outras linguagens de programação. ToPS foi testado nos ambientes Linux, Mac OS X, e Windows.

6.1 Estrutura de classes

Como evidenciado na Figura 6.1, o núcleo do ToPS consiste numa hierarquia de três níveis. Na raiz, a classe abstrata *ProbabilisticModel* que fornece uma interface para possibilitar o uso individual dos modelos de forma intercambiável. No segundo nível, há quatro classes abstratas:

- *FactorableModel* representa modelos em que a verossimilhança de uma sequência é fatorável em um produto de termos, sendo um termo para cada posição da sequência.
- *InhomogeneousFactorableModel* representa modelos que são não homogenêos.
- *DecodableModel* representa modelos decodificadores.
- *PairDecodableModel* representa os modelos decodificadores que lidam com um par de sequências simultaneamente. Esta classe foi parte de extensão do arcabouço ToPS feita neste projeto.

O último nível da hierarquia contém classes concretas que representam os modelos em si. As classes concretas atualmente implementadas no ToPS são:

- *DiscreteIIDModel.*
- *InhomogeneousMarkovChain.*
- *VariableLengthMarkovChain.*
- *HiddenMarkovModel.*
- *GeneralizedHiddenMarkovModel.*
- *PairHiddenMarkovModel.* Esta foi parte da extensão do arcabouço feita neste projeto.

A implementação de outros modelos envolve estender esta hierarquia, dependendo da característica do modelo a ser implementado. Utilizando esta hierarquia é garantido que os modelos funcionarão com outras componentes do arcabouço.

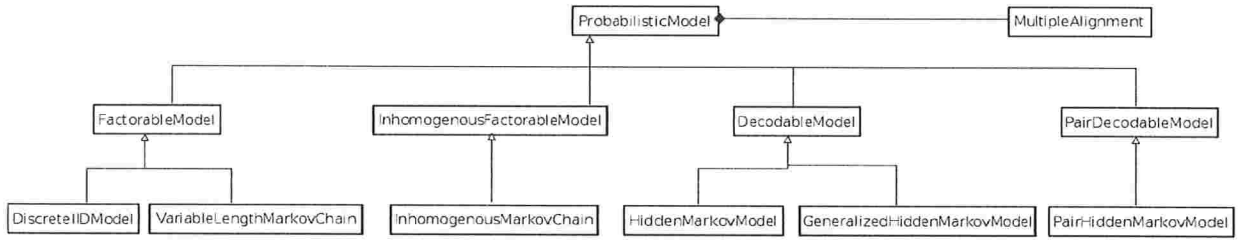

Figura 6.1: *Diagrama de classes do ToPS*

Uma instância de algum modelo pode ser obtida a partir de muitos algoritmos distintos. O padrão de projeto *Factory Method* fornece meios de implementar tal característica. Ele define uma interface de criação de objetos, permitindo que subclasses escolham qual classe instanciar (Kashiwabara e Durham , 2011).

Existe neste arcabouço arcabouço a classe abstrata *ProbabilisticModelCreator* fornece o método abstrato *create(parameters)*, que permite que o método de criação de cada um dos modelos seja chamado utilizando sempre esta mesma função.

Neste projeto foram criada ainda a classe *MultipleAlignment*, que envolve a criação de um alinhamento múltiplo entre sequências. É a classe onde são implementados os métodos de construção por sequence annealing, e também as transformações de consistência. Ela contém instâncias de modelos probabilísticos, que permitem calcular as probablidades *a posteriori* de alinhamento entre pares de posições, assim como probabilidades das posições terem sido emitidas por diferentes estados de um GHMM.

6.2 Descrevendo modelos no ToPS

Para qualquer uma das aplicações mencionadas acima é necesária a utilização de algum tipo de modelo probabilístico. Aqui mostraremos a maneira como são especificados estes modelos em arquivos de configuração. Utilizaremos como exemplo um arquivo de configuração para um pairHMM com dois estados de inserção, dois de deleção, um de pareamento, um de início e um de final.

pairHMM5S.txt

```
model_name="PairHiddenMarkovModel"
state_names = ("M", "I1", "D1", "I2", "D2", "B", "E")
observation_symbols = ("A","C","G","T")
transitions = ("M" | "B" :0.9615409374;
               "I1" | "B" : 4.537999985e-07;
               "D1" | "B" : 4.537999985e-07;
               "I2" | "B" : 0.01922916807;
               "D2" | "B" : 0.01922916807;
               "I1" | "M" : 0.01075110921;
               "D1" | "M" : 0.01075110921;
               "I2" | "M" : 0.008213998383;
               "D2" | "M" : 0.008213998383;
               "M" | "M" : 0.9619031182;
               "I1" | "I1" : 0.3209627509;
               "D1" | "D1" : 0.3209627509;
               "I2" | "I2" : 0.3297395944;
               "D2" | "D2" : 0.3297395944;
               "M" | "I1" : 0.6788705825;
               "M" | "D1" : 0.6788705825;
               "M" | "I2" : 0.670093739;
               "M" | "D2" : 0.670093739;
               "E" | "M" : 0.000166667;
               "E" | "I1" : 0.000166667;
               "E" | "D1" : 0.000166667;
               "E" | "I2" : 0.000166667;
               "E" | "D2" : 0.000166667;)
emission_probabilities = ("AA" | "M" : 0.1487240046;
                          "AT" | "M" : 0.0238473993;
                          "AC" | "M" : 0.0184142999;
                          "AG" | "M" : 0.0361397006;
                          "TA" | "M" : 0.0238473993;
                          "TT" | "M" : 0.1557479948;
                          "TC" | "M" : 0.0389291011;
                          "TG" | "M" : 0.0244289003;
                          "CA" | "M" : 0.0184142999;
                          "CT" | "M" : 0.0389291011;
                          "CC" | "M" : 0.1583919972;
                          "CG" | "M" : 0.0275536999;
                          "GA" | "M" : 0.0361397006;
                          "GT" | "M" : 0.0244289003;
                          "GC" | "M" : 0.0275536999;
                          "GG" | "M" : 0.1979320049;
                          "A-" | "I1" : 0.2270790040;
                          "T-" | "I1" : 0.2464679927;
                          "C-" | "I1" : 0.2422080040;
                          "G-" | "I1" : 0.2839320004;
                          "-A" | "D1" : 0.2270790040;
                          "-T" | "D1" : 0.2464679927;
                          "-C" | "D1" : 0.2422080040;
                          "-G" | "D1" : 0.2839320004;
                          "A-" | "I2" : 0.2270790040;
                          "T-" | "I2" : 0.2464679927;
                          "C-" | "I2" : 0.2422080040;
                          "G-" | "I2" : 0.2839320004;
                          "-A" | "D2" : 0.2270790040;
                          "-T" | "D2" : 0.2464679927;
                          "-C" | "D2" : 0.2422080040;
                          "-G" | "D2" : 0.2839320004;)
number_of_emissions = ("M" : "1,1";
                       "I1" : "1,0";
                       "D1" : "0,1";
                       "I2" : "1,0";
                       "D2" : "0,1";
                       "B" : "0,0";
                       "E" : "0,0";)
```

Podemos observar no arquivo de configuração deste pairHMM que são descritos os se-

guintes parâmetros deste modelo:

- **state_names** é lista de rótulos associados aos estados.
- **observation_symbols** é uma lista que descreve os símbolos observados.
- **transitions** especifica as probabilidades de transições entre os estados. As transições envolvendo os estados de início (B) e de final (E), representam as probabilidades de se começar e terminar a emissão em cada um dos estados.
- **emission_probabilities** especifica as probabilidades de emissões em cada um dos estados. Podemos observar que os estados de inserção (I1 e I2) emitem símbolos alinhados com um gap na segunda sequência, enquanto estados de deleção (D1 e D2) emitem símblos alinhados com um gap na primeira sequência. O estado de pareamento (M) emite símbolos alinhados, sendo um em cad uma das sequências.
- **number_of_emissions** especifica quantos caracteres são emitidos por cada estado em cada uma das sequências.

Com este arquivo de configuração, é possível alterar qualquer parâmetro do modelo sem que seja necessário alterar o código do ToPS. Pode-se não só alterar os valores das probabilidades de transição e emissão, mas também a estrutura de estados do modelo. Da mesma forma que este modelo especifica um pairHMM, é possível utilizar um arquivo de configuração similar a este para especificar qualquer outro modelo já implementado no arcabouço.

Capítulo 7

Conclusões e trabalhos futuros

Neste trabalho foram propostas uma série de novas abordagens para lidar com os problemas de alinhamentos de múltiplas sequências e predição de genes. Essas propostas envolveram modificações em técnicas populares de construção de alinhamento múltiplo, dois novos algoritmos para construir e avaliar a qualidade de predições de genes, assim como uma metodologia que visava combinar informações sobre alinhamento e predições de genes para gerar predições de maior qualidade.

Em relação ao problema de alinhamentos múltiplos propusemos várias possíveis alterações na formulação original da técnica de transformação de consistência. Essas modificações envolviam um tratamento mais adequado das probabilidades de alinhamento entre elementos das sequências sendo alinhadas e gaps em outras sequências. Vimos que a formulação com melhor performance foi aquela que não usava as probabilidades de alinhamento com gaps em posições específicas em outras sequências, mas sim tratava da possibilidade de uma base não estar alinhada com nenhuma outra em alguma das sequências. Confirmamos ainda que a utilização de pesos relacionados à similaridade entre as sequências de fato é benéfica.

Em decorrência desta melhora estatisticamente significativa nos alinhamentos presentes em *benchmarks* consagrados para avaliar ferramentas de alinhamento múltiplo, sugerimos que outras ferramentas que utilizam a técnica de transformação de consistência implementem esta nova formulação.

Ainda tratando de alinhamentos múltiplos, apresentamos uma modificação na técnica de construção por *sequence annealing* visando maximizar a função de precisão que penaliza o alinhamento entre posições que deveriam estar alinhadas com gaps. Esse tipo de erro em geral não é levado em consideração na avaliação de alinhamentos múltiplos, já que na maioria dos trabalhos fazem comparação entre ferramentas, as medidas de qualidade do alinhamento envolvem apenas o alinhamento correto entre os elementos das sequências. Consideramos que o alinhamento entre posições que deveriam estar alinhadas com gaps é de fato um erro que deve ser levado em consideração no alinhamento múltiplo, já que isso significaria inferir homologia entre duas posições que na realidade não têm homólogos na outra sequência.

Mostramos que quando utilizamos essa nova abordagem de *sequence annealing*, há de

fato uma melhora estatisticamente significativa na qualidade do alinhamento se utilizarmos um critério de avaliação que penaliza esse tipo de erro.

Nesta nova formulação do algoritmo de *sequence annealing* tratamos o alinhamento entre posições e gaps de maneira genérica, ou seja, quando adicionamos um par posição-gap no alinhamento, o gap não está em uma posição específica da sequência. A implicação deste fato é que a sequência onde aparece o gap poderá ter qualquer uma de suas bases inserida antes ou depois deste gap. Entretanto, é possível que se tratássemos do alinhamento entre uma posição de uma sequência e um gap entre duas posições específicas de uma outra sequência, isso pudesse ajudar a posicionar os próximos pares inseridos no alinhamento. Isso pois os próximos pares teriam que estar consistentes com o par posição-gap, mais provável segundo o modelo, inserido anteriormente. Assim, como um trabalho futuro sugerimos a avaliação desta possibilidade.

Tratando das inovações deste trabalho em relação à predição de genes, pudemos observar que o cálculo de probabilidades *a posteriori* em um GHMM pode ser bastante útil como uma medida de confiabilidade nas predições feitas para cada uma das posições de uma sequência. Observamos que a proporção de falsos positivos entre aquelas posições preditas como éxons que apresentam probabilidade *a posteriori* maiores que 0.9, é aproximadamente 13% menor que a proporção de posições falsamente preditas como éxons tratando a predição como um todo. Observamos ainda que se usarmos esse corte de probabilidade em 0.05, já haveria uma diminuição de 10% na proporção de falsos positivos.

Vimos ainda na análise dessas probabilidades *a posteriori* que a quantidade de posições corretamente preditas como éxons com probabilidade menor que 5% é bastante reduzida, justificando ainda mais a desconfiança nas predições feitas com probabilidade *a posteriori* próximas a zero.

Este é um resultado importante já que em muitos aplicações de predição de genes *ab-initio* saber que determinadas posições de uma predição estão corretas, é mais importante do que saber a predição para todas as posições de uma dada sequência. Para esses casos a utilização dessas probabilidades *a posteriori* seria bastante adequada. Além disso, apesar de a utilização do corte em 0,90 levar a uma diminuição na sensibilidade, descartando posições que foram preditas corretamente, o grande aumento no PPV pode ser desejável em casos onde a precisão das predições é mais importante do que a sua completude.

A utilização das probabilidades *a posteriori* para construir a predição de genes consistente de máxima precisão esperada mostrou-se não tão precisa quanto o algoritmo de Viterbi. O principal problema com essa abordagem foi a quantidade de éxons incorretamente preditos. No caso de nucleotídeos as predições MPE e Viterbi tiveram resultados muito próximos.

Já esperávamos que o contexto de nucleotídeos fosse aquele onde o algoritmo MPE obteria sua melhor performance em relação ao algoritmo de Viterbi, já que este MPE visa maximizar a quantidade de posições corretamente preditas. Entretanto, pudemos observar pelos histogramas da figura 4.2 que a distribuição de probabilidades *a posteriori* calculadas pelos modelos GHMM utilizados evidenciava que as posições que eram éxons e não foram

encontradas pelo algoritmo de Viterbi (falsos negativos), tínham probabilidade *a posteriori* muito baixas, com mais de 60% de tais posições tendo probabilidade menor do que 0.05. Observamos ainda uma grande quantidade de falsos positivos encontrados pelo algoritmo de Viterbi com probabilidade entre 0,95 e 1. Essa forma de distribuição pode ter levado à piora na performance deste algoritmo em relação ao algoritmo de Viterbi.

É claro que o fato de termos feito a avaliação das probabilidades *a posteriori* de predição de genes apenas para um organismo (camundongo) pode ter levado algum tipo de viés em nossas conclusões. Assim, como uma continuação deste trabalho seria importante fazer esse mesmo tipo de análise para uma quantidade maior de organismos, para verificar se os resultados se mantêm.

Os resultados obtidos através da utilização da técnica de combinação de alinhamentos múltiplos e predição de genes não indicam que esta metodologia traga grandes benefícios para a predição de genes. Entretanto, não pudemos chegar a uma conclusão definitiva a respeito da utilidade desta técnica. Isso pois os testes que foram feitos neste trabalho utilizaram apenas genes menores do que 8000nt. Neste conjunto de genes, a dificuldade de predição destes não é tão elevada.

É possível que se aplicada a sequências com genes longos, para os quais os preditores de genes *ab initio* apresentam mais dificuldade de determinar a anotação, a utilização destas transformações de consistência trouxesse alguma vantagem. Até o momento, no entanto, esse tipo de teste não pode ser feito, já que o consumo de memória do algoritmo necessário para calcular probabilidades *a posteriori* de alinhamentos é demasiadamente alto (complexidade $O(N^2)$ em memória), se quisermos alinhar os maiores genes de organismos eucariotos.

Esse custo em termos de memória, assim como de velocidade do algoritmo poderia ser reduzido drasticamente através da utilização da heurística de alinhamentos ancorados (Paten *et al.* , 2009). Este seria um passo importante para tornar a ferramenta ToPS competitiva com outras ferramentas de alinhamento múltiplo. Isso permitiria ainda avaliar de forma mais adequada a proposta de combinação de alinhamentos múltiplos e predição de genes.

Além disso, uma análise que não foi feita neste trabalho em relação a esta abordagem de combinação de predição de genes e alinhamentos, foi verificar se essa técnica tem consequências benéficas em relação à qualidade do alinhamento múltiplo gerado com as probabilidades *a posteriori* modificadas. Faria sentido que a informação a respeito da classe de estrutura gênica predita para cada posição das sequências sendo alinhadas fosse vantajosa para o alinhamento entre elas.

Este teste não foi realizado neste trabalho devido à dificuldade de avaliação da qualidade dos alinhamento entre sequências genômicas. Em especial, o fato de estarmos utilizando a estrutura predita dos genes presentes nas sequências, qualquer simulação de evolução utilizada para avaliação dos alinhamentos teria que modelar a evolução de cada elemento de um gene de maneira adequada. Analisamos algumas possibilidades de avaliação destes alinhamentos utilizando sequências reais. A utilização da ferramenta StatSigMA-w foi a que

pareceu mais promissora, assim esse seria um possível trabalho futuro.

Por fim, é importante destacar que os testes relacionados às inovações apresentadas neste trabalho foram feitos sobre uma mesma plataforma. Isso permitiu que fossem feitas análises apenas das técnicas em si, já que os resultados não são influenciados por heurísticas presentes em uma das ferramentas e não em outra, ou por diferenças na maneira como determinados algoritmos são implementados em duas ferramentas diferentes. Temos desta maneira comparações justas entre os algoritmos.

Referências Bibliográficas

Alexanderson *et al.* **(2003)** Marina Alexanderson, Simon Cawley e Lior Pachter. SLAM: Cross-Species Gene Finding and Alignment with a Generalized Pair Hidden Markov Model. *Genome Research*, 13(3):496–502. doi: 10.1101/gr.424203. Citado na pág. 47

Axelson-Fisk (2010) Marina Axelson-Fisk. *Comparative gene finding: models and algorithms and implementation.* Springer-Verlag London Limited. Citado na pág. 2

Blanchette *et al.* **(2004)** Mathieu Blanchette, W James Kent, Cathy Riemer, Laura Elnitski, Arian FA Smit, Krishna M Roskin, Robert Baertsch, Kate Rosenbloom, Hiram Clawson, Eric D Green, David Haussler e Webb Miller. Aligning multiple genomic sequences with the threaded blockset aligner. *Genome Research*, 14(4):708–715. doi: 10.1101/gr.1933104. Citado na pág. 1, 19

Bradley *et al.* **(2009)** Robert K Bradley, Adam Roberts, Michael Smoot, Sudeep Juvekar, Jaeyoung Do, Colin Dewey, Ian Holmes e Lior Pachter. Fast Statistical Alignment. *PLoS Computational Biology*, 5(5). doi: 10.1371/journal.pcbi.1000392. Citado na pág. 1, 2, 7, 14, 15, 23, 24, 27

Do *et al.* **(2005)** Chuong B Do, Mahathi S P Mahabhashyam, Michael Brudno e Serafim Batzoglou. PROBCONS: Probabilistic Consistency-based Multiple Sequence Alignment. *Genome Research*, 15(2):330–340. doi: 10.1101/gr.2821705. Citado na pág. 1, 7, 9, 13, 16, 17, 23

Durbin *et al.* **(1998)** Richard Durbin, Sean Eddy, Anders Krogh e Graeme Mitchison. *Biological sequence analysis: Probabilistic models of proteins and nucleic acids.* The press syndicate of the University of Cambridge. Citado na pág. 1, 7, 9, 10, 33, 39

Edgar (2004) Robert C. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797. doi: 10.1093/nar/gkh340. Citado na pág. 13

Geer *et al.* **(2010)** L. Y. Geer, A. Marchler-Bauer, R. C. Geer, L. Han, J. He, S. He, C. Liu, W. Shi e S. H. Bryant. The NCBI BioSystems database. *Nucleic Acids Research*, 38:492–496. Citado na pág. 51

Kashiwabara e Durham (2011) Andre Yoshiaki Kashiwabara e Alan Mitchell Durham. *MYOP/ToPS/SGEval: Um ambiente computacional para estudo sistematico de predicao de genes.* Tese de Doutorado, Instituto de Matematica e Estatistica - Universidade de Sao Paulo. Citado na pág. 3, 6, 36, 39, 40, 42, 51, 55, 57

Kemena e Notredame (2009) C. Kemena e C. Notredame. Upcoming challenges for multiple sequence alignment methods in the high-throughput era. *Bioinformatics*, 25(19): 2455–2465. doi: 10.1093/bioinformatics/btp452. Citado na pág. 1, 13, 14, 16, 19

Lee *et al.* (2002) C. Lee, C. Grasso e M. Sharlow. Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18(3):452–,Äì464. doi: 10.1093/bioinformatics/18.3.452. Citado na pág. 14

Margulies e et al. (2007) EH Margulies e et al. Analyses of deep mammalian sequence alignments and constraint predictions for 1% of the human genome. *Genome Research*, 21:760–774. doi: 10.1101/gr.6034307. Citado na pág. 19

Needleman e Wunsch (1970) S. B. Needleman e C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453. Citado na pág. 7

Notredame e et al. (2000) C. Notredame e et al. T-Coffee: a novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 302(1):205–217. Citado na pág. 1, 13

Paten *et al.* (2009) B. Paten, J. Herrero, K. Beal e E. Birney. Sequence progressive alignment and a framework for practical large-scale probabilistic consistency alignment. *Bioinformatics*, 25(3):295–301. doi: 10.1093/bioinformatics/btn630. Citado na pág. 1, 7, 51, 54, 63

Pearce e Kelly (2006) D. J. Pearce e P. H. J. Kelly. A dynamic topological sort algorithm for directed acyclic graphs. *Journal of Experimental Algorithmics*, 11(1.7). doi: 10.1145/1187436.1210590. Citado na pág. 15

Rabiner (1989) L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recoginition. Em *Proccedings of the IEEE*, volume 77, páginas 257–286. Citado na pág. 3

Sahraeian e Yoon (2010) S. M. Sahraeian e B. J. Yoon. PicXAA: greedy probabilistic construction of maximum expected accuracy alignment of multiple sequences. *Nucleic Acids Research*, 38(15):4917–4928. doi: 10.1093/nar/gkq255. Citado na pág. 1, 2, 7, 9, 14, 15, 16, 18, 23, 52

Stoye *et al.* (1998) J. Stoye, D. Evers e F. Meyer. Rose: generating sequence families. *Bioinformatics*, 14(2):157–163. Citado na pág. 21

Subramanian *et al.* (2008) A. R. Subramanian, M. Kaufmann e B. Morgenstern. DIALIGN-TX: greedy and progressive approaches for segment-based multiple sequence alignment. *Algorithms for Molecular Biology*, 3(6):1792–1797. doi: 10.1186/1748-7188-3-6. Citado na pág. 20, 28

Tamura e Nei (1993) K. Tamura e M. Nei. Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Molecular Biology and Evolution*, 10(3):512–526. Citado na pág. 52

Thompson *et al.* (1994) J. D. Thompson, D. G. Higgins e T. J. Gibson. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting and position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680. Citado na pág. 1, 13

Thompson *et al.* **(2005)** Julie Thompson, Patrice Koehl, Raymond Ripp e Olivier Poch. BAliBASE 3.0: Latest developments of the multiple sequence alignment benchmark. *Proteins: Structure, Function, and Bioinformatics*, 61(1):127–136. doi: 10.1002/prot.20527. Citado na pág. 19, 28

Varadarajan *et al.* **(2008)** A. Varadarajan, R. K. Bradley e I. H. Holmes. Tools for simulating evolution of aligned genomic regions with integrated parameter estimation. *Genome Biology*, 9(10). doi: 10.1186/gb-2008-9-10-r147. Citado na pág. 19

Wilcoxon (1945) Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83. Citado na pág. 21, 29

Wong *et al.* **(2008)** Karen M Wong, Marc A Suchard e John P Huelsenbeck. Alignment Uncertainty and Genomic Analysis. *Science*, 319(473):473–476. doi: 10.1126/science.1151532. Citado na pág. 1