Alinhamento múltiplo de genomas e sequências de proteínas com repetições e rearranjos

Laécio Freitas Chaves

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa: Mestrado em Ciência da Computacão

Orientador: Prof. Dr. Alan Mitchell Durham

São Paulo, Janeiro de 2016

Alinhamento múltiplo de genomas e sequências de proteínas com repetições e rearranjos

Esta é a versão original da dissertação/tese elaborada pelo
candidato Laécio Freitas Chaves, tal como
submetida à Comissão Julgadora.

Resumo

Alinhamento múltiplo de genomas e sequências de proteínas com repetições e rearranjos.

Neste trabalho, foram abordados o problema de alinhamento de sequências menores, correspondentes a genes ou proteínas individuais, como para sequências grandes do tamanho de cromossomos. Assim, uma proposta desse trabalho foi implementar um alinhador de sequências de granularidade variável, chamado de Multigran. Nesse alinhador, implementamos o algoritmo de alinhamento ancorado e baseando em segmentos, para possibilitar o alinhamento de sequências grandes. Apresentamos duas modificações nesse algoritmo: utilização do algoritmo de *sequence annealing* para manter a consistência entre os segmentos ou âncoras alinhadas e o algoritmo de refinamento dos segmentos para evitar que ocorra intersecção ou sobreposição entre os segmentos. Para o alinhamento de sequências com multidomínios, implementamos o algoritmo de alinhamento baseado em repetições e rearranjos. Esse algoritmo é utilizado para alinhar sequências de proteínas com domínios recombinantes, embaralhados ou repetidos. As modificações nesse algoritmo visam manter a consistência dos alinhamentos locais dos domínios e filtrar as partes conservadas do alinhamento. Em particular, implementamos uma versão modificada da técnica de refinamento iterativo discriminativo para melhorar a qualidade dos alinhamentos construídos pelo Multigran. Mostramos que as modificações citadas acima melhoram estatisticamente os resultados dos alinhamentos em relação as técnicas originais. Para evidenciar as melhorias, utilizamos *benchmarks* para avaliar as ferramentas de alinhamento múltiplo. Finalmente, por nossa ferramenta permitir o alinhamento de sequências com granularidade variável e os erros que podem ser gerados na construção dos alinhamentos, o Multigran foi integrado com uma ferramenta de edição, visualização e análise de alinhamentos de sequências conhecida como Jalview.

Palavras-chave: alinhamento múltiplo, multidomínios, âncoras, segmentos, refinamento iterativo.

Abstract

Multiple alignment of genomes and protein sequences with repetition and rearrangement.

This study has addressed the alignment problem smaller sequences corresponding to individual genes or proteins, such as for large size sequences of the chromosomes. Thus, a proposal of this study was to implement a variable granularity sequences aligner, called Multigran. In this aligner, implement the alignment algorithm based on segments and anchored to enable the alignment of large sequences. Here two changes in this algorithm: using the sequence annealing algorithm to maintain consistency between the line segments or anchors and refinement algorithm of segments to avoid the occurrence intersection or overlapping between segments. For sequence alignment with multi-domain implement alignment algorithm based on repetition and rearrangement. This algorithm is used to align sequences with recombinant protein, shuffled or repeated domains. The changes in this algorithm are intended to maintain the consistency of local alignments of fields and filter parts preserved alignment. In particular, we implemented a modified version of discriminative iterative refinement technique to improve the quality of alignments built by Multigran. We show that the changes mentioned above statistically improve the results of alignments regarding the original techniques. To demonstrate the improvements, we use benchmarks to evaluate the multiple alignment tools. Finally, our tool allows you to align sequences with variable granularity and errors that can be generated in the construction of alignments, the Multigran has been integrated with an editing tool, visualization and analysis of sequence alignments known as jalview.

Keywords: multiple alignment, multi-domain, anchors, segments, iterative refinement.

Sumário

Lista de Símbolos

$x_i \sim y_j$: posições alinhadas entre as sequências (posição i da sequência $\mathbf{x}$ alinhada com a 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 $\mathbf{y}$).
$\forall x_i \sim y_j \in A$: todos os pares de posições alinhadas que fazem parte do alinhamento A.
$\forall x_i \sim y_- \in A$: todos os pares de 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.
$P(\mathbf{x} <> \mathbf{y})$: probabilidade média dos pares alinhados no alinhamento ótimo entre duas sequências.

Lista de Figuras

Capítulo 1

Introdução

O alinhamento múltiplo de sequências (MSA, Multiple Sequence Alignment) é uma técnica essencial para a biologia molecular, a biologia computacional e a bioinformática. Alinhamentos múltiplos são usados na anotação genômica, na predição de estruturas de proteínas e na detecção de homologia entre novas sequências e famílias de sequências existentes (Russell, 2014). Essa técnica envolve o alinhamento de duas ou mais sequências de nucleotídeos ou aminoácidos, onde resíduos são pareados e os eventos mutacionais são inferidos, como as substituições e *indels* (exclusões e inserções) que ocorrem durante a evolução dos organismos.

A construção de um alinhamento com mais de duas sequências é um problema NP-completo (Just, 2001) (Wang e Jiang, 1994). Por essa razão, os alinhamentos múltiplos de sequências são em geral construídos através de algum tipo de heurística. O algoritmo progressivo é a heurística mais utilizada para alinhar sequências, onde pares e grupos de sequências são sucessivamente alinhados para construir o MSA. Outras heurísticas utilizam pares de segmentos conservados (Rausch *et al.* , 2008) (Subramanian *et al.*, 2005) ou pares de resíduos (Bradley *et al.*, 2009) (Sahraeian e Yoon, 2010) das sequências que se deseja alinhar, e assim constroem o alinhamento através de similaridades locais.

Com a grande quantidade de dados genômicos e proteômicos que precisam ser analisados, processados e avaliados, além da necessidade de se extrair informações desses dados, o interesse por técnicas de alinhamento múltiplo tem aumentado (Ortuño *et al.*, 2013). Além disso, alinhamento múltiplo de sequências tem sido utilizado na comparação de estruturas de proteínas (Gelly *et al.*, 2011) e na reconstrução de árvores filogenéticas (Wang *et al.*, 2011).

Os métodos tradicionais de alinhamento múltiplo têm complexidade de tempo proporcional ao produto do comprimento das sequências para o alinhamento de pares de sequências, o que é muito lento para o alinhamento de sequências genômicas grandes e o consumo de memória é demasiadamente alto (Russell, 2014). O alinhamento ancorado é uma técnica para alinhar sequências grandes e reduzir a complexidade de tempo e memória, onde são criados segmentos alinhados entre pares ou múltiplas sequências chamados de âncoras, e as regiões entre essas âncoras são alinhadas de maneira independente. Entre as ferramentas que implementam o alinhamento ancorado para alinhar sequências genômicas grandes, estão: FSA (Bradley *et al.*, 2009), MAVID (Bray e Pachter, 2004), Pecan (Paten *et al.*, 2009) e DIALIGN (Morgenstern *et al.*, 2006).

O alinhamento baseado em segmentos permite segmentos alinhados (âncoras) com substituições e *indels*. Assim, um segmento alinhado pode ser encontrado utilizando-se um algoritmo para alinhamento local entre pares de sequências. Entre as ferramentas que se beneficiam dessa heurística para alinhar sequências com blocos conservados, estão: T-Coffee (Rausch *et al.*, 2008) e DIALIGN (Morgenstern *et al.*, 1998)

Grande parte das heurísticas de alinhamento múltiplo inferem eventos mutacionais que podem ocorrer durante a evolução das sequências, e assumem implicitamente que as sequências são globalmente alinháveis (Phuong *et al.*, 2006). No entanto, famílias de proteínas com arquitetura de multidomínios podem ter domínios recombinantes, embaralhados (domínios em ordem diferente em diferentes proteínas) ou domínios repetidos (Raphael *et al.*, 2004). Entre as ferramentas de MSA

que abordam o problema de alinhar sequências de proteínas com uma arquitetura de multidomínios, estão: ABA (Raphael *et al.*, 2004) e ProDA (Phuong *et al.*, 2006).

Nos últimos anos, o grupo de pesquisa do Prof. Alan Durham, meu orientador, vem desenvolvendo um arcabouço computacional para a implementação de modelos probabilísticos markovianos visando análise e classificação de sequências (Kashiwabara *et al.*, 2013). Esse arcabouço, chamado de ToPS (do inglês *Toolkit of Probabilistc Model of Sequence*), foi estendido pelo mestrando Vitor Onuchic para inclusão de modelos que permitem a construção de alinhamentos (Onuchic, 2012). O ToPS utiliza diversas técnicas da literatura para alinhar sequências, tais como uma variação dos modelos ocultos de Markov (HMM) conhecida como PairHMM (Durbin *et al.*, 1998), alinhamento de máxima precisão esperada, transformações de consistência probabilística e implementa o algoritmo de *sequence annealing*. Estes modelos permitem o desenvolvimento de ferramentas de alinhamento de proteínas ou genes individuais. O problema de alinhamento de sequências grandes, do tamanho de cromossomos em eucariotos não é coberto por estas técnicas.

Neste trabalho apresentamos um alinhador de sequências, chamado de Multigran, que pode ser aplicado tanto a sequências menores, correspondentes a genes ou proteínas individuais, como a sequências grandes, do tamanho de cromossomos. Multigran foi integrado com o ToPS, assim os modelos probabilísticos do arcabouço ficam disponíveis para serem utilizados nos algoritmos do alinhador. Além disso, esta ferramenta apresenta resultados competitivos em relação a outras ferramentas de MSA e resolve vários problemas típicos de alinhamento de sequências.

Para possibilitar que nossa ferramenta possa alinhar sequências grandes, implementaremos no Multigran o algoritmo de alinhamento baseado em segmentos. O algoritmo é divido em 6 passos e pode ser utilizado para alinhar sequências genômicas utilizando âncoras ou alinhar sequências de nucleotídeos e proteínas utilizando o alinhamento local entre pares de sequências como segmentos alinhados. A diminuição da complexidade de tempo fornecida pelo alinhamento ancorado permite que nossa ferramenta trabalhe com grandes sequências de nucleotídeos e sequências genômicas, e tenha resultados competitivos em relação a outras ferramentas de alinhamento múltiplo.

Para sequências de proteínas com multidomínios implementamos o algoritmo baseado em repetições (repetição de um domínio em uma proteína) e rearranjos (mudança na ordem de dois domínios em uma única sequência). O algoritmo é divido em 6 passos baseados no algoritmo do alinhador ProDA. Em particular, implementamos a técnica de refinamento iterativo para melhorar a qualidade dos alinhamentos construídos pelo Multigran.

Além dos algoritmos implementados, nossa ferramenta é beneficiada pela implementação de duas versões do algoritmo de *sequence annealing* e 6 variações de algoritmos de consistência de alinhamentos desenvolvidas no trabalho de Vitor Onuchic (Onuchic, 2012), fornecendo várias opções para melhorar a qualidade dos alinhamentos. O usuário pode alinhar sequências combinando a versões do algoritmo de *sequence annealing* com as versões das transformações de consistência. Além disso, poderá utilizar os novos algoritmos implementados. Devido a quantidade de opções para alinhar sequências e aos erros que podem ser gerados pelos mais acurados algoritmos de alinhamento (Blackshields *et al.*, 2006), integramos o Multigran com uma ferramenta de edição, visualização e análise de alinhamentos de sequências conhecida como Jalview (Waterhouse *et al.* , 2009).

Finalmente, comparamos as implementações entre diferentes ferramentas de MSA. Assim, avaliamos em um mesmo contexto computacional, diferentes heurísticas implementadas por outras ferramentas, como alinhamento progressivo e *sequence annealing*, que utilizam ou não técnicas para melhorar a qualidade dos seus alinhamentos, como refinamento iterativo e as transformações de consistência probabilísticas.

Capítulo 2

Algoritmos de alinhamento

Nesta sessão será feita uma revisão de alguns métodos para alinhar pares e múltiplas sequências. Além disso, discorreremos sobre as propostas feitas neste projeto, e sobre os conjuntos de teste para avaliar a qualidade dos alinhamentos construídos por ferramentas de MSA.

2.1 Alinhamento de pares de sequências

As sequências divergem umas das outras durante o curso da evolução dos organismos, no qual ocorrem substituições, remoções e inserções de nucleotídeos no genoma. O alinhamento de sequências pode ser utilizado para inferir o processo mutacional que ocorre entre as sequências. Para construir e avaliar o resultado de um alinhamento, é necessário definir um sistema de pontuação. Assim, uma maneira de pontuar um alinhamento é usar uma matriz de substituição, como a BLOSUM50 (Durbin *et al.*, 1998). Para as proteínas, a BLOSUM50 forma uma matriz 20 x 20, com $s(x_i, y_j)$ sendo a pontuação na posição x_i e y_j da matriz, onde x_i é o i_{th} aminoácido da sequência $\mathbf{x}$ e y_j é o j_{th} aminoácido da sequência $\mathbf{y}$.

Dado um sistema de pontuação, é preciso ter um algoritmo para encontrar um alinhamento ótimo para um par de sequências. Um dos algoritmos que utiliza a técnica de programação dinâmica para resolver esse problema é conhecido como algoritmo de Needleman e Wunsch (1970). Neste algoritmo a pontuação para o melhor alinhamento das sequências $\mathbf{x} = x_1, x_2, ..., x_n$ e $\mathbf{y} = y_1, y_2, ..., y_m$, é dada por $F_{x,y}(n, m)$, que pode ser calculada pela recursão abaixo:

$$F_{x,y}(i,j) = \max \begin{cases} F_{x,y}(i-1, j-1) + s(x_i, y_j), \\ F_{x,y}(i-1, j) - d, \\ F_{x,y}(i, j-1) - d. \end{cases}$$

onde $s(x_i, y_j)$ são valores encontrados na matriz de substituição e d é uma penalidade para os *gaps* (correspondentes a inserções e remoções).

Pair Hidden Markov Models (PairHMM) fornecem uma abordagem probabilística para o problema de alinhar sequências (Durbin *et al.*, 1998). Estes modelos trabalham com duas fitas de leitura/emissão e a geração do alinhamento está modelada como um processo de Markov de primeira ordem associado a emissões de resíduos e pares de resíduos associadas a cada estado. A Figura 2.1 mostra um modelo de PairHMM de três estados, sendo M um estado que emite dois símbolos (um para cada cadeia) e que corresponde a um pareamento de nucleotídeos (*match*), e os estados I_x e I_y emitem um símbolo (em uma das cadeias) e um *gap* que corresponde as inserções ou remoções. As probabilidades são atribuídas para as emissões de símbolos e para transições entre os estados.

Neste modelo, duas sequências alinhadas $\mathbf{x} = x_1x_2...x_L$ e $\mathbf{y} = y_1y_2...y_L$ são geradas. O estado M emite o par alinhado x_i:y_j com probabilidade $P_{x_iy_j}$, o estado I_x emite o par x_i:*gap*, sendo x_i da sequência $\mathbf{x}$ e o *gap* da sequência $\mathbf{y}$ com probabilidade q_{x_i} e o estado I_y emite o par y_j:*gap* com o símbolo y_j sendo emitido da sequência $\mathbf{y}$ e o *gap* da sequência $\mathbf{x}$, com probabilidade q_{y_j} (Durbin *et al.*, 1998). As transições pelos estados desse modelo serão, portanto, alinhamentos entre

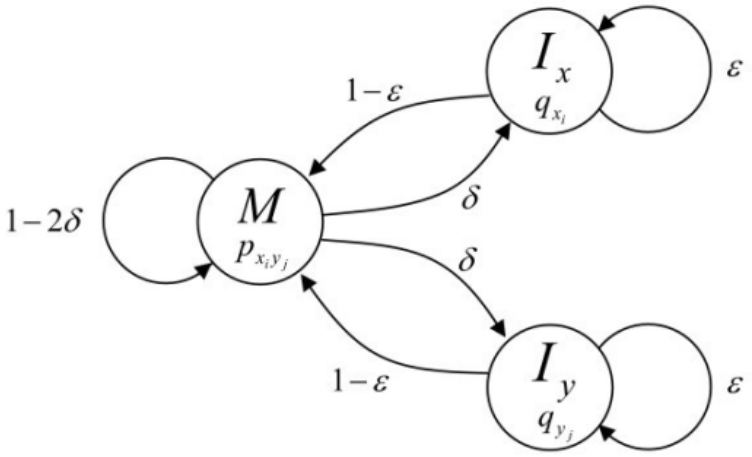

Figura 2.1: *Exemplo de modelo PairHMM. Figura extraída de Durbin et al. (1998).*

um par de sequências e terão valores de probabilidade associados a eles.

O modelo de PairHMM calcula as probabilidades dependendo dos valores dos parâmetros no modelo (transição e emissão). Desta maneira, precisamos ajustar os parâmetros (otimizar as probabilidades de transição entre os estados e emissão), de modo que o modelo represente a variação observada em um grupo de sequências de proteínas ou de DNA (Durbin *et al.*, 1998).

É possível encontrar o alinhamento mais provável entre duas sequências utilizando o modelo de PairHMM através do algoritmo de programação dinâmica chamado de Viterbi (Durbin *et al.* , 1998). Outra forma de alinhar duas sequências utilizando esse modelo é encontrar o alinhamento que maximize o número esperado de bases alinhadas corretamente ou o alinhamento de máxima precisão esperada (Durbin *et al.*, 1998) como definido na equação abaixo:

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

onde A é o alinhamento entre as sequências $\mathbf{x}$ e $\mathbf{y}$.

Para encontrar o alinhamento que maximiza o número esperado de bases alinhadas corretamente entre duas sequências $\mathbf{x}$ e $\mathbf{y}$, é necessário estimar as probabilidades $P(x_i \sim y_j | \mathbf{x}, \mathbf{y})$ de cada par de bases x_i e y_j estar corretamente alinhada. O algoritmo forward-backward pode ser utilizado para calcular essas probabilidades (Durbin *et al.*, 1998). Podemos agora encontrar o alinhamento de máxima precisão esperada para o alinhamento de um par de sequências através do algoritmo de Needleman e Wunsch (1970), mas que usa a probabilidade $P(x_i \sim y_j | \mathbf{x}, \mathbf{y})$ como o seu sistema de pontuação, como mostra a equação de recursão abaixo:

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

Outra abordagem para alinhar um par de sequências consiste em encontrar similaridades locais, como domínios repetidos em sequências de proteínas, sem a necessidade de alinhar toda a extensão das sequências. Smith e Waterman (1981) definiram um algoritmo que encontra o alinhamento com maior pontuação entre subsequências das duas sequências que desejamos alinhar. A diferença desse algoritmo em relação ao algoritmo de Needleman e Wunsch está no preenchimento da matriz, como mostra a recorrência baixo:

$$F(i,j) = \max \begin{cases} 0, \\ F(i-1, j-1) + s(x_i, y_j), \\ F(i-1, j) - d, \\ F(i, j-1) - d. \end{cases}$$

Assim, a recuperação do alinhamento local pode ser feita traçando o caminho a partir da célula com a maior pontuação até a célula com valor zero na matriz preenchida pela recursão acima.

Waterman e Eggert (1987) descreveram um algoritmo para calcular um alinhamento subótimo local entre duas sequências que não compartilha pares de resíduos com alinhamentos encontrados previamente. Em cada iteração desse algoritmo os seguintes passos são executados:

1. Um alinhamento subótimo local distinto é encontrado (não compartilha pares de resíduos com outros alinhamentos).
2. As células que representam o caminho do alinhamento anterior (1) na matriz de pontuação são atualizadas para zero.
3. A matriz de pontuação é recalculada.

A grande vantagem do algoritmo de Waterman e Eggert é que não existe a necessidade de recalcular toda a matriz de pontuação em cada iteração, pois somente as células do alinhamento subótimo encontrado na iteração anterior são atualizadas para zero. Isso evita que o pares de resíduos de um alinhamento contribuam para o próximo alinhamento encontrado.

2.2 Alinhamento progressivo

O alinhamento progressivo foi um dos primeiros algoritmos utilizados para construir alinhamentos múltiplos (Hogeweg e Hesper, 1984). Esse algoritmo consiste em alinhar as sequências progressivamente segundo um critério de proximidade que determina uma matriz de distâncias. Esta matriz de distâncias é utilizada para o estabelecimento de uma árvore guia. Esta árvore é utilizada para determinar a ordem em que sequências (ou alinhamentos de sequências) são agrupadas. Vários algoritmos para alinhamento progressivo têm sido propostos e as variações entre essas propostas ocorrem em relação às seguintes etapas:

1. Alinhamento de cada par de sequências.
2. Cálculo da matriz de distâncias entre as sequências.
3. Construção da árvore guia utilizando algum algoritmo de clusterização.
4. Construção do alinhamento múltiplo.

Cada uma das etapas acima podem ser implementadas com diferentes algoritmos. O alinhamento entre pares de sequências, por exemplo, pode ser encontrado utilizando o algoritmo de Needleman-Wunsch (Needleman e Wunsch, 1970). Assim, as pontuações dos alinhamentos entre pares de sequências são convertidas para uma medida de distância e os valores dessa medida são utilizados para preencher a matriz de distâncias.

A árvore guia é construída através de algum algoritmo de clusterização que é aplicado na matriz de distâncias, para assim gerar uma árvore binária enraizada. Os algoritmos UPGMA (Mount, 2004) e *Neighbor-Joining* (Saitou e Nei, 1987) são os mais utilizados para construir a árvore. As folhas da árvore guia representam cada uma das sequências sendo alinhadas e o nós internos representam alinhamentos entre pares de sequências (ou alinhamentos) dos seus nós filhos.

Uma vez que a árvore guia foi construída, o alinhamento múltiplo de sequências é iniciado a partir do alinhamento entre pares de sequências (folhas da árvore), e segue alinhando sucessivamente as sequências até a raiz da árvore (sequências com menor distância até as mais distantes).

Os algoritmos de cada etapa do alinhamento progressivo são, em geral, simples de implementar e têm bom desempenho em relação ao alinhamento de até 10.000 sequências (Sievers *et al.*, 2011). No entanto, um dos problemas do alinhamento progressivo está no fato de se construir o alinhamento com base no alinhamento entre pares e grupos de sequências e não levar em conta todas as informações comuns do conjunto de sequências que desejamos alinhar. Na sessão 2.6, descrevemos outros problemas e duas técnicas para minimizar erros que podem ocorrer durante qualquer etapa do alinhamento progressivo e que são propagados para o alinhamento final.

Entre as ferramentas que constroem o alinhamento múltiplo utilizando o alinhamento progressivo estão: ProbCons (Do *et al.*, 2005), Probalign (Roshan e Livesay, 2006), MAFFT (Katoh *et al.*, 2005), GLProbs (Ye *et al.*, 2013), MAVID (Bray e Pachter, 2004), Pecan (Paten *et al.*, 2009), Clustal Omega (Sievers *et al.*, 2011), entre outras.

2.3 Sequence annealing

O algoritmo de *sequence annealing*, em cada etapa da construção do alinhamento, se compromete em alinhar somente um par de resíduos, ao invés de levar em conta pares de sequências (ou alinhamentos). Assim, a chance de erros serem propagados para o resultado final do alinhamento é reduzida em relação ao alinhamento progressivo.

Em qualquer etapa do alinhamento, o par de resíduos selecionado para ser adicionado no alinhamento é aquele com o maior peso. Esse peso pode ser definido como a probabilidade *a posteriori* de um par de resíduos estar alinhado corretamente no alinhamento ótimo entre duas sequências. Essas probabilidades podem ser calculadas a partir de diferentes modelos probabilísticos, como: PairHMM (Do *et al.*, 2005) e função de partição (Roshan e Livesay, 2006).

Para evitar conflitos entre o próximo par a ser alinhado e os pares de resíduos previamente alinhados é utilizado um grafo acíclico direcionado para representar o alinhamento, onde cada vértice corresponde a uma coluna do alinhamento e as arestas representam a ordem das colunas do alinhamento final (Figura 2.2). Um conflito significa, neste contexto, que a inserção de um par de resíduos leva ao aparecimento de um ciclo no grafo como mostra a Figura 2.2F. Existem três maneiras de adicionar um par de resíduos no grafo, são elas: adicionar uma nova coluna (Figura 2.2C), estender uma coluna que já existe (Figura 2.2G) e mesclar duas colunas (Figura 2.2I).

O grafo representa os *gaps* do alinhamento de forma implícita e não impõe a ordem topológica dos vértices. Isso implica em um relacionamento de um-para-muitos entre o grafo e a matriz do alinhamento (Figura 2.2O e Figura 2.2P). O mapeamento do grafo para a matriz do alinhamento múltiplo pode ser feito de maneira linear no número de vértices e arestas do grafo (Bradley *et al.*, 2009).

Por iniciar o alinhamento pelas regiões mais confiáveis (maior probabilidade) e ir gradualmente progredindo para regiões menos confiáveis (menor probabilidade), esse algoritmo tem bons resultados quando o conjunto de sequências que desejamos alinhar tem alta similaridade local. Ferramentas que implementam esse algoritmo são: AMAP (S. Schwartz e Pachter, 2007), FSA (Bradley *et al.*, 2009), PicXAA (Sahraeian e Yoon, 2010), ToPS (Kashiwabara *et al.*, 2013).

2.4 Alinhamento ancorado e baseado em segmentos

Os métodos tradicionais de alinhamento múltiplo têm complexidade de tempo $O(L^2)$ para o alinhamento de pares de sequências (onde L é o tamanho das sequências), o que é muito lento para o alinhamento de sequências genômicas grandes (Russell, 2014). Uma técnica utilizada para reduzir essa complexidade é o alinhamento ancorado, onde são criados segmentos alinhados entre pares ou múltiplas sequências chamados de âncoras. O alinhamento ancorado consiste em encontrar

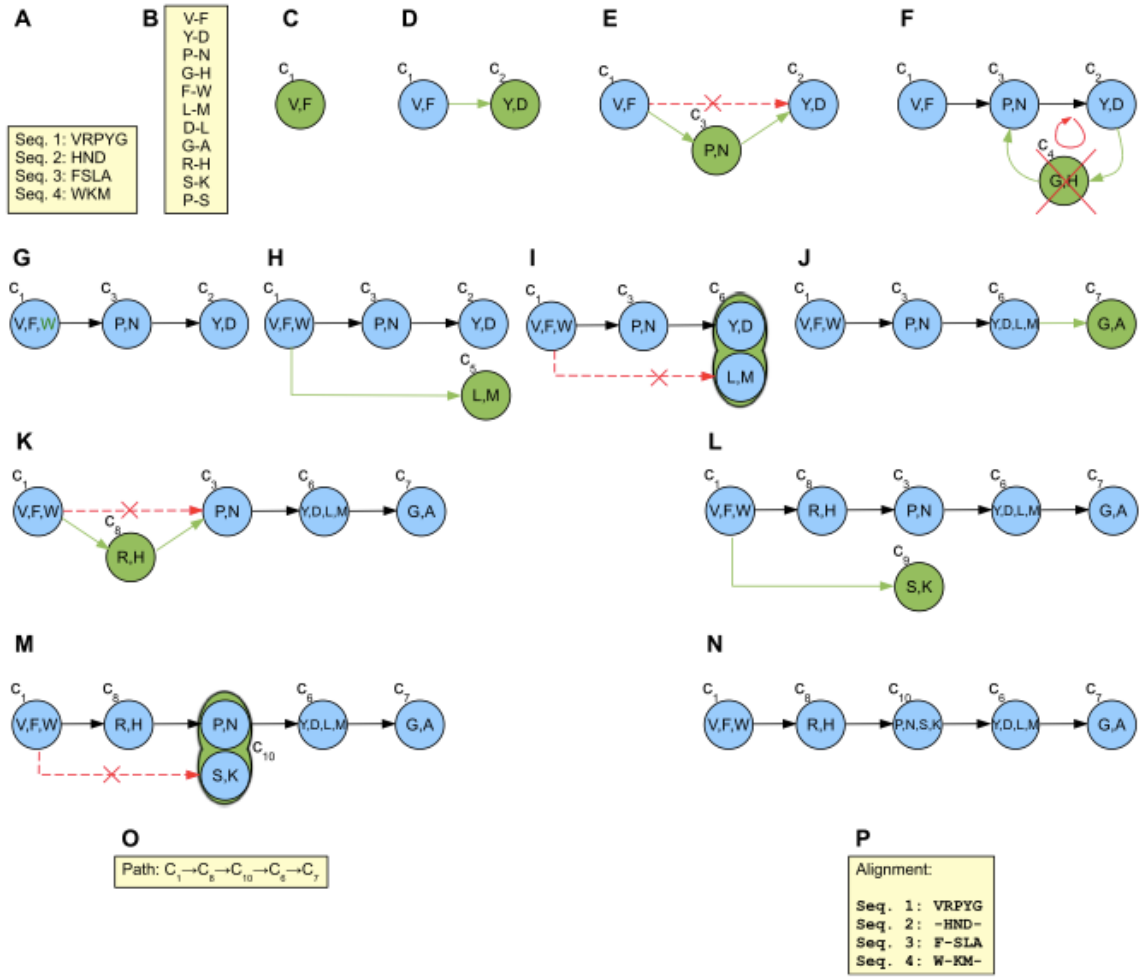

Figura 2.2: *Algoritmo de sequence annealing: (A) sequências que desejamos alinhar, (B) pares de resídios em ardem decrescente em relação ao peso, (C) adicionado um novo vértice para o par de resíduos V-F, (D) adicionado um novo vértice para o par de resíduos Y-D, (E) adicionado um novo vértice para o par de resíduos P-N e a aresta redundante entre os vértices c_1 e c_2 é excluída, (F) o par de resíduos G-H é descartado por introduzir um ciclo no grafo, (G) o vértice c_1 é estendido para adicionar o par de resíduos F-W, (H) adicionado um novo vértice para o par de resíduos L-M, (I) os vértices c_2 e c_5 são mesclados para adicionar o par de resíduos D-L, (J) adicionado um novo vértice para o par de resíduos G-A, (K) adicionado um novo vértice para o par de resíduos R-H e a aresta redundante entre os vértices c_1 e c_3 é excluída, (L) adicionado um novo vértice para o par de resíduos S-K, (M) os vértices c_3 e c_9 são mesclados para adicionar o par de resíduos P-S, (N) grafo do alinhamento, (O) ordem topológica do grafo e (P) alinhamento múltiplo final. Figura extraída de Sahraeian e Yoon (2010)*

.

e selecionar um conjunto consistente dessas âncoras que são utilizadas como um mapa de restrições. Assim, regiões entre as âncoras são alinhadas de maneira independente (Figura 2.3).

Inúmeras ferramentas utilizam o alinhamento ancorado para alinhar sequências genômicas. FSA utiliza uma heurística de *"anchor annealing"* (Bradley *et al.*, 2009), MAVID e Pecan alinham progressivamente as sequências respeitando as restrições impostas pelas âncoras (Bray e Pachter, 2004) (Paten *et al.*, 2009) e o alinhador DIALIGN permite que os usuários especifiquem as âncoras manualmente (Morgenstern *et al.*, 2006).

O alinhamento baseado em segmentos difere do alinhamento ancorado por permitir segmentos alinhados com várias substituições e *indels*. Segmento alinhado, neste caso, é definido como um alinhamento local entre pares de sequências encontrado por algoritmos de alinhamento local, como o algoritmo de Smith-Waterman (Smith e Waterman, 1981).

T-Coffee (Rausch *et al.*, 2008) e DIALIGN (Morgenstern *et al.*, 1998) são duas ferramentas que

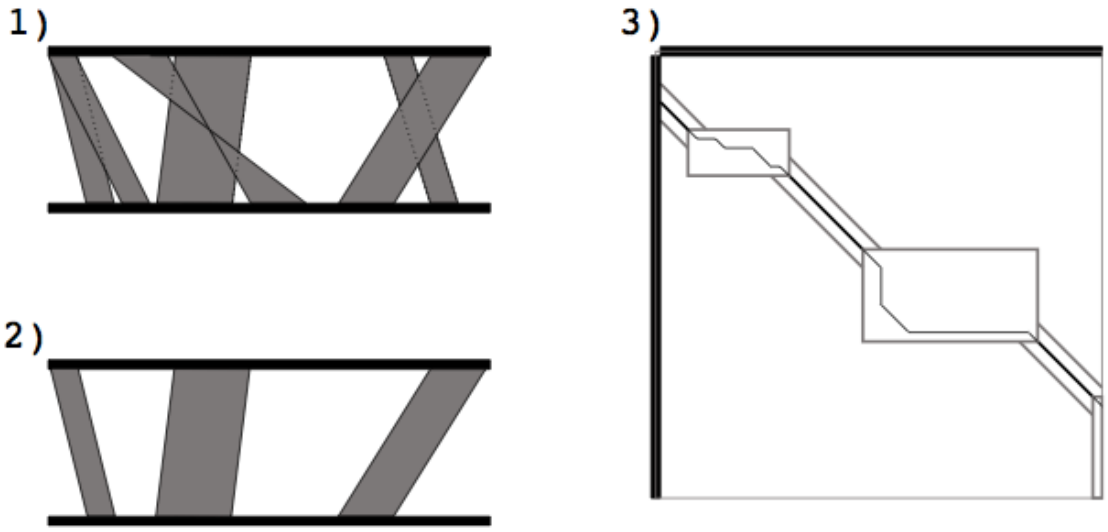

Figura 2.3: *Alinhamento ancorado: (1) âncoras são encontradas entre um par de sequências (2) refinamento das âncoras (sem intersecção) e (3) alinhamento das regiões entre as âncoras. Figura extraída de Rausch (2010).*

se beneficiam do alinhamento baseado em segmentos para identificar e alinhar sequências com blocos conservados. T-Coffee melhorou significativamente seus resultados ao utilizar segmentos em vez de resíduos individuais no alinhamento (Rausch *et al.*, 2008). DIALIGN, na maioria das casos, tem os melhores resultados em relação a outras ferramentas de MSA quando as sequências têm regiões localmente conservadas ou pouca similaridade (Subramanian *et al.*, 2008).

Com a grande quantidade de dados biológicos disponibilizados através de banco de dados de sequências de nucleotídeos, famílias e anotação de sequência de proteínas, fontes externas também podem ser utilizadas para melhorar a qualidade dos alinhamentos construídos pelos algoritmos baseados em âncoras ou em segmentos, como domínios homólogos às sequências que se deseja alinhar encontrados no *Pfam* (Ait *et al.*, 2012) e âncoras encontradas com o pós-processamento de consultas Blast (Thompson *et al.*, 2000).

2.5 Alinhamento baseado em repetições e rearranjos

Uma limitação das ferramentas de alinhamento múltiplo é assumir implicitamente que as sequências são globalmente alinháveis ou que as regiões localmente conservadas entre as sequências ocorrem na mesma ordem (Phuong *et al.*, 2006)(Do e Katoh, 2008). Essas ferramentas inferem eventos mutacionais que ocorrem durante a evolução das sequências, tais como: substituições e *indels*. No entanto, famílias de proteínas com multidomínios podem ter domínios recombinantes, embaralhados (domínios em ordem diferente em diferentes proteínas) ou repetidos (repetição de um domínio em uma proteína) (Raphael *et al.*, 2004).

Domínios repetidos em uma única sequência de proteína podem ser encontrados e alinhados através dos algoritmos de *Repeat finding*. Repetições e rearranjos em várias sequências de proteínas são detectados por *Motif finders* que constroem o alinhamento múltiplo local com blocos de segmentos pequenos e conservados (com mesmo tamanho e sem *gaps*).

Algumas ferramentas de alinhamento múltiplo abordam o problema de alinhar sequências com repetições e rearranjos. O algoritmo consiste basicamente em dois passos: (1) dividir as sequências em blocos de fragmentos, onde cada bloco contém um conjunto de fragmentos de um domínio qualquer e (2) construir o alinhamento múltiplo de cada bloco (Figura 2.4).

ABA é uma ferramenta de MSA que utiliza um grafo de A-Bruijn que permite ciclos e detecta e alinha sequências de proteínas com multidomínios (Raphael *et al.*, 2004). ProDA é um alinhador de domínios de proteínas que encontra repetições e rearranjos e retorna o alinhamento múltiplo local

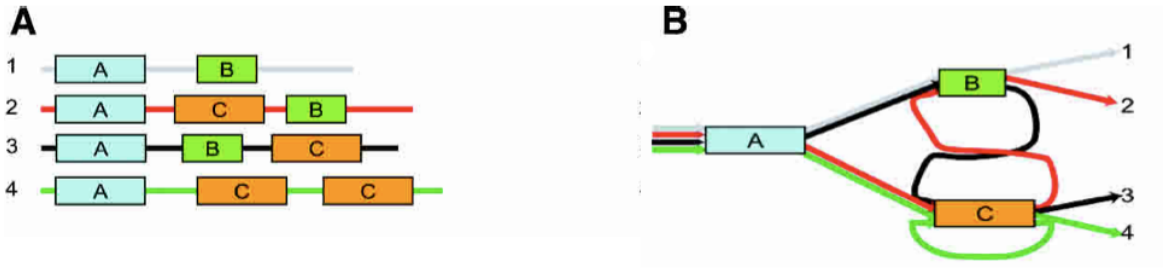

Figura 2.4: *Alinhamento baseado em repetições e rearranjos: (A) quatro sequências de proteínas, onde os fragmentos A, B e C representam um domínio e (B) fragmentos do mesmo domínio são alinhados (blocos). Figura extraída e modificada de Raphael et al. (2004).*

dessas regiões (Phuong *et al.*, 2006).

Para sequências de proteínas com multidomínios, ProDA tem resultados significativamente melhores que outras ferramentas de alinhamento múltiplo e *Motif finders*. No entanto, quando o alinhamento é de fato global, ferramentas que levam em conta repetições e rearranjos têm menor acurácia em relação ao algoritmos de alinhamento global.

2.6 Técnicas para melhorar a qualidade dos alinhamentos múltiplos

O aspecto guloso dos algoritmos de alinhamento múltiplo e as características das sequências que desejamos alinhar podem propagar erros para o resultado final do alinhamento (Chakrabarti *et al.*, 2010). O refinamento iterativo é uma técnica de pós-processamento que é aplicada após a construção do alinhamento e que torna o resultado do alinhamento mais acurado.

Outro problema dos algoritmos de MSA advém do fato de se basearem no alinhamento de pares ou grupos de sequências e não levarem em conta todas as informações das sequências que estão sendo alinhadas. A transformação de consistência é uma técnica de pré-processamento que é aplicada antes do alinhamento das sequências e que torna os alinhamentos entre pares de sequências consistentes uns com os outros.

Apresentaremos estas duas técnicas para melhorar a qualidade dos alinhamentos múltiplos de sequências.

2.6.1 Refinamento iterativo

O aspecto guloso dos algoritmos de alinhamento múltiplo influencia diretamente no resultado do alinhamento. No algoritmo progressivo, por exemplo, em cada etapa do alinhamento se compromete com o alinhamento de pares ou grupos de sequências; assim, se um *gap* é adicionado de forma incorreta em uma etapa do alinhamento, esse mesmo gap não pode ser mais corrigido em qualquer outra etapa (Russell, 2014). Já o algoritmo de *sequence anneling* tem menor acurácia quando os pares de posições têm probabilidades pequenas, fazendo com que a seleção do par mais provável a ser alinhado tenha muitos outros pares de posições concorrentes (Sahraeian e Yoon, 2010).

As características das sequências que desejamos alinhar também podem influenciar os algoritmos de MSA a propagar erros para o alinhamento. Os alinhamentos de sequências com baixa porcentagem de identidade ou com similaridades locais tendem a ter menor acurácia quando alinhados com algoritmos que assumem que as sequências são globalmente alinháveis (Subramanian *et al.* , 2008). Já quando as sequências têm similaridade global ocorre o contrário, ou seja, algoritmos de alinhamento local tendem a ter menor acurácia.

O refinamento iterativo é uma técnica aplicada após a construção do alinhamento que corrige ou minimiza erros propagados para o resultado final do alinhamento. A finalidade dessa técnica é aumentar a pontuação da função objetivo do alinhamento, tal como a soma de pares (Durbin *et al.*, 1998).

Alguns algoritmos de refinamento iterativo, como remove primeiro, melhor primeiro e refinamento iterativo aleatório foram comparados no trabalho de Hirosawa *et al.* (1995) e revisados no trabalho de Wallace *et al.* (2005). Os algoritmos remove primeiro e melhor primeiro são descritos abaixo:

1. *Remove primeiro*
   (a) Seja $S = \{S_1, ..., S_m\}$ o conjunto de sequências que foram alinhadas e A o alinhamento entre as sequências.
   (b) Uma sequência S_x é retirada do alinhamento A. Colunas do alinhamento compostas somente por gaps também são removidas.
   (c) A sequência S_x é realinhada com o restante das sequências do alinhamento $R(= A - \{S_x\})$, construindo assim o alinhamento A'.
   (d) O alinhamento A' é mantido se sua pontuação aumenta em relação a pontuação do alinhamento anterior A. Em outro caso o alinhamento A' é descartado. Os passos (b) e (c) são repetidos.

2. *Melhor primeiro*
   (a) Seja $S = \{S_1, ..., S_m\}$ o conjunto de sequências que foram alinhadas e A o alinhamento entre as sequências.
   (b) Cada uma das sequências $S_x \in S$ são retiradas do alinhamento A e realinhadas com o restante das sequências $R(= A - \{S_x\})$, construindo assim novos alinhamentos $A' = \{A'_1, ..., A'_N\}$, onde N é a quantidade de sequências do alinhamento A.
   (c) Se A'_x é o alinhamento com maior pontuação do conjunto A' e tem pontuação maior que o alinhamento anterior A, então A'_x é mantido como novo alinhamento A. Em outro caso o alinhamento A'_x é descartado.
   (d) Os passos (b) e (c) são repetidos.

Ferramentas como ProbCons (Do *et al.*, 2005), Probalign (Roshan e Livesay, 2006), MAFFT (Katoh *et al.*, 2005) e GLProbs (Ye *et al.*, 2013) implementam o refinamento iterativo aleatório, onde o MSA é dividido em dois grupos aleatoriamente e o algoritmo de alinhamento de perfil é utilizando para realinhar esses grupos (Durbin *et al.*, 1998). O algoritmo aleatório é formalmente definido abaixo:

3. *Refinamento iterativo aleatório*
   (a) O alinhamento A é particionado aleatoriamente em dois grupos de sequências disjuntos G_1 e G_2.
   (b) Os dois grupos G_1 e G_2 são realinhados um com o outro e, se a pontuação desse realinhamento for maior do que a pontuação do alinhamento anterior A, o mesmo é mantido para a próxima iteração.
   (c) Os passos (a) e (b) são repetidos.

Outra abordagem, chamada de refinamento iterativo discriminativo é implementada no trabalho de Sahraeian e Yoon (2010), onde o algoritmo de clusterização k-means é utilizado para particionar o alinhamento em grupos de sequências similares e o algoritmo de alinhamento de perfil alinha os distintos grupos encontrados (Theodoridis e Koutroumbas, 2009) (Durbin *et al.*, 1998). Dessa forma, a pontuação do alinhamento aumenta em regiões com baixa similaridade (entre os grupos) e é preservada em regiões com alta similaridade (nos grupos).

Em relação ao refinamento iterativo aleatório, a versão discriminativa é capaz de aumentar de 5% a 10% a acurácia de um alinhamento (Sahraeian e Yoon, 2010). Formalmente, o refinamento iterativo discriminativo é descrito abaixo.

3. *Refinamento iterativo discriminativo*

   (a) Seja $S = \{S_1, ..., S_m\}$ o conjunto de sequências alinhadas. Para cada sequência $S_x \in S$ é encontrado um grupo de sequências similares G_x e um grupo de sequências que não são similares a S_x, utilizando o algoritmo de clusterização k-means.

   (b) S_x é alinhada com seu respectivo grupo G_x, construindo assim o alinhamento $A'(= G_x \cup \{S_x\})$.

   (c) A' é alinhado com grupo $R(= S - \{G_x\})$ de sequências alinhadas que não são similares a S_x.

   (d) Outra sequência é escolhida e o passo (b) é repetido para cada sequência $S_x \in S$.

A iteração dos algoritmos descritos acima pode ser finalizada quando a pontuação do alinhamento convergir ou pode ser executada um número predefinido de vezes. O refinamento iterativo pode ser utilizado em alinhamentos construídos por qualquer heurística de MSA.

2.6.2 Transformações de consistência

Os algoritmos de MSA, descritos até agora, constroem o alinhamento utilizando as pontuações ou probabilidades entre pares de resíduos, e não levam em conta toda a informação contida nas sequências que estão sendo alinhadas. Dessa forma, erros podem ser propagados para o resultado do alinhamento múltiplo final, pois a pontuação ou probabilidade de dois resíduos estarem alinhados corretamente no alinhamento entre duas sequências pode não ser a mesma quando consideramos o alinhamento de três ou mais sequências (Onuchic, 2012).

A transformação de consistência é uma técnica aplicada antes da construção do MSA que torna os alinhamentos entre pares de sequências consistentes uns com os outros. Isso significa que, se x_i está alinhado com z_k no alinhamento entre as sequências $\mathbf{x}$ e $\mathbf{z}$ com pontuação $w(x_i, z_k)$, e se z_k está alinhado com y_j no alinhamento entre as sequências $\mathbf{z}$ e $\mathbf{y}$ com pontuação $w(z_k, y_j)$, então x_i está alinhado com y_j no alinhamento entre $\mathbf{x}$ e $\mathbf{y}$, com pontuação $w(x_i, y_j) = w(x_i, y_j) + min\{w(x_i, z_k), w(z_k, y_j)\}$.

A ferramenta T-Coffee foi uma das primeiras a implementar a técnica de transformação de consistência utilizando as pontuações do alinhamento entre pares de sequências (Notredame *et al.*, 2000). Do *et al.* (2005) introduziram uma abordagem probabilística, onde são utilizadas as probabilidades *a posteriori* $P(x_i \sim y_j \in a^* | \mathbf{x}, \mathbf{y})$ de cada par de resíduos x_i e y_j no alinhamento ótimo a^* entre as sequências $\mathbf{x}$, $\mathbf{y} \in \mathbf{S}$. Essas probabilidades podem ser calculadas através de diferentes métodos, como PairHMM implementado pela ferramenta ProbCons (Do *et al.*, 2005) e função de partição implementado pela ferramenta Probalign (Roshan e Livesay, 2006).

Dadas probabilidades *a posteriori*, a transformação de consistência probabilística visa atualizar as probabilidades de cada par de resíduos $x_i \in \mathbf{x}$ e $y_j \in \mathbf{y}$ incorporando informações de outras sequências que estão sendo alinhadas $\mathbf{z} \in \mathbf{S} - \{\mathbf{x}, \mathbf{y}\}$. Assim os alinhamento entre os pares de sequências serão consistentes uns com os outros. Essa transformação é dada por:

$$P^*(x_i \sim y_j) = \frac{1}{|S|} \sum_{z \in S} \sum_{k=1}^{|z|} P(x_i \sim z_k) P(z_k \sim y_j) \tag{2.1}$$

onde S é conjunto de sequências sendo alinhadas, $|S|$ é a quantidade de sequências e $|\mathbf{z}|$ é o tamanho da sequência $\mathbf{z}$.

Um problema específico da transformação acima é quando o conjunto de sequências S contém sequências pouco similares ou que pertencem a diferentes subfamílias, assim se a sequência $\mathbf{z}$ não é similar em relação às sequências $\mathbf{x}$ e $\mathbf{y}$ a tendência é que ocorra uma deterioração na qualidade do alinhamento.

Para contornar esse problema, Sahraeian e Yoon (2010) implementaram uma nova versão dessa técnica na ferramenta PicXAA, em que pesos são calculados para cada par de sequências $\mathbf{x}$, $\mathbf{y} \in \mathbf{S}$. O cálculo do peso é dado por:

$$P(\mathbf{x} <> \mathbf{y}) = \frac{1}{|A|} \sum_{x_i \sim y_j \in A} P(x_i \sim y_j), \tag{2.2}$$

onde A é alinhamento de máxima precisão entre $\mathbf{x}$ e $\mathbf{y}$ e $|A|$ é a quantidade de pares de resíduos do alinhamento A. Assim, o peso $P(\mathbf{x} <> \mathbf{y})$ é calculado baseando-se na probabilidade do pares de resíduos e representa a probabilidade das sequências $\mathbf{x}$ e $\mathbf{y}$ serem similares. A nova transformação de consistência com os pesos é dada por:

$$P^*(x_i \sim y_j) = \frac{\sum_{z \in S} P(x <> z) P(z <> y) \sum_{k=1}^{|z|} P(x_i \sim z_k) P(z_k \sim y_j)}{\sum_{z \in S} P(x <> z) P(z <> y)} \tag{2.3}$$

A transformação acima tem melhores resultados que a formulação original da equação 2.1, pois evita que os pares de resíduos $x_i \in \mathbf{x}$ e $y_j \in \mathbf{y}$ incorporem probabilidades de sequências que não são similares a $\mathbf{x}$ e $\mathbf{y}$. Assim, essa nova formulação permite que os alinhamento entre os pares de sequências sejam consistentes uns com os outros, baseando-se na similaridade ou características das sequências.

Na dissertação de Onuchic (2012), foram apresentadas duas inovações que corrigem o problema de negligenciar as probabilidades do alinhamento com *gaps* da formulação original da técnica de transformação de consistência. A primeira inovação foi levar em conta as probabilidades do alinhamento com *gaps* em posições específicas nas sequências intermediárias. A segunda inovação torna as probabilidades consistentes com gaps quaisquer nas sequências intermediárias. Os pesos definidos na equação 2.2, também podem ser adicionados nas duas inovações citadas acima.

Comparações entre as versões das transformações de consistência citadas nesta sessão, utilizando os *benchmarks* BaliBase e IRMBASE, mostraram que a transformação que leva em conta *gaps* quaisquer nas sequências intermediárias é a melhor opção para ser aplicada antes da construção do alinhamento. Devido as melhorias significativas ao utilizar as transformações de consistência modificadas, foi sugerido que outras ferramentas as implementem, e assim possam melhorar os resultados dos seus alinhamentos (Onuchic, 2012).

Apesar da técnica de transformação de consistência deixar os algoritmos de MSA lentos, devido aos cálculos das transformações, essa técnica tem permitido que os algoritmos de MSA construam alinhamentos mais acurados. Assim como o refinamento iterativo, cada uma das formulações citadas acima podem ser executadas iterativamente um número predefinido de vezes.

2.7 Avaliação de alinhamentos múltiplos

A qualidade do alinhamento de sequências tem influência direta no estudo sobre anotação de genomas, comparação de estruturas de proteínas e reconstrução de árvores filogenéticas como mencionado anteriormente neste documento. Sendo assim, *benchmarks* são utilizados para avaliar a qualidade dos alinhamentos construídos pelos algoritmos de MSA. Entre os *benchmarks* para alinhamento de proteínas podemos citar: BaliBase e IRMBASE.

O BaliBase foi um dos primeiros *benchmarks* para avaliação de alinhadores de sequências de proteínas (Thompson *et al.*, 1999). Na sua versão 3.0, é fornecido um conjunto de teste com 218 alinhamentos divididos em seis categorias (Thompson *et al.*, 2005). Cada categoria avalia a performance e a qualidade dos alinhamentos construídos pelas ferramentas de MSA em diferentes cenários. Abaixo são descritas cada uma das categorias:

- RV11: Contém alinhamentos em que as sequências têm menos de 20% de identidade.
- RV12: Contém alinhamentos em que as sequências compartilham entre 20% a 40% de identidade.
- RV20: Contém alinhamentos com sequências de alta similaridade e sequências órfãs.

- RV30: Contém alinhamentos em que sequências dentro de uma subfamília têm mais de 40% de identidade, mas sequências de subfamílias diferentes têm menos de 20% de identidade.
- RV40: Sequências com longas inserções nas extremidades N/C-terminal.
- RV50: Sequências com longas inserções internas.

A versão 2.0 do *benchmark* BaliBase contém conjuntos de testes para problemas específicos, como proteínas com repetições e permutação circular (Bahr *et al.*, 2001). Esses conjuntos podem ser utilizados para avaliar algoritmos de alinhamento múltiplo que levam em conta repetições e rearranjos. A categoria 6 desse *benchmark* está organizada em sete conjuntos de acordo com as repetições de domínios nas sequências: C1a contém sequências com a mesma quantidade de repetições e C1b contém sequências com a quantidade de repetições variável; C2a tem sequências com a mesma quantidade de repetições com subtipos diferentes, na mesma ordem, C2b tem sequências com a mesma quantidade de repetições, em ordem diferente e C2c tem uma quantidade variada de repetições com subtipos diferentes; C3 contém sequências com domínios repetidos e um domínio não repetido e C4 tem sequências com várias repetições com subtipos diferentes. Os subtipos das repetições são definidos de acordo com a similaridade dos seus resíduos.

O IRMBASE é um *benchmark* utilizado para avaliar a capacidade das ferramentas de MSA em capturar similaridades locais (Subramanian *et al.*, 2008). Esse *benchmark* foi construído através da inserção de segmentos altamente conservados gerados pelas ferramentas de simulação de evolução ROSE (Stoye *et al.*, 1998), em longas sequências aleatórias. São 192 alinhamentos divididos em quatro conjuntos de referência, onde os segmentos conservados e as sequências aleatórias têm diferentes tamanhos.

Várias medidas de qualidade podem ser utilizadas para avaliar a similaridade entre esses alinhamentos. As duas medidas mais utilizadas são: a pontuação SP (*Sum of pairs*), que é a porcentagem de pares de resíduos presentes no alinhamento referência que estão corretamente alinhados no alinhamento que desejamos avaliar, e a pontuação CS (*Column Score*), que é a porcentagem de colunas idênticas em ambos os alinhamentos.

Para avaliação de alinhamentos de sequências genômicas são geralmente utilizados conjuntos de dados que simulam regiões gênicas (Bradley *et al.*, 2009). Outra forma de avaliar esses alinhamentos foi definida no estudo de Chen e Tompa (2010), onde as ferramentas para o alinhamento de sequências genômicas Pecan, MAVID, TBA e MLAGAN alinharam 1% do genoma humano contra 27 outros mamíferos. Nesse teste foi utilizada a ferramenta StatSigMA-w para medir a acurácia e a cobertura do alinhamento em quatro categorias de localização genômica: codificante, UTR, intrônica e intergênica.

Capítulo 3

Visualização dos alinhamentos

O alinhamento múltiplo de sequências é representado como uma matriz, onde cada linha representa uma sequência e as colunas são os resíduos alinhados. Novas abordagens para visualização de alinhamentos têm surgido, como a representação do alinhamento como um grafo acíclico direcionado (Grasso *et al.*, 2003) (Sahraeian e Yoon, 2010) ou como um perfil, onde resíduos do alinhamento são visualizados de acordo com sua frequência (Roca *et al.*, 2013). O estudo de Procter *et al.* (2010), mostrou que a visualização de alinhamentos com várias sequências ainda é problema em aberto e que exite uma carência de novas formas de visualizar esses alinhamentos. Nesse mesmo estudo, as ferramentas de visualização foram divididas em 4 categorias, são elas: renderizadores, visualizadores, edição e análise, e os *workbenchs*.

3.1 Renderizadores

Os renderizadores foram as primeiras ferramentas para visualização gráfica e formatação dos alinhamentos múltiplos de sequências. Essas ferramentas recebem um conjunto de parâmetros que controlam como o alinhamento vai ser renderizado, como por exemplo, a fonte, a coloração e o sombreamento do alinhamento (Barton, 1993). Os alinhamentos renderizados podem ser gerados em diferentes formatos e utilizados em publicações, *posters* e apresentações.

3.2 visualizadores

Os visualizadores surgiram como uma ferramenta de interface gráfica para visualização de alinhamentos integrada com algum algoritmo de MSA, como o ClustalX, que é um visualizador de alinhamentos construídos pela ferramenta ClustalW (Larkin *et al.*, 2007) (Figura 3.1). Mais recentemente, foi desenvolvido o primeiro visualizador interativo de alinhamentos com interface web (Gille *et al.*, 2014). Essas ferramentas diferem dos renderizadores por terem uma interface gráfica e facilitarem a navegação por regiões dos alinhamentos utilizando recursos como coloração, gráficos e recursos para ampliar e reduzir a matriz do alinhamento.

Com os visualizadores é possível destacar regiões do alinhamento utilizando esquemas de cores, como o de Taylor (LIN *et al.*, 2002) e do Clustal (Larkin *et al.*, 2007). Esses esquemas são estáticos e atribuem uma cor para cada aminoácido ou nucleotídeo do alinhamento. O visualizador da ferramenta FSA (Fast Statistical Alignment) utiliza uma abordagem dinâmica, onde cada resíduo do alinhamento é classificado de forma binária por meio de cinco medidas: acurácia, sensibilidade, especificidade, certeza e consistência (Bradley *et al.*, 2009). Essas medidas são calculadas com as probabilidades *a posteriori* de um resíduo x_i estar alinhado com outros resíduos em uma mesma coluna de um alinhamento. Com essas medidas, o visualizador consegue destacar as regiões mais confiáveis do alinhamento. Outros recursos fornecidos pelos visualizadores são a geração de gráficos de qualidade, consenso e conservação do alinhamento, e o sombreamento de regiões não conservadas. Devido à integração com algoritmos de MSA, as sequências podem ser alinhadas e realinhadas diretamente na interface gráfica.

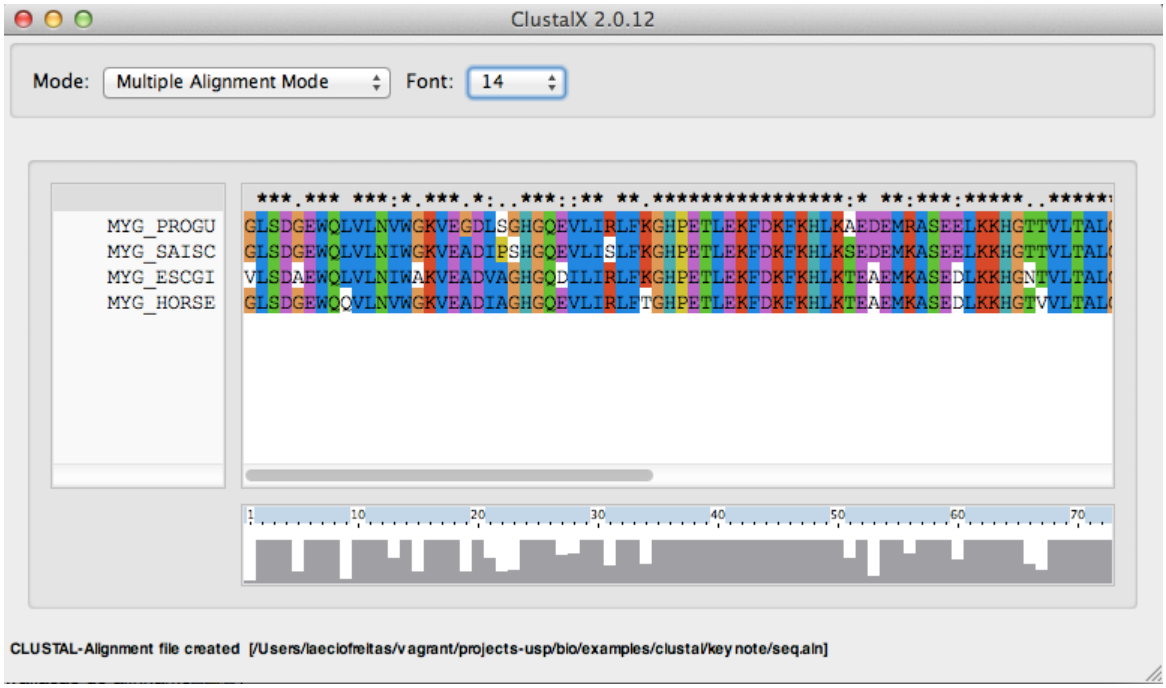

Figura 3.1: *ClustalX: visualizador de alinhamentos.*

3.3 Edição, visualização e análise

Mesmo os algoritmos de MSA mais acurados podem propagar erros para o resultado final do alinhamento; assim, as ferramentas de edição, visualização e análise fornecem recursos para editar os alinhamentos interativamente. Edição significa, neste contexto, que *gaps* podem ser inseridos e removidos em uma única sequências ou em um grupo delas, todas as sequências ou partes delas podem ser selecionadas e editadas, colunas e sequências podem ser escondidas da visualização do alinhamento e grupos podem ser criados para rotular sequências, regiões ou subfamílias de sequências que pertencem ao alinhamento.

Os esquemas de cores das ferramentas de edição e visualização permitem que os usuários restrinjam suas análises a regiões mais confiáveis e editem regiões não confiáveis manualmente (Figura 3.2). Outros recursos são a análise e construção de árvores filogenéticas, integração com bancos de dados de proteínas e a combinação entre alinhamento e a estrutura 3D das sequências. Ferramentas que se enquadram na categoria de edição, visualização e análise são: CINEMA (Lord *et al.*, 2002), GeneDoc (Nicholas *et al.*, 1997), Base-By-Base (Hillary *et al.*, 2011), PFFAT (Caffrey *et al.*, 2007), JEvTrace (Joachimiak e Cohen, 2002), SeaView (Gouy *et al.*, 2010) e Jalview (Waterhouse *et al.*, 2009).

Figura 3.2: *Ferramenta de edição, visualização e análise Jalview.*

Capítulo 4

Multigran: um sistema de alinhamento de granularidade variável

Neste capítulo apresentaremos as propostas feitas neste projeto. O objetivo geral é implementar um alinhador de sequências de granularidade variável, chamado de Multigran, para resolver problemas típicos de alinhamento de sequências, suportar algoritmos de MSA e técnicas para melhorar a qualidade dos alinhamentos. Multigran utiliza o arcabouço ToPS para manipular o modelo probabilístico de PairHMM. Em particular, implementaremos os algoritmos de alinhamento ancorado, baseado em segmentos e baseado em repetições e rearranjos de proteínas, além da técnica de refinamento iterativo.

Integramos também o Multigran com uma ferramenta de interface gráfica para edição, visualização e análise de sequências e alinhamentos. Os alinhamentos e os resultados obtidos pela nossa ferramenta serão comparados com outras ferramentas de MSA existentes.

4.1 TOPS

O ToPS é arcabouço orientado a objetos que oferece uma grande variedade de modelos probabilísticos, que podem ser combinados e utilizados para analisar sequências (Kashiwabara *et al.*, 2013). O arcabouço implementa o algoritmo de *sequence annealing*, onde inicialmente é calculada a probabilidade posterior de cada par de resíduos para cada par de sequências utilizando um modelo de PairHMM. Através dessas probabilidades e do modelo de PairHMM o arcabouço encontra o alinhamento de cada par de sequências que maximiza o maior número esperado de resíduos alinhados corretamente, baseado no alinhamento de máxima precisão esperada, definido por Durbin (Durbin *et al.*, 1998). Antes de alinhar as sequências, qualquer uma das transformações de consistência probabilísticas que serão apresentadas mais à frente na sessão 2.6.2, podem ser calculadas. Utilizando um grafo acíclico direcionado e as probabilidades atualizadas, as sequências são alinhadas através do algoritmo de *sequence annealing*.

A arquitetura de classes do ToPS (Figura 4.1) é dividida em três níveis, onde a classe abstrata *ProbabilisticModel* está no primeiro nível e fornece uma interface que possibilita a construção de instâncias dos modelos através do polimorfismo. O segundo nível é composto pelas seguintes classes abstratas:

- *FactorableModel* representa modelos em que a verossimilhança de uma sequência é fatorável em um produto de termos, com um termo para cada posição da sequência.
- *InhomogeneousFactorableModel* representa modelos não homogêneos.
- *DecodableModel* representa modelos decodificadores.
- *DiscriminativeModel* representa modelos discriminativos, como *Linear-Chain Conditional Random Field*.

- *GenerativeModel* representa modelos geradores, como HMM e GHMM.
- *PairDecodableModel* representa modelos decodificadores que trabalham com um par de sequências.

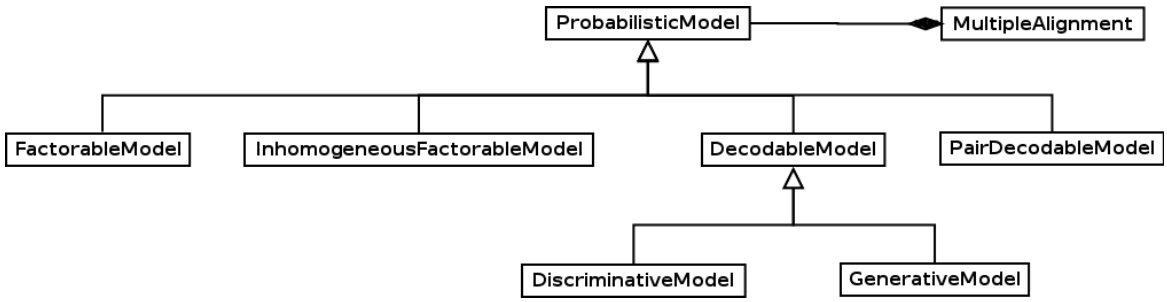

Figura 4.1: *Primeiro e segundo nível da arquitetura de classes do ToPS e a classe MultipleAlignment.*

O terceiro nível contém as implementações dos modelos abstratos e a implementação da classe *MultipleAlignment*. Entre os modelos implementados estão: cadeias ocultas de Markov, cadeias generalizadas ocultas de Markov, cadeias de Markov não homogêneas e cadeias de alcance variável de Markov:

- *DiscreteIIDModel.*
- *VariableLengthMarkovChain.*
- *InhomogeneousMarkovChain.*
- *LinearChainCRF.*
- *HiddenMarkovModel.*
- *GeneralizedHiddenMarkovModel.*
- *PairHiddenMarkovModel.*
- *ProfileHiddenMarkovModel.*
- *MultipleAlignment é a classe que implementa o algoritmo de sequence annealing.*

4.2 Modificações na construção de alinhamentos múltiplos

Nesta sessão apresentaremos as modificações nos algoritmos de alinhamento, que foram implementados no Multigran.

4.2.1 Alinhamento ancorado e baseado em segmentos

Um dos desafios desse algoritmo é manter a consistência entre os segmentos, ou seja, manter a ordem e a posição na qual os segmentos serão adicionados no alinhamento. Para manter a consistência usaremos o algoritmo de *sequence anneling* citado na sessão 2.3. Outro desafio é evitar que ocorra intersecção ou sobreposição entre os segmentos. Para permitir somente segmentos alinhados disjuntos utilizaremos o algoritmo de refinamento de segmentos.

O algoritmo para alinhamento baseado em segmentos descrito nas subseções seguintes é divido em 6 passos e pode ser utilizado para alinhar sequências genômicas utilizando âncoras ou alinhar sequências de nucleotídeos e proteínas utilizando o alinhamento local entre pares de sequências como segmentos alinhados.

Geração dos segmentos

Para cada par de sequências é encontrado um conjunto de segmentos alinhados, que são armazenados no conjunto $M = \{M_1, M_2, ..., M_m\}$. Os segmentos alinhados podem ser encontrados através do alinhamento local, e para sequências genômicas grandes, âncoras são encontradas utilizando MuMMer (Kurtz *et al.*, 2004) ou exonerate (Slater e Birney, 2005).

MuMMer encontra a subsequência localmente mais longa, sem substituições ou *indels*, que ocorre apenas uma vez entre um par de sequências. Essa subsequência é chamada de *maximal unique matches* (MUM) e é encontrada de forma eficiente por uma árvore de sufixo. Para espécies distantes o MuMMer encontra poucos MUMs, assim o exonerate será utilizado para encontrar subsequência localmente mais longa com substituições.

Refinamento dos segmentos

Os segmentos alinhados podem se sobrepor ou se intersectar, sendo necessário definir um conjunto de segmentos alinhados entre pares de sequências que são consistentes com todas as sequências e não se sobrepõem. Usamos o algoritmo de refinamento de múltiplos segmentos alinhados para encontrar uma subdivisão mínima de segmentos, tal que os segmentos possam ser adicionados ao alinhamento.

Dado o conjunto de segmentos alinhados $M = \{M_1, M_2, ..., M_m\}$, existe a possibilidade de sobreposição entre $M_x, M_y \in M$. O algoritmo de refinamento encontra um conjunto de refinamento $R = \{M'_1, M'_2, ..., M'_n\}$, onde n é a quantidade de segmentos alinhados sem sobreposição. Cada segmento alinhado $M'_x \in R$ é disjunto de qualquer outro segmento alinhado $M'_k \in R$ (Figura 4.2).

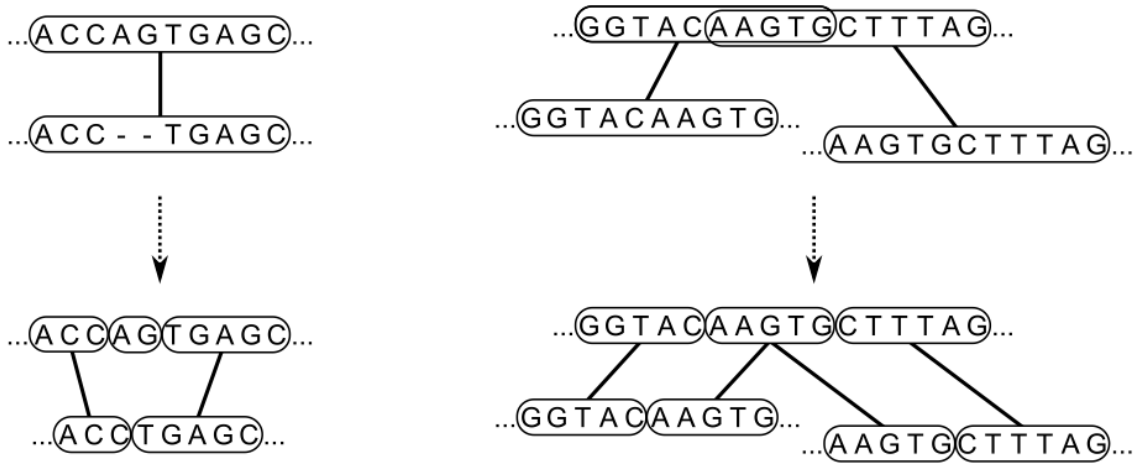

Figura 4.2: *Refinamento dos segmentos alinhados. Figura extraída de Gogol-Döring e Reinert (2009).*

O algoritmo de refinamento para pares de sequências foi inicialmente definido por Halpern e seus colaboradores (Halpern *et al.*, 2002) e estendido para várias sequências pela ferramenta T-Coffee (Rausch *et al.*, 2008). Usaremos a versão definida por Rausch e seus colaboradores (Rausch *et al.*, 2008) para encontrar o conjunto de refinamento R para qualquer número de sequências.

Construção do grafo do alinhamento

O algoritmo de refinamento dos segmentos retorna um grafo $G = (V, E)$ que só permite alinhamentos entre segmentos do mesmo tamanho (Figura 4.2). O grafo contém todos os segmentos refinados $R = \{M'_1, M'_2, ..., M'_n\}$, onde cada $v \in V$ corresponde a um segmento sem *gaps* e cada aresta $e \in E$ representa um segmento alinhado (Figura 4.2).

Cada aresta $e = (v_1, v_2) \in E$ com $v_1, v_2 \in V$ recebe um peso que pode ser definido de acordo com a ferramenta ou algoritmo utilizado para encontrar os segmentos.

Transformações de consistência

Dados os pesos para cada aresta do grafo $G = (V, E)$ construído no passo anterior, a transformação de consistência visa transformar esses pesos de forma que os alinhamentos entre os segmentos sejam consistentes uns com os outros.

Consistência neste caso significa que, se existe a aresta $e_1 = (v_1, v_2) \in E$ com os vértices $v_1, v_2 \in V$ e peso $w(e_1)$ entre as sequências x e y, e se existe a aresta $e_2 = (v_2, v_3) \in E$ com os vértices $v_2, v_3 \in V$ e peso $w(e_2)$ entre as sequências y e z, então o peso $e_3 = (v_1, v_3)$ dos vértices $v_1, v_3 \in V$ será incrementado com $min\{w(e1), w(e2)\}$. Se a aresta e_3 não existe, ela é criada com o peso $min\{w(e1), w(e2)\}$. Essa transformação foi inicialmente definida pela ferramenta T-Coffee (Rausch *et al.*, 2008) e será utilizada neste algoritmo.

Assim, temos um novo grafo $G' = (V, E')$, com $E' \in E$. O pesos transformados serão utilizados para construir o alinhamento com os segmentos alinhados utilizando o algoritmo de *sequence annealing* no passo abaixo.

Alinhamento dos segmentos

O algoritmo de *sequence annealing* será utilizado para a construção do grafo do alinhamento final. Para isso, o conjunto $P = \{w(e_1), w(e_2), ..., w(e_n)\}$ é construído com os pesos das arestas do grafo $G' = (V, E')$ em ordem decrescente, onde $w(e_1)$ é o maior peso de uma aresta do grafo G' e $w(e_n)$ é o menor peso.

O alinhamento inicia do maior peso para o menor. Seja $w(e_1)$ o peso para aresta $e_1 = (v_1, v_2)$, então e_1 é adicionado ao grafo do alinhamento final $A(V, E'')$, desde que seja compatível com as arestas de E''. Ser compatível significa que os vértices v_1 e v_2 não pertencem a E''. Esse processo é repetido para cada $w(e_i) \in P$.

Alinhamento das regiões entre os segmentos

A partir do passo anterior, temos um conjunto de segmentos alinhados entre pares de sequências que são consistentes com todas as sequências. Esses segmentos alinhados são utilizados como um mapa de restrições para alinhamento das regiões entre os segmentos. Nesse passo assumimos que os segmentos alinhados têm probabilidade ~ 1 (Bradley *et al.*, 2009) e as regiões entre os segmentos serão alinhadas com a versão do algoritmo de alinhamento múltiplo implementado pelo ToPS definido na sessão 2.3 sobre o algoritmo de *sequence annealing*. Os resíduos entre os segmentos consistentes podem ser adicionados ao alinhamento final sem a necessidade de alinhá-los. Esse passo pode ser definido como uma opção para o usuário.

O algoritmo de *sequence annealing* retorna um grafo que precisa ser mapeado para o alinhamento múltiplo final. Existe um relacionamento de um-para-muitos entre o grafo e o alinhamento. Encontrar a ordem topológica dos vértices e arestas é feita de maneira linear, como descrito na sessão 2.3.

4.2.2 Alinhamento baseado em repetições e rearranjos

Um dos desafios desta heurística é manter a consistência entre os pares de fragmentos alinhados, pois domínios da mesma família podem ter diferentes graus de conservação e diferentes tamanhos entre as sequências. Utilizaremos o algoritmo de *sequence annealing* para manter a consistência entre pares de alinhamentos locais e filtrar as partes conservadas desses alinhamentos. O resultado final do alinhamento é um conjunto de regiões alinhadas com um comprimento mínimo (Figura 4.3).

Nas subseções seguintes descrevemos os passos do algoritmo que leva em conta repetições e rearranjos, e será integrado no ToPS. São 6 passos baseados no algoritmo do alinhador ProDA. Os passos para geração e alinhamento de um bloco, extração do alinhamento final e remoção de PLAs usados, são executados iterativamente enquanto existirem pares de fragmentos alinhados que não pertencem a qualquer bloco.

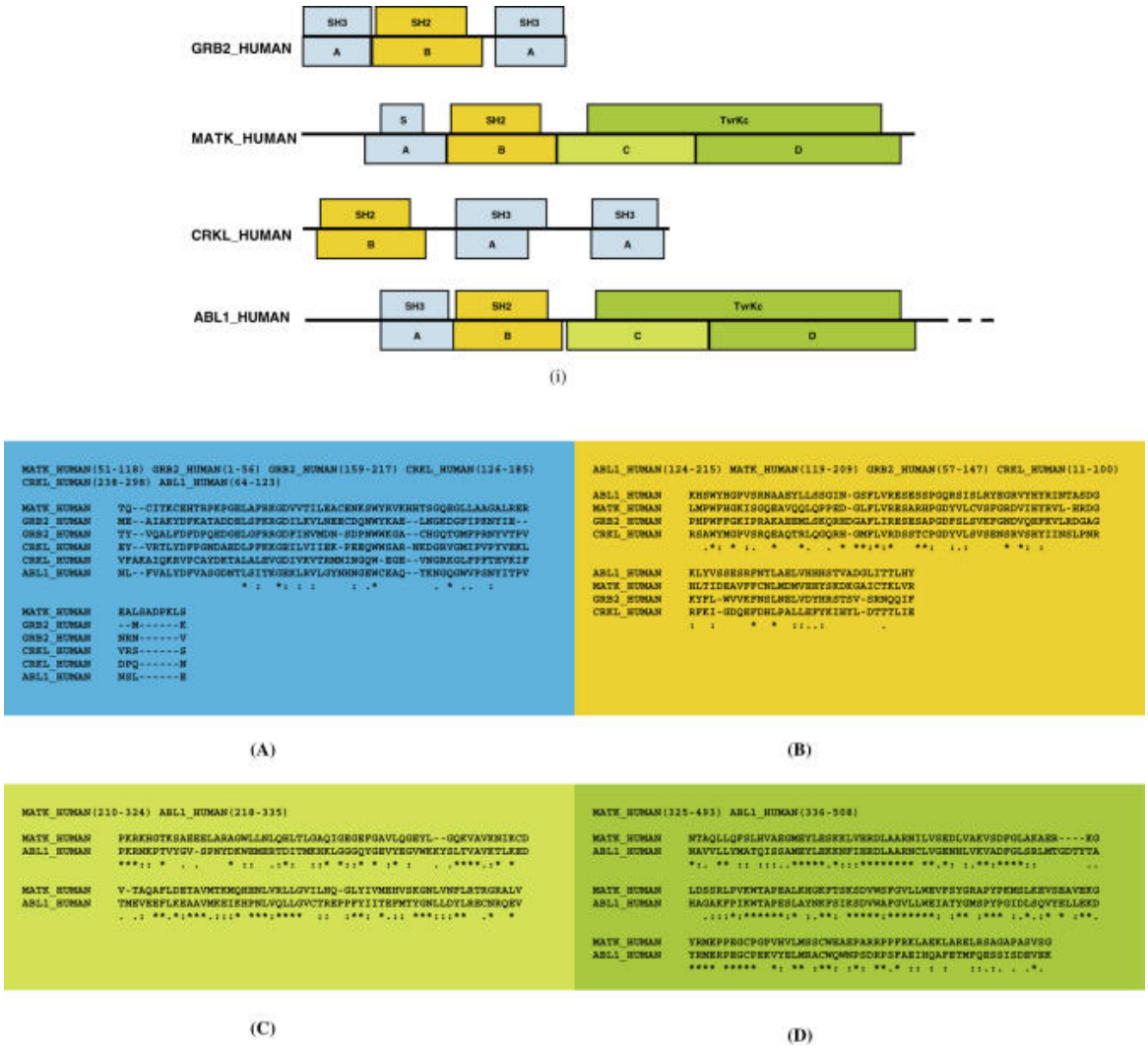

Figura 4.3: *Alinhamento baseado em repetições e rearranjos: (i) quatro sequências de proteínas, com domínios repetidos (SH3 na sequência CRKL_HUMAN) e rearranjos (SH2-SH3 na sequência CRKL_HUMAN e SH3-SH2 na sequência ABL1_HUMAN), e (B) fragmentos do mesmo domínio são alinhados em um mesmo bloco (A-D). Figura extraída de Phuong et al. (2006).*

Construção do alinhamentos locais entre pares de sequências

As sequências de proteínas podem ter diferentes domínios, com diferentes tamanhos e diferentes graus de conservação uns com os outros (Raphael *et al.*, 2004). Poucos alinhamentos exatos são encontrados em sequências de proteínas distantemente relacionadas, assim nos alinhamentos locais são permitidas substituições e *indels*.

Dado s sequências de proteínas $S = \{S_1, ..., S_s\}$ e o comprimento mínimo de um alinhamento local L_{min}. Para cada par de sequências $x, y \in S$, são calculados todos os alinhamentos locais de comprimento no mínimo L_{min}. Os alinhamentos podem ser calculados através dos algoritmos *viterbi deconding*, Smith e Waterman, e Waterman e Eggert descritos na sessão 2.1 sobre alinhamento de pares. Esses algoritmos retornam um conjunto de alinhamento locais $M = \{M_1, M_2, ..., M_m\}$ entre todas as sequências de S, onde m é a quantidade de alinhamentos encontrados.

Domínios grandes podem ter regiões pouco conservadas, fazendo com que os alinhamentos locais encontrados sejam somente fragmentos de domínios e domínios pequenos próximos uns dos outros podem fazer com que os alinhamentos locais encontrados sejam uma concatenação de domínios vizinhos (Phuong *et al.*, 2006).

ProDa utiliza o modelo de PairHMM específico, para calcular o alinhamento local de cada par de sequências $x, y \in S$. ToPS é mais flexível por permitir que os modelos sejam especificados

em arquivos de configuração, em que é possível alterar qualquer parâmetro sem a necessidade da alteração do código do arcabouço. Assim, diferentes modelos de PairHMM podem ser definidos para modelar tanto sequências com domínios grandes como sequências com domínios pequenos.

Inferência de repetições dos alinhamentos de pares de sequências

Os alinhamentos locais encontrados no passo anterior são pós-processados para que seja possível encontrar alinhamentos que se sobrepõem entre um par de sequências. Essa sobreposição pode ocorrer devido aos alinhamentos locais pequenos e próximos uns dos outros inferidos como um único alinhamento, como mencionado anteriormente. Com este pós-processamento, os alinhamentos locais que contêm vários outros alinhamento pequenos, podem ser divididos em alinhamento menores.

Este processamento é executado para cada par de sequências. Após a inferência das repetições os passos subsequentes são executados.

Geração de um bloco de fragmentos das sequências

Bloco é definido como um conjunto B de alinhamentos locais que representam um domínio específico. O conjunto de sequências S pode ter uma arquitetura com multidomínios e assim vários blocos podem ser inferidos.

Dado m alinhamentos locais que ainda não pertencem a nenhum bloco $M = \{M_1, M_2, ..., M_m\}$, dois segmentos de um par de sequências são alinháveis se pertencem a qualquer alinhamento local $M_x \in M$ ou ambos os fragmentos de $M_x \in M$ são alinháveis com um terceiro fragmento $M_y \in M$. O bloco B é calculado com o número máximo de segmentos alinháveis, no qual pelo menos um dos segmentos é alinhável com todos os outros segmentos diretamente ou através de um terceiro segmento (Phuong *et al.*, 2006).

Os alinhamentos locais do bloco B podem ser inconsistentes uns com os outros e ter diferentes tamanhos, assim o alinhador ProDa implementa um filtro para definir o limite inicial e final de cada segmento do bloco B. Utilizaremos somente o alinhamento de um bloco descrito abaixo para manter a consistência entre os alinhamentos locais do bloco B e encontrar as similaridades locais entre os alinhamentos sem a necessidade de definir filtros heurísticos.

Alinhamento de um bloco

Em vez de alinhar sequências inteiras, somente os alinhamentos locais encontrados no passo anterior são alinhados. Para cada par de alinhamentos locais $M_x \in B$ são calculadas as probabilidades posteriores utilizando o modelo de PairHMM. As probabilidades são reestimadas com qualquer uma das transformações de consistência fornecidas pelo ToPS e descritas na sessão 2.6.2.

Após o cálculo das probabilidades e das transformações de consistência, todos os alinhamentos locais do bloco B são alinhados usando o algoritmo de alinhamento múltiplo implementado pelo arcabouço ToPS e descrito na sessão 2.3 sobre o algoritmo de *sequence annealing*.

Extração do alinhamento final

As colunas iniciais e finais do alinhamento construído pelo passo anterior podem conter **gaps**, indicando erros no cálculo dos limites do bloco. O algoritmo refina o alinhamento retornando a maior fração do bloco, em que a coluna inicial e final não contêm *gaps*.

Remoção de alinhamentos locais usados

Dois fragmentos iguais de alinhamentos locais distintos não podem pertencer a mais de um bloco, pois cada bloco possui um conjunto de PLAs alinhados que representam um domínio específico. Seja $M = \{M_1, M_2, ..., M_m\}$ o conjunto de alinhamentos locais que não ainda não pertencem a nenhum bloco e $N = \{N_1, N_2, ..., N_n\}$ o conjunto de alinhamentos locais que pertencem ao bloco

construído no passo anterior. O algoritmo verifica se algum alinhamento local $N_x \in N$ compartilha algum par de fragmentos com M.

A Figura 4.3 mostra um exemplo, onde um par de fragmentos é compartilhado por um alinhamento local $N_x \in N$ e um alinhamento local $M_y \in M$. O par de fragmentos compartilhado é removido de M_x e o restante do alinhamento local é mantido se o seu comprimento é maior que L_{min}. Se ainda existirem alinhamentos locais o passo 4.2.2 é executado novamente.

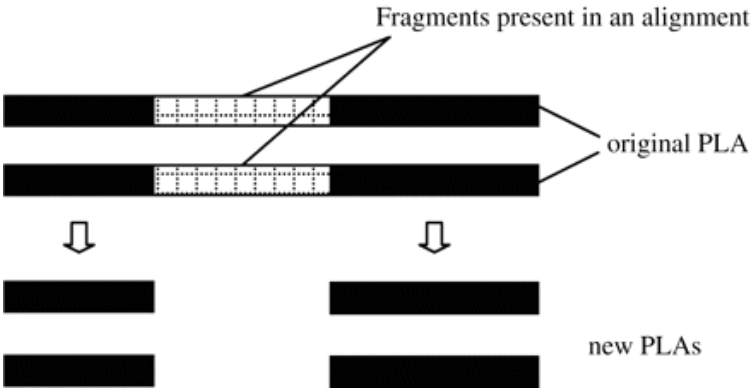

Figura 4.4: *Remoção de fragmento de um alinhamento local já utilizado usado. Figura extraída de Phuong et al. (2006).*

4.2.3 Refinamento iterativo

O algoritmo de *sequence annealing* pode não gerar um alinhamento acurado ao alinhar sequências que pertencem a diferentes subfamílias, pois poderão existir vários pares de posições com probabilidades pequenas concorrendo entre si e a seleção do par mais provável e consistente a ser alinhado entre esses pares concorrentes pode produzir resultados ruins (Sahraeian e Yoon, 2010).

Para alinhamentos que contêm subfamílias de sequências o algoritmo progressivo separa cada subfamília em grupos, utilizando algum algoritmo de clusterização, depois as sequências de cada grupo são alinhadas e por fim os grupos são alinhados entre si, usando o algoritmo de alinhamento de perfil. Por alinhar inicialmente as sequências similares que pertencem a cada grupo e depois alinhar os grupos de cada subfamília, algumas ferramentas que implementam uma variação do algoritmo progressivo produzem os melhores resultados para esse conjunto de dados (Sahraeian e Yoon, 2010) (Altschul *et al.*, 1997). Sendo assim, para o Multigran, que implementa o algoritmo de *sequence annealing*, ter resultados competitivos em relação às outras ferramentas é necessária a implementação da técnica de refinamento iterativo para melhorar a qualidade dos alinhamentos.

Baseado no algoritmo de refinamento iterativo discriminativo (Sahraeian e Yoon, 2010) e na desvantagem do algoritmo de *sequence annealing* citada acima, propomos a seguinte variação da heurística de refinamento iterativo.

Dado m sequências $S = \{S_1, ..., S_m\}$ e o número máximo de sequências C_{max} que serão realinhadas com seus respectivos grupos:

1. *Divisão de S em grupos.* Usando o algoritmo de clusterização k-means (Theodoridis e Koutroumbas , 2009), para cada sequência $S_x \in S$ é encontrado um conjunto de sequências similares N_x. Baseado nos conjuntos de sequências similares, o conjunto S é particionado em grupos $G = \{G_1, ..., G_n\}$, sendo n a quantidade de grupos que representam cada subfamília ou conjunto de sequências similares que pertencem a S. R é o conjunto de sequências que não pertencem a nenhum grupo. A definição de máxima precisão esperada é utilizada como medida de similaridade entre duas ou mais sequências em um alinhamento A. Se A for um alinhamento múltiplo é necessário considerar todos os pares de posições alinhadas.

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

2. *Repetição para cada* $S_x \in S$. São escolhidas no máximo C_{max} sequências de S.

   (a) Cada uma das sequências escolhidas são realinhadas com seus respectivos grupos.

   (b) Os grupos que representam cada uma das sequências escolhidas e o grupo R são alinhados. A definição de máxima precisão esperada citada acima é utilizada para definir a similaridade e a ordem dos alinhados de cada grupo.

   (c) Outras C_{max} sequências são escolhidas e o passo (a) é repetido.

O ciclo de iteração é finalizado quando a pontuação do alinhamento converge, ou se um limite de $2N^2$ iterações é atingido, onde N é o número de sequências.

4.3 Implementação

A arquitetura do ToPS foi construída em cima de padrões de projeto, como *Factory Method* e *Composite*, descritos por Gamma E. *et al.* (1994). Essa característica permite que novos modelos sejam adicionados mais facilmente no arcabouço.

Nossa proposta consistiu em transformar a classe concreta *MultipleAlignment*, que implementa o algoritmo de *sequence annealing*, em uma classe abstrata que será utilizada para facilitar a adição de algoritmos de alinhamento múltiplo. Outra modificação na arquitetura é a construção da classe abstrata *PairwiseAlignment* que funcionará como uma interface para a implementação dos algoritmos para alinhamento de um par de sequências.

A implementação dos algoritmos de MSA e de alinhamento de pares envolve a extensão das classes abstratas *MultipleAlignment* e *PairwiseAlignment*, como mostra Figura 4.5. Abaixo, estão as classes concretas dos algoritmos que estão sendo implementados e que foram descritos neste documento:

- *Sequence annealing* (2.3),
- *Anchored* (4.2.1),
- *RepeatsAndRearrangements* (4.2.2),
- *IterativeRefinement* (4.2.3),
- *NeedlemanWunsch* (2.1),
- *SmithWaterman* (2.1),
- *WatermanEggert* (2.1).

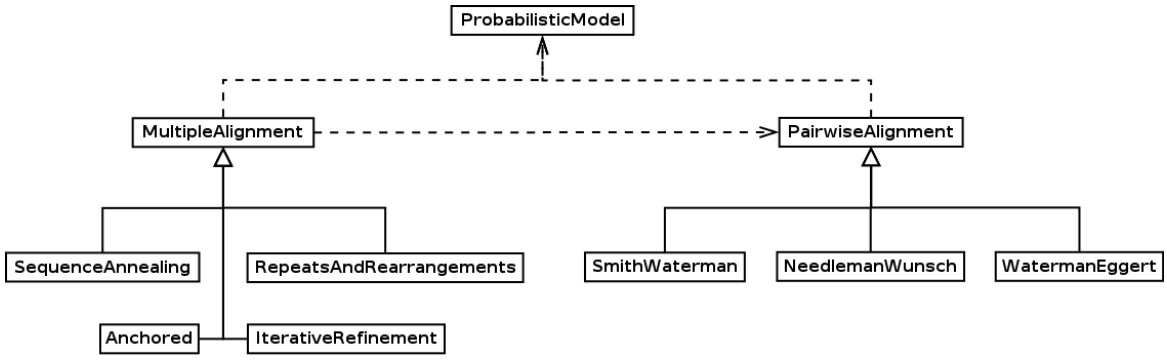

Figura 4.5: *Modelagem proposta para a adição dos algoritmos de alinhamento múltiplo e alinhamento de pares de sequências. As classes abstratas MultipleAlignment e PairwiseAlignment dependem de um modelo probabilístico para o cálculo das probabilidades a posteriori, das transformações de consistência, entre outros.*

Outra modificação, na classe concreta *MultipleAlignment*, será a extração do código das versões de transformações de consistência e do grafo acíclico direcionado do algoritmo de *sequence annealing* para classes específicas.

4.4 Visualização

Multigran tem duas versões do algoritmo de *sequence annealing*. Uma é a formulação original deste método definida no artigo de Sahraeian e Yoon (2010) e a outra é uma versão modificada que maximiza a função de precisão esperada levando em conta *gaps* no alinhamento (Onuchic , 2012). A ferramenta também fornece opções para melhorar a qualidade dos alinhamentos, como seis versões de transformações de consistência (todas as transformações da sessão 2.6.2), sendo que quatro dessas versões são inovações que modificam a formulação original deste método (Onuchic, 2012). As outras duas versões são definidas nos artigos de Do *et al.* (2005) e de Sahraeian e Yoon (2010). Portanto, o usuário pode alinhar sequências de 14 formas diferentes, além disso, pode utilizar as técnicas de refinamento iterativo, alinhamento ancorado e alinhamento baseado em segmentos, repetições e rearranjos formuladas neste documento.

O algoritmo de *sequence annealing* usa diferentes critérios matemáticos para encontrar uma solução aproximada para o problema de alinhamento de sequências, sem levar em conta as características bioquímicas das sequências. Isso pode gerar erros difíceis de serem corrigidos automaticamente, através das transformações de consistência e refinamento iterativo. Esses erros podem ser corrigidos manualmente com ajuda de informações sobre as sequências, tais como estrutura das famílias de proteínas e genes.

Devido à quantidade de opções para alinhar sequências fornecidas pelo Multigran e aos erros que podem ser gerados pelos mais acurados algoritmos de alinhamento (Blackshields *et al.*, 2006), integramos nossa ferramenta com uma ferramenta de edição, visualização e análise de alinhamentos de sequências.

Dentre as ferramentas sobre visualização de alinhamentos múltiplos, CINEMA e GeneDoc não parecem estar em desenvolvimento ativo, pois não são atualizadas desde 2004 e 2007 respectivamente. Base-By-Base é uma ferramenta com foco no alinhamento de genomas, enquanto PFFAT, JEvTrace e SeaView focam na visualização e análise de proteínas e árvores filogenéticas. Jalview fornece recursos para edição, visualização e análise de sequências de proteínas e árvores filogenéticas, mas também inclui recursos específicos para análise de sequências e alinhamentos de DNA.

Dessa maneira, integramos o ToPS com à ferramenta Jalview (Figura 3.2). A escolha se deve ao foto de a nossa ferramenta alinhar sequências de DNA e proteínas e a ferramenta Jalview fornecer recursos para edição, visualização e análise desses tipos de alinhamentos. Através da integração do Multigran com JalView, o usuário pode editar os alinhamentos e utilizar as diferentes opções de alinhamento fornecidas pela nossa ferramenta (Figura 4.6).

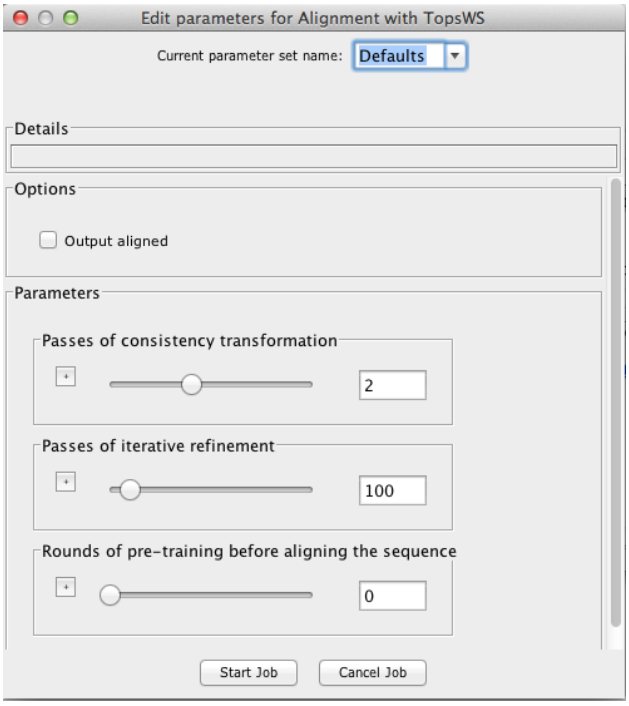

Figura 4.6: *Opções fornecidas para o alinhamento de sequências utilizando os algoritmos de sequence annealing do Multigran. O usuário pode escolher a versão do algoritmo e informar o número de vezes que as transformações de consistência, o algoritmo iterativo e o treinamento das sequências serão executados.*

Capítulo 5

Validações e comparações

Neste capítulo descreveremos como foram feitas as validações e as comparações com outras ferramentas em relação aos algoritmos de alinhamento ancorado e baseado em segmentos, alinhamento baseado em repetições e rearranjos, refinamento iterativo e *sequence annealing*.

Como mencionado na sessão 4.4, ToPS implementa as transformações de consistência clássicas e as modificadas, além da versão original do algoritmo de *sequence annealing* e uma versão modificada desse método que leva em conta o posicionamento correto dos *gaps*. Comparações feitas entre os pares de versões da ferramenta (qualquer versão do algoritmo de *sequence annealing* mais qualquer versão das transformações de consistência), utilizando os *benchmarks* BaliBase e IRMBASE, mostraram que a transformação de consistência que considera o alinhamento entre uma base e um *gap* qualquer na outra sequência tem as melhores performances utilizando os critérios de avaliação citados na sessão 2.7 (Onuchic, 2012). Já o algoritmo de *sequence annealing* modificado tem uma melhora significativa, em relação à versão original, quando o critério de qualidade maximiza o posicionamento correto dos *gaps*.

O motivo de comparar pares de versões implementados pelo ToPS foi avaliar as técnicas independentemente de outras heurísticas e melhorias de performance implementadas por outras ferramentas, fazendo assim uma comparação honesta em uma única plataforma. Devido às melhorias significativas ao utilizar as transformações de consistência modificadas, foi sugerido que outras ferramentas pudessem implementá-las, e assim melhorarem o resultados dos seus alinhamentos (Onuchic, 2012).

Neste trabalho, faremos uma comparação dos resultados entre diferentes ferramentas de MSA. Assim, serão avaliadas diferentes heurísticas implementadas por outras ferramentas, como alinhamento progressivo e *sequence annealing* que utilizam ou não as transformações de consistência e o refinamento iterativo.

Em estudo recente, o Clustal Omega superou outras ferramentas de alinhamento múltiplo em relação ao tempo de execução e à qualidade do alinhamento quando são alinhadas mais de 10000 sequências (Sievers *et al.*, 2011). Utilizaremos as ferramentas que tiveram seus resultados avaliados no artigo do Clustal Omega e os benchmarks BaliBase2.0, BaliBase3.0 e IRMBASE para avaliar os resultados do ToPS em relação ao alinhamento de sequências de proteínas, utilizando os algoritmos já implementados e os algoritmos propostos neste documento. Os critérios de qualidade definidos na sessão 2.7 serão usados para a avaliação dos resultados.

Nos trabalhos de Chen e Tompa (2010) e Kim e Sinha (2010) a ferramenta Pecan é a que tem os melhores resultados no alinhamento de sequências genômicas. Utilizaremos o teste, as ferramentas e os resultados encontrados por Chen e Tompa descritos na sessão 2.7, para avaliar o alinhamento ancorado 4.2.1 e o alinhamento baseado em segmentos 4.2.1 em relação ao alinhamento de sequências genômicas.

Capítulo 6

Conclusões e considerações finais

Neste trabalho propomos novas abordagens para lidar com os problemas de alinhamentos de múltiplas sequências. Essas propostas envolveram modificações em técnicas para construção de alinhamento múltiplo e na forma de editar, visualizar e analisar o alinhamento de sequências.

Em relação ao alinhamento de sequências grandes, propusemos duas modificações no algoritmo de alinhamento ancorado e baseado em segmentos. Essas modificações envolviam a utilização do algoritmo de *sequence annealing* para manter a consistência entre os segmentos ou âncoras alinhadas e o algoritmo de refinamento dos segmentos para evitar que ocorra intersecção ou sobreposição entre os segmentos. Confirmamos que a utilização dessas modificações de fato melhora o resultado final dos alinhamentos.

Agora tratando de alinhamentos de sequências com multidomínios, apresentamos uma modificação no algoritmo de alinhamento baseado em repetições e rearranjos. Mostramos que utilizar o algoritmo *sequence annealing* na construção do alinhamento melhora a consistência dos alinhamentos dos domínios e filtra as partes conservadas do alinhamento.

Mostramos que o alinhamento progressivo tem bons resultados no alinhamento de sequências que contêm subfamílias, e que utilizando o algoritmo de refinamento iterativo modificado nos alinhamentos construidos pelo algoritmo de *sequence annealing* em alinhamentos de sequências que contêm subfamílias, o resultado final do alinhamento é compatível com o resultado dos alinhamentos construidos por ferramentas que implementam o algoritmo progressivo.

Por fim, é importe ressaltar que o alinhador Multigran foi integrado com uma ferramenta de edição, visualização e analise de sequências, chamada de Jalview.

Referências Bibliográficas

Ait *et al.* **(2012)** L.A. Ait, E. Corel e B. Morgenstern. Using protein-domain information for multiple sequence alignment. Em *Bioinformatics Bioengineering (BIBE), 2012 IEEE 12th International Conference on*, páginas 163–168. doi: 10.1109/BIBE.2012.6399667. Citado na pág. 8

Altschul *et al.* **(1997)** Stephen F. Altschul, Thomas L. Madden, Alejandro A. Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller e David J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402. doi: 10.1093/nar/25.17.3389. Citado na pág. 25

Bahr *et al.* **(2001)** Anne Bahr, Julie Dawn Thompson, Jean-Claude Thierry e Olivier Poch. Balibase (benchmark alignment database): enhancements for repeats, transmembrane sequences and circular permutations. *Nucleic Acids Research*, 29(1):323–326. Citado na pág. 13

Barton (1993) G. J. Barton. ALSCRIPT — a tool to format multiple sequence alignments. *Prot. Eng.*, 6. Citado na pág. 15

Blackshields *et al.* **(2006)** Gordon Blackshields, WallaceIain M. Wallace, Mark Larkin e Desmond G. Higgins. Analysis and comparison of benchmarks for multiple sequence alignment. *In Silico Biology*, 6(4):321–339. Citado na pág. 2, 27

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 Comput Biol*, 5(5):e1000392. doi: 10.1371/journal.pcbi.1000392. Citado na pág. 1, 6, 7, 13, 15, 22

Bray e Pachter (2004) Nicolas Bray e Lior Pachter. Mavid: Constrained ancestral alignment of multiple sequences. *Genome Research*, 14(4):693–699. doi: 10.1101/gr.1960404. Citado na pág. 1, 6, 7

Caffrey *et al.* **(2007)** Daniel Caffrey, Paul Dana, Vidhya Mathur, Marco Ocano, Eun-Jong Hong, Yaoyu Wang, Shyamal Somaroo, Brian Caffrey, Shobha Potluri e Enoch Huang. Pfaat version 2.0: A tool for editing, annotating, and analyzing multiple sequence alignments. *BMC Bioinformatics*, 8(1):381. Citado na pág. 16

Chakrabarti *et al.* **(2010)** Saikat Chakrabarti, Christopher Lanczycki, Anna Panchenko, Teresa Przytycka, Paul Thiessen e Stephen Bryant. State of the art: refinement of multiple sequence alignments. *BMC Bioinformatics*, 11(1):3. ISSN 1471-2105. doi: 10.1186/1471-2105-11-3. Citado na pág. 9

Chen e Tompa (2010) Xiaoyu Chen e Martin Tompa. Comparative assessment of methods for aligning multiple genome sequences. *Nature Biotechnology*, 28(6):567 – 572. ISSN 1546-1696. Citado na pág. 13, 29

Do e Katoh (2008) Chuong B. Do e Kazutaka Katoh. Protein Multiple Sequence Alignment. Em *Functional Proteomics*, volume 484 of *Methods in Molecular Biology*, páginas 379–413. Citado na pág. 8

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. 6, 10, 11, 27

Durbin *et al.* **(1998)** Richard Durbin, Sean R Eddy, Anders Krogh e Graeme Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press. Citado na pág. ix, 2, 3, 4, 9, 10, 19

Gamma E. *et al.* **(1994)** Helm R. Gamma E., Johnson R. e Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley. Citado na pág. 26

Gelly *et al.* **(2011)** Jean-Christophe Gelly, Agnel Praveen Joseph, Narayanaswamy Srinivasan e Alexandre G. de Brevern. ipba: a tool for protein structure comparison using sequence alignment strategies. *Nucleic Acids Research*, 39(suppl 2):W18–W23. doi: 10.1093/nar/gkr333. Citado na pág. 1

Gille *et al.* **(2014)** Christoph Gille, Weyand Birgit e Andreas Gille. Sequence alignment visualization in html5 without java. *Bioinformatics*, 30:121–122. Citado na pág. 15

Gogol-Döring e Reinert (2009) Andreas Gogol-Döring e Knut Reinert. *Biological sequence analysis using the SeqAn C++ library*. CRC. Citado na pág. ix, 21

Gouy *et al.* **(2010)** Manolo Gouy, Stéphane Guindon e Olivier Gascuel. Seaview version 4: A multiplatform graphical user interface for sequence alignment and phylogenetic tree building. *Molecular Biology and Evolution*, 27(2):221–224. Citado na pág. 16

Grasso *et al.* **(2003)** Catherine Grasso, Michael Quist, Kevin Ke e Christopher Lee. Poaviz: a partial order multiple sequence alignment visualizer. *Bioinformatics*. Citado na pág. 15

Halpern *et al.* **(2002)** Aaron L. Halpern, Daniel H. Huson e Knut Reinert. Segment match refinement and applications. Em *Proceedings of the Second International Workshop on Algorithms in Bioinformatics*, WABI '02, páginas 126–139. Springer-Verlag. ISBN 3-540-44211-1. Citado na pág. 21

Hillary *et al.* **(2011)** William Hillary, Song-Han Lin e Chris Upton. Base-by-base version 2: single nucleotide-level analysis of whole viral genome alignments. *Microbial Informatics and Experimentation*, páginas 1–6. Citado na pág. 16

Hirosawa *et al.* **(1995)** Makoto Hirosawa, Yasushi Totoki, Masaki Hoshida e Masato Ishikawa. Comprehensive study on iterative algorithms of multiple sequence alignment. *Computer Applications in the Biosciences*, 11:13–18. Citado na pág. 10

Hogeweg e Hesper (1984) P. Hogeweg e B. Hesper. The alignment of sets of sequences and the construction of phyletic trees: An integrated method. *Journal of Molecular Evolution*. doi: 10.1007/BF02257378. URL http://dx.doi.org/10.1007/BF02257378. Citado na pág. 5

Joachimiak e Cohen (2002) Marcin Joachimiak e Fred Cohen. Jevtrace: refinement and variations of the evolutionary trace in java. *Genome Biology*, 3(12). Citado na pág. 16

Just (2001) Winfried Just. Computational complexity of multiple sequence alignment with sp-score. *Journal of Computational Biology*, 8(6):615–623. Citado na pág. 1

Kashiwabara *et al.* **(2013)** A. Y. Kashiwabara, A. M. Durham, V. Onuchic, I. Bonadio, R. Mathias e F. Amado. Tops: A framework to manipulate probabilistic models of sequence data. *PLoS computational biology (accepted for publication)*. Citado na pág. 2, 6, 19

Katoh *et al.* **(2005)** Kazutaka Katoh, Kei-ichi Kuma, Hiroyuki Toh e Takashi Miyata. Mafft version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Research*, 33. Citado na pág. 6, 10

Kim e Sinha (2010) Jaebum Kim e Saurabh Sinha. Towards realistic benchmarks for multiple alignments of non-coding sequences. *BMC Bioinformatics*, 11:54. Citado na pág. 29

Kurtz *et al.* **(2004)** Stefan Kurtz, Adam Phillippy, Arthur Delcher, Michael Smoot, Martin Shumway, Corina Antonescu e Steven Salzberg. Versatile and open software for comparing large genomes. *Genome Biology*, 5(2):R12. ISSN 1465-6906. doi: 10.1186/gb-2004-5-2-r12. Citado na pág. 21

Larkin *et al.* **(2007)** M.A. Larkin, G. Blackshields, N.P. Brown, R. Chenna, P.A. McGettigan, H. McWilliam, F. Valentin, I.M. Wallace, A. Wilm, R. Lopez, J.D. Thompson, T.J. Gibson e D.G. Higgins. Clustal w and clustal x version 2. *Bioinformatics*, páginas 2947–2948. Citado na pág. 15

LIN *et al.* **(2002)** KUANG LIN, ALEX C.W. MAY e WILLIAM R. TAYLOR. Amino acid encoding schemes from protein structure alignments: Multi-dimensional vectors to describe residue types. *Journal of Theoretical Biology*, 216(3):361 – 365. Citado na pág. 15

Lord *et al.* **(2002)** P.W. Lord, J.N. Selley e T.K. Attwood. Cinema-mx: a modular multiple alignment editor. *Bioinformatics*, 18(10):1402–1403. Citado na pág. 16

Morgenstern *et al.* **(1998)** B Morgenstern, K Frech, A Dress e T Werner. Dialign: finding local similarities by multiple sequence alignment. *Bioinformatics*, 14(3):290–294. doi: 10.1093/bioinformatics/14.3.290. Citado na pág. 1, 7

Morgenstern *et al.* **(2006)** Burkhard Morgenstern, Sonja Prohaska, Dirk Pohler e Peter Stadler. Multiple sequence alignment with user-defined anchor points. *Algorithms for Molecular Biology*, 1(1):6. ISSN 1748-7188. doi: 10.1186/1748-7188-1-6. Citado na pág. 1, 7

Mount (2004) David W. Mount. *Bioinformatics: sequence and genome analysis*, chapter Phylogenetic Prediction. CSHL press. Citado na pág. 5

Needleman e Wunsch (1970) Saul B. Needleman e Christian 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. ISSN 0022-2836. doi: http://dx.doi.org/10.1016/0022-2836(70)90057-4. Citado na pág. 3, 4, 5

Nicholas *et al.* **(1997)** K. B. Nicholas, H. B. Nicholas e D. W. Deerfield. GeneDoc: analysis and visualization of genetic variation. *EMBNEW. NEWS*, 4:14. Citado na pág. 16

Notredame *et al.* **(2000)** Cédric Notredame, Desmond G Higgins e Jaap Heringa. T-coffee: a novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 302(1):205 – 217. ISSN 0022-2836. doi: http://dx.doi.org/10.1006/jmbi.2000.4042. Citado na pág. 11

Onuchic (2012) Vitor Onuchic. Inovações em técnicas de alinhamentos múltiplos e predições de genes., 2012. Citado na pág. 2, 11, 12, 27, 29

Ortuño *et al.* **(2013)** Francisco M. Ortuño, Olga Valenzuela, Hector Pomares, Fernando Rojas, Javier P. Florido, Jose M. Urquiza e Ignacio Rojas. Predicting the accuracy of multiple sequence alignment algorithms by using computational intelligent techniques. *Nucleic Acids Research*, 41(1):e26. doi: 10.1093/nar/gks919. Citado na pág. 1

Paten *et al.* **(2009)** Benedict Paten, Javier Herrero, Kathryn Beal e Ewan Birney. Sequence progressive alignment, a framework for practical large-scale probabilistic consistency alignment. *Bioinformatics*, 25(3):295–301. doi: 10.1093/bioinformatics/btn630. Citado na pág. 1, 6, 7

Phuong *et al.* **(2006)** Tu Minh Phuong, Chuong B. Do, Robert C. Edgar e Serafim Batzoglou. Multiple alignment of protein sequences with repeats and rearrangements. *Nucleic Acids Research*, 34(20):5932–5942. doi: 10.1093/nar/gkl511. Citado na pág. ix, x, 1, 2, 8, 9, 23, 24, 25

Procter *et al.* **(2010)** James B Procter, Julie Thompson, Ivica Letunic, Chris Creevey, Fabrice Jossinet e Geoffrey J Barton. Visualization of multiple alignments, phylogenies and gene family evolution. *Nature Methods*, 7:S16 – S25. ISSN 1548-7105. Citado na pág. 15

Raphael *et al.* **(2004)** Benjamin Raphael, Degui Zhi, Haixu Tang e Pavel Pevzner. A novel method for multiple alignment of sequences with repeated and shuffled elements. *Genome Research*, 14 (11):2336–2346. doi: 10.1101/gr.2657504. Citado na pág. ix, 1, 2, 8, 9, 23

Rausch (2010) Tobias Rausch. *Dissecting Multiple Sequence Alignment Methods*. Tese de Doutorado, Universidade Livre de Berlim. Citado na pág. ix, 8

Rausch *et al.* **(2008)** Tobias Rausch, Anne-Katrin Emde, David Weese, Andreas Döring, Cedric Notredame e Knut Reinert. Segment-based multiple sequence alignment. *Bioinformatics*, 24(16): i187–i192. doi: 10.1093/bioinformatics/btn281. Citado na pág. 1, 7, 8, 21, 22

Roca *et al.* **(2013)** Alberto I. Roca, Aaron C. Abajian e David J. Vigerust. ProfileGrids solve the large alignment visualization problem: influenza hemagglutinin example. *F1000 Research*. ISSN 2046-1402. Citado na pág. 15

Roshan e Livesay (2006) Usman Roshan e Dennis R. Livesay. Probalign: multiple sequence alignment using partition function posterior probabilities. *Bioinformatics*, 22:2715–2721. Citado na pág. 6, 10, 11

Russell (2014) David J. Russell. *Multiple Sequence Alignment Methods*. Humana Press. Citado na pág. 1, 6, 9

S. Schwartz e Pachter (2007) Ariel S. Schwartz e Lior Pachter. Multiple alignment by sequence annealing. *Bioinformatics*, 23(2):e24–e29. doi: 10.1093/bioinformatics/btl311. Citado na pág. 6

Sahraeian e Yoon (2010) Sayed Mohammad Ebrahim Sahraeian e Byung-Jun 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. ix, 1, 6, 7, 9, 10, 11, 15, 25, 27

Saitou e Nei (1987) N Saitou e M Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425. Citado na pág. 5

Sievers *et al.* **(2011)** Fabian Sievers, Andreas Wilm, David Dineen, Toby J Gibson, Kevin Karplus, Weizhong Li, Rodrigo Lopez, Hamish McWilliam, Michael Remmert, Johannes Söding, Julie D Thompson e Desmond G Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Molecular Systems Biology*, 7(1). doi: 10.1038/msb.2011.75. Citado na pág. 6, 29

Slater e Birney (2005) Guy Slater e Ewan Birney. Automated generation of heuristics for biological sequence comparison. *BMC Bioinformatics*, 6(1):31. ISSN 1471-2105. doi: 10.1186/1471-2105-6-31. Citado na pág. 21

Smith e Waterman (1981) T.F. Smith e M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195 – 197. ISSN 0022-2836. doi: http://dx.doi.org/10.1016/0022-2836(81)90087-5. Citado na pág. 4, 7

Stoye *et al.* **(1998)** J Stoye, D Evers e F Meyer. Rose: generating sequence families. *Bioinformatics*, 14(2):157–163. Citado na pág. 13

Subramanian *et al.* **(2005)** Amarendran Subramanian, Jan Weyer-Menkhoff, Michael Kaufmann e Burkhard Morgenstern. Dialign-t: An improved algorithm for segment-based multiple sequence alignment. *BMC Bioinformatics*, 6(1):66. ISSN 1471-2105. doi: 10.1186/1471-2105-6-66. Citado na pág. 1

Subramanian *et al.* **(2008)** Amarendran R. Subramanian, Michael Kaufmann e Burkhard Morgenstern. Dialign-tx: greedy and progressive approaches for segment-based multiple sequence alignment. *Algorithms for Molecular Biology*, 3. Citado na pág. 8, 9, 13

Theodoridis e Koutroumbas (2009) Sergios Theodoridis e Konstantinos Koutroumbas. *Pattern Recognition, Fourth Edition*. Academic Press. ISBN 9781597492720. Citado na pág. 10, 25

Thompson *et al.* **(1999)** J D Thompson, F Plewniak e O Poch. Balibase: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, 15(1):87–88. Citado na pág. 12

Thompson *et al.* **(2000)** J. D. Thompson, F. Plewniak, J.-C. Thierry e O. Poch. Dbclustal: rapid and reliable global multiple alignments of protein sequences detected by database searches. *Nucleic Acids Research*, 28(15):2919–2926. doi: 10.1093/nar/28.15.2919. Citado na pág. 8

Thompson *et al.* **(2005)** Julie D. 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. Citado na pág. 12

Wallace *et al.* **(2005)** Iain M. Wallace, O'Sullivan Orla e Desmond G. Higgins. Evaluation of iterative alignment algorithms for multiple alignment. *Bioinformatics*, 21(8):1408–1414. doi: 10.1093/bioinformatics/bti159. Citado na pág. 10

Wang *et al.* **(2011)** Li-San Wang, James Leebens-Mack, P. Kerr Wall, Kevin Beckmann, Claude W. dePamphilis e Tandy Warnow. The impact of multiple protein sequence alignment on phylogenetic estimation. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 8(4):1108–1119. Citado na pág. 1

Wang e Jiang (1994) Lusheng Wang e Tao Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348. Citado na pág. 1

Waterhouse *et al.* **(2009)** Andrew M. Waterhouse, James B. Procter, David M. A. Martin, Michèle Clamp e Geoffrey J. Barton. Jalview version 2?a multiple sequence alignment editor and analysis workbench. *Bioinformatics*, 25(9):1189–1191. Citado na pág. 2, 16

Waterman e Eggert (1987) Michael S. Waterman e Mark Eggert. A new algorithm for best subsequence alignments with application to trna-rrna comparisons. *Journal of Molecular Biology*, 197(4):723 – 728. Citado na pág. 5

Ye *et al.* **(2013)** Yongtao Ye, David W. Cheung, Yadong Wang, Siu-Ming Yiu, Qing Zhan, Tak Wah Lam e Hing-Fung Ting. Glprobs: Aligning multiple sequences adaptively. ACM. ISBN 978-1-4503-2434-2. Citado na pág. 6, 10