Novas abordagens para o problema do alinhamento múltiplo de seqüências

André Atanasio Maranhão Almeida

Resumo

Alinhamento de seqüências é reconhecidamente uma das tarefas de maior importância em biologia computacional, principalmente por ser uma operação básica utilizada por diversos outros procedimentos na área. Da mesma forma, alinhamento múltiplo de seqüências também possui grande relevância, sendo empregado em tarefas como detecção de padrões para caracterizar famílias proteicas e predição de estruturas secundárias e terciárias de proteínas. É importante salientar que existem algoritmos eficientes para o problema do alinhamento de duas seqüências, porém o problema do alinhamento múltiplo é NP-Difícil [117]. O objetivo deste projeto é desenvolver métodos eficientes, quanto a tempo e espaço, para o problema do alinhamento múltiplo de seqüências, que apresentem resultados de alta relevância biológica. Estudaremos as principais abordagens e métodos existentes, realizando uma série de implementações e testes. Ao final do trabalho, esperamos dispor de diversos algoritmos, usando as mais distantas abordagens, e um meta-método capaz de combinar os resultados gerados por eles para construir alinhamentos de mais alta qualidade.

1 Introdução

Quando um genoma é seqüenciado um dos principais objetivos é entender o conjunto de genes do qual ele é composto. Em outras palavras, entender a estrutura, função e evolução dos genes. Tradicionalmente, investiga-se a função de um gene através de análise de fenótipos mutantes, porém isso tem mudado desde a década passada. Análise comparativa de seqüências homólogas tem mostrado ser uma abordagem muito eficiente para estudar função de gene [59]. Estudar padrões de mutação através de análise de seqüências homólogas é útil não apenas para o estudo de relacionamentos evolucionários, mas também para identificar restrições estruturais ou funcionais nas seqüências de RNA, DNA ou proteína [59]. Alinhamento de seqüências é sem dúvida a tarefa mais comum em bioinformática [162]. Diversos procedimentos necessitam de comparação de seqüências, desde buscas em bancos de dados [9] até predição de estrutura secundária [183]. Seqüências podem ser comparadas aos pares numa varredura por seqüências homólogas em uma base de dados ou elas podem ser alinhadas simultaneamente, construindo o chamado MSA (do inglês, *Multiple Sequence Alignment*), para visualização do efeito da evolução em uma família inteira de proteínas.

MSAs tornam possível a indagação de uma grande variedade de questões biológicas importantes. Por exemplo, árvore filogenética é um dos instrumentos utilizado para elucidar

relações evolucionárias que existem entre organismos. Atualmente, a construção de árvores filogenéticas altamente precisas fazem uso de dados moleculares e a geração de um MSA é uma etapa essencial de suas construções. O papel do alinhamento múltiplo é prover uma estimativa muito precisa de distância de pares e possibilitar estimar a confiabilidade de cada ramo através de *bootstrapping* [66]. MSAs também possibilitam a identificação de *motifs* preservados pela evolução. *Motifs* desempenham um papel importante na estrutura e função de um grupo de proteínas relacionadas. Quando trabalhando com dados experimentais, estes *motifs* constituem um meio muito poderoso de caracterizar seqüências de função desconhecida [162].

Predição de estrutura é outro importante uso de alinhamentos múltiplos. Predição de estrutura secundária e terciária objetivam a predição do papel que um dado resíduo desempenha na estrutura da proteína (oculto ou exposto, hélice ou fita, etc.). Predições de estruturas secundárias baseadas em uma única seqüência possuem uma precisão baixa [71], enquanto que predições baseadas em MSAs são mais precisas [116, 182, 183]. Isso ocorre porque padrões de substituição observados em uma coluna refletem diretamente o tipo de restrições impostas naquela posição no curso da evolução. No contexto de determinação de estrutura terciária ou predição de contatos não locais, alinhamentos múltiplos podem também ajudar na identificação de mutações correlacionadas. Esta abordagem tem dado resultados limitados quando aplicada a proteínas [72], mas tem tido sucesso em análises de RNA [92].

O alinhamento de seqüências homólogas consiste na tentiva de posicionar resíduos (nucleotídeos ou aminoácidos) em colunas, que derivam de um resíduo de um ancestral comum [59]. Isso é obtido pela introdução de *gaps*, que representam *indels* (inserções ou remoções), nas seqüências. Desta forma, um alinhamento é um modelo hipotético de mutações (substituições, inserções ou remoções) que ocorreram durante a evolução da seqüência. O melhor alinhamento será aquele mais parecido com um cenário evolucionário.

No contexto do alinhamento de duas seqüências, Needleman e Wunsch em 1970 [158] descreveram um algoritmo que constrói um alinhamento global ótimo através de programação dinâmica. Para seqüências de tamanhos m e n tal algoritmo tem $O(mn)$ como complexidade de tempo e memória. Posteriormente, Smith e Waterman [199] descreveram uma variação deste algoritmo para alinhamento local, que tem a mesma complexidade. Por alinhamento global entende-se o alinhamento das seqüências em suas extensões completas. No alinhamento local só é alinhada a região de maior similaridade entre as seqüências. Hoje, além destes algoritmos, dispomos de soluções heurísticas que, apesar de não garantirem a melhor solução, obtêm bons resultados em um tempo menor. Um exemplo é o FASTA [174], que limita a programação dinâmica a uma faixa, de largura parametrizada, na diagonal da matriz. Ele parte do pressuposto de que o alinhamento ótimo não foge muito da diagonal principal da matriz. Outro exemplo é o algoritmo de Miller e Myers [147], que reduz a complexidade de espaço para $O(m+n)$, mas demanda mais tempo. Um último exemplo é o BLAST [9].

Para alinhar mais que duas seqüências, é possível generalizar facilmente a abordagem de Needleman e Wunsch, porém esta solução ficará limitada a um pequeno número de seqüências não muito longas [136]. Este algoritmo requer $\Omega(2^n l^n)$ de tempo e $\Omega(l^n)$ de espaço, onde n é o número de seqüências e l o comprimento de cada uma delas, para calcular um alinhamento ótimo. Esta complexidade pode ainda ser mais elevada caso a

penalidade para *gap* não seja linear [59].

Alinhamento múltiplo de seqüências é uma tarefa complexa [117,231], que implica em três dificuldades técnicas distintas: escolha das seqüências, escolha da função objetivo e otimização da função. Os métodos tratados neste contexto, geralmente, fazem sentido apenas se assumido que o conjunto de seqüências é composto por seqüências homólogas. E quando tratamos de métodos de alinhamento global, com exceção apenas do DiAlign [152], há ainda o requisito de que as seqüências sejam semelhantes em todo, ou quase todo, o seu comprimento para que o alinhamento gerado tenha alguma relevância. Para os casos em que não se cumprem estes requisitos, deve se considerar o uso de métodos para computação de MSAs locais, tais como: Gibbs Sampler [131], Match-Box [51] e MACAW [191]. Para maiores informações sobre os problemas que podem surgir devido a uma má escolha das seqüências a alinhar, consultar trabalho de Henikoff [94]. É comum o uso de uma das ferramentas BLAST (WU-BLAST, PSI-BLAST, Gapped BLAST, etc.) [11] para avaliar o nível de relacionamento entre as seqüências do conjunto. Tais ferramentas utilizam modelos estatísticos, desenvolvidos por Altschul e Karlin [119], para estimar o grau de relacionamento entre as seqüências. É importante salientar que estes modelos são uma mera aproximação da realidade biológica.

A função objetivo (OF, do inglês *objective function*) deve permitir uma quantificação da qualidade de um alinhamento. Como conseqüência, permita comparações e a seleção daquele alinhamento que tenha um maior significado biológico. Dada uma OF "perfeita", o alinhamento ótimo matemático deveria ser um alinhamento ótimo biológico [162]. Na prática, tal OF "perfeita" não existe e mesmo que existisse, a tarefa de demonstrar que é "perfeita" já se trata de um grande problema. Na teoria, uma OF deveria incorporar tudo o que é conhecido acerca das seqüências, incluindo sua estrutura, função e história evolucionária. Entretando, este tipo de informação raramente está disponível e, mesmo que esteja, é de difícil utilização [162]. Desta forma, é comum utilizar similaridade de seqüência através da simples função de soma-dos-pares com pesos e função afim para penalidade de *gaps* [10]. Embora esta OF seja claramente incorreta do ponto de vista evolucionário [10], por assumir que toda seqüência é ancestral de toda outra no conjunto, é popular nos métodos MSA mais utilizados [80,136,215], devido a facilidade de implementação. Em 2000, Gonnet e colegas propuseram uma variação de soma-dos-pares que parece ser mais fidedigna aos eventos evolucionários [76].

Trabalhos recentes têm feito uso de OFs que parecem ser menos sensíveis à penalidade de *gap* graças a incorporação de informações locais. Neste contexto podemos citar o DiAlign [152] e o T-COFFEE [165]. O primeiro inclui avaliação baseada em segmento e o segundo usa uma OF baseada em consistência. Outros trabalhos fazem uso de HMMs [23,93], que descrevem o MSA em um contexto estatístico usando uma abordagem Bayesiana. Embora de um ponto de vista formal estes métodos (HMM) parecerem soluções mais atrativas, suas performances para alinhamentos *ab initio* decepcionam [162]. É importante, entretando, salientar que um trabalho recente [120] exibiu uma solução que faz uso de HMM e foi capaz de produzir melhores resultados que o Clustal W, método para computação de MSA largamente utilizado [162]. Outros métodos baseados em estatística que tentam associar um *P-value* ao alinhamento múltiplo foram descritos [100, 131]. Infelizmente estas medidas são restritas a MSAs sem *gaps*. Em um mundo ideal, uma OF perfeita deveria existir para todas as situações. Na prática, isso não ocorre [162] e, assim, o usuário terá

sempre de tomar a decisão de qual o método mais apropriado para o conjunto de seqüências do qual dispõe.

Assumindo que temos um conjunto adequado de seqüências e uma OF biologicamente perfeita, a computação de um alinhamento múltiplo matematicamente ótimo é ainda uma tarefa complexa. O custo computacional de um método exato de MSA é tão elevado que inviabiliza seu uso para conjuntos de dados reais [117, 231]. Desta forma, os algoritmos para computação de MSAs utilizados na prática são heurísticas que não garantem um alinhamento ótimo como resultado.

No texto que segue a proposta de doutorado é apresentada. Na Seção 2 são expostos alguns conceitos básicos para MSA. As abordagens, algoritmos e ferramentas relacionadas a MSA são descritas na Seção 3. A Seção 4 apresenta como são feitas as avaliações das soluções, assim como expõe algumas comparações entre os métodos. O projeto é detalhado na Seção 5 e, finalmente, a Seção 6 enumera as tarefas planejadas para o doutorado e os respectivos períodos de execução.

2 Conceitos básicos

Duas seqüencias são chamadas homólogas quando derivam de um ancestral comum. Geralmente, homologia é inferida por similaridade de seqüências, porém, similaridade não reflete, necessariamente, homologia. O acaso ou convergência evolucionária [57], por exemplo, podem levar a similaridade entre fragmentos curtos de seqüências. É possível também haver similaridade em trechos longos de seqüências como em repetições CA em moléculas de DNA ou domínios ricos em prolina em proteínas [236]. Geralmente, similaridades entre tais seqüências com "regiões de baixa complexidade" não refletem relacionamento evolucionário, mas na ausência de tais composições tendenciosas similaridade em um região extensa freqüentemente implica em homologia. Testes estatísticos podem ser usados para avaliar as chances de uma similaridade observada ocorrer ao acaso e, assim, aceitar ou não a homologia hipotética [8].

Buscar pelo melhor alinhamento consiste de buscar por aquele que aparente um melhor cenário evolucionário. Assim, a probabilidade de ocorrência de diferentes eventos mutacionais durante a evolução deve ser levada em consideração quando computando um alinhamento múltiplo. Em alinhamentos, tipicamente três tipos de mutações são consideradas: substituições, inserções ou remoções. Os dois últimos eventos são freqüentemente indistingüíveis e são comumente referenciados como *indels*. A probabilidade de substituição de um aminoácido por outro depende da estrutura do código genético (por exemplo: o número de mutações necessárias para transformar um *codon* em outro) e também do efeito fenotípico da mutação. Substituições de um aminoácido por outro com propriedades bioquímicas similares geralmente não têm grande efeito na estrutura e conseqüentemente na função da proteína. Assim, durante a evolução, substituições conservativas são relativamente freqüentes se comparadas a outras substituições.

É importante notar que a probabilidade de substituição de um aminoácido por outro depende da distância evolucionária entre as seqüências. Vários métodos foram propostos para a construção de séries de matrizes que estimam a probabilidade de todas as possíveis substituições para diferentes distâncias evolucionárias [50,74,95,193]. As mais utilizadas são as matrizes PAM [50,193] e BLOSUM [95]. Geralmente os alinhadores permitem ao usuário

selecionar a matriz de substituição a utilizar. Probabilidades de substituição também variam ao longo das seqüências de acordo com ambientes locais de aminoácidos na proteína dobrada (por exemplo, α-hélice ou folha-β). Matrizes de substituição de ambiente específico foram desenvolvidas [171], porém são raramente utilizadas nas ferramentas disponíveis atualmente. Em seqüências de DNA, probabilidades de susbtituições variam de acordo com as bases. Transições, trocas de bases de uma mesma classe (púricas ou pirimídicas), são geralmente mais freqüentes que transversões, trocas de bases de classes distintas. Assim, alinhadores múltiplos geralmente propõem um parâmetro de peso mais rígido para transversões que transições. Probabilidades de substituições de nucleotídeos também dependem das bases vizinhas. Por exemplo, em vertebrados, C no dinucleotídeo CG é extremamente mutável [19, 101]. Os alinhadores correntes ainda não fazem uso de tal informação.

A probabilidade de ocorrência de um *indel* depende do tamanho. Assim, quando computando um alinhamento, freqüentemente estima-se penalidades (p) associadas com *gaps* usando um modelo linear ou "afim" tal como $p = a + bL$, onde L é o tamanho do *gap*, a a penalidade para abertura de *gap* e b a penalidade para extensão de *gap*. Entretanto, análises de alinhamentos têm mostrado que este modelo subestima a probabilidade de *indels* longos [74, 75, 89]. Penalidades mais realísticas para *indels* podem ser estimadas com modelos tais como $p = a + b \times log(L)$ [59]. Devido à complexidade computacional, tais modelos não têm sido implementados nos alinhadores comumente utilizados. Felizmente outras abordagens para alinhar seqüências com grandes *indels* foram propostas, como poderá ser visto na Seção 3.4, onde apresentamos os modelos baseados em blocos. Em proteínas a probabilidade de ocorrência de *indels* também depende do grau de divergência entre as seqüências [74, 75] e das composições da molécula. Por exemplo, *indels* são mais freqüentes em laços externos que no núcleo da estrutura. É importante salientar que, na maioria dos programas, parâmetros padrões para penalidades de *gap* têm sido definidos para proteínas globulares típicas, que podem não ser os melhores para outras seqüências [59].

Considere que uma seqüência é um conjunto ordenado de letras de um alfabeto Σ. Um alinhamento de n seqüências $S_1, S_2, \ldots, S_n$ pode ser definido como uma matriz $A = a(S_1, S_2, \ldots, S_n)$, onde cada entrada A_{ij} é uma uma letra de Σ ou um símbolo nulo (freqüentemente representado por "-"). A linha i de A é a seqüência S_i depois dos *gaps* removidos. Em princípio, a pontuação de um alinhamento múltiplo deve refletir o grau de semelhança de suas seqüências, de acordo com um modelo evolucionário dado. Há diversas formas de mensurar a pontuação (ou custo) de um alinhamento. No modelo mais simples, a pontuação de um alinhamento de n seqüências é definido como a soma das pontuações de suas colunas. Um defeito deste modelo é considerar cada coluna do alinhamento independentemente de seu contexto. Por exemplo, um *gap* de tamanho L é considerado o mesmo que L *indels* independentes.

Em modelos mais realísticos, um *gap* é considerado um único evento mutacional e associado com uma penalidade proporcional ao seu tamanho. Em tais modelos, pontuação para alinhamento de par é definida como a soma das pontuações de substituições e penalidades para *gaps*. Para alinhamentos múltiplos, uma solução bastante utilizada, conhecida como Soma dos Pares (SP) [43], consiste em calcular a pontuação do alinhamento múltiplo a partir da soma das pontuações das projeções de $n(n-1)/2$ alinhamentos de pares, desprezando-se os alinhamentos de dois *gaps*. Pontuação SP simples, entretanto, pode ser inapropriada quando alguns grupos de seqüências são extremamente ou pouco represen-

tados em uma família. Este defeito pode ser corrigido pela introdução de um sistema de pesos apropriado [7, 81], que associa um peso a cada seqüência. Assim, pode-se dar um peso menor a seqüências de grupos super-representados. Outra solução consistem em usar uma função de pontuação baseada em uma árvore evolucionária. As folhas da árvore são as seqüências que queremos alinhar, e os nós internos são suas seqüências ancestrais hipotéticas. Apesar de parecer uma função melhor que a SP, o problema de alinhamento usando árvore evolucionária é também difícil [114].

3 Alinhamento múltiplo de seqüências

Na tentativa de computar alinhamentos múltiplos exatos, ou muito próximos a eles, Carillo e Lipman [43] propuseram um algoritmo *branch and bound* para computar um alinhamento de pontuação SP máxima. Este faz uso de um limiar superior para a pontuação e, assim, impõe limites para o espaço e tempo. Esta abordagem foi implementada no alinhador MSA [136]. Observe que, desta forma, já não garante um resultado ótimo. Stoye e colegas [206] descreveram um novo algoritmo por divisão e conquista, chamado DCA, que estendeu as capacidades do MSA original. O algoritmo DCA fragmenta as seqüências em segmentos que são pequenos o suficiente para utilizar o MSA. Depois, resta ao DCA a tarefa de montar o alinhamento a partir dos sub-alinhamentos gerados pelo MSA. Testes no BAliBASE [162, 217] indicaram que a estratégia DCA produz resultados ligeiramente melhores que o Clustal W [162]. É importante salientar, entretanto, que limitações ainda existem e que o DCA não foi capaz de realizar o alinhamento de quatro conjuntos de testes do BAliBASE, exatamente por estas limitações. Posteriormente, uma implementação iterativa do DCA, chamada OMA [181], foi descrita. Com ela pretende-se tornar a estratégia DCA mais rápida e reduzir os requisitos de memória. Ainda assim continuamos a ter fortes limitações, apesar de amenizadas, para o conjunto de seqüências de entrada.

Computar MSAs já foi provado ser um problema NP-Difícil [117, 231]. Desta forma, a construção de MSAs exatos - mesmo que utilizando soluções como MSA, DCA ou OMA - fica bastante limitada no que se refere ao tamanho da entrada (número e comprimento das seqüências). Sendo assim, algoritmos de aproximação e diversas heurísticas, usando as mais diversas abordagens, têm sido descritos para o problema de construção de MSAs.

O primeiro algoritmo de aproximação foi proposto por Gusfield [91] e tem um fator de aproximação $2-2/n$, onde n é o número de seqüências. Pevzner [178] melhorou o algoritmo e alcançou um fator de $2-3/n$. Posteriormente, Bafna, Lawler e Pevzner [16] apresentaram um algoritmo com fator $2-l/n$, para qualquer $l < n$.

Dentre os algoritmos heurísticos, destacam-se as seguintes classes: progressivos, iterativos e os métodos baseados em consenso, modelos, consistência ou blocos. As soluções não pertencem necessariamente a uma única classe. Embora não hajam garantias, observa-se que os resultados são muito úteis na prática e freqüentemente estão muito próximos da solução exata [59].

Nos últimos anos houve uma grande evolução da área com a introdução de diversos novos algoritmos e novos métodos de avaliação dos algoritmos. Nesta evolução destaca-se o crescente interesse em estratégias iterativas e uso de esquemas de pontuação baseados em consistência [162]. Outro ponto importante em MSA nos trabalhos recentes são métodos baseados em HMM [23,93]. Para maiores informações acerca de métodos HMM para MSA,

consulte o livro de Durbin e colegas [58].

Na Seção 3.1 apresentamos a abordagem progressiva. A abordagem iterativa é descrita na Seção 3.2. A Seção 3.3 aborda os métodos baseados em consistência. Já os métodos baseados em consenso, modelos e blocos são tratados na Seção 3.4. A Seção 3.5 expõe trabalhos brasileiros acerca de alinhamento múltiplo. Finalmente, comentamos sobre formatos de arquivos e ferramentas para visualização e edição de MSAs na Seção 3.6.

3.1 Algoritmos progressivos

Alinhamento progressivo é uma das maneiras mais simples e efetivas para a realização de alinhamentos múltiplos com um pequeno requisito de tempo e memória. Esta abordagem foi inicialmente descrita por Hogeweg e Hesper [105] e depois reinventada por Feng e Doolittle [67] e Taylor [209]. Diversos pacotes para MSA são baseados nesta abordagem, dentre eles: Pileup (componente do pacote GCG [52]), MultiAlign [48] e Clustal W [215]. Esta abordagem oferece um bom desempenho quando as seqüências são homólogas e relativamente bem conservadas [67, 209]. O principal problema da abordagem progressiva é a sua natureza gulosa, onde qualquer erro cometido em etapas iniciais do processo não são mais corrigidos [59]. Estratégias de otimização iterativas têm sido propostas para resolver tal problema, tais como: RIW ou DNR [82]. Em testes, estes métodos apresentaram melhores desempenhos que o Clustal W, por exemplo. Porém, mesmo sendo mais rápido que algoritmos ótimos, estes são ainda muito lentos para entradas grandes.

Alinhamento progressivo consiste em construir um alinhamento múltiplo a partir de alinhamentos de pares. Nesta abordagem é possível identificar três passos: computação dos alinhamentos de pares, construção da árvore guia a partir das distâncias de pares e construção do alinhamento múltiplo guiado pela árvore. As diferenças entre as ferramentas para MSA que fazem uso da abordagem progressiva estão nos métodos que utilizam em cada um dos três passos.

Clustal W [215], por exemplo, permite o uso de programação dinâmica ou métodos heurísticos para a computação do alinhamento de pares. Com programação dinâmica temos resultados mais precisos, mas com os métodos heurísticos é possível ganhar em velocidade de processamento. Para a construção da árvore guia há diversos métodos, dentre eles tem-se o UPGMA [201] e o Neighbor-Joining [186]. É conhecido um problema no UPGMA, onde é dada uma ordem de ramificação incorreta quando as taxas de substituição variam em diferentes linhagens. Este problema levou a utilização de Neighbor-Joining, ao invés de UPGMA, nas versões mais recentes do Clustal [59]. No terceiro passo é preciso, inicialmente, determinar um método para a seleção do par de seqüências (ou alinhamentos) a agrupar. O principal problema deste passo, entretanto, consiste no alinhamento de dois alinhamentos. O método mais simples é reduzir cada alinhamento a uma seqüência consenso e usar um algoritmo de alinhamento de pares comum. Outro método, utilizado pela maioria dos programas, é representar alinhamento com perfis, onde colunas são reduzidas a distribuições das freqüências de cada letra. Alinham-se dois perfis de forma similar a de duas seqüências por programação dinâmica.

O esquema de pontuação é o componente de maior influência nos resultados dos algoritmos progressivos. Tais esquemas podem ser divididos em duas categorias: baseados em matriz ou baseados em consistência. Os algoritmos baseados em matriz usam uma matriz

de substituição para avaliar o custo de relacionar dois símbolos. São exemplo de algoritmos baseados em matriz: Clustal W [215], MUSCLE [62] e Kalign [130]. Os algoritmos baseados em consistência incorporam um maior conjunto de informações na avaliação. Inicialmente tal método foi utilizado no T-COFFEE [165], inspirado no DiAlign [152]. Depois surgiram outros algoritmos que utilizam a mesma idéia. O PCMA [177] reduz os requisitos computacionais exigidos pelo T-COFFEE. O ProbCons [54] adiciona o uso de consistência Bayesiana e HMM. MUMMALS [175] combina o esquema de pontuação do ProbCons com a estratégia do PCMA. Outro exemplo ainda de algoritmo baseado em consistência é o MAFFT [121], que utilizando transformada rápida de Fourier para gerar uma árvore guia e, assim, gera alinhamentos precisos [122] em um tempo curto. Estudos têem indicado que os métodos baseados em consistência são mais precisos que os métodos baseados em matriz, mas eles geralmente requerem um maior tempo de processamento.

MUSCLE [62, 63] é um novo pacote de alinhamento progressivo que é extremamente rápido e preciso. Seu primeiro passo é gerar um esboço de um alinhamento através de uma árvore guia imatura. As distâncias entre os pares de seqüências de entrada são estimadas usando contagem k-mer para um alfabeto restrito [61] e, juntamente com um algoritmo de agrupamento, dão origem à árvore guia inicial. MUSCLE implementa a pontuação LE (do inglês, *log expectation*) para alinhar perfis durante o alinhamento progressivo. Uma vez gerado o MSA inicial, passa-se ao refinamento do alinhamento pela geração de uma árvore guia mais precisa, baseando-se no alinhamento inicial. LE tem apresentado bons resultados em buscas por homologia [221]. Nas versões mais recentes do MUSCLE foi adicionado um refinamento iterativo utilizado pelo ProbCons.

Dentre os algoritmos progressivos, destaca-se o Clustal W [215], que inclui uma variedade de heurísticas altamente especializadas destinadas a máxima exploração de informações acerca das seqüências. Com isso, diferencia-se da maioria dos outros alinhadores progressivos por prover penalidade de *gap* local, escolha automática da matriz de substituição, ajuste automático de penalidade para *gap* e retardo do alinhamento de seqüência com baixo grau de relacionamento.

3.2 Algoritmos iterativos

Os algoritmos iterativos dependem de um algoritmo capaz de produzir alinhamentos iniciais. Cabe a eles a tarefa de refinar tais alinhamentos através de uma série de ciclos até que não seja mais possível incrementar a qualidade do alinhamento. Métodos iterativos podem ser determinísticos ou estocásticos. Os métodos mais simples são determinísticos. Sua estratégia implica em extrair seqüências, uma por vez, de um alinhamento múltiplo e então realinhá-las novamente ao restante das seqüências [25, 82]. Métodos iterativos estocásticos incluem HMM [127], *simulated annealing* (SA) [125] e algoritmos genéticos (GA, do inglês genetic algorithm) [12, 41, 45, 77, 164, 237]. A principal vantagem destes é permitir uma boa separação conceitual entre processo de otimização e OF. É possível a existência de métodos que usam estratégias mistas, progressiva e iterativa ao mesmo tempo [97].

O primeiro algoritmo iterativo não estocástico data da origem de MSAs [25]. Sua estratégia consiste em realinhamentos para corrigir possíveis erros em estágios anteriores, para tanto faz uso de algoritmos padrões de alinhamento com programação dinâmica [158]. O procedimento é encerrado quando as iterações falham consistentemente na tentativa de

melhorar o alinhamento. Este é o procedimento utilizado pela maioria das estratégias descritas até o início da década de 1990. A principal variação nestes algoritmos encontra-se na forma como as seqüências são divididas em dois grupos antes de serem realinhadas. Por exemplo, em AMPS [25] as seqüências são escolhidas de acordo com sua ordem de entrada e realinhadas uma a uma. Já no algoritmo de Berger e Munsen [32] a escolha é feita de maneira aleatória e as seqüências são divididas em dois grupos que podem conter mais que uma seqüência. A introdução da aleatoriedade na seleção dos grupos torna o algoritmo mais robusto e incrementa sua precisão. Poucos destes métodos iterativos iniciais têm sido efetivamente avaliados por ferramentas de *benchmark*, o que torna difícil uma estimativa de suas verdadeiras significâncias biológicas.

O mais sofisticado algoritmo iterativo baseado em programação dinâmica foi descrito por Gotoh [82] e é conhecido por PRRP. Trata-se de uma estratégia iterativa de duplo aninhamento com aleatorização que otimiza uma pontuação de soma-de-pares com peso e função afim para penalidade de *gap*. Sua originalidade está na otimização simultânea de pesos e alinhamento. A iteração interna otimiza a soma-de-pares com peso enquanto a iteração externa otimiza os pesos que são calculados em uma árvore filogenética estimada do alinhamento corrente [7]. O algoritmo é encerrado quando os pesos convergem. PRRP foi o primeiro programa para alinhamento múltiplo a ser extensivamente avaliado por *benchmark*, usando JOY [148], um banco de dados de alinhamentos estruturais. Posteriormente os resultados foram confirmados no BAliBASE [165, 218]. PRRP apresenta resultados significativamente melhores que a maioria dos métodos progressivos e iterativos [162].

SA [146] foi o primeiro método iterativo estocástico descrito para alinhamento múltiplo. Vários esquemas foram publicados [111, 125] e todos envolvem a mesma cadeia de processos: modifica aleatoriamente um alinhamento, calcula a pontuação, decide se mantém ou descarta a modificação de acordo com a função de aceitação que torna-se mais exigente com o aumento no número de iterações. O procedimento é repetido até que seja satisfeito um critério de término. É importante observar que SA tem se mostrado pouco eficiente, em termos de tempo, para a construção de alinhamentos *ab initio*, mas tem se mostrado uma boa ferramenta para refino de alinhamentos. GAs [106] constituem uma interessante alternativa aos SAs. SAGA [164], assim como SA, é uma caixa preta de otimização na qual qualquer OF pode ser testada. Seu esquema é direto e segue a risca GA simples [73]. SAGA usa a OF definida como sua função de aptidão, suas mutações inserem ou deslocam *gaps* e cruzamentos combinam o conteúdo de dois alinhamentos. Ao todo são 20 operadores competindo pelo uso. Testes têm mostrado que é capaz de produzir resultados próximos ou até melhores que o MSA (usando a mesma OF). Os testes também mostraram que GAs são lentos para alinhamentos múltiplos, no uso do dia-a-dia. SAGA foi paralelizado por dois grupos independentemente [12, 167] em busca de eficiência e novos métodos foram implementados baseando-se em princípios descritos no SAGA [41, 45, 77]. Zhang e Wong [237] implementaram uma nova ferramenta que faz uso de GAs. Foi observado um alto grau de eficiência da ferramenta, mas deve-se notar que o método é dirigido pela presença de segmentos completamente conservados para guiar a montagem do alinhamento, algo nem sempre realístico.

O pacote Gibbs Sampler [131] faz uso de *Gibbs Sampling*, que é uma outra interessante estratégia iterativa estocástica. Ele implementa um método de alinhamento local que encontra *motifs* sem *gaps* ao longo de um conjunto de seqüências não alinhadas. Da perspectiva

de alinhamento múltiplo, sua característica mais interessante está em sua OF. O algoritmo objetiva construir um alinhamento com um bom *P-value* (por exemplo, baixa probabilidade de ter sido gerado por acaso). A cada iteração, adicionam-se ou removem-se segmentos de acordo com a probabilidade do modelo corrente poder gerar eles. Se aquela probabilidade é suficientemente alta, o modelo é atualizado com os novos segmentos e o algoritmo passa a próxima iteração.

Dentre as estratégias iterativas estocásticas ainda destaca-se HMM, que tenta simultaneamente maximizar os dados e o modelo através de um modelo Bayesiano [23,93]. HMMs, que podem ser treinadas por maximização de expectativa [60, 127], também apresentaram resultados decepcionantes para a montagem de alinhamentos *ab initio* [162]. Hoje, HMMs tais como aqueles encontrados no Pfam [27] não são mais gerados a partir de seqüências não alinhadas. Os métodos, agora, estão mais inclinados a transformar um alinhamento pré-computado em um HMM e então submetê-lo a um refinamento usando HMMER [127] ou SAM [60].

Dois outros métodos de alinhamento iterativo foram recentemente descritos: PRALINE [97] e IterAlign [37], que compartilham protocolos muito similares. Quanto a suas potenciais performances, nenhum deles foi adequadamente avaliado. Todavia deve ser dado ênfase a novos conceitos que eles incorporaram: uso de informação local no IterAlign, que objetiva um decremento na sensibilidade quanto a parametrização de penalidade de *gap*; e o conceito de consistência. A busca por consistência tem se tornado um dos pontos mais fortes no desenvolvimentos recentes em MSA. Tal conceito é, também, central nos métodos não-iterativos.

3.3 Métodos baseados em consistência

O primeiro método MSA baseado em consistência foi descrito por Kececioglu na década de 1990 [124]. Dado um conjunto de seqüências, o MSA ótimo é definido como aquele que está de acordo com a maioria de todos os possíveis alinhamentos ótimo de pares. Há, pelo menos, três boas razões que fazem OFs baseadas em consistência muito interessantes, são elas: não dependem de uma matriz de substituição específica, mas de qualquer método, ou coleção de métodos, capaz de alinhar duas seqüências por vez; o esquema baseado em consistência é dependente de posição; e experiências mostraram que dado um conjunto de observações independentes, a maioria dos consistentes estão freqüentemente próximos da verdade [59]. Embora a primeira OF baseada em consistência tenha sido descrita em 1993, passaram-se ainda alguns anos para o desenvolvimento de algoritmos heurísticos capazes de tratar sua otimização. Um GA (SAGA [164]) foi usado para mostrar os avanços biológicos de tal função, denominada COFFEE [166], que emula o problema do *trace* de peso máximo. No SAGA-COFFEE, a coleção de alinhamentos de pares com peso é nomeada uma biblioteca e o SAGA é utilizado para computar o alinhamento que tem o mais alto nível de consistência com a biblioteca. Na prática, a biblioteca pode conter mais que um alinhamento para cada par de seqüências. A informação que ela contém pode ser redundante, conflitante e pode se originar de fontes que variam tanto quanto se desejar (análise de estrutura, comparação de seqüências, busca em bases de dados, conhecimento experimental, etc.).

O SAGA-COFFEE obteve resultados interessantes, mas o problema do desempenho ainda persistia. Isso levou ao desenvolvimento de um novo algoritmo heurístico para

otimizar a função COFFEE de maneira que fosse eficiente em termos de tempo, o T-COFFEE [165]. Neste a biblioteca COFFEE é transformada na biblioteca estendida, uma matriz de substituição para posição específica onde a pontuação associada a cada par de resíduos depende da compatibilidade daquele par com o resto da biblioteca. T-COFFEE faz uso de um procedimento reminiscente da multiplicação de matrizes de pontos de Vingron [224] e da sobreposição de pesos de Morgenstern [153]. O alinhamento múltiplo é montado usando um algoritmo de alinhamento progressivo semelhante ao Clustal W. Uma característica importante do T-COFFEE é que a biblioteca primária é criada de uma combinação de alinhamentos globais (produzidos pelo Clustal W) e alinhamentos locais (produzidos pelo Lalign). Testes realizados através do BAliBASE mostraram que a combinação de informação global e local permite ao T-COFFEE superar o PRRP, Clustal W e DiAlign nas cinco categorias de conjuntos de testes contidos neste banco de dados de referência [165]. T-COFFEE é de grande interesse não apenas por causa da forma como permite que dados heterogêneos sejam combinados em alinhamentos, como pode ser observado em seu uso no 3D-COFFEE [169], mas também como um precursor para o método baseado em probabilidade do ProbCons [54].

ProbCons é atualmente o método MSA mais preciso, conforme avaliação realizada pelo BAliBASE [227]. Ele obtém os melhores resultados em todos os cinco conjuntos de referência providos pela ferramenta de avaliação. De uma forma geral, ProbCons funciona de forma semelhante ao T-COFFEE, mas usa probabilidade ao invés da heurística para pesos de pares de resíduos. Para tanto usa HMM de pares de seqüências gerados por uma função objetivo de máxima precisão esperada [108]. Faz uso de um algoritmo de alinhamento progressivo, seguido de um procedimento iterativo de refinamento.

3.4 Métodos baseados em consenso, modelos e blocos

M-COFFEE [228] implementa um meta-método consenso baseado no T-COFFEE. Inicialmente são utilizados diversos métodos MSA para calcular alinhamentos alternativos. Então utiliza-se o T-COFFEE para combiná-los em um MSA final que seja consistente com os alinhamentos de entrada. Uma vez gerados os alinhamentos pelos métodos, o procedimento de combinação destes MSAs realizado pelo M-COFFEE executa mais rápido que qualquer outro método conhecido para construção de MSAs. Desconsiderando os algoritmos meta-métodos, o ProbCons é o mais rápido alinhador múltiplo [228]. É importante salientar, entretanto, que o M-COFFEE enfrenta problemas a medida que as seqüências se tornam menos homólogas [163].

Para tentar resolver os problemas decorrentes de seqüências de entrada com baixa homologia, uma alternativa apontada é incorporar informações relativas às seqüências. Seguindo esta abordagem temos os métodos de alinhamento baseados em modelos [14], que podem ser implementados com extensão estrutural ou extensão por homologia. Extensão estrutural foi inicialmente descrita por Taylor [208], onde tenta-se identificar um modelo estrutural, no Protein Data Bank [33], para cada seqüência usando BLAST [9]. Então alinham-se modelos usando um método para superposição de estruturas e mapeam-se as seqüências originais em seus alinhamentos de modelos. Tais resultados compõem uma biblioteca primária, que é enviada a um método baseado em consistência para calcular o resultado final.

3D-COFFEE [169] foi projetado para criar alinhamentos de seqüências de proteína que

incorporam informação de estruturas tridimensionais, quando estas existem. Em um conjunto de seqüências de entrada, é comum encontrar entradas PDB [33] para uma ou mais destas seqüências. O objetivo de utilizar as estruturas tridimensionais é facilitar o alinhamento de seqüências com relacionamentos distantes, pois elementos estruturais são geralmente mais conservados que seqüências primárias e assim permitem um bom alinhamento mesmo na *twilight zone* (menos de 25% de identidade). Na prática, usar este tipo de informação pode ser complicado, mas há trabalhos descrevendo seu uso [4]. 3D-COFFEE é um método rápido, simples e preciso que explora a habilidade do T-COFFEE em incorporar informações heterogêneas para adicionar dados estruturais no alinhamento e, assim, melhorar sua precisão, mesmo quando não se dispõe de todas as estruturas. Quando é dada apenas uma estrutura, o 3D-COFFEE combina cada seqüência com a estrutura usando uma ferramenta externa (FUGUE [195]) e converte a saída para um alinhamento de duas seqüências. Desta forma temos um conjunto de alinhamentos de pares que indicam como as seqüências alinham com a estrutura e, indiretamente, como cada seqüência se alinha a cada outra. Caso duas ou mais estruturas estejam disponíveis, é possível usar um pacote de superposição de estrutura completa (SAP [211]). Apesar de usar FUGUE e SAP por padrão, o 3D-COFFEE permite o uso de outras ferramentas do gênero.

A extensão por homologia foi originalmente aplicada no pacote DbClustal [220] e trabalha de forma semelhante. A diferença está no fato de usar um perfil (construído com PSI-BLAST [11], no caso do DbClustal) ao invés da estrutura. Outros pacotes que fazem alinhamento baseado em modelos são: Expresso [14], PROMALS [176], PRALINE [97], SPEM [241] e T-Lara [28]. A maioria dos métodos baseados em modelos são baseados em consistência. PRALINE e SPEM são exceções. Eles usam a abordagem progressiva. PROMALS, que implementa uma extensão por homologia, tem mostrado um forte desempenho nos *benchmarks* mais recentes. Isso sugerem um bom potencial para a abordagem de extensão por homologia.

Outra abordagem para computação de MSAs é a utilização de blocos conservados (vistos como âncoras) para guiar o alinhamento que, desta forma, não irá depender tanto dos parâmetros para penalidades de *gaps* quando as seqüências compartilharem blocos conservados separados por regiões não conservadas contendo longos *indels*. Blocos são alinhamentos de fragmentos de seqüências (alinhamentos locais). A maioria dos métodos que fazem uso desta abordagem consideram apenas blocos sem *gaps*. Os blocos permitidos podem ser exatos (composto por segmentos idênticos) ou não exatos e podem ser uniformes (encontrado em todas as seqüências) ou não. O primeiro programa de alinhamento múltiplo de blocos [202] usou um algoritmo de ordenação para computar blocos exatos uniformes. Algoritmos mais rápidos, baseados em árvores de sufixo [143], ou estruturas de dados equivalentes, podem também ser utilizados para computar blocos exatos. ASSEMBLE [224] realiza uma análise em uma matriz de pontos em todos os pares de seqüências e então compara estas matrizes de pontos para encontrar blocos uniformes, não necessariamente exatos. Na prática, é comum alguns blocos que não estão presentes em todas as seqüências. Melhorias têm consistido no desenvolvimento de métodos que permitam blocos que não sejam necessariamente uniformes.

No contexto dos métodos baseados em blocos, a ferramenta DiAlign é a de maior destaque. Para computar um "bom" conjunto consistente de diagonais (numa matriz de pontos), faz uso de um algoritmo heurístico, que adiciona diagonais pela ordem decrescente

de pontuação em um conjunto consistente de diagonais. Aquelas diagonais que não sejam consistentes com o conjunto corrente são rejeitadas.

3.5 Soluções brasileiras

Há também algumas soluções brasileiras para MSA, dentre elas destacam-se um trabalho de Meidanis e Setubal (Unicamp) [145], um de Lopes e Moritz (UTFPR) [138] e outro de Santos e Vieira (UFAL) [189]. Lopes (UTFPR) ainda participou da elaboração de uma solução para MSA que faz uso de processamento paralelo [137] e outra que faz uso de computação reconfigurável [135]. Adi e Ferreira (USP) [3] apresentaram um algoritmo para predição de genes, que faz uso de alinhamento múltiplo.

Meidanis e Setubal [145] apresentaram em 1995 uma abordagem heurística para computação de MSAs que faz uso de grafos orientados acíclicos para representar alinhamentos. Com tais grafos, denominados TLGs (do inglês, *Trace Layout Graphs*), eles conseguem codificar diversos alinhamentos ótimos com uma estrutura compacta. Neste trabalho apresenta-se como, a partir de alinhamentos de pares construídos com um algoritmo de programação dinâmica padrão para alinhamento global [158], constrói-se TLGs de pares de seqüências usando apenas os alinhamentos ótimos *upmost* e *downmost*. Depois descreve-se como juntar estes TLGs de pares em um só e como otimizar esta estrutura. No final detalha o algoritmo de alinhamento que, neste caso, recebe um único TLG como entrada e gera o alinhamento associando colunas aos caracteres e, conseqüentemente, posicionando os *gaps*. A abordagem utilizada neste trabalho deriva da noção de *trace* introduzida por Sankoff e Kruskal [188] e posteriormente generalizada para seqüências múltiplas por Kececioglu [123]. Não há registros de uma avaliação comparativa deste algoritmo com outras soluções.

Lopes e Moritz [138] apresentaram em 2006 uma solução para alinhamento múltiplo que faz uso de algoritmos genéticos e representa um alinhamento através de grafos orientados multidimensionais, reduzindo os requisitos de memória. Indivíduos no GA representam um caminho na matriz n-dimensional, na qual cada dimensão corresponde a uma seqüência. Um caminho válido sempre inicia na célula correspondente ao primeiro aminoácido de todas as seqüências e acaba na célula correspondente ao último aminoácido de todas as seqüências. Para representar o grafo usa-se um cromossomo de tamanho variável, composto por m genes, cujo número será no mínimo eqüivalente ao tamanho da maior das seqüências e no máximo a soma dos tamanhos da seqüências decrescido de um. Cada gene é composto de dígitos binários, que apontam para a direção da próxima célula e que, desta forma, indicam como os aminoácidos, associados às células de origem e destino, são alinhados. O número de dígitos nos genes será diretamente proporcional ao número de seqüências. Para avaliar uma solução candidata, decodifica-se o cromossomo em um MSA, calcula-se a pontuação e normaliza-se este valor. São definidas três operações (cruzamento, mutação aleatória e mutação dinâmica), que foram implementadas de forma a manter a validade do caminho. O algoritmo precisa ainda de uma melhor avaliação, principalmente comparando com outras soluções que façam uso de GAs (como o SAGA), e evoluções. Ele só foi comparado com o Clustal W usando BAliBASE e os resultados foram, no geral, piores.

Em sua dissertação de mestrado, Santos [189] também apresentou uma solução que faz uso de algoritmos genéticos. Ela descreveu uma instanciação de um algoritmo genético baseado em tipos abstratos de dados para MSA, de acordo com as definições apresentadas

por Roberta Vieira (orientadora do mestrado) em sua tese de doutorado [222]. Nesta solução, cromossomos representam possíveis soluções e são compostos por genes, que por sua vez são compostos por bases. Uma base é um par composto por um aminoácido (ou um *gap*) e um número natural. As bases de uma coluna do alinhamento (ou cromossomo) compõem um gene. Na montagem de um cromossomo cada seqüência é alocada em uma linha e são inseridos *gaps* para deixar todas as linhas com o mesmo comprimento. A *gaps* sempre há o natural 0 associado. A aminoácidos são associados números naturais maiores que 0, que indicam a posição do aminoácido na seqüência original (sem *gaps*). Tal número é utilizado pelos operadores para não permitir que a ordem dos aminoácidos em uma seqüência seja alterada. O grau de adaptação de um cromossomo é dado por uma função de soma dos pares simples que atribui o mesmo peso para abertura e extensão de *gaps* e usa a matriz Dayhoff para determinar pesos para *matches* e *mismatches*. A população inicial, com 100 indivíduos, é gerada aleatoriamente e no decorrer das gerações tem no máximo 200 indivíduos. A cada iteração são selecionados os cromossomos, que serão preservados para a próxima geração, usando como critério um ponto de corte. São mantidos aqueles que possuem um grau de adaptação maior que a média da população. Os selecionados, então, irão compor o conjunto que será submetido a operações de cruzamento. Para tanto subdivide-se este conjunto em dois conjuntos sem intersecção. Cruzamentos são feitos sempre entre cromossomos de conjuntos distintos e preservam-se apenas os cromossomos resultantes de cruzamento que possuam um grau de adaptação maior que média da população corrente. Para operações de mutação são selecionados cromossomos da população corrente que foram descartados pelo ponto de corte. Estes só irão para a próxima geração se melhorarem o grau de adaptação com a mutação. Fez-se apenas uma avaliação da solução com 5 conjuntos de referência da mesma classe do BAliBASE e os resultados foram inferiores às outras soluções em quase todas as situações.

3.6 Formatos de arquivo e ferramentas para visualização e edição de MSAs

Resultados de alinhadores múltiplo são geralmente salvos em arquivos texto. Não há um formato padrão para MSA. Entretanto, o formato MSF é provido pelos mais populares programas de alinhamento e é reconhecido por muitos programas que requerem alinhamentos como entrada, tal como filogenia molecular ou buscas de perfis. O formato MASE apresenta a vantagem de permitir a inclusão de anotações do alinhamento como um todo ou específicas a cada seqüência.

Interfaces gráficas têm sido desenvolvidas para manipular e editar alinhamentos múltiplos. Geralmente permitem aos usuários colorir ou sombrear resíduos de acordo com vários critérios, tais como propriedades físico-químicas, grau de conservação no alinhamento, hidrofobicidade ou estrutura secundária. O uso das cores ajuda muito na interpretação do MSA, dando uma visão muito mais abrangente da informação contida em um alinhamento múltiplo. Além disso, estas interfaces propõem diversos recursos interessantes, tais como verificação ou refino manual de alinhamentos, adição de anotações e extração de subalinhamentos. São ferramentas para visualização e edição de MSAs: Jalview, CINEMA, SeaView, MPSA e Se-Al. Há ainda o Clustal X, que adiciona uma interface ao Clustal W e permite uma visualização gráfica, mas não fornece ferramentas para a edição dos alinha-

mentos. Já para visualização e geração de figuras com os alinhamentos para impressão, há também uma série de ferramentas, tais como: BoxShade, WebLogo, MView e AMAS.

4 Avaliação dos algoritmos

Existem diversas ferramentas para *benchmark* em MSA na forma de bases de dados de alinhamentos pré-compilados aos quais os alinhamento gerados pelos métodos testados são comparados. Estas comparações são freqüentemente realizadas pelo cálculo da fração de colunas de resíduos alinhados que são idênticas em ambos os alinhamentos (teste e referência). Este método é conhecido como pontuação de coluna [120]. Outra forma comum de comparar os alinhamentos é através da SPS (do inglês, *Sum-of-Pairs Score*), onde a pontuação é incrementada sempre que duas bases forem alinhadas da mesma forma em ambos os alinhamentos [218]. Tal abordagem para *benchmark* é atrativa devido a sua simplicidade, mas deve-se tomar cuidado para, na busca por melhorar o alinhador, não parametrizá-lo de forma a apresentar bons resultados nos conjuntos de referência das ferramentas para *benchmark* e degradar a performance em situações gerais. Tais ferramentas para *benchmark* têm sido efetivas na evolução dos algoritmos, levando a alinhamentos estruturalmente corretos [163]. É importante salientar entretando que os *benchmarks* correntes têm pecado pela ausência de conjuntos de referência com grande número de seqüências, o que inviabiliza uma avaliação apropriada dos diversos métodos disponíveis quanto a suas capacidades diante de tal situação [163].

Dentre as soluções para *benchmark* de MSAs, BAliBASE [216] foi o primeiro construído com o propósito de *benchmarking* em larga escala e continua sendo regularmente utilizado hoje em dia [227]. Seus conjuntos de dados são sub-divididos em diversos conjuntos de referência manualmente refinados, sendo que cada um deles destina-se a avaliação de um problema específico em MSA (alinhamentos globais/locais de diferentes tamanhos e identidade de seqüências, longos *gaps* internos, etc.). O refino manual e a sub-divisão em conjuntos de referência são suas principais vantagens e grandes diferenciais, que o tornam a principal ferramenta para avaliação de alinhadores múltiplos utilizada pelo comunidade científica [227].

HOMSTRAD [149] compreende um conjunto de mais de 1000 alinhamentos, cada um deles baseado em uma família de proteínas em particular. É exclusivamente baseado em seqüências com estruturas tridimensionais e arquivos PDB conhecidos. Em cada entrada, um alinhamento estrutural das proteínas é automaticamente gerado. Pode-se utilizar estes alinhamentos para *benchmark*. Possui uma menor cobertura, se comparado ao BAliBASE. PREFAB [63] é também automaticamente gerado. Duas proteínas com estruturas conhecidas são estruturalmente alinhadas e suas seqüências são usadas para consultar um banco de dados, onde resultados com altas pontuações são selecionados. Os parâmetros de consulta e seus resultados são combinados e então alinhados usando o *software* que está sendo testado. A precisão é calculada apenas no par original, pela comparação com seu alinhamento/superposição estrutural.

O banco de dados SABmark [229] contém dois grandes sub-bancos de dados de alinhamentos de até 25 seqüências de entrada cada. Há 425 grupos representando famílias de proteína com identidade entre 25% e 50%. Há ainda 209 grupos para avaliação de *twilight zone*. Cada um destes grupos representa uma dobra de proteína definida pelo SCOP [155]

e as seqüências compartilham menos de 25% de identidade. Já a ferramenta IRMbase [207] constrói seu conjunto de referência implantando *motifs* gerados pelo ROSE [205] em seqüências aleatórias. Provê diversos grupos de alinhamento que variam no tamanho e no número de implantes.

Diferente de todos os métodos de avaliação citados, APDB [170] não recai na comparação de alinhamentos de referência pré-existentes. Ao invés disso, a qualidade é mensurada com base na superposição de estruturas PDB conhecidas, estas relativas às seqüências de entrada do alinhamento múltiplo de teste. Sua grande vantagem está na independência de alinhamento de referência e, assim, evita qualquer chance de definir métodos MSA otimizados para conjuntos específicos de referência. Sua grande desvantagem encontra-se na dependência da existência das entradas PDB.

Resultados de estudos recentes sugerem que os melhores métodos têm se tornado indistingüíveis, exceto em situações de homologia remota (menos que 25% de identidade). Infelizmente, homologias remotas são fracamente indicadas para a geração de alinhamentos de referência, pois suas sobreposições freqüentemente aceitam diversos alinhamentos alternativos com estruturas eqüivalentes [128]. Visando eliminar tal dificuldade na avaliação de um método, é possível comparar diretamente o alinhamento avaliado com alguma superposição 3D idealizada, que tem sido adotada por diversos autores [14, 170, 175].

Um problema, levantado recentemente, refere-se a suposição de que alinhamentos estruturalmente corretos são os melhores MSAs para modelar qualquer tipo de sinal biológico (evolução, homologia ou função) [163]. Um relatório acerca de construção de perfil [88] mostrou que alinhamentos estruturalmente corretos não resultavam necessariamente em melhores perfis. É necessário sistematicamente questionar e quantificar o relacionamento entre a precisão de MSAs e a relevância biológica de qualquer modelo oriundo deles.

Diversas análises comparativas de programas para alinhamento múltiplo têm sido publicadas [36, 82, 142, 153]. Estas comparações são baseadas na habilidade de detectar padrões de *motifs* em diversas famílias de proteínas ou baseadas em alinhamentos de referência derivados de estruturas tridimensionais de proteínas. Análises comparativas podem também ser baseadas no efeito dos programas de alinhamento múltiplo na filogenia. Quando as seqüências são similares (mais que 50% de identidade de pares de proteínas e mais de 70% para DNA) e são homólogas em seu tamanho completo, todos os métodos de alinhamento global produzem resultados bons. Além disso, em tais casos, qualquer conjunto razoável de parâmetros geram alinhamentos similares. Porém, quando pelo menos duas seqüências em uma dada família compartilham uma identidade menor, ou as regiões homólogas são interrompidas por *gaps* longos de tamanhos diferentes, o resultado do alinhamento pode variar consideravelmente de acordo com os programas e parâmetros utilizados [59].

Métodos progressivos enfrentam grandes dificuldades com *gaps* longos devido ao esquema linear para peso de *gap*, que tende a aplicar uma penalização excessiva a *indels* longos. Métodos globais baseados em blocos, como o DiAlign [152, 153] ou IterAlign [37], são menos sensíveis a esses *gaps* longos e são particularmente apropriados para tais situações [59]. Um outro defeito dos métodos progressivos, mas que também ocorre com a maioria dos métodos de alinhamento global, é que sempre produzem alinhamento mesmo quando as seqüências não são relacionadas. Já o DiAlign e IterAlign não tentam gerar um alinhamento global se as seqüências são apenas localmente relacionadas. Em testes de *benchmark*, realizados com o BAliBASE [218], observou-se que, em geral, o Clustal W

executa melhor quando a árvore filogenética é relativamente densa. Para ele não importa o quão longa são as seqüências, mas sim o quão similares elas são. Métodos globais são apropriados apenas se todos os blocos conservados são consistentes. Se alguns domínios estão duplicados ou ordenados diferentemente ao longo das seqüências é necessário usar um método de alinhamento local para alinhar todos os domínios relacionados.

Um grande obstáculo na construção de MSAs é o processamento de repetições, que causam confusão a todos os métodos globais existentes para MSA. No caso de existirem repetições nas seqüências de entrada, a única solução é o pré-processamento das seqüências com extração destes e apenas realizar o alinhamento de regiões homólogas [162]. Esta extração pode ser realizada por ferramentas de alinhamento múltiplo local tal como Gibbs Sampler [131], Mocca [161] ou Repro [99]. Infelizmente nenhuma delas está bem integrada com um procedimento de alinhamento múltiplo global. Dentre todos os métodos apresentados, não há um em particular que seja o melhor. Todos podem ser superados em determinadas situações. Por esta razão, mostra-se ser mais importante ter a capacidade de calcular o nível exato de precisão de um alinhamento ao invés de melhorar as performances médias de cada método [162]. Dentre os métodos tem-se conhecimento apenas de quatro que são capazes de realizar tal estimativa: Clustal X [214], PRALINE [97], T-COFFEE [165] e Match-Box [51]. Entretanto, é importante salientar que em nenhum dos casos estas estimativas foram apropriadamente avaliadas [162].

5 Projeto

Apesar de existirem diversas heurísticas para alinhamento múltiplo de seqüências, ainda há espaço para o desenvolvimento de novas que sejam capazes de gerar resultados de alta qualidade e que requisitem menos tempo e/ou memória para processamento. Pretendemos ao final do trabalho ter projetado, implementado e testado algoritmos para MSA, usando as mais diversas abordagens, que gerem bons alinhamentos do ponto de vista biológico e que sejam eficientes em termos de tempo e espaço. Dentre estes algoritmos estará, pelo menos, um que implemente um meta-método. Este tem como característica a capacidade de combinar alinhamentos gerados por outros métodos e, assim, gerar alinhamentos de maior qualidade que métodos individuais. A seguir apresentamos uma visão geral dos estudos que serão realizados no decorrer do projeto.

Numa etapa inicial do projeto, pretendemos desenvolver heurísticas para MSA utilizando a abordagem progressiva de Feng e Doolittle [67]. Por ser uma das abordagens mais exploradas para MSA deve facilitar nossa busca por métodos para sua implementação e, assim, nos ajudar a ganhar experiência e um melhor entendimento do problema. Além disso, nos estudos atuais em MSA tem-se dado muita atenção a abordagem progressiva devido a seus requisitos muito baixos para tempo. Neste período faremos um extenso estudo acerca de *gaps* e matrizes de pontuação tanto para seqüenciamento de DNAs como de proteínas.

A seguir iniciaremos estudos acerca de métodos iterativos estocásticos - tais como algoritmos genéticos [106] e *simulated annealing* [2] - e determinísticos. Pretendemos conduzir o trabalho de forma que, a medida que estudamos novas abordagens e métodos, revisamos os algoritmos já implementados e produzimos variações destes adicionando os novos recursos. Quanto aos método iterativos, o foco do trabalho estará na tarefa de refinamento, devido a alta demanda por tempo dos métodos iterativos para a geração de alinhamentos *ab initio*.

Os algoritmos iterativos devem ser adicionados às soluções da etapa anterior visando um incremento na qualidade dos resultados gerados por elas.

Uma vez que já tenhamos ganho bastante experiência na área, pretedemos trabalhar em busca de métodos alternativos para aplicar no problema de MSA. Um bom exemplo aqui é o método aplicado no DiAlign [152], que inspirou métodos como T-Coffee [165] e ProbCons [54]. Como descrevemos acima, tal método utiliza matriz de pontos e define alinhamentos como diagonais. Assim, ele não usa o conceito de *gap* e reduz sensivelmente a dependência dos resultados nos parâmetros definidos pelo usuário. Destaque aqui para estudos e experiências usando o conceito de consistência. Neste ponto variações dos algoritmos já implementados podem surgir, assim como novos algoritmos devem ser desenvolvidos.

Os métodos baseados em modelos têm sido bastante empregados nos últimos trabalhos em MSA. Eles surgem como uma boa alternativa quando tentamos alinhar seqüências de baixa homologia, uma vez que incorporam novas informações relativas às seqüências que tentamos alinhar. Pretendemos incorporar o uso de tais modelos nos algoritmos desenvolvidos em etapas anteriores. Para manipular tais modelos poderemos utilizar pacotes externos, como: FUGUE [195], PSI-BLAST [11] ou SAP [211]. Ainda é possível fazer uso de modelos como HMM.

Para finalizar o trabalho, pretendemos fazer uma integração dos algoritmos que desenvolvemos em etapas anteriores do estudo. O objetivo é extrair o que há de melhor em cada um através de um meta-método capaz de combinar os alinhamentos de diversos algoritmos e, assim, produzir resultados de maior qualidade. É importante salientar que o uso de meta-métodos implica na execução de uma série de outros algoritmos e isso vai implicar em uma grande demanda por tempo.

O nosso trabalho concentra-se no desenvolvimento de métodos e implementação de algoritmos para o problema de MSA. A eficácia de nossas soluções poderá ser avaliada tendo como parâmetros sua simplicidade, qualidade de resultados e o custo computacional em comparação às técnicas já existentes na literatura. Compararemos nossas soluções com pacotes de domínio público fazendo uso dos chamados "gold standards" [34], como o BAliBASE, para avaliar a qualidade dos resultados. Apesar do nosso foco estar voltado para a alta qualidade dos alinhamentos gerados, haverá uma grande preocupação com o tempo demandado e a capacidade de tratar grandes entradas. Realizaremos também análises comparativas quanto aos requisitos de tempo e a capacidade de alinhar grandes entradas.

Como resultado da tese, além dos algoritmos e suas avaliações, haverá também uma série de informações acerca das estratégias e métodos que mostraram maior ou menor potencial e os respectivos custos computacionais. Este tipo de conhecimento virá a agregar ao que temos disponível na bibliografia.

No decorrer do trabalho daremos importância à publicação dos resultados parciais em congressos e revistas internacionais. Vemos na publicação científica a forma mais concreta de avaliar o andamento do projeto e mensurar o êxito das soluções, além de propiciar uma efetiva interação com a comunidade especializada.

6 Cronograma

As Tabelas 1 e 2 apresentam a distribuição das atividades a serem realizadas durante a execução deste trabalho.

Tabela 1: Cronograma de Atividades: Março/2007 - Fevereiro/2009

2007										2008											2009	
M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F
1										2													
						3																	
																					4		

Tabela 2: Cronograma de Atividades: Março/2009 - Fevereiro/2011

2009										2010												2011	
M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F
4																							
5																							
						6																	
										7													
															8								
										9													
																						10	

1. Disciplinas obrigatórias;
2. Estágio docente;
3. Revisão bibliográfica das principais referências de MSA;
4. Preparação para o Exame de Qualificação Específico;
5. Estudo da abordagem progressiva seguido de projeto, implementação e teste dos novos algoritmos;
6. Estudo de métodos iterativos e alternativos seguido de projeto, implementação e teste dos novos algoritmos;
7. Estudo de métodos baseados em modelo seguido de projeto, implementação e teste dos novos algoritmos;
8. Projeto, implementação e teste de um meta-método que combine os resultados de uma seleção dos métodos desenvolvidos nas etapas anteriores;
9. Escrita e revisão da tese; e
10. Defesa da tese.

Referências

[1] BAliBASE 3.0 Web Site, February 2009. http://www-bio3d-igbmc.u-strasbg.fr/balibase/.

[2] E. Aart and P. Laarhoven. *Simulated Annealing: a review of theory and applications*. Kluwer Academic Publishers, Amsterdam, 1987.

[3] S. Adi and C. Ferreira. Gene prediction by multiple syntenic alignment. *Journal of Integrative Bioinformatics*, 2:19–28, 2005.

[4] B. Al-Lazikani, F. Sheinerman, and B. Honig. Combining multiple structure and sequence alignments to improve sequence detection and alignment: application to the SH2 domains of Janus kinases. *Proc. Natl. Acad. Sci. USA*, 98:14796–14801, 2001.

[5] E. Althaus, A. Caprara, H. Lenhof, and K. Reinert. Multiple sequence alignment with arbitrary gap costs: computing an optimal solution using polyhedral combinatorics. *Bioinformatics*, 18(90002):S4–S16, 2002.

[6] S. Altschul, M. Boguski, W. Gish, and J. Wootton. Issues in searching molecular sequence databases. *Nat Genet*, 6(2):119–129, 1994.

[7] S. Altschul, R. Carroll, and D. Lipman. Weights for data related by a tree. *J Mol Biol*, 207:647–653, 1989.

[8] S. Altschul and W. Gish. Local alignment statistics. *Methods Enzymol.*, 266:460–480, 1996.

[9] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–410, 1990.

[10] S. Altschul and D. Lipman. Trees, stars and multiple biological sequence alignment. *SIAM J. Appl. Math*, 49:197–209, 1989.

[11] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Jhang, W. Miller, and D. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*, 25(17):3389–3402, 1997.

[12] L. Anabarasu. Multiple sequence alignment using parallel genetic algorithms. In *The Second Asia-Pacific Conference on Simulated Evolution (SEAL-98)*, Canberra, Australia, 1998.

[13] P. Argos. A sensitive procedure to compare amino acid sequences. *J. Mol. Biol.*, 193:385–396, 1987.

[14] F. Armougom, S. Moretti, O. Poirot, S. Audic, P. Dumas, B. Schaeli, V. Keduas, and C. Notredame. Expresso: automatic incorporation of structural information in multiple sequence alignments using 3D-COFFEE. *Nucleic Acids Res*, 34:W604–608, 2006.

[15] T. Attwood, M. Croning, D.Flower, and et al. PRINTS-S: the database formerly known as PRINTS. *Nucleic Acids Res.*, 28(1):225–227, 2000.

[16] V. Bafna, E. Lawler, and P. Pevzner. Approximation algorithms for multiple sequence alignment. In *CPM*, pages 43–53, 1994.

[17] T. Bailey and C. Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 2:28–36, 1994.

[18] T. Bailey and C. Elkan. The value of prior knowledge in discovering motifs with MEME. In *ISMB*, volume 3, pages 21–29, 1995.

[19] W. Bains. Local sequence dependence of rate of base replacement in mammals. *Mutat. Res.*, 267:43–54, 1992.

[20] A. Bairoch and R. Apweiler. The Swiss-Prot protein sequence data bank and its supplement TrEMBL in 1999. *Nucleic Acids Research*, 27(1):49–54, 1999.

[21] A. Bairoch, B. Boeckmann, S. Ferro, and E. Gasteiger. Swiss-Prot: Juggling between evolution and stability. *Brief. Bioinform.*, 5:39–55, 2004.

[22] A. Bairoch, P. Bucher, and K. Hofmann. The PROSITE database - its status in 1997. *Nucleic Acids Res.*, 25:217–221, 1997.

[23] P. Baldi, Y. Chauvin, T. Hunkapiller, and M. McClure. Hidden Markov models of biological primary sequence information. *Proc. Nat. Acad. Sci.*, 91:1059–1063, 1994.

[24] W. Barker, J. Garavelli, P. McGarvey, C. Marzec, B. Orcutt, G. Srinivasarao, L. Yeh, R. Ledley, H. Mewes, Pfeiffer, A. Tsugita, and C. Wu. The PIR - international protein sequence database. *Nucleic Acids Research*, 27:39–43, 1999.

[25] G. Barton and M. Sternberg. A strategy for the rapid multiple alignment of protein sequences: confidence levels from tertiary structure comparisons. *J. Mol. Biol.*, 198:327–337, 1987.

[26] D. Bashford, C. Chothia, and A. Lesk. Determinants of a protein fold: Unique features of the globin amino acid sequences. *J. Mol. Biol.*, 196:199–216, 1987.

[27] A. Bateman, E. Birney, R. Durbin, S. Eddy, K. Howe, and E. Sonnhammer. The Pfam protein families database. *Nucleic Acids Res.*, 28(1):263–266, 2000.

[28] M. Bauer, G. Klau, and K. Reinert. Multiple structural RNA alignment with lagrangian relaxation. volume 3692, pages 303–314, 2005.

[29] L. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math. Statist.*, 41(1):164–171, 1970.

[30] A. Baxevanis and D. Landsman. Histone sequence database: new histone fold family members. *Nucleic Acids Research*, 26(1):372–375, 1998.

[31] D. Benson, I. Karsch-Mizrachi, D. Lipman, J. Ostell, and D. Wheeler. GenBank. *Nucleic Acids Res.*, 36:25–30, 2008.

[32] M. Berger and P. Munson. A novel randomized iterative strategy for aligning multiple protein sequences. *Comput. Applic. Biosci.*, 7:479–484, 1991.

[33] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. Bourne. The Protein Data Bank. *Nucleic Acids Res*, 28(1):235–242, 2000.

[34] G. Blackshields, I. Wallace, M. Larkin, and D. Higgins. Analysis and comparison of benchmarks for multiple sequence alignment. *In Silico Biol*, 6(4):321–339, 2006.

[35] N. Bray and L. Pachter. MAVID: constrained ancestral alignment of multiple sequences. *Genome Res.*, 14:693–699, 2004.

[36] P. Briffeuil, G. Baudoux, C. Lambert, X. de Bolle, C. Vinals, E. Feytmans, and E. Depiereux. Comparative analysis of seven multiple protein sequence alignment servers: clues to enhance reliability of predictions. *Bioinformatics*, 14(4):357–366, 1998.

[37] L. Brocchieri and S. Karlin. A symmetric-iterated multiple alignment of protein sequences. *J. Mol. Biol.*, 276(1):249–264, 1998.

[38] M. Brudno, M. Chapman, B. Gottgens, S. Batzoglou, and B. Morgenstern. Fast and sensitive multiple alignment of large genomic sequences. *BMC Bioinformatics*, 4:66, 2004.

[39] M. Brudno, C. Do, G. Cooper, M. Kim, E. Davydov, , E. Green, A. Sidow, and S. Batzoglou. LAGAN and multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res.*, 13:721–731, 2003.

[40] K. Bucka-Lassen, O. Caprani, and J. Hein. Combining many multiple alignments in one improved alignment. *Bioinformatics*, 15:122–130, 1999.

[41] L. Cai, D. Juedes, and E. Liakhovitch. Evolucionary computation techniques for multiple sequence alignment. In *Congress on Evolutionary Computation*, 2000.

[42] L. Cardon and G. Stormos. Expectation maximization algorithm for identifying protein-binding sites with variable lengths from unaligned DNA fragment. *Journal of Molecular Biology*, 223:159–170, 1992.

[43] H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SAIM J. Appl. Math.*, 48(5):1073–1082, 1988.

[44] S. Chan, A. Wong, and D. Chiu. A survey of multiple sequence comparison methods. *Bull. Math. Biol.*, 54:563–598, 1992.

[45] K. Chellapilla and G. Fogel. Multiple sequence alignment using evolutionary programming. In *Congress on Evolucionary Computation*, 1999.

[46] R. Chenna, H. Sugawara, T. Koike, R. Lopez, T. Gibson, D. Higgins, and J. Thompson. Multiple sequence alignment with the Clustal series of programs. *Nucleic Acids Res.*, 31:3497–3500, 2003.

[47] The UniProt Consortium. The Universal Protein Resource (UniProt). *Nucleic Acids Res.*, 36:D190–D195, 2008.

[48] F. Corpet. Multiple sequence alignment with hierarchical clustering. *Nucleic Acids Res.*, 16:10881–10890, 1988.

[49] L. Davis. Adapting operator probabilities in genetic algorithms. In *ICGA*, pages 61–69, 1989.

[50] M. Dayhoff, R. Schwartz, and B. Orcutt. A model for evolutionary change in proteins. *Atlas of Protein Sequence and Structure*, 5(3):345–352, 1978.

[51] E. Depiereux, G. Baudoux, P. Briffeuil, I. Reginster, X. de Bolle, C. Vinals, and E. Feytmans. Match-Box server: a multiple sequence alignment tool placing emphasis on reliability. *Comput. Appl. Biosci.*, 13(3):249–256, 1997.

[52] J. Devereux, P. Haeberli, and O. Smithies. GCG package. *Nucleic Acids Res.*, 12:387–395, 1984.

[53] S. Dietmann, J. Park, C. Notredame, A. Heger, M. Lappe, and L. Holm. A fully automatic evolucionary classification of protein fold: DALI domain dictionary version 3. *Nucleic Acids Res.*, 29(1):55–57, 2001.

[54] C. Do, M. Mahabhashyam, M. Brudno, and S. Batzoglou. ProbCons: Probabilistic consistency-based multiple sequence alignment. *Genome Res.*, 15:330–340, 2005.

[55] C. Do, M. Mahabhashyam, M. Brudno, and S. Batzoglou.

ProbCons Web Site, February 2009. http://probcons.stanford.edu.

[56] R. Doolittle. Similar amino acid sequences: chance and common ancestry? *Science*, 214:149–159, 1981.

[57] R. Doolittle. Convergent evolution: the need to be explicit. *Trends Biochem Sci.*, 19:15–18, 1994.

[58] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis*. Cambridge University Press, Cambridge, UK, 1998.

[59] L. Duret and S. Abdeddaim. *Bioinformatics: sequence, structure and databanks*, chapter Multiple alignment for structural functional or phylogenetic analyses of homologous sequences. Oxford University Press, Oxford, UK, 2000.

[60] S. Eddy. Multiple alignment using hidden Markov models. In *Third International Conference on Intelligent Systems for Molecular Biology (ISMB)*, Cambridge, England, 1995.

[61] R. Edgar. Local homology recognition and distance measures in linear time using compressed amino acid alphabets. *Nucleic Acids Res.*, 32:380–385, 2004.

[62] R. Edgar. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *Bioinformatics*, 5:113, 2004.

[63] R. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.*, 32:1792–1797, 2004.

[64] I. Eidhammer, I. Jonassen, and W. Taylor. Structure comparison and structure patterns. *J. Comput. Biol.*, 7(5):685–716, 2000.

[65] T. Etzold and P. Argos. SRS - an indexing and retrieval tool for flat file data libraries. *Comput. Appl. Biosci.*, 9:49–57, 1993.

[66] J. Felsenstein. Confidence limits on phylogenies: an approach using the bootstrap. *Evolution*, 39:783–791, 1985.

[67] D. Feng and R. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Biol.*, 25:351–360, 1987.

[68] W. Fitch and K. Yasunobu. Phylogenies from amino acid sequences aligned with gaps: the problem of gap weighting. *J. Mol. Evol.*, 5:1–24, 1974.

[69] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *J. Am. Stat. Assoc.*, 32:675–701, 1937.

[70] M. Frith, U. Hansen, J. Spouge, and Z. Weng. Finding functional sequence elements by multiple local alignment. *Nucleic Acids Res.*, 32:189–200, 2004.

[71] J. Garnier, J. Gibrat, and B. Robson. GOR method for predicting protein secondary structure from amino acid sequence. *Meth. Enzymol*, 266:540–553, 1996.

[72] U. Goebel, C. Sander, R. Schneider, and A. Valencia. Correlated mutations and residue contacts in proteins. *Proteins: Structure Function and Genetics*, 18(4):309–317, 1994.

[73] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, New York, 1989.

[74] G. Gonnet, M. Cohen, and S. Benner. Exhaustive matching of the entire protein sequence database. *Science*, 256:1443–1445, 1992.

[75] G. Gonnet, M. Cohen, and S. Benner. Empirical and structural models for insertions and deletions in the divergent

evolution of proteins. *Journal of Molecular Biology*, 229:1065–1082, 1993.

[76] G. Gonnet, C. Korostensky, and S. Benner. Evaluation measures of multiple sequence alignments. *J. Comput. Biol.*, 7:261–276, 2000.

[77] R. Gonzalez. Multiple protein sequence comparison by genetic algorithms, 1999.

[78] O. Gotoh. Consistency of optimal sequence alignments. *Bull Math Biol.*, 52(4):509–525, 1990.

[79] O. Gotoh. Optimal alignment between groups of sequences and its application to multiple sequence alignment. *Computer Applications in the Biosciences*, 9:361–370, 1993.

[80] O. Gotoh. Further improvement in methods of group-to-group sequence alignment with generalized profile operation. *CABIOS*, 10(4):379–387, 1994.

[81] O. Gotoh. A weighting system and algorithm for aligning many phylogenetically related sequences. *Computer Applications in the Biosciences*, 11(5):543–551, 1995.

[82] O. Gotoh. Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. *J. Mol. Biol.*, 264:823–838, 1996.

[83] M. Gouy, C. Gautier, M. Attimonelli, C. Lanave, and G. di Paola. ACNUC - a portable retrieval system for nucleic acid sequence databases: logical and physical designs and usage. *Computer Applications in the Biosciences*, 1(3):167–172, 1985.

[84] J. Gracy and P. Argos. Automated protein sequence database classification. *Bioinformatics*, 14(2):164–173, 1998.

[85] C. Grasso and C. Lee. Combining partial order alignment and progressive multiple sequence alignment increases alignment speed and scalability to very large alignment problems. *Bioinformatics*, 20:1546–1556, 2004.

[86] M. Gribskov, R. Luethy, and D. Eisenberg. Profile analysis. *Meth. Enzymol.*, 183:146–159, 1990.

[87] M. Gribskov, A. McLachlan, and D. Eisenberg. Profile analysis: detection of distantly related proteins. *Proc. Natl. Acad. Sci.*, 84:4355–4358, 1987.

[88] S. Griffiths-Jones and A. Bateman. The use of structure information to increase alignment accuracy does not aid homologue detection with profile-HMMs. *Bioinformatics*, 18:1243–1249, 2002.

[89] X. Gu and W. Li. The size distribution of insertions and deletions in human and rodent pseudogenes suggests the logarithmic gap penalty for sequence alignment. *Journal of Molecular Evolution*, 40:464–473, 1995.

[90] S. Gupta, J. Kececioglu, and A. Schaffer. Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. *J. Computational Biology*, 2:459–472, 1995.

[91] D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bull Math Biol*, 55(1):141–154, 1993.

[92] R. Gutell, B. Weiser, C. Woese, and H. Noller. Comparative anatomy of 16S-like ribosomal RNA. *Prog. Nucleic Acid Res. Mol. Biol.*, 32:155–216, 1985.

[93] D. Haussler, A. Krogh, I. Mian, and K. Sjölander. Protein modeling using hidden Markov models: analysis of

globins. In *Proceedings for the 26th Hawaii International Conference on Systems Sciences*, Wailea, 1993.

[94] S. Henikoff. Playing with blocks: some pitfalls of forcing multiple alignments. *The New Biologist*, 3(12):1148–1154, 1991.

[95] S. Henikoff and J. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci.*, 89(22):10915–10919, 1992.

[96] S. Henikoff, J. Henikoff, W. Alford, and S. Pietrokovski. Automated construction and graphical presentation of protein blocks from unaligned sequences. *Gene-COMBIS, Gene*, 163:17–26, 1995.

[97] J. Heringa. Two strategies for sequence comparison: profile-preprocessed and secondary structure-induced multiple alignment. *Computers and Chemistry*, 23:341–364, 1999.

[98] J. Heringa. Local weighting schemes for protein multiple sequence alignment. *Comput. Chem.*, 5:459–477, 2002.

[99] J. Heringa and P. Argos. A method to recognize distant repeats in protein sequences. *Proteins*, 17(4):391–411, 1993.

[100] G. Hertz and G. Stormo. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15:563–577, 1999.

[101] S. Hess, J. Blake, and R. Blake. Wide variations in neighbor-dependent substitution rates. *J. Mol. Biol*, 236:1022–1033, 1994.

[102] D. Higgins, A. Bleasby, and R. Fuchs. CLUSTAL V: improved software for multiple sequence alignment. *Computer Applications in the Biosciences*, 8(2):189–191, 1992.

[103] D. Higgins and P. Sharp. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene*, 73(1):237–244, 1988.

[104] M. Hirosawa, M. Hoshida, M. Ishikawa, and T. Toya. MASCOT: multiple alignment system for protein sequences based on three-way dynamic programming. *Computer Applications in the Biosciences*, 9(2):161–167, 1993.

[105] P. Hogeweg and B. Hesper. The alignment of sets of sequences and the construction of phyletic trees: An integrated method. *J. Mol. Evol.*, 20:175–186, 1984.

[106] J. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Michigan, USA, 1975.

[107] I. Holmes and W. Bruno. Evolutionary HMMs: a bayesian approach to multiple alignment. *Bioinformatics*, 9:803–820, 2001.

[108] I. Holmes and R. Durbin. Dynamic programming alignment accuracy. *J. Comput. Biol.*, 5:493–504, 1998.

[109] I. Holmes and G. Rubin. An expectation maximization algorithm for training hidden substitution models. *J. Mol. Biol.*, 5:753–764, 2002.

[110] X. Huang and W. Miller. A time-efficient linear-space local similarity algorithm. *Adv. Appl. Math.*, 12:337–357, 1991.

[111] M. Ishikawa, T. Toya, M. Hoshida, K. Nitta, A. Ogiwara, and M. Kanehisa. Multiple sequence alignment by parallel simulated annealing. *Computer Applications in the Biosciences*, 9(3):267–273, 1993.

[112] A. Jennings, C. Edge, and M. Sternberg. An approach to improving multiple alignments of protein sequences using predicted secondary structure. *Protein Eng.*, 14(4):227–231, 2001.

[113] J. Jiang and H. Jacob. EbEST: an automatic tool using expressed sequence tags to delineate gene structure. *Genome Research*, 8(3):268–275, 1998.

[114] T. Jiang, E. Lawler, and L. Wang. Aligning sequences via an evolutionary tree: complexity and approximation. In *ACM Sumpos. Theory Comput.*, volume 26, pages 760–769, 1994.

[115] I. Jonassen. *Bioinformatics: sequence, structure and databanks*, chapter Methods for discovering conserved patterns in protein sequences and structures. Oxford University Press, Oxford, UK, 2000.

[116] D. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *J. Mol. Biol.*, 292(2):195–202, 1999.

[117] W. Just. Computational complexity of multiple sequence alignment with SP-score. *J Comput Biol*, 8(6):615–623, 2001.

[118] W. Kabsch and C. Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 12:2577–2637, 1983.

[119] S. Karlin, P. Bucher, V. Brendel, and S. Altschul. Statistical methods and insight for protein and DNA sequences. *Biophys. Chem.*, 20:175–203, 1991.

[120] K. Karplus and B. Hu. Evaluation of protein multiple alignments by SAM-T99 using the BAliBASE multiple alignment test set. *Bioinformatics*, 8:713–720, 2001.

[121] K. Katoh, K. Kuma, H. Toh, and T. Miyata. MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Res*, 33(2):511–518, 2005.

[122] K. Katoh, K. Misasa, K. Kuma, and T. Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nucleic Acids Res.*, 30:3059–3066, 2002.

[123] J. Kececioglu. *Exact and approximation algorithms for DNA sequence reconstruction.* PhD thesis, University of Arizona, 1991.

[124] J. Kececioglu. The maximum weight trace problem in multiple sequence alignment. *Lecture Notes In Computer Science*, 684:106–119, 1993.

[125] J. Kim, S. Paramanik, and M. Chung. Multiple sequence alignment using simulated annealing. *Computer Applications in the Biosciences*, 10(4):419–426, 1994.

[126] K. Koretke, R. Russell, R. Copley, and A. Lupas. Fold recognition using sequence and secondary structure information. *Proteins*, 37:141–148, 1999.

[127] A. Krogh, M. Brown, I. Mian, K. Sjlander, and D. Haussler. Hidden Markov models in computational biology. Applications to protein modeling. *J. Mol. Biol.*, 235(5):1501–1531, 1994.

[128] P. Lackner, W. Koppensteiner, M. Sippl, and F. Domingues. ProSup: a refined tool for protein structure alignment. *Protein Eng.*, 13:745–752, 2000.

[129] T. Lassmann and E. Sonnhammer. Quality assessment of multiple alignment programs. *FEBS Lett*, 529:126–130, 2002.

[130] T. Lassmann and E. Sonnhammer. Kalign - an accurate and fast multiple se-

quence alignment algorithm. *BMC bioinformatics*, 6:298, 2005.

[131] C. Lawrence, S. Altschul, M. Boguski, J. Liu, A. Neuwald, and J. Wootton. Detecting subtle sequence signals: A Gibbs Sampling strategy for multiple alignment. *Science*, 262:208–214, 1993.

[132] C. Lawrence and A. Reilly. An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins*, 7(1):41–51, 1990.

[133] C. Lee, C. Grasso, and M. Sharlow. Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18:452–464, 2002.

[134] H. Lenhof, B. Morgenstern, and K. Reinert. An exact solution for the sequence-to-sequence multiple sequence alignment problem. *Bioinformatics*, 15(3):203–210, 1999.

[135] C. Lima, H. Lopes, M. Moroz, and R. Meneses. Multiple sequence alignment using reconfigurable computing. *Lecture Notes in Computer Science*, 4419:379–384, 2007.

[136] D. Lipman, S. Altschul, and J. Kececioglu. A tool for multiple sequence alignment. *Proc. Natl. Acad. Sci.*, 86:4412–4415, 1989.

[137] H. Lopes and G. Moritz. A distributed approach for a multiple sequence alignment algorithm using a parallel virtual machine. In *Proceedings of the 27th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, pages 2843–2846, Shanghai, China, 2005.

[138] H. Lopes and G. Moritz. A graph-based genetic algorithm for the multiple sequence alignment problem. *Lecture Notes in Artificial Intelligence*, 4029:420–429, 2006.

[139] A. Lukashin, J. Engelbrecht, and S. Brunak. Multiple alignment using simulated annealing: branch point definition in human mRNA splicing. *Nucleic Acids Res.*, 20(10):2511–2516, 1992.

[140] R. Lüthy, I. Xenarios, and P. Bucher. Improving the sensitivity of the sequence profile method. *Protein Sci.*, 3:139–146, 1994.

[141] B. Marsden and R. Abagyan. SAD - a normalized structural alignment database: improving sequence-structure alignments. *Bioinformatics*, 15:2333–2344, 2004.

[142] M. McClure, T. Vasi, and W. Fitch. Comparative analysis of multiple protein-sequence alignment methods. *Mol. Biol. Evol.*, 11:571–592, 1994.

[143] E. McCreight. A space-economical suffix tree construction algorithm. *JACM*, 23(2):262–272, 1976.

[144] J. Meidanis. *Algorithms for problems in computacional genetics*. PhD thesis, University of Winsconsin-Madison, 1992.

[145] J. Meidanis and J. Setubal. Multiple alignment of biological sequences with gap flexibility. In R. Baeza-Yates and U. Manber, editors, *Proceedings of Second South American Workshop on String Processing*, pages 138–153, Valparaíso, Chile, 1995.

[146] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1092, 1953.

[147] W. Miller and E. Myers. Sequence comparison with concave weighting

functions. *Bull. Math. Biol.*, 50:97–120, 1988.

[148] K. Mizuguchi, C. Deane, T. Blundell, M. Johnson, and J. Overington. Joy: protein sequence-structure representation and analysis. *Bioinformatics*, 14:617–623, 1998.

[149] K. Mizuguchi, C. Deane, T. Blundell, and J. Overington. HOMSTRAD: a database of protein structure alignments for homologous families. *Protein Sci.*, 7:2469–2471, 1998.

[150] B. Morgenstern. DiAlign 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, 15:211–218, 1999.

[151] B. Morgenstern. DiAlign: multiple DNA and protein sequence alignment at BiBiServ. *Nucleic Acids Res.*, 32:33–36, 2004.

[152] B. Morgenstern, A. Dress, and T. Wener. Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proc. Natl. Acad. Sci.*, 93:12098–12103, October 1996.

[153] B. Morgenstern, K. Frech, A. Dress, and T. Werner. DiAlign: finding local similarities by multiple sequence alignment. *Bioinformatics*, 14:290–294, 1998.

[154] M. Murata, J. Richardson, and J. Sussman. Simultaneous comparison of three protein sequences. *Proc. Natl. Acad. Sci.*, 82:3073–3077, 1985.

[155] A. Murzin, S. Brenner, T. Hubbard, and C. Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, 247:536–540, 1995.

[156] E. Myers and W. Miller. Optimal alignments in linear-space. *Comput. Appl. Biosci.*, 4:11–17, 1988.

[157] E. Myers and W. Miller. Chaining multiple-alignment fragments in sub-quadratic time. In *SODA*, pages 38–47, 1995.

[158] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.

[159] A. Neuwald, J. Liu, D. Lipman, and C. Lawrence. Extracting protein alignment models from the sequence database. *Nucleic Acid Research*, 25:1665–1677, 1997.

[160] H. Nguyen, I. Yoshihara, K. Yamamori, and M. Yasunaga. Aligning multiple protein sequences by parallel hybrid genetic algorithm. *Genome Inform Ser*, 13:123–132, 2002.

[161] C. Notredame. Mocca: semi-automatic method for domain hunting. *Bioinformatics*, 17(4):373–374, 2001.

[162] C. Notredame. Recent progress in multiple sequence alignment: a survey. *Pharmacogenomics*, 3:131–144, 2002.

[163] C. Notredame. Recent evolutions of multiple sequence alignment algorithms. *PLoS Computational Biology*, 3(8):1405–1408, 2007.

[164] C. Notredame and D. Higgins. SAGA: sequence alignment by genetic algorithm. *Nucl. Acid. Res.*, 24(8):1515–1524, 1996.

[165] C. Notredame, D. Higgins, and J. Heringa. T-COFFEE: A novel method for fast and accurate multiple sequence alignment. *J Mol Biol*, 302(1):205–217, 2000.

[166] C. Notredame, L. Holm, and D. Higgins. COFFEE: an objective function

for multiple sequence alignments. *Bioinformatics*, 14:407–422, 1998.

[167] C. Notredame, E. O'Brien, and D. Higgins. RAGA: RNA sequence alignment by genetic algorithm. *Nucleic Acids Res.*, 25(22):4570–4580, 1997.

[168] C. Orengo and W. Taylor. A rapid method of protein structure alignment. *J. Theor. Biol.*, 147:517–551, 1990.

[169] O. O'Sullivan, K. Suhre, C. Abergel, D. Higgins, and C. Notredame. 3D-COFFEE: combining protein sequences and structures within multiple sequence alignments. *J. Mol. Biol.*, 340:385–395, 2004.

[170] O O'Sullivan, M. Zehnder, D. Higgins, P. Bucher, A. Grosdidier, and et al. APDB: a novel measure for benchmarking sequence alignment methods without reference alignments. *Bioinformatics*, 19:1215–1221, 2003.

[171] J. Overington, D. Donnelly, M. Johnson, A. Sali, and T. Blundell. Environment-specific amino acid substitution tables: Tertiary templates and prediction of protein folds. *Protein Science*, 2:216–226, 1992.

[172] S. Pascarella and P. Argos. Analysis of insertions/deletions in protein structures. *J. Mol. Biol.*, 224:461–471, 1992.

[173] L. Patthy. Exon shuffling and other ways of module exchange. *Matrix Biol*, 15:301–310, 1996.

[174] W. Pearson and D.Lipman. Improved tools for biological sequence comparison. *Proc Natl Acad Sci USA*, 85(8):2444–2448, 1988.

[175] J. Pei and N. Grishin. MUMMALS: multiple sequence alignment improved by using hidden Markov models with local structural information. *Nucleic Acids Res*, 34:4364–4374, 2006.

[176] J. Pei and N. Grishin. PROMALS: towards accurate multiple sequence alignments of distantly related proteins. *Bioinformatics*, 23:802–808, 2007.

[177] J. Pei, R. Sadreyev, and N. Grishin. PCMA: fast and accurate multiple sequence alignment based on profile consistency. *Bioinformatics*, 19:427–428, 2003.

[178] P. Pevzner. Multiple alignment, communication cost, and graph matching. *SIAM J. Appl. Math.*, 52(6):1763–1779, 1992.

[179] F. Plewniak, J. Thompson, and O. Poch. Ballast: BLAST post-processing based on locally conserved segments. *Bioinformatics*, 16(92000):750–759, 2000.

[180] G. Raghava, S. Searle, P. Audley, J. Barber, and G. Barton. OXBench: a benchmark for evaluation of protein multiple sequence alignment accuracy. *BMC Bioinformatics*, 4(47), 2003.

[181] K. Reinert, J. Stoye, and T. Will. An iterative method for faster sum-of-pair multiple sequence alignment. *Bioinformatics*, 16(9):808–814, 2000.

[182] B. Rost. Review: protein secondary structure prediction continues to rise. *J. Struct. Biol.*, 134:204–218, 2001.

[183] B. Rost, C. Sander, and R. Schneider. PHD - an automatic server for protein secundary structure prediction. *CABIOS*, 10:53–60, 1994.

[184] R. Rughey and A. Krogh. Hidden Markov models for sequence analysis: extension and analysis of the basic method. *Computer Application in Biological Science*, 12:95–107, 1996.

[185] N. Saitou. Maximum likelihood methods. *Meth. Enzymol.*, 183:584–598, 1990.

[186] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4:406–425, 1987.

[187] C. Sander and R. Schneider. Database of homology derived protein structures and the structural meaning of sequence alignment. *Proteins*, 9:56–58, 1991.

[188] D. Sankoff and J. Kruskal, editors. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley, Reading, MA, 1983.

[189] D. Santos. Alinhamento múltiplo de proteínas via algoritmo genético baseado em tipos abstratos de dados. Master's thesis, Instituto de Computação - Universidade Federal de Alagoas, 2008.

[190] A. Schaffer, L. Aravind, T. Madden, S. Shavirin, J. Spouge, Y. Wolf, E. Koonin, and S. Altschul. Improving the accuracy of PSI-BLAST protein database searches with composition-based statistics and other refinements. *Nucleic Acids Res.*, 14:2994–3005, 2001.

[191] G. Schuler, S. Altschul, and D. Lipman. A workbench for multiple alignment construction and analysis. *Proteins*, 9(3):180–190, 1991.

[192] G. Schuler, J. Epstein, H. Ohkawa, and J. Kans. Entrez: molecular biology database and retrieval system. *Methods Enzymol*, 266:141–162, 1996.

[193] R. Schwartz and M. Dayhoff. Matrices for detecting distant relationships. *Atlas of Protein Sequence and Structure*, 5:353–358, 1978. suppl. 3.

[194] P. Sellars. On the theory and computation of evolutionary distances. *SIAM J. Appl. Math.*, 26:787–793, 1974.

[195] J. Shi, T. Blundell, and K. Mizuguchi. FUGUE: sequence-structure homology recognition using environment-specific substitution tables and structure-dependent gap penalties. *J. Mol. Biol.*, 310:243–257, 2001.

[196] V. Simossis. and J. Heringa. PRALINE: a multiple sequence alignment toolbox that integrates homology-extended and secondary structure information. *Nucleic Acids Res*, 33:289–294, 2005.

[197] S. Sinha, M. Blanchette, and M. Tompa. PhyME: a probabilistic algorithm for finding motifs in sets of orthologous sequences. *BMC Bioinformatics*, 5:170, 2004.

[198] R. Smith and T. Smith. Pattern-Induced Multi-sequence Alignment (PIMA) algorithm employing secondary structure-dependent gap penalties for comparative protein modelling. *Protein Engng*, 5:35–41, 1992.

[199] T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[200] T. Smith, M. Waterman, and W. Fitch. Comparative biosequence metrics. *J. Mol. Evol.*, 18:38–46, 1981.

[201] P. Sneath and R. Sokal. *Numerical Taxonomy*. Freeman, San Francisco, 1973.

[202] E. Sobel and H. Martinez. A multiple sequence alignment program. *Nucleic Acids Research*, 14(1):363–374, 1986.

[203] G. Srinivasarao, L. Yeh, C. Marzec, B. Orcutt, and W. Barker. Database of protein sequence alignments: PIR-ALN. *Bioinformatics*, 15:382–390, 1999.

[204] G. Stoesser, M. Tuli, R. Lopez, and P. Sterk. The EMBL nucleotide se-

quence database. *Nucleic Acids Research*, 27:18–24, 1999.

[205] J. Stoye, D. Evers, and F. Meyer. ROSE: generating sequence families. *Bioinformatics*, 14:157–163, 1998.

[206] J. Stoye, V. Moulton, and A. Dress. DCA: an efficient implementation of the divide-and-conquer approach to simultaneous multiple sequence alignment. *Comput. Appl. Biosci.*, 13(6):625–626, 1997.

[207] A. Subramanian, J. Weyer-Menkhoff, M. Kaufmann, and B. Morgenstern. DiAlign-T: an improved algorithm for segment-based multiple sequence alignment. *BMC Bioinformatics*, 6:66, 2005.

[208] W. Taylor. Identification of protein sequence homology by consensus template alignment. *J Mol Biol*, 188(2):233–258, 1986.

[209] W. Taylor. A flexible method to align a large number of sequences. *J. Mol. Evol.*, 28:161–169, 1988.

[210] W. Taylor. Dynamic sequence databank searching with templates and multiple alignments. *J. Mol. Biol.*, 280:375–406, 1998.

[211] W. Taylor. Protein structure comparison using SAP. *Methods Mol. Biol.*, 143:19–32, 2000.

[212] W. Taylor, G. Saelensminde, and I. Eidhammer. Multiple protein sequence alignment using double-dynamic programming. *Comput. Chem.*, 24(1):3–12, 2000.

[213] J. Thompson. Introducing variable gap penalties to sequence alignment in linear space. *Computer Applications in the Biosciences*, 11(2):181–186, 1995.

[214] J. Thompson, T. Gibson, F. Plewniak, F. Jeanmougin, and D. Higgins. The Clustal X window interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Res.*, 24:4876–4882, 1997.

[215] J. Thompson, D. Higgins, and T. Gibson. CLUSTAL W: improving the sesitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucl. Acid. Res.*, 22:4673–4680, 1994.

[216] J. Thompson, P. Koehl, R. Ripp, and O. Poch. BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark. *Proteins*, 61:127–136, 2005.

[217] J. Thompson, F. Plewniak, and O. Poch. BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, 15(1):87–88, 1999.

[218] J. Thompson, F. Plewniak, and O. Poch. A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Res.*, 27(13):2682–2690, 1999.

[219] J. Thompson, F. Plewniak, R. Ripp, J. Thierry, and O. Poch. Towards a reliable objective function for multiple sequence alignments. *J. Mol. Biol.*, 314(4):937–951, 2001.

[220] J. Thompson, F. Plewniak, J. Thierry, and O. Poch. DbClustal: rapid and reliable global multiple alignments of protein sequences detected by database searches. *Nucleic Acids Res*, 28(15):2919–2926, 2000.

[221] N. van Ohsen and R. Zimmer. Improving profile-profile alignments via log average scoring. In *WABI*, pages 11–26, 2001.

[222] R. Vieira. *Um Algoritmo Genético Baseado em Tipos Abstratos de Dados e sua Especificação em Z.* PhD thesis, Universidade Federal de Pernambuco, 2003.

[223] M. Vingron and P. Argos. A fast and sensitive multiple sequence alignment algorithm. *Comput. Appl. Biosci.*, 5:115–121, 1989.

[224] M. Vingron and P. Argos. Motif recognition and alignment for many sequences by comparison of dot-matrices. *J. Mol. Biol.*, 218:33–43, 1991.

[225] M. Vingron and M. Waterman. Sequence alignment and penalty choice. *J. Mol. Biol.*, 235:1–12, 1994.

[226] A. Viterbi. Error bounds for computacional codes and an symptotically optimal decoding algorithm. *IEEE Trans. Inf. Theory*, IT-13:260–269, 1967.

[227] I. Wallace, G. Blackshields, and D. Higgins. Multiple sequence alignments. *Curr. Opin. Struct. Biol.*, 15:261–266, 2005.

[228] I. Wallace, O. O'Sullivan, D. Higgins, and C. Notredame. M-COFFEE: combining multiple sequence alignment methods with T-COFFEE. *Nucleic Acids Res*, 34:1692–1699, 2006.

[229] I. Walle, I. Lasters, and L. Wyns. SABmark - a benchmark for sequence alignment that covers the entire known fold space. *Bioinformatics*, 21(7):1267–1268, 2005.

[230] I. Van Walle, I. Lasters, and L. Wyns. Align-m: a new algorithm for multiple alignment of highly divergent sequences. *Bioinformatics*, 20:1428–1435, 2004.

[231] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *J. Comput. Biol.*, 1(4):337–348, 1994.

[232] Y. Wang and K. Li. An adaptive and iterative algorithm for refining multiple sequence alignment. *Comput. Biol. Chem.*, 2:141–148, 2004.

[233] M. Waterman and R. Jones. Consensus methods for DNA and protein sequence alignment. *Meth. Enzym.*, 183:221–236, 1990.

[234] M. Waterman, T. Smith, and W. Bayer. Some biological sequence metrics. *Adv. Math.*, 20:267–287, 1976.

[235] F. Wilcoxon. Probability tables for individual comparisons by ranking methods. *Biometrics*, 3:119–122, 1947.

[236] J. Wootton and S. Federhen. Statistics of local complexity in amino acid sequences and sequence databases. *Computers in Chemistry*, 17:149–163, 1993.

[237] C. Zhang and A. Wong. A genetic algorithm for multiple molecular sequence alignment. *Comput. Appl. Biosci.*, 13(6):565–581, 1997.

[238] Y. Zhang and M. Waterman. An Eulerian path approach to local multiple alignment for DNA sequences. *Proc. Natl. Acad. Sci. USA*, 102:1285–1290, 2005.

[239] Z. Zhang, B. He, and W. Miller. Local multiple alignment via subgraph enumeration. *Discrete Appl. Math.*, 71:337–365, 1996.

[240] Z. Zhang, B. Raghavachari, R. Hardison, and W. Miller. Chaining multiple-aligned blocks. *Journal of Computational Biology*, 1:217–226, 1994.

[241] H. Zhou and Y. Zhou. SPEM: improving multiple sequence alignment with sequence profiles and predicted secondary structures. *Bioinformatics*, 21(18):3615–3621, 2005.

A Revisão Bibliográfica

Nesta revisão bibliográfica apresentamos trabalhos de grande relevância para os estudos a serem desenvolvidos durante o doutorado. A escolha destes trabalhos foi realizada de forma a englobar aspectos relacionados, seja direta ou indiretamente, ao nosso estudo. Com isso temos o objetivo de adiquirir um amplo conhecimento sobre o contexto do nosso trabalho e as soluções já existentes para os problemas relacionados a MSA.

A.1 Multiple alignments for structural, functional, or phylogenetic analyses of homologous sequences

Este trabalho [59] expõe uma extensa revisão acerca de alinhamento múltiplo de seqüências. Apresenta desde conceitos básicos até ferramentas de suporte a MSA. Claro que passando pelos métodos existentes (até 2000, época da publicação).

Entender a estrutura, função e evolução dos genes é um dos principais objetivos de projetos de seqüenciamento de genoma. Classicamente, a função de um gene é investigada experimentalmente através da análise de fenótipos mutantes. Mais recentemente, análise comparativa de seqüências homólogas tem mostrado ser uma abordagem muito eficiente para estudar função de gene. Estudar padrões de mutação através de análise de seqüências homólogas é útil não apenas para estudar relacionamentos evolucionários entre seqüências, mas também para identificar restrições estruturais ou funcionais nas seqüências de RNA, DNA ou proteína.

O alinhamento de seqüências homólogas consiste de tentar posicionar resíduos (nucleotídeos ou aminoácidos) em colunas, que derivam de um resíduo de um ancestral comum. Isto é obtido pela introdução de *gaps*, que representam inserções ou remoções, nas seqüências. Desta forma, um alinhamento é um modelo hipotético de mutações (substituições, inserções ou remoções) que ocorreram durante a evolução da seqüência. O melhor alinhamento será aquele mais parecido com um cenário evolucionário.

Devido a complexidade computacional deste problema, algoritmos de alinhamento utilizáveis na prática não podem garantir a melhor solução. E, mesmo com um algoritmo ideal, encontrar o melhor alinhamento poderá não ser garantido porque o conhecimento corrente de probabilidade de ocorrência de diferentes tipos de mutações é ainda limitado. Entretanto, quanto menos divergentes forem as seqüências homólogas, mais confiáveis tornam-se as heurísticas e algoritmos aproximados. Na prática, tais alinhadores são comumente utilizados em biologia molecular ou evolucionária. A seguir listamos exemplos típicos de uso de MSA.

- Busca por similaridades fracas, mas significantes, em bases de dados de seqüências;
- Demonstração de homologia entre seqüências;
- Filogenia molecular;
- Identificação de genes relacionados;
- Identificação de sítios funcionalmente importantes;
- Predição de estrutura;
- Predição de função; ou

- Projeto de *primers* para PCR.

O procedimento geral para a computação de um alinhamento múltiplo de seqüências homólogas consiste de três passos:

1. Busca por homólogas em bases de dados de seqüências
2. Computação dos alinhamentos
3. Verificação e edição dos alinhamentos

Neste texto o foco estará nos dois últimos passos. Inicialmente serão definidos alguns conceitos gerais acerca da metodologia de alinhamento múltiplo. Então descreve-se e compara-se os diferentes métodos que têm sido desenvolvidos para alinhamento de seqüências. Uma lista de URLs para sítios na *web* de alinhadores está disponível no seguinte endereço: `http://pbil.univ-lyon1.fr/alignment.html`.

Alguns problemas relacionados a alinhamento múltiplo, tal como montagem de *contigs*, não serão tratatos neste texto. Apenas serão descritos métodos para alinhamento de seqüências homólogas. Sendo assim, também não será abordado o problema da busca por *motifs* comuns em um conjunto de seqüências não relacionadas neste texto. Para maiores detalhes sobre este assunto em particular, consulte o texto de Jonassen [115].

A.1.1 Conceitos básicos

Duas seqüências são chamadas homólogas quando derivam de um ancestral comum. Geralmente, homologia é inferida por similaridade de seqüências. É importante salientar, entretanto, que similaridade não reflete, necessariamente, homologia. Similaridade entre fragmentos curtos de seqüência podem resultar de convergência evolucionária [57] ou podem resultar simplesmente do acaso.

Há ainda o fato de muitas seqüências conterem fragmentos relativamente longos de composições de nucleotídeos ou aminoácidos muito tendenciosas, por exemplo: repetições CA em DNA ou domínios ricos em prolina em proteínas [236]. Geralmente, similaridades entre tais seqüências com "regiões de baixa complexidade" não refletem relacionamento evolucionário.

Na ausência de tais composições tendenciosas, similaridade em uma região extensa freqüentemente implica em homologia. Testes estatísticos podem ser usados para avaliar as chances de uma similaridade observada ocorrer ao acaso e, assim, aceitar ou rejeitar a hipotética homologia [8]. Tais testes são geralmente fornecidos por programas de busca por similaridade.

Alinhamentos múltiplos podem ser úteis para ajudar a demonstrar homologia. Uma pequena similaridade, que poderia ser considerada não significante em um alinhamento de par de seqüências, pode se mostrar altamente significante se os mesmos resíduos são conservados em outras seqüências com relação distante. Deve-se, entretanto, ser enfatizado que, se as seqüências divergiram muito, a homologia pode não ser reconhecida apenas com similaridade de seqüências.

Em muitos casos, homologia está restrita a regiões limitadas das seqüências. Muitas proteínas consistem de uma combinação de "módulos" discretos que foram deslocados durante a evolução. É claro que muitos genes codificadores de proteínas resultam da recombinação

entre diferentes fragmentos de outros genes. Esta evolução modular tem um papel de maior importância na evolução de proteína e tem sido particularmente facilitado em eucariotos graças a presença de *introns* nos genes [173].

Cópias múltiplas de um dado módulo podem ser repetidas em uma seqüência e um conjunto de módulos pode ocorrer em diferentes posições relativas em genes diferentes. Em tais casos, não é possível alinhar seqüências em seu tamanho completo (alinhamento global) e é assim necessário executar alinhamentos apenas em regiões homólogas (alinhamento local).

Como indicado anteriormente, buscar pelo melhor alinhamento consiste de buscar por aquele que aparente um melhor cenário evolucionário. Assim, a probabilidade de ocorrência de diferentes eventos mutacionais durante a evolução deve ser levada em consideração quando computando um alinhamento múltiplo. Em alinhamentos, tipicamente três tipos de mutações são consideradas: substituições, inserções ou remoções. Os dois últimos eventos são freqüentemente indistingüíveis e são comumente referenciados como *indels*.

A probabilidade de substituição de um aminoácido por outro depende da estrutura do código genético (por exemplo: o número de mutações necessárias para transformar um *codon* em outro) e também do efeito fenotípico da mutação. Substituições de um aminoácido por outro com propriedades bioquímicas similares geralmente não têm grande efeito na estrutura e conseqüentemente na função da proteína. Assim, durante a evolução, substituições conservativas são relativamente freqüentes se comparadas a outras substituições.

É importante notar que a probabilidade de substituição de um aminoácido por outro depende da distância evolucionária entre as seqüências. Vários métodos foram propostos para a construção de séries de matrizes que estimam a probabilidade de todas as possíveis substituições para diferentes distâncias evolucionárias [50, 74, 95, 193]. As mais utilizadas são as matrizes PAM [50, 193] e BLOSUM [95].

Nas matrizes PAM os índices das matrizes crescem de acordo com as distâncias evolucionárias. Por exemplo, a PAM80 é adequada para pequenas distâncias evolucionárias e a PAM250 para maiores distâncias. Já as matrizes BLOSUM adotam uma convenção oposta, assim BLOSUM80 é adequada para pequenas distâncias evolucionárias e BLOSUM45 para maiores distâncias. Geralmente os alinhadores permitem ao usuário selecionar a matriz de substituição a utilizar. No Clustal W [215] as matrizes de substituição são automaticamente selecionadas e variam nos diferentes estágios do alinhamento de acordo com o grau de divergência entre as seqüências.

Probabilidades de substituição também variam ao longo das seqüências de acordo com ambientes locais de aminoácidos na proteína dobrada (por exemplo, hélice-α ou folha-β). Matrizes de substituição de ambiente específico foram desenvolvidas [171]. É importante salientar, entretanto, que estas matrizes são raramente utilizadas para alinhamento múltiplo nas ferramentas disponíveis atualmente.

Ao nível de DNA, probabilidades de susbtituições variam de acordo com as bases. Transições são geralmente mais freqüentes que transversões. Transições são substituições entre bases púricas ou entre bases pirimídicas e transversões substituições entre uma base púrica e uma pirimídica. São bases púricas A e G e são bases pirimídicas C e T (U para RNA). Assim, alinhadores múltiplos geralmente propõem um parâmetro de peso mais rígido para transversões que transições.

Probabilidades de substituições de nucleotídeos também dependem das bases vizinhas,

por exemplo: em vertebrados, C no dinucleotídeo CG é extremamente mutável [19, 101]. Os alinhadores correntes ainda não fazem uso de tal informação.

A probabilidade de ocorrência de um *indel* depende do tamanho. Assim, quando computando um alinhamento, freqüentemente estima-se penalidades (p) associadas com *gaps* usando um modelo linear ou "afim" tal como:

$$p = a + bL$$

onde L é o tamanho do *gap*, a a penalidade para abertura de *gap* e b a penalidade para extensão de *gap*. Entretanto, análises de alinhamentos de seqüências homólogas têm mostrado que este modelo, para seqüências de proteínas e nucleotídeos, subestima a probabilidade de *indels* longos [74, 75, 89]. Penalidades mais realísticas para *indels* podem ser estimadas com modelos tais como o seguinte:

$$p = a + b \times log(L).$$

Devido à complexidade computacional, tais modelos não têm sido implementados nos alinhadores comumente utilizados. Felizmente, outras abordagens para alinhar seqüências com grandes *indels* foram propostas, como poderá ser visto na Seção A.1.2.

Em proteínas a probabilidade de ocorrência de *indels* também depende do grau de divergência entre as seqüências [74, 75]. Tais probabilidades variam de acordo com a natureza das seqüências: proteína, RNA estrutural, DNA não codificante, etc. Em proteínas, *indels* são mais freqüentes em laços externos que no núcleo da estrutura. Desta forma, é possível utilizar o conhecimento da estrutura da proteína para valorar pesos de *indels*.

No Clustal W, por exemplo, implementa-se penalidade para *indel* específica para resíduo e esta é localmente reduzida para encorajar novos *gaps* em potenciais regiões de laço. Em casos onde informações acerca da estrutura secundária estão disponíveis, máscaras para penalidades de *indels* também podem ser usadas para guiar o alinhamento.

É importante salientar que, na maioria dos programas, parâmetros padrões para penalidades de *gap* têm sido definidos para proteínas globulares típicas, que podem não ser os melhores para outras seqüências.

O primeiro passo na análise de uma família de seqüências homólogas consiste da busca por todos os membros disponíveis daquela família. Seqüências publicadas são armazenadas em bancos de dados públicos, tais como GenBank [31] e EMBL [204] para seqüências de nucleotídeo e Swiss-Prot/TrEMBL [20] e PIR [24] para seqüências de proteína. Sistemas de recuperação de informação tais como Entrez [192], SRS [65] e ACNUC [83] foram desenvolvidos para a consulta destas bases de dados e extração de seqüências de acordo com as anotações associadas (tais como palavras chaves, taxonomia ou autores).

Infelizmente, não é possível utilizar as anotações para identificar no banco de dados todas as seqüências homólogas pertencentes a uma dada família. Atualmente, a forma mais eficiente de identificar estes homólogos consiste em comparar um membro da família com todas as outras seqüências no banco de dados, utilizando-se para isso um programa de busca por similaridade tal como FASTA [174] ou BLAST [9, 11]. Para uma discussão abrangente acerca de busca por similaridade de seqüências, consulte a revisão de Altschul e colegas [6].

A.1.2 Métodos para alinhamento múltiplo

O problema do alinhamento múltiplo de seqüências é algoritmicamente difícil, ou seja, métodos que garantam o melhor alinhamento requerem tanto tempo e espaço de memória que não é possível utilizá-los na prática. Torna-se impraticável utilizar algoritmos exatos com entradas de 10 a 15 seqüências de tamanho 100.

Desta forma, algoritmos alternativos têm sido desenvolvidos usando heurísticas para ganhar em velocidade e reduzir os requisitos de memória. Embora não hajam garantias, observa-se que os resultados são muito úteis na prática e freqüentemente seus alinhamentos estão muito próximos da solução exata. Neste trabalho divide-se os algoritmos para MSA em quatro categorias:

1. Algoritmos que garantem o alinhamento ótimo para um dado esquema de pontuação. Estes possuem fortes limitações quanto ao número e tamanho das seqüências;
2. Algoritmos heurísticos baseados na abordagem de alinhamento progressivo de pares;
3. Algoritmos heurísticos que constroem alinhamentos globais baseados em alinhamentos locais;
4. Algoritmos heurísticos que constroem alinhamentos múltiplos locais.

Esta lista não é exaustiva. Há outros métodos para MSA tais como os baseados em modelos ocultos de Markov [184] ou algoritmos genéticos [164]. Para uma revisão dos algoritmos de alinhamento múltiplo, consulte o trabalho de Chan e colegas [44].

Deve-se chamar a atenção que o termo ótimo aqui refere-se a um ótimo matemático. Se um alinhamento matematicamente ótimo corresponde ou não a um alinhamento biologicamente correto irá depender da escolha dos parâmetros e da forma que se calcula a pontuação do alinhamento múltiplo.

Em princípio, a pontuação de um alinhamento múltiplo deve refletir o grau de semelhança de suas seqüências, de acordo com um modelo evolucionário dado. Há diversas formas de mensurar a pontuação (ou custo) de um alinhamento múltiplo.

Considere que uma seqüência é um conjunto ordenado de letras de um alfabeto Σ. Um alinhamento de n seqüências $S_1, S_2, \ldots, S_n$ pode ser definido como uma matriz $A = a(S_1, S_2, \ldots, S_n)$, onde cada entrada A_{ij} é uma uma letra de Σ ou um símbolo nulo (freqüentemente representado por "-"). A linha i de A é a seqüência S_i, depois dos *gaps* removidos.

No modelo mais simples, a pontuação de um alinhamento de n seqüências é definido como a soma das pontuações de suas colunas. Entretanto, este modelo é grosseiro, uma vez que cada coluna do alinhamento é considerada independentemente de seu contexto. Por exemplo, um *gap* de tamanho L é considerado o mesmo que L *indels* independentes.

Em modelos mais realísticos, um *gap* é considerado um único evento mutacional e associado com uma penalidade proporcional ao seu tamanho. Em tais modelos, pontuação para alinhamento de par é definida como a soma das pontuações de substituições e penalidades para *gaps*. Porém, a definição da pontuação de um alinhamento múltiplo é mais complexa.

Uma solução possível, conhecida como Soma dos Pares (SP) [43], consiste em calcular a pontuação do alinhamento múltiplo a partir das pontuações dos alinhamentos de pares. Um alinhamento múltiplo $a(S_1, S_2, \ldots, S_n)$ contém $n(n-1)/2$ alinhamentos de pares $a(S_i, S_j)$, onde $1 \leq i < j \leq n$ e onde desprezam-se todas as colunas onde há duas letras nulas. A

pontuação SP de um alinhamento múltiplo é definida como a soma de todas estas pontuações de projeções [43].

Pontuação SP simples, entretanto, pode ser inapropriada quando alguns grupos de seqüências são extremamente ou pouco representados em uma família. Este defeito pode ser corrigido pela introdução de um sistema de pesos apropriado [7, 81], que associa um peso a cada seqüência. Assim, pode-se dar um peso menor a seqüências de grupos super-representados. Outra solução consistem em usar uma função de pontuação baseada em uma árvore evolucionária. As folhas da árvore são as seqüências que nós queremos alinhar, e os nós internos são suas hipotéticas seqüências ancestrais.

Um alinhamento ótimo é um que possua pontuação máxima. Needleman e Wunsch [158] propuseram um algoritmo eficiente, baseado em programação dinâmica, para computar alinhamentos ótimos para pares de seqüências. Esta abordagem pode ser facilmente generalizada para mais que duas seqüências, porém o custo para alinhar, n seqüências com comprimento l cada, é muito elevado. Ele requer $O(2^n l^n)$ em tempo e $O(l^n)$ em espaço. Esta complexidade pode ainda ser mais elevada caso a penalidade para *gap* não seja linear.

Carillo e Lipman [43] propuseram um algoritmo *branch and bound* para computar um alinhamento de pontuação SP máxima. Faz uso de um limiar superior para a pontuação e, assim, impõe limites para o espaço e tempo. Esta abordagem foi implementada pelo programa MSA [136].

Conforme comentado anteriormente, uma função de pontuação alternativa poderia fazer uso de uma árvore evolucionária. Em princípio, seria uma função melhor que a SP, entretanto o problema de alinhamento usando uma árvore evolucionária é também difícil, mesmo quando a árvore é dada [114].

Alinhamento progressivo é o método mais comumente utilizado para alinhamento múltiplo de seqüências biológicas. Esta abordagem heurística é muito rápida, requer pouca memória e oferece um bom desempenho quando as seqüências são homólogas e relativamente bem conservadas [67, 209].

Alinhamento progressivo consiste em construir um alinhamento múltiplo a partir de alinhamentos de pares. Nesta abordagem é possível identificar três passos: computação dos alinhamentos de pares, construção da árvore guia a partir das distâncias obtidas no passo anterior e construção do alinhamento múltiplo guiado pela árvore.

As diferenças entre as ferramentas para MSA que fazem uso da abordagem progressiva estão nos métodos que utilizam em cada um dos três passos.

Clustal W [215], por exemplo, permite o uso de programação dinâmica ou métodos heurísticos para a computação do alinhamento de pares. Com programação dinâmica temos resultados mais precisos, mas com os métodos heurísticos é possível ganhar em velocidade de processamento.

Para a construção da árvore guia há diversos métodos, sendo UPGMA [201] e Neighbor-Joining [186] os mais conhecidos. É conhecido um problema no UPGMA, onde é dada uma ordem de ramificação incorreta quando as taxas de substituição variam em diferentes linhagens. Este problema levou a utilização de Neighbor-Joining, ao invés de UPGMA, nas versões mais recentes do Clustal (a partir do Clustal W).

O principal problema no terceiro passo consiste no alinhamento de dois alinhamentos. O método mais simples é reduzir cada alinhamento a uma seqüência consenso e usar um algoritmo de alinhamento de pares comum. Na maioria dos programas, representa-se ali-

nhamento com perfis, onde colunas são reduzidas a distribuições das freqüências de cada letra. Alinham-se dois perfis de forma similar a de duas seqüências por programação dinâmica. O alinhamento de dois perfis de tamanho l tem complexidade $O(a^2l^2)$, onde a é o tamanho do alfabeto. Clustal [103] usa alinhamento de consenso, mas Clustal W usa alinhamento de perfis com penalidades de *gap* para posição específica.

O principal problema da abordagem progressiva oriunda em sua natureza gulosa, onde qualquer erro cometido em etapas iniciais do processo não são mais corrigidos.

Estratégias de otimização iterativas têm sido propostas para resolver tal problema, tais como: RIW ou DNR [82]. Em testes, estes métodos apresentaram melhores desempenhos que o Clustal W, por exemplo. Porém, mesmo sendo mais rápido que algoritmos ótimos, estes são ainda muito lentos para entradas grandes.

Outra limitação para a abordagem progressiva é o requisito de computação das distâncias de pares entre todas as seqüências para construir a árvore guia. Pode-se em alguns casos ter de alinhar seqüências homólogas e alguns fragmentos não sobrepostos (por exemplo, seqüências parciais de proteína). Quando as seqüências não se sobrepõem, estas obviamente são não relacionadas e assim a árvore guia gerada pode ser completamente incorreta. O alinhamento produzido, neste caso, pode ser imprevisível.

As seqüências a comparar podem compartilhar blocos conservados, separados por regiões não conservadas contendo longos *indels*. Em tais casos, o resultado do alinhamento global ótimo ou progressivo irá depender fortemente da escolha dos parâmetros para penalidade de *gap*.

Uma alternativa consiste em buscar por blocos conservados para usar como âncoras no alinhamento das seqüências. Blocos são alinhamentos de fragmentos de seqüências (alinhamentos locais). A maioria dos métodos que fazem uso desta abordagem consideram apenas blocos sem *gaps*. Os blocos permitidos podem ser exatos (composto por segmentos idênticos) ou não exatos e podem ser uniformes (encontrado em todas as seqüências) ou não.

O primeiro programa de alinhamento múltiplo de blocos [202] usou um algoritmo de ordenação para computar blocos exatos uniformes. Algoritmos mais rápidos baseados em árvores de sufixo [143], ou estruturas de dados equivalentes, podem também ser utilizados para computar blocos exatos. Porém regiões homólogas raramente são perfeitamente conservadas.

ASSEMBLE [224] realiza uma análise em uma matriz de pontos em todos os pares de seqüências e então compara estas matrizes de pontos para encontrar blocos uniformes, não necessariamente exatos. Na prática, é comum alguns blocos que não estão presentes em todas as seqüências. Melhorias têm consistido no desenvolvimento de métodos que permitam blocos que não sejam necessariamente uniformes.

Um conjunto de blocos uniformes é consistente quando cada par de blocos está em ordem, ou seja, eles não se cruzam. O problema de selecionar um conjunto de blocos consistente e ótimo pode ser reduzido a um algoritmo clássico para caminho ótimo em um grafo [202], que requer tempo $O(M^2)$ para M blocos. Algoritmos mais rápidos (sub-quadráticos) para computação de conjuntos de blocos uniformes e consistentes ótimos foram propostos [157,240]. Porém, quando os blocos são não uniformes o problema é intratável [239]. A consistência de blocos não uniformes não pode ser reduzida para uma relação binária entre eles. Um conjunto de três blocos não uniformes, tal que todos seus três pares de blocos são consistentes,

não é necessariamente consistente.

Para computar um "bom" conjunto consistente de diagonais (numa matriz de pontos), DiAlign usa um algoritmo heurístico, que adiciona diagonais pela ordem decrescente de pontuação em um conjunto consistente de diagonais. Diagonais que não sejam consistentes com o conjunto corrente são rejeitadas.

As seqüências a comparar podem compartilhar regiões similares, mas não serem globalmente relacionadas. Estes módulos homólogos podem ocorrer em diferentes posições relativas e podem ser duplicados em diferentes seqüências. Em tais casos não é possível computar um alinhamento global, mas pode-se procurar por "bons" alinhamentos locais.

Neste contexto, há diversas ferramentas disponíveis. Dentre estas, estão as ferramentas PRALIGN [233] e MACAW [191]. Destaca-se que o primeiro possui um requisito elevado para espaço e o segundo requisito elevado para tempo de processamento.

Os programas para alinhamento local mais recentes são baseados em métodos estatísticos, que usam heurísticas computacionalmente eficientes para resolver problemas de otimização. Por exemplo, Gibbs Sampler [131] usa *Gibbs sampling* iterativo para encontrar blocos. O tempo de computação desta abordagem cresce linearmente com o número de seqüências de entrada. A ferramenta está disponível nos pacotes MACAW e Block Maker [96]. A ferramenta MEME [18] usa um algoritmo de maximização de expectativa (EM) [42, 132] para localizar padrões repetidos.

A.1.3 Comparação dos diferentes métodos

Quando as seqüências são similares (mais que 50% de identidade de pares de proteínas e mais de 70% para DNA) e são homólogas em seu tamanho completo, todos os métodos de alinhamento global produzem resultados bons. Além disso, em tais casos, qualquer conjunto razoável de parâmetros geram alinhamentos similares. Porém, quando pelo menos duas seqüências em uma dada família compartilham uma identidade menor, ou as regiões homólogas são interrompidas por *gaps* longos de tamanhos diferentes, o resultado do alinhamento pode variar consideravelmente de acordo com os programas e parâmetros utilizados.

Diversas análises comparativas de programas para alinhamento múltiplo têm sido publicadas [36, 82, 142, 153]. Estas comparações são baseadas na habilidade de detectar padrões de *motifs* em diversas famílias de proteínas ou baseadas em alinhamentos de referência derivados de estruturas tridimensionais de proteínas. Análises comparativas podem também ser baseadas no efeito dos programas de alinhamento múltiplo na filogenia.

Dependendo do tipo de entrada, há métodos mais adequados para se utilizar. Para alinhar duas seqüências, deve-se usar um método ótimo de alinhamento de pares tal como Lalign ou SIM [110].

Para mais que duas seqüências, geralmente tem-se de utilizar uma abordagem heurística. Como um primeiro passo, o usuário deve tentar computar o alinhamento múltiplo com um alinhador progressivo, pois são rápidos e demandam uma capacidade menor de memória. Dentre as ferramentas que utilizam esta abordagem, recomenda-se o Clustal W, que tem a capacidade de selecionar automaticamente a matriz de substituição durante o alinhamento e ajustar o peso de *gaps* ao longo do alinhamento, de acordo com a composição das seqüências. Caso este primeiro alinhamento mostre que todas as seqüências estão relacionadas a cada outra em seu comprimento completo, dificilmente qualquer outro método gerará um resultado melhor.

Porém, se há algumas seqüências altamente divergentes, *gaps* longos ou regiões pouco conservadas, recomenda-se comparar o resultado de diferentes métodos e/ou conjuntos de parâmetros. Métodos progressivos enfrentam grandes dificuldades com longos *gaps* devido ao esquema linear para peso de *gap*, que tende a aplicar uma penalização excessiva a *indels* longos. Métodos globais baseados em blocos, como o DiAlign [152,153] ou IterAlign [37], são menos sensíveis a esses *gaps* longos e são particularmente apropriados para tais situações.

Um outro defeito dos métodos progressivos, mas que também ocorre com os métodos de alinhamento globais ótimos, é que sempre produzem alinhamento mesmo quando as seqüências não são relacionadas. Já o DiAlign e IterAlign não tentam gerar um alinhamento global se as seqüências são apenas localmente relacionadas. Outra característica interessante destes alinhadores é a indicação de significância do alinhamento. No DiAlign, por exemplo, regiões que não são consideradas alinhadas (tal como em regiões não conservadas entre dois blocos alinhados) são impressas com letras minúsculas, enquanto que os resíduos alinhados estão em letras maiúsculas.

Métodos globais (ótimos, progressivos ou baseados em blocos) são apropriados apenas se todos os blocos conservados são consistentes. Se alguns domínios estão duplicados ou ordenados diferentemente ao longo das seqüências é necessário usar um método de alinhamento local para alinhar todos os domínios relacionados.

A.1.4 Visualizando e editando alinhamentos múltiplos

Resultados de alinhadores múltiplo são geralmente salvos em arquivos texto. Não há um formato padrão para MSA. Entretanto, o formato MSF é provido pelos mais populares programas de alinhamento e é reconhecido por muitos programas que requerem alinhamentos como entrada, tal como filogenia molecular ou buscas de perfis. O formato MASE apresenta a vantagem de permitir a inclusão de anotações do alinhamento como um todo ou específicas a cada seqüência.

Interfaces gráficas têm sido desenvolvidas para manipular e editar alinhamentos múltiplos. Geralmente permitem os usuários colorir ou sombrear resíduos de acordo com vários critérios, tais como propriedades físico-químicas, grau de conservação no alinhamento, hidrofobicidade ou estrutura secundária. O uso das cores ajuda muito na interpretação do MSA, dando uma visão muito mais abrangente da informação contida em um alinhamento múltiplo. Além disso, estas interfaces propõem diversos recursos interessantes, tais como verificação ou refino manual de alinhamentos, adição de anotações e extração de sub-alinhamentos.

Na Tabela 3 são listadas ferramentas para visualização e edição de MSAs. É importante salientar que a ferramenta Clustal X adiciona uma interface ao Clustal W, que permite uma visualização gráfica do alinhamento. Entretanto, esta não permite a edição dos alinhamentos.

Se o objetivo é apenas visualização e geração de figuras para implessão, há também uma série de ferramentas com este propósito, que são listadas na Tabela 4.

A.1.5 Conclusões

A grosso modo, o melhor alinhamento consiste naquele que seja o mais parecido com o cenário evolucionário (substituições, inserções e remoções). Diversos algoritmos de alinhamento

Jalview	http://www.jalview.org/
CINEMA	http://utopia.cs.manchester.ac.uk/cinema
SeaView	http://pbil.univ-lyon1.fr/software/seaview.html
MPSA	http://mpsa.ibcp.fr/
Se-Al	http://tree.bio.ed.ac.uk/software/seal/

Tabela 3: Editores de MSAs.

BoxShade	http://sourceforge.net/projects/boxshade/
WebLogo	http://weblogo.berkeley.edu/
MView	http://bio-mview.sourceforge.net/index.html
AMAS	http://www.compbio.dundee.ac.uk/www-amas/

Tabela 4: Ferramentas para visualização e geração de figuras para impressão de MSAs.

foram desenvolvidos, mas nenhum deles é ideal, ou seja, bom em qualquer situação.

Por causa de requisitos de tempo e memória, algoritmos que garantem o melhor alinhamento para um dado modelo evolucionário podem, na prática, ser utilizados apenas para um número muito restrito de pequenas seqüências. Assim, diversos algoritmos heurísticos têm sido propostos. Estes reduzem os requisitos de tempo e memória, mas não garantem alinhamentos ótimos.

A.2 Recent progresses in multiple sequence alignment: a survey

Este trabalho [162] apresenta uma breve descrição das técnicas existentes (até 2002) e expõe os pontos fortes e fracos dos pacotes mais utilizados para alinhamento múltiplo. Como citado pelo próprio autor, o texto pode ser visto como um complemento ao texto de Duret e Abdeddaim [59]. Há um outro trabalho de Cédric Notredame [163] onde ele apresenta o que foi visto de novidade no contexto de MSA entre 2002 e 2007.

Alinhamento de seqüências é sem dúvida a tarefa mais comum em bioinformática. Diversos procedimentos necessitam de comparação de seqüências, desde buscas em bancos de dados [9] até predição de estrutura secundária [183]. Seqüências podem ser comparadas aos pares numa varredura por seqüências homólogas em uma base de dados ou elas podem ser alinhadas simultaneamente para visualização do efeito da evolução em uma família de proteínas inteira. Neste estudo o foco estará na comparação global simultânea de mais que duas seqüências. Será dada uma maior ênfase nas técnicas mais recentes.

Alinhamentos múltiplos constituem um meio extremamente poderoso de revelar restrições impostas por estrutura e função na evolução de uma família de proteínas. Eles tornam possível a indagação de uma grande variedade de questões biológicas importantes, tais como as comentadas nos parágrafos a seguir.

Árvore filogenética é um dos instrumentos utilizado para elucidar relações evolucionárias existem entre organismos. Atualmente, a construção de árvores filogenéticas altamente

precisas fazem uso de dados moleculares. Para a produção de tais árvores, geralmente temos os seguintes passos:

- coleta de um conjunto de seqüências ortólogas;
- alinhamento múltiplo destas seqüências;
- mensuração das distâncias filogenéticas de pares no alinhamento múltiplo e computação da matriz de distâncias; e
- computação da árvore pela aplicação de um algoritmo de agrupamento [186, 201] utilizando a matriz de distâncias como entrada.

Alternativamente, os dois últimos passos podem ser substituidos por uma computação de máxima semelhança [185]. Em qualquer dos casos, o papel do MSA é prover uma estimativa muito precisa de distância de pares e possibilitar estimar a confiabilidade de cada ramo através de *bootstrapping* [66].

MSAs possibilitam a identificação de *motifs* preservados pela evolução. *Motifs* desempenham um papel importante na estrutura e função de um grupo de proteínas relacionadas. Quando trabalhando com dados experimentais, estes *motifs* constituem um meio muito poderoso de caracterizar seqüências de função desconhecida. Bancos de dados como PROSITE [22] e PRINTS [15] fazem uso deste princípio. No caso de um *motif* ser muito sutil para ser definido com modelos padrões, pode-se fazer uso de outros tipos de descritores tais como perfil [86] ou HMM (do inglês, *hidden Markov model*) [93]. Estes permitem uma sumarização exaustiva (coluna por coluna) das propriedades de uma família ou domínio de proteínas. Perfis e HMMs permitem a identificação de membros muito distantes de uma família de proteínas. Suas sensibilidade e especificidade são muito maiores que as providas por uma única seqüência ou um modelo. Na prática, pode-se derivar um perfil de alinhamentos múltiplos usando ferramentas tais como: PFTOOLS [140], coleções pré-definidas como Pfam [27], ou computar os perfis com PSI-BLAST [11]. É importante salientar que a sensibilidade e especificidade dos perfis dependerão da qualidade biológica dos alinhamentos múltiplos dos quais oriundam.

Predição de estrutura é outro importante uso de alinhamentos múltiplos. Predição de estrutura secundária e terciária objetivam a predição do papel que um dado resíduo desempenha na estrutura da proteína (oculto ou exposto, hélice ou fita, etc.). Predições de estruturas secundárias baseadas em uma única seqüência possuem uma precisão baixa (cerca de 60%) [71], enquanto que predições baseadas em MSAs são muito mais precisas (cerca de 75%) [116, 182, 183]. Isso ocorre porque padrões de substituição observados em uma coluna refletem diretamente o tipo de restrições impostas naquela posição no curso da evolução.

No contexto de determinação de estrutura terciária ou predição de contatos não locais, alinhamentos múltiplos podem também ajudar na identificação de mutações correlacionadas. Esta abordagem tem dado resultados limitados quando aplicada a proteínas [72], mas tem tido sucesso em análises de RNA [92].

Como pode ser notado, estas importantes aplicações explicam o porque se dedica tanta atenção ao problema de MSA. Infelizmente, as ferramentas disponíveis são heurísticas que muitas das vezes provêem soluções boas, mas não necessariamente ótimas, para o problema.

Alinhamento múltiplo de seqüências (MSA) é uma tarefa complexa, que implica em três dificuldades técnicas distintas: escolha das seqüências, escolha da função objetivo e otimização da função.

Os métodos tratados neste texto só fazem sentido se assumido que o conjunto de seqüências é composto por seqüências homólogas. E quando tratarmos de métodos de alinhamento global, com exceção do DiAlign [152], há ainda o requisito de que as seqüências sejam semelhantes em todo o seu comprimento, ou pelo menos quase todo, para que o alinhamento de resultado tenha alguma relevância. Para os casos em que não se cumpre este requisito, deve se considerar o uso de métodos para MSA locais, tais como: Gibbs Sampler [131], Match-Box [51] e MACAW [191].

Para maiores informações sobre os problemas que podem surgir devido a uma má escolha das seqüências a alinhar, consultar trabalho de Henikoff [94]. É comum o uso de uma das ferramentas BLAST (WU-BLAST, PSI-BLAST, Gapped BLAST, etc.) [11] para avaliar o nível de relacionamento entre as seqüências do conjunto. Tais ferramentas utilizam modelos estatísticos, desenvolvidos por Altschul e Karlin [119], para estimar o grau de relacionamento entre as seqüências. Claro que estes modelos são uma mera aproximação da realidade biológica.

A função objetivo (OF, do inglês *objective function*) deve permitir uma quantificação da qualidade de um alinhamento, ou seja, dado um alinhamento de entrada, gerar um valor, indicativo da qualidade, de saída e assim permitir a comparação de alinhamentos. Como conseqüência permita a seleção daquele alinhamento que tenha um maior significado biológico. Dada uma OF "perfeita", o alinhamento ótimo matemático deveria ser um alinhamento ótimo biológico.

Na prática tal OF "perfeita" não existe e mesmo que existisse, a tarefa de demonstrar que é "perfeita" já se trata de um grande problema. Na teoria, uma OF deveria incorporar tudo o que é conhecido acerca das seqüências, incluindo sua estrutura, função e história evolucionária. Entretando, este tipo de informação raramente está disponível e, mesmo que esteja, é de difícil utilização. Desta forma, é comum utilizar similaridade de seqüência através da simples função de soma-dos-pares com pesos e função afim para penalidade de *gaps* [10].

Neste modelo, cada seqüência recebe um peso proporcional ao quanto de informação independente ela contém [7] e a pontuação do alinhamento múltiplo é equivalente à soma das pontuações de todas as substituições de pares com peso. As pontuações para substituições, por sua vez, são calculadas usando um modelo evolucionário predefinido conhecido como matrizes de substituição [50,74,95,193], no qual uma pontuação é associada a toda possível substituição ou conservação de acordo com a realidade biológica observada. Inserções ou remoções são pontuadas através de funções afim para penalidade de *gap*, que diferenciam abertura e extensão de *gap*. Normalmente há um custo elevado para a abertura e um custo inferior para a extensão do *gap*, que aumenta lentamente de acordo com o incremento no comprimento.

Um problema que surge então é a determinação destes custos (abertura e extensão). Estes valores são determinados empiricamente e podem variar de um conjunto de seqüências para outro [225]. Embora esta OF seja claramente incorreta do ponto de vista evolucionário [10], por assumir que toda seqüência é ancestral de toda outra no conjunto, é popular nos métodos MSA mais utilizados [80,136,215], devido a facilidade de implementação. Em

2000, Gonnet e colegas propuseram uma variação de soma-dos-pares que parece ser mais fidedigna aos eventos evolucionários [76].

Trabalhos recentes têm feito uso de OFs que parecem menos sensíveis à penalidade de *gap* graças a incorporação de informações locais. Neste contexto podemos citar o DiAlign [152] e o T-COFFEE [165]. O primeiro inclui avaliação baseada em segmento, já o segundo usa uma OF baseada em consistência. Outros trabalhos fazem uso de modelos ocultos de Markov (HMMs, do inglês *hidden Markov models*) [23,93]. HMMs descrevem o alinhamento múltiplo em um contexto estatístico usando uma abordagem Bayesiana. Embora de um ponto do vista formal estes métodos parecerem soluções mais atrativas, suas performances para alinhamentos *ab initio* decepcionam. É importante, entretando, salientar que um trabalho recente [120] exibiu uma solução que faz uso de HMM e foi capaz de produzir melhores resultados que o Clustal W. Outros métodos baseados em estatística que tentam associar um *P-value* ao alinhamento múltiplo foram descritos [100, 131]. Infelizmente estas medidas são restritas a MSAs sem *gaps*.

Em um mundo ideal, uma OF perfeita deveria existir para todas as situações. Na prática, isso não ocorre e, assim, o usuário terá sempre de tomar a decisão de qual o método mais apropriado para o conjunto de seqüências que dispõe.

Assumindo que temos um conjunto adequado de seqüências e uma OF biologicamente perfeita, a computação de um alinhamento múltiplo matematicamente ótimo é ainda uma tarefa complexa. O custo computacional de um método exato de MSA é tão elevado que inviabiliza seu uso para conjuntos de dados reais [117, 231]. Desta forma os algoritmos utilizados, na prática, para MSA são heurísticas que não garantem um alinhamento ótimo como resultado.

Considerando as propriedades mais óbvias dos algoritmos, pode-se classificá-los em três categorias: exato, progressivo e iterativo. Os algoritmos exatos são heurísticas de alto grau de qualidade que entregam freqüentemente alinhamentos muito próximos ao ótimo [136,206], em alguns casos, com limitantes bem definidos. Estes possuem fortes restrições quanto ao número de seqüências e OFs.

Alinhadores progressivos são, sem dúvida, os mais utilizados [48, 103, 165]. Estes têm como principal característica a dependência de uma montagem progressiva do alinhamento múltiplo [67,105,209] onde, duas a duas, seqüências (ou alinhamentos) são alinhadas através de programação dinâmica [158]. Dentre as vantagens desta abordagem estão velocidade, simplicidade e sensibilidade razoável.

Já os algoritmos iterativos dependem de um algoritmo capaz de produzir alinhamentos iniciais. Cabe a eles a tarefa de refinar tais alinhamentos através de uma série de ciclos (iterações) até que não seja mais possível incrementar a qualidade do alinhamento. Métodos iterativos podem ser determinísticos ou estocásticos, dependendo da estratégia utilizada. Os métodos mais simples são determinísticos. Sua estratégia implica em extrair seqüências, uma a uma, de um alinhamento múltiplo e então realinhar [25,82]. É possível ainda a existência de métodos que usam estratégias mistas progressiva e iterativa ao mesmo tempo [97]. Métodos iterativos estocásticos incluem HMM [127], *simulated annealing* [125] e algoritmos genéticos [12,41,45,77,164,237]. A principal vantagem destes é permitir uma boa separação conceitual entre processo de otimização e OF.

A.2.1 Revisão

Nos últimos anos, MSA tem sofrido drásticas evoluções com a introdução de diversos novos algoritmos e novos métodos de avaliação destes algoritmos. A Tabela 5 lista alguns dos métodos para MSA. Dentre estes, dois novos pontos têm emergido:

- o crescente interesse em estratégias de otimização iterativa; e
- o uso de esquemas de pontuação baseados em consistência.

Nome	**Algoritmo**	**Ref.**
MSA	Exato	[136]
DCA	Exato (requer MSA)	[206]
OMA	DCA iterativo	[181]
Clustal W	Progressivo	[215]
MultiAlign	Progressivo	[48]
DiAlign	Baseado em consistência	[152]
ComAlign	Baseado em consistência	[40]
T-COFFEE	Baseado em consistência/progressivo	[165]
PRALINE	Iterativo/progressivo	[97]
IterAlign	Iterativo	[37]
PRRP	Iterativo/estocástico	[82]
SAM	Iterativo/estocástico/HMM	[60]
HMMER	Iterativo/estocástico/HMM	[127]
SAGA	Iterativo/estocástico/GA	[164]
GA	Iterativo/estocástico/GA	[237]

Tabela 5: Alguns métodos para MSA.

Nesta seção, revisa-se alguns destes novos algoritmos com suas principais características e potenciais falhas. Outro ponto importante em MSA nos trabalhos recentes são métodos baseados em HMM [23,93]. Para maiores informações acerca de métodos HMM para MSA, consulte o livro de Durbin e colegas [58].

Alinhamento progressivo é uma das maneiras mais simples e efetivas para a realização de alinhamentos múltiplos com pouco requisito de tempo e memória. Esta abordagem foi inicialmente descrita por Hogeweg e Hesper [105] e depois reinventada por Feng e Doolittle [67] e Taylor [209]. Os pacotes mais utilizados para MSA são baseados nesta abordagem, incluindo: Pileup (componente do pacote GCG [52]), MultiAlign [48] e Clustal W [215]. Estes tornaram-se os métodos padrões para alinhamento múltiplo.

Clustal W implementa um algoritmo determinístico não-iterativo que tenta otimizar o peso de uma soma-de-pares com função afim para penalidade de *gap*. É uma estratégia de alinhamento progressivo direto onde as seqüências são adicionadas uma a uma de acordo com a ordem indicada por um dendrograma pré-computado. A adição da seqüência é realizada através de um algoritmo de alinhamento de pares [158]. Seu principal defeito é que uma vez que uma seqüência foi alinhada, nunca mais o alinhamento será modificado, mesmo

em caso de conflito com uma seqüência a ser adicionada ao alinhamento. Clustal W inclui uma variedade de heurísticas altamente especializadas destinadas a máxima exploração de informações acerca da seqüência. Com isso, o Clustal W diferencia-se da maioria dos outros alinhadores progressivos por prover:

- penalidade de *gap* local;
- escolha automática da matriz de substituição;
- ajuste automático de penalidade para *gap*; e
- retardo do alinhamento de seqüência com baixo grau de relacionamento.

Em testes de *benchmark*, realizados com o BAliBASE [218], observou-se que, em geral, o Clustal W executa melhor quando a árvore filogenética é relativamente densa, sem qualquer esboço óbvio. Para ele não importa o quão longa são as seqüências, mas sim o quão similares elas são. Longas inserções ou deleções causam problema, devido a uma limitação intrinseca do esquema de penalidade de *gap* usado.

O último avanço realizado em algoritmos progressivos é o T-COFFEE, uma nova estratégia onde utiliza-se uma OF baseada em consistência. Tal função permite minimizar potenciais erros, especialmente em estágios iniciais da montagem do alinhamento. Maiores detalhes na seção sobre algoritmos baseados em consistência.

Como mencionado, alinhamento progressivo é apenas uma solução aproximada. Para obter uma solução precisa, é necessário alinhar todas as seqüências simultaneamente. Este tipo de procedimento pode ser especialmente útil quando estiver tratando de um conjunto de seqüências extremamente divergente, onde todos os alinhamentos de pares parecem estar incorretos. Infelizmente, para realizar tal tarefa necessita-se generalizar o algoritmo de Needleman e Wunsch [158] para um espaço multidimensional e por razões práticas (tempo e memória) isto torna o número de seqüências aceitável no conjunto de entrada extremamente restrito.

O programa MSA faz exatamente isso, mas tenta minimizar as restrições por identificar e desprezar porções do hiperespaço que não contribuem para a solução. MSA implementa o algoritmo de Carillo e Lipman [43]. É importante salientar que MSA é apenas uma implementação heurística do algoritmo de Carillo e Lipman e não garante o ótimo matemático. Este pacote não é muito usado devido a seu alto requisito de tempo e memória, além das limitações no número de seqüências.

Stoye e colegas [206] descreveram um novo algoritmo por divisão e conquista, DCA, que estendeu as capacidades do MSA original. O algoritmo DCA fragmenta as seqüências em segmentos que são pequenos o suficiente para utilizar o MSA. Posteriormente, resta ao DCA montar o alinhamento a partir dos sub-alinhamentos. Testes no BAliBASE indicaram que a estratégia DCA produz resultados ligeiramente melhores que o Clustal W. É importante salientar, entretanto, que limitações ainda existem e que o DCA não foi capaz de realizar o alinhamento de quatro conjuntos de testes do BAliBASE, exatamente por estas limitações.

Recentemente, uma implementação iterativa do DCA, OMA [181], foi descrita. Através do OMA pretende-se tornar a estratágia DCA mais rápida e reduzir os requisitos de memória.

Estes são baseados na idéia de que a solução para um dado problema pode ser computada por modificar uma solução sub-ótima já existente. Cada passo de "modificação" é uma

iteração e modificações podem ser realizadas utilizando programação dinâmica ou vários protocolos aleatórios. Faz-se, nestes, uma distinção entre estocásticos e determinísticos. Por estocástico aqui referimo-nos a métodos estocásticos iterativos tradicionais tais como SA (do inglês, *simulated annealing*) [146] ou GA (do inglês, genetic algorithm) [106].

SA foi o primeiro método iterativo estocástico descrito para alinhamento múltiplo. Vários esquemas foram publicados [111, 125] e todos envolvem a mesma cadeia de processos: modifica aleatoriamente um alinhamento, calcula a pontuação, decide se mantém ou descarta a modificação de acordo com função de aceitação que torna-se mais exigente com o aumento no número de iterações. O procedimento é repetido até que seja satisfeito um critério de término. É importante observar que SA tem se mostrado pouco eficiente, em termos de tempo, para a construção de alinhamentos *ab initio*, mas tem se mostrado um bom *improver* de alinhamentos.

GAs constituem uma interessante alternativa aos SAs como mostrado em SAGA [164]. Assim como SA, SAGA é uma caixa preta de otimização na qual qualquer OF pode ser testada. Seu esquema é direto e segue a risca o GA simples [73]. SAGA usa a OF definida como sua função de aptidão, suas mutações inserem ou deslocam *gaps* e cruzamentos combinam o conteúdo de dois alinhamentos. Ao todo possui 20 operadores, que competem pelo uso. SAGA não garante otimalidade, mas testes têm mostrado que é capaz de produzir resultados próximos ou até melhores que o MSA (usando a mesma OF). Os testes também mostraram que GAs são lentos para alinhamentos múltiplos, no uso do dia-a-dia. É importante salientar que os operadores implementados independem da OF utilizada e, assim, é fácil realizar a substituição da OF no SAGA.

Posteriormente, Zhang e Wong [237] implementaram uma nova ferramente que faz uso de GAs. Eles reportaram um alto grau de eficiência da ferramenta, mas deve-se observar que o método é dirigido pela presença de segmentos completamente conservados para guiar a montagem do alinhamento. A suposição da sempre existência de tais segmentos não é realística. Este método parece ser apropriado para alinhamento de seqüências muito longas e com alto grau de similaridade.

SAGA foi paralelizado por dois grupos independentemente [12, 167] em busca de eficiência. Novos métodos para MSA foram implementados baseando-se em princípios descritos no SAGA [41, 45, 77].

O pacote Gibbs Sampler [131] faz uso de *Gibbs Sampling*, que é uma outra interessante estratégia iterativa estocástica. Ele implementa um método de alinhamento local que encontra *motifs* sem *gaps* ao longo de um conjunto de seqüências não alinhadas. Da perspectiva de alinhamento múltiplo, sua característica mais interessante está em sua OF. O algoritmo objetiva construir um alinhamento com um bom *P-value* (por exemplo, baixa probabilidade de ter sido gerado por acaso). A cada iteração, adicionam-se ou removem-se segmentos de acordo com a probabilidade do modelo corrente poder gerar eles. Se aquela probabilidade é suficientemente alta, o modelo é atualizado com os novos segmentos e o algoritmo passa a próxima iteração.

Esta idéia Bayesiana de simultaneamente maximizar os dados e o modelo é também central a HMMs [23, 93] e, desta forma, HMMs também podem podem ser treinadas por maximização de expectativa [60, 127]. Assim como GAs, HMMs apresentaram resultados decepcionantes para a montagem de alinhamentos *ab initio*. Hoje, HMMs tais como aqueles encontrados no Pfam [27] não são mais gerados a partir de seqüências não alinhadas. Atu-

almente, os métodos estão mais inclinados a transformar um alinhamento pré-computado em um HMM e então submetê-lo a um refinamento usando HMMER [127] ou SAM [60].

O primeiro algoritmo iterativo não estocástico data da origem de MSAs [25]. Sua estratégia consiste em re-alinhamentos para corrigir possíveis erros em estágios anteriores, para tanto faz uso de algoritmos padrões de alinhamento com programação dinâmica [158]. O procedimento é encerrado quando as iterações falham consistentemente na tentativa de melhorar o alinhamento. Este é o procedimento utilizado pela maioria das estratégias descritas até o início da década de 1990.

A principal variação nestes algoritmos encontra-se na forma como as seqüências são divididas em dois grupos antes de serem re-alinhadas. Por exemplo, em AMPS [25] as seqüências são escolhidas de acordo com sua ordem de entrada e re-alinhadas uma a uma. Já no algoritmo de Berger e Munsen [32] a escolha é feita de maneira aleatória e as seqüências são divididas em dois grupos que podem conter mais que uma seqüência. A introdução da aleatoriedade na seleção dos grupos torna o algoritmo mais robusto e incrementa sua precisão. Poucos destes métodos iterativos iniciais têm sido efetivamente avaliados por ferramentas de *benchmark*, o que torna difícil uma estimativa de suas verdadeiras significâncias biológicas.

O mais sofisticado algoritmo iterativo baseado em programação dinâmica foi descrito recentemente por Gotoh [82] e é conhecido por PRRP. Trata-se de uma estratégia iterativa de duplo aninhamento com aleatorização que otimiza uma pontuação de soma-de-pares com peso e função afim para penalidade de *gap*. Sua originalidade está na otimização simultânea de pesos e alinhamento. A iteração interna otimiza a soma-de-pares com peso enquanto a iteração externa otimiza os pesos que são calculados em uma árvore filogenética estimada do alinhamento corrente [7]. O algoritmo é encerrado quando os pesos convergem. PRRP foi o primeiro programa para alinhamento múltiplo a ser extensivamente avaliado por *benchmark*, usando JOY [148], um banco de dados de alinhamentos estruturais. Posteriormente os resultados foram confirmados no BAliBASE [165, 218]. PRRP apresenta resultados significativamente melhores que a maioria dos métodos progressivos tradicionais, assim como a maioria dos métodos iterativos recentes como podemos observar na Tabela 6.

Método	**Ref1**	**Ref2**	**Ref3**	**Ref4**	**Ref5**	**Total**
DiAlign	71.0	25.2	35.1	74.7	80.4	57.3
Clustal W	78.5	32.2	42.5	65.7	74.3	58.7
PRRP	78.6	32.5	50.2	51.1	82.7	59.0
T-COFFEE	80.7	37.3	52.9	83.2	88.7	68.7

Tabela 6: Avaliação do DiAlign, Clustal W, PRRP e T-COFFEE nos 5 conjuntos de referência do BAliBASE. A pontuação aqui utilizada indica a porcentagem de colunas confiáveis, do alinhamento de referência, que foram corretamente alinhadas. As colunas Ref1 a Ref5 da tabela correspondem a uma média das pontuações obtidas pelo alinhador nos conjuntos de teste da respectiva categoria do BAliBASE. A coluna total apresenta uma média do alinhador nestas 5 categorias.

Dois outros métodos de alinhamento iterativo foram recentemente descritos: PRALINE [97] e IterAlign [37], que compartilham protocolos muito similares. Quanto a suas potenciais performances, nenhum deles foi adequadamente avaliado. Todavia deve ser dado ênfase a novos conceitos que eles incorporaram:

- uso de informação local no IterAlign, que objetiva um decremento na sensibilidade a parametrização de penalidade de *gap*; e
- o conceito de consistência.

A busca por consistência tem se tornado um dos pontos mais fortes no desenvolvimentos recentes em MSA. Tal conceito é, também, central nos métodos não-iterativos.

O primeiro método MSA baseado em consistência foi descrito por Kececioglu na década de 1990 [124]. Dado um conjunto de seqüências, o MSA ótimo é definido como aquele que está de acordo com a maioria de todos os possíveis alinhamentos ótimo de pares. Computar tal alinhamento é um problema NP-Completo que pode ser resolvido apenas para um pequeno número de seqüências relacionadas, usando um algoritmo tal como MSA. Todavia, há pelo menos três boas razões que fazem OFs baseadas em consistência muito interessantes:

- não dependem de uma matriz de substituição específica, mas de qualquer método, ou coleção de métodos, capaz de alinhar duas seqüências por vez;
- o esquema baseado em consistência é dependente de posição, dada a coleção de alinhamentos de pares. Isso significa que a pontuação associada com o alinhamento de dois resíduos dependem de seus índices ao invés de sua natureza individual; e
- experiências mostraram que dado um conjunto de observações independentes, a maioria dos consistentes estão freqüentemente próximos da verdade.

Embora a primeira OF baseada em consistência tenha sido descrita em 1993 [124], passaram-se ainda alguns anos para o desenvolvimento de algoritmos heurísticos capazes de tratar sua otimização. Apenas recentemente um GA (SAGA [164]) foi usado para mostrar os avanços biológicos de tal função, denominada COFFEE [166], que emula o problema do *trace* de peso máximo. No SAGA-COFFEE, a coleção de alinhamentos de pares com peso é nomeada uma biblioteca e o SAGA é utilizado para computar o alinhamento que tem o mais alto nível de consistência com a biblioteca. Na prática, a biblioteca pode conter mais que um alinhamento para cada par de seqüências. A informação que ela contém pode ser redundante, conflitante e pode se originar de fontes que variam tanto quanto se desejar (análise de estrutura, comparação de seqüências, busca em bases de dados, conhecimento experimental, etc.).

Embora o SAGA-COFFEE tenha obtido resultados interessantes, o problema do desempenho ainda persistia. Isso levou ao desenvolvimento de um novo algoritmo heurístico para otimizar a função COFFEE de maneira que fosse eficiente em termos de tempo: T-COFFEE [165]. No T-COFFEE, a biblioteca COFFEE é transformada na biblioteca estendida, uma matriz de substituição para posição específica onde a pontuação associada a cada par de resíduos depende da compatibilidade daquele par com o resto da biblioteca. T-COFFEE faz uso de um procedimento reminiscente da multiplicação de matrizes de pontos de Vingron [224] e da sobreposição de pesos de Morgenstern [153]. O alinhamento múltiplo é montado usando um algoritmo de alinhamento progressivo semelhante ao Clustal W.

A principal diferença entre T-COFFEE e Clustal W está na biblioteca estendida, que assume o lugar da matriz de substituição. Outra característica importante do T-COFFEE é que a biblioteca primária é criada de uma combinação de alinhamentos globais (produzidos pelo Clustal W) e alinhamentos locais (produzidos com Lalign). Testes realizados através do BAliBASE mostraram que a combinação de informação global e local permite ao T-COFFEE superar o PRRP, Clustal W e DiAlign nas cinco categorias de conjuntos de testes contidos neste banco de dados de referência [165].

Embora diferente nos detalhes, T-COFFEE e DiAlign [152], um outro algoritmo baseado em constência que tenta usar informação local para guiar um alinhamento múltiplo global, possuem alguma similaridade. No geral, DiAlign não é tão preciso quanto o Clustal W ou o PRRP, mas é superior nas categorias 4 e 5 do BAliBASE, que requer que inserções muito longas sejam alinhadas adeqüadamente [165,218]. O algoritmo do DiAlign já sofreu modificações para melhorar a eficiência [134].

Um outro método que trabalha com consistência é o ComAlign [40], que é capaz de combinar diversos alinhamentos múltiplos em um único alinhamento, freqüentemente mais preciso.

A.2.2 Conclusões

No início da década de 1990, quando os métodos para MSA começavam a se desenvolver, havia poucos dados disponíveis e o principal problema era usar todo o tipo de informação disponível. Esta situação tem se alterado dramaticamente. Hoje dispomos de uma grande quantidade de dados de uma grande variedade de tipos e facilmente acessíveis através de bancos de dados públicos disponíveis na *web*. Como conseqüência surgiu o desafio de ter de selecionar o que deve ser adicionado ou não como entrada no seu método. Há duas boas razões para não usar todas as seqüências disponíveis:

- alinhar um grande número de seqüências requer muito tempo de computação, além de ser difícil de analisar; e
- embora as soluções existente usem esquemas de peso que visam minimizar o efeito de seqüências similares ou altamente correlacionadas, nenhuma delas é inteiramente satisfatória e sub-grupos super-representados acabam por dominar o alinhamento ou o perfil.

Uma trimagem cuidadosa feita pelo usuário é ainda a melhor forma disponível de minimizar tais efeitos.

Outra grande alteração que ocorreu nos últimos anos refere-se ao crescente número de estruturas 3D disponíveis. Embora a proporção de seqüências de proteínas com uma estrutura 3D conhecida esteja ficando cada vez menor, a situação é muito diferente sob a perspectiva de famílias de proteínas onde, pelo menos, um membro tem a estrutura 3D conhecida, esta cresce regularmente. Isso significa que na maioria dos casos o alinhamento múltiplo pode se beneficiar da incorporação da informação da estrutura 3D objetivando reconhecer homologias remotas ou guiar a escolha de penalidades locais [112].

Poucas implementações disponíveis são capazes de combinar seqüências e estruturas em um alinhamento múltiplo. Enquanto o Clustal W é capaz de usar informações de estrutura secundária do Swiss-Prot [21] para estimativa de penalidade de *gap*, falta uma

ferramenta apropriada para alinhamento simutâneo de seqüências e estruturas. Dois dos métodos introduzidos aqui são bons candidatos para tal tarefa. Os algoritmos baseados em consistência têm a vantagem de possuírem poucos requisitos na composição de suas bibliotecas. Por exemplo, o banco de dados de alinhamento múltiplo estrutural DALI [53] usa T-COFFEE para montar a coleção de alinhamentos de pares produzidos pelo algoritmo DALI no alinhamento múltiplo. O algoritmo de programação dinâmica dupla introduzido por Taylor [212] é outro bom candidato para este propósito. Já havia sido mostrado que tal solução era adequada para alinhamentos de estruturas [168]. Resultados recentes indicam que ela poderia ser usada no contexto de MSA e possivelmente no contexto da combinação de seqüências e estruturas [64].

Um grande obstáculo na construção de MSAs é o processamento de repetições. Repetições (*in tandem* ou não) causam confusão a todos os métodos existentes para MSA. No caso de existir repetições nas seqüências de entrada, a única solução é o pré-processamento das seqüências com extração destes e apenas realizar o alinhamento de regiões homólogas. Esta extração pode ser realizada por ferramente de alinhamento múltiplo local tal como Gibbs Sampler [131], Mocca [161] ou Repro [99]. Infelizmente nenhum deles está bem integrado com um procedimento de alinhamento múltiplo global. Conta a favor do Gibbs Sampler e do Mocca o fato de proverem uma estimativa de relevância biológica de suas saídas.

Um quarto ponto de relevância no contexto de MSA refere-se a sua computação em si. A paralelização de algoritmos para MSA continua sendo uma tarefa difícil. As operações contidas na implementação de tais algoritmos requerem esquemas complexos de compartilhamento de memória que não são suportados por alguns *clusters*, tais como os `Linux`. Quando trabalha-se com grandes conjuntos de dados compostos por seqüências longas, ainda são necessários super-computadores para a computação do alinhamento múltiplo.

Um último ponto importante é a estimativa de precisão local. Dentre todos os métodos apresentados neste texto, não há um em particular que seja o melhor. Todos podem ser superados em determinadas situações. Por esta razão, mostra-se ser mais importante ter a capacidade de calcular o nível exato de precisão de um alinhamento ao invés de melhorar as performances médias de cada método. Dentre os métodos tem-se conhecimento apenas de quatro que são capazes de realizar tal estimativa: Clustal X [214], PRALINE [97], T-COFFEE [165] e Match-Box [51]. Entretanto, é importante salientar que em nenhum dos casos estas estimativas foram apropriadamente avaliadas.

Um alinhamento múltiplo é um modelo muito restrito e uma maneira poderosa de visualizar inconsistência em conjuntos de dados, assim como permite a visualização de relacionamentos que podem existir em pedaços de informação aparentemente independentes.

Métodos para MSA têm de evoluir. Eles precisam integrar informações heterogêneas tais como estruturas, resultados de buscas de bancos de dados, dados experimentais e qualquer informação que venha de dados de expressão e análise proteômica, incluindo informação regulatória. Integrar tais informações é uma tarefa complexa, pois quando trabalha-se com dados heterogêneos, conhecer quem está certo e quem está errado torna-se uma arte. Resolver este tipo de questão é difícil ao mesmo tempo que essencial. Métodos apropriados terão de fazer isso de uma forma transparente, permitindo ao usuário controlar todas as informações extras que serão utilizadas durante o alinhamento. Este método ideal deveria também permitir, ao usuário, adicionar ao modelo informações oriundas de seu próprio conhecimento.

De qualquer forma, estes métodos futuros continuarão a requerer muita memória e processamento. Comparado com buscas em bancos de dados, alinhamento mútiplo é difícil de otimizar. Poderá ser necessário o desenvolvimento de um *hardware* especial e conseqüentemente, reprojetar as implementações correntes.

Na área de genômica comparativa, a comparação simultânea de um grande número de objetos biológicos homólogos irá se tornar cada vez mais importante e continuará sendo central para análises biológicas.

A.3 Recent evolutions of MSA algorithms

Este trabalho [163] aborda os últimos trabalhos na área, onde aparecem métodos tais como meta-métodos e baseados em modelos. O mesmo autor havia publicado em 2002 um *survey* [162] com os trabalhos mais antigos.

Um número crescente de métodos para modelagem biológica têm uma forte dependência com um MSA preciso, mas a tarefa de construir um bom MSA não é trivial e nenhum dos métodos existentes consegue construir MSAs biologicamente perfeitos.

Calcular MSAs exatos é computacionalmente inviável, assim na prática há heurísticas que maximizam a similaridade. Freqüentemente a relevância biológica dos resultados de um alinhador é avaliada pela comparação do alinhamento com coleções pré-estabelecidadas de alinhamentos. Tais comparações são baseadas na estrutura e as coleções são chamadas "*gold standards*" [34]. Tais ferramentas têm dado efeito na evolução dos algoritmos, levando a alinhamentos estruturalmente corretos. Esta evolução também tem coincidido com uma harmonização nos algoritmos, onde agora a maioria utiliza a abordagem progressiva [105].

O esquema de pontuação é o componente de maior influência nos resultados dos algoritmos progressivos. Tais esquemas podem ser divididos em duas categorias: baseados em matriz ou baseados em consistência. Os algoritmos baseados em matriz usam uma matriz de substituição para avaliar o custo de relacionar dois símbolos. São exemplo de algoritmos baseados em matriz o Clustal W [215], MUSCLE [62] e Kalign [130]. Os algoritmos baseados em consistência incorporam um maior conjunto de informações na avaliação. Inicialmente tal método foi utilizado no T-COFFEE [165], inspirado no DiAlign [152]. Depois surgiram outros algoritmos que utilizam a mesma idéia. O PCMA [177] reduz os requisitos computacionais exigidos pelo T-COFFEE. O ProbCons [54] adiciona o uso de consistência Bayesiana e modelos ocultos de Markov. MUMMALS [175] combina o esquema de pontuação do ProbCons com a estratégia do PCMA. Outro exemplo ainda de algoritmo baseado em consistência é o MAFFT [121]. Estudos têem indicado que os métodos baseados em consistência são mais precisos que os métodos baseados em matriz, mas eles tipicamente requerem um maior tempo de processamento.

Há também os métodos de consenso, tal como usado pelo M-COFFEE [228]. Este algortimo é um meta-método consenso baseado no T-COFFEE. Inicialmente são utilizados diversos métodos MSA para calcular alinhamentos alternativos. Então utiliza-se o T-COFFEE para calcular um MSA final, que seja consistente, utilizando-se os alinhamentos gerados inicialmente. M-COFFEE tem como vantagem uma execução mais rápida. Desconsiderando os algoritmos meta-métodos, o ProbCons é o mais rápido [228]. Quando inserimos o M-COFFEE neste grupo de algoritmos, M-COFFEE é mais rápido que o ProbCons. É importante salientar, entretanto, que o M-COFFEE enfrenta problemas a medida que as

seqüências se tornam menos homólogas. Sua principal vantagem é a habilidade de estimar a consistência local entre o alinhamento final e os MSAs combinados.

Para tentar resolver este problema com seqüências pouco homólogas, uma alternativa apontada é incorporar informações relativas às seqüências. Seguindo esta abordagem temos os métodos de alinhamento baseados em modelos [14], que podem ser implementados com extensão estrutural ou extensão por homologia. Extensão estrutural foi inicialmente descrita por Taylor [208]. Nele tenta-se identificar um modelo estrutural, no Protein Data Bank [33], para cada seqüência usando BLAST [9]. Passamos a alinhar modelos usando um método para superposição de estruturas e mapear as seqüências originais em seus alinhamentos de modelos. Tais resultados compõem uma biblioteca primária, que é enviada a um método baseado em consistência para calcular o resultado final. Há também a extensão por homologia, que foi originalmente aplicada no pacote DbClustal [220] e trabalha de forma semelhante. A diferença está no fato de usar um perfil (construído com PSI-BLAST [11]) ao invés da estrutura.

Outros pacotes que fazem alinhamento baseado em modelos são: Expresso [14], PROMALS [176], PRALINE [97], SPEM [241] e T-Lara [28]. A maioria dos métodos baseados em modelos são baseados em consistência. PRALINE e SPEM são exceções. Eles usam a abordagem progressiva. PROMALS implementa uma extensão por homologia. Os *benchmarks* mais recentes mostram um forte desempenho do método (PROMALS) e sugerem um bom potencial para a abordagem de extensão por homologia.

Resultados de estudos recentes sugerem que os melhores métodos têm se tornado indistingüíveis, exceto em situações de homologia remota (menos que 25% de identidade). Infelizmente, homologias remotas são fracamente indicadas para a geração de alinhamentos de referência, pois suas sobreposições freqüentemente aceitam diversos alinhamentos alternativos com estruturas eqüivalentes [128]. Visando eliminar tal dificuldade na avaliação de um método, é possível comparar diretamente o alinhamento avaliado com alguma superposição 3D idealizada, que tem sido adotada por diversos autores [14, 170, 175].

Os *benchmarks* correntes têm pecado pela ausência de conjuntos de refência com grande número de seqüências, o que inviabiliza uma avaliação apropriada dos diversos métodos disponíveis quanto a suas capacidades diante de tal situação.

Um problema, levantado recentemente, refere-se a suposição de que alinhamentos estruturalmente corretos são os melhores MSAs para modelar qualquer tipo de sinal biológico (evolução, homologia ou função). Um relatório acerca de construção de perfil [88] mostrou que alinhamentos estruturalmente corretos não resultavam necessariamente em melhores perfis. É necessário sistematicamente questionar e quantificar o relacionamento entre a precisão de MSAs e a relevância biológica de qualquer modelo oriundo deles.

Alinhamento baseado em modelo é mais que uma extensão trivial de métodos baseados em consistência. Neste novo método, o propósito de um MSA é utilizar uma entrada expandida, com uma série de informações adicionais como estruturas das seqüências a alinhar, como um ponto de partida para explorar e recuperar todas as informações relacionadas (às seqüências) contidas em bancos de dados públicos. Desta forma, utilizando todo este conhecimento adiquirido acerca das seqüências tanto com o objetivo de mapeamento quanto de guia para a computação do MSA. O uso de tais informações tornam métodos baseados em modelos uma verdadeira mudança de paradigma e um grande passo na direção de uma integração global de dados biológicos.

A.4 Multiple sequence alignments

Este trabalho [227] apresenta uma revisão, realizada por Wallace, Blackshields e Higgins, acerca das soluções para alinhamento múltiplo de seqüências disponíveis até meados desta década.

Os principais métodos, ainda em uso, são baseados em alinhamento progressivo e datam da metade da década de 1980 em diante. Recentemente houveram melhorias dramáticas na metodologia no que se refere a velocidade e capacidade de trabalhar com grande número de seqüências. Tem-se conquistado avanços muito significativos no que se refere a precisão dos resultados. Houve também avanços práticos na definição de métodos para combinar informações de estruturas tridimensionais com seqüências primárias, que é um dos principais fatores para a melhoria na qualidade dos alinhamentos.

Até meados da década de 1980, MSAs eram construídos manualmente, pois até então a única solução de que dispunham era uma extensão do algoritmo de programação dinâmica para alinhamento de pares [158]. Tal método era muito limitado devido a suas altas requisições de processamento e memória, o que o tornava proibitivo para casos reais.

T-COFFEE é de grande interesse não apenas por causa da forma como permite que dados heterogêneos sejam combinados em alinhamentos, como pode ser observado em seu uso no 3D-COFFEE [169], mas também como um precursor para o método baseado em probabilidade do ProbCons [54], que é o mais preciso hoje em dia. ProbCons funciona de forma semelhante ao T-COFFEE, mas usa probabilidades para as pontuações de resíduos.

3D-COFFEE [169] foi projetado para criar alinhamentos de seqüências de proteína que incorporam informação de estruturas tridimensionais, quando estas existem. Em um conjunto de seqüências de entrada, é comum encontrar entradas PDB [33] para uma ou mais destas seqüências. O objetivo de utilizar as estruturas tridimensionais é facilitar o alinhamento de seqüências com relacionamentos distantes, pois elementos estruturais são geralmente mais conservados que seqüências primárias e assim permitem um bom alinhamento mesmo na *twilight zone* (menos de 25% de identidade). Na prática, usar este tipo de informação pode ser complicado, mas há trabalhos descrevendo seu uso [4].

3D-COFFEE é um método rápido, simples e preciso que explora a habilidade do T-COFFEE em incorporar informações heterogêneas para adicionar dados estruturais no alinhamento e, assim, melhorar sua precisão, mesmo quando não se dispõe de todas as estruturas. Quando é dada apenas uma estrutura, o 3D-COFFEE combina cada seqüência com a estrutura usando uma ferramenta externa (FUGUE [195]) e converte a saída para um alinhamento de duas seqüências. Desta forma temos um conjunto de alinhamentos de pares que indicam como as seqüências alinham com a estrutura e, indiretamente, como cada seqüência se alinha a cada outra.

Caso duas ou mais estruturas estejam disponíveis, é possível usar um pacote de superposição de estrutura completa. O 3D-COFFEE usa por padrão o SAP [211]. Apesar de usar FUGUE e SAP por padrão, o 3D-COFFEE permite o uso de outras ferramentas do gênero.

A.4.1 Avaliação da qualidade dos alinhadores

Existem diversas ferramentas para *benchmark* em MSA na forma de bases de dados de alinhamentos pré-compilados aos quais os alinhamento gerados pelos métodos testados são

comparados. Estas comparações são freqüentemente realizadas pelo cálculo da fração de colunas de resíduos alinhados que são idênticas em ambos os alinhamentos (teste e referência). Este método é conhecido como pontuação de coluna [120]. Existem diversos outros métodos. Tal abordagem para *benchmark* é atrativa devido a sua simplicidade, mas deve-se tomar cuidado para, na busca por melhorar o alinhador, não parametrizá-lo de forma a apresentar bons resultados nos conjuntos de referência das ferramentas para *benchmark* e degradar a performance em situações gerais.

Dentre as soluções para *benchmark*, BAliBASE [216] foi o primeiro construído com o propósito de *benchmarking* em larga escala e continua sendo regularmente utilizado hoje em dia. Seus conjuntos de dados são sub-divididos em diversos conjuntos de referência manualmente refinados, sendo que cada um deles destina-se a avaliação de um problema específico em MSA (alinhamentos globais/locais de diferentes tamanhos e identidade de seqüências, longos *gaps* internos, etc.). Estas são suas principais vantagens e grandes diferenças, que o tornam a principal ferramenta utilizada pelo comunidade científica.

HOMSTRAD [149] compreende um conjunto de mais de 1000 alinhamentos, cada um deles baseado em uma família de proteínas em particular. É exclusivamente baseado em seqüências com estruturas tridimensionais e arquivos PDB conhecidos. Em cada entrada, um alinhamento estrutural das proteínas é automaticamente gerado. Pode-se utilizar estes alinhamentos para *benchmark*. Possui uma menor cobertura, se comparado ao BAliBASE.

PREFAB [63] é também automaticamente gerado. Duas proteínas com estruturas conhecidas são estruturalmente alinhadas e suas seqüências são usadas para consultar um banco de dados, onde resultados com altas pontuações são selecionados. Os parâmetros de consulta e seus resultados são combinados e então alinhados usando o *software* que está sendo testado. A precisão é calculada apenas no par original, pela comparação com seu alinhamento/superposição estrutural.

O banco de dados SABmark [229] contém dois grandes sub-bancos de dados de alinhamentos de até 25 seqüências de entrada cada. Há 425 grupos representando famílias de proteína com identidade entre 25% e 50%. Há ainda 209 grupos para avaliação de *twilight zone*. Cada um destes grupos representa uma dobra de proteína definida pelo SCOP [155] e as seqüências compartilham menos de 25% de identidade.

Já a ferramenta IRMbase [207] constrói seu conjunto de referência implantando *motifs* gerados pelo ROSE [205] em seqüências aleatórias. Provê diversos grupos de alinhamento que variam no tamanho e no número de implantes.

Diferente de todos os métodos de avaliação citados, APDB [170] não recai na comparação de alinhamentos de referência pré-existentes. Ao invés disso, a qualidade é mensurada com base na superposição de estruturas PDB (das seqüências de entrada) conhecidas. Sua grande vantagem está na independência de alinhamento de referência e, assim, evita qualquer chance de definir métodos MSA otimizados para conjuntos específicos de referência. Sua grande desvantagem encontra-se na dependência da existência das entradas PDB.

A.4.2 Novos pacotes para alinhamento

MAFFT [121] usa transformada rápida de Fourier para gerar uma árvore guia para alinhamento progressivo. Uma estratégia de iteração baseada em árvore é então utilizada para refinar o alinhamento pela otimização de uma função objetivo de soma-dos-pares com peso.

Este protocolo resulta em alinhamentos muito rápidos e precisos, conforme avaliado pelo BAliBASE [122].

PSI-PRALINE [196] melhora a qualidade dos resultados do PRALINE [97] pela inclusão de seqüências homólogas (àquelas do conjunto de seqüências de entrada) obtidas pelo PSI-BLAST [11].

ProbCons [54] é atualmente o método MSA mais preciso, conforme avaliação realizada pelo BAliBASE. Ele obtém os melhores resultados em todos os cinco conjuntos de referência providos pela ferramenta de avaliação. De uma forma geral, ProbCons funciona de forma semelhante ao T-COFFEE, mas usa probabilidade ao invés da heurística para pesos de pares de resíduos. Para tanto usa HMM de pares de seqüências gerados por uma função objetiva de máxima precisão esperada [108]. Faz uso de um algoritmo de alinhamento progressivo, seguido de um procedimento iterativo de refinamento.

MUSCLE (do inglês, *MUltiple Sequence Comparison by Log Expectation*) [62, 63] é um novo pacote de alinhamento progressivo que é extremamente rápido e preciso. Seu primeiro passo é gerar um alinhamento grosseiro através de uma árvore guia muito imatura. As distâncias entre os pares de seqüências de entrada são estimadas usando contagem k-mer para um alfabeto restrito [61]. São estas distâncias que, através de um algoritmo de agrupamento, geram a árvore guia inicial. MUSCLE implementa a pontuação LE (do inglês, *log expectation*) para alinhar perfis durante o alinhamento progressivo. O próximo estágio no processo é o refinamento do alinhamento pela geração de uma árvore guia mais precisa, baseando-se no alinhamento inicial. LE tem apresentado bons resultados em buscas por homologia [221]. Nas versões mais recentes do MUSCLE foi adicionado um refinamento iterativo utilizado pelo ProbCons.

A.4.3 Conclusões

MSAs são largamente utilizados para diversos procedimentos e qualquer melhoria em seus métodos pode resultar em um impacto significativo na comunidade científica. Entretando, há um limite para a precisão que os algoritmos baseados em seqüência podem chegar.

Assim, para o caso particular de alinhamento de proteínas, os métodos tendem e devem ser sensíveis e flexíveis de tal forma que permitam a incorporação de uma diversidade de tipos de informações oriundas de diversas fontes, tais como coleções de alinhamentos múltiplos existentes ou estruturas de proteína.

No caso de alinhamentos de seqüências de RNA e/ou DNA, a situação é completamente diferente. Não há ferramentas para *benchmark* apropriadamente estabelecidas para avaliações e comparações das soluções.

Neste contexto, a situação melhora bastante quando se trata de seqüências de DNA longas, que recaem em problemas como alinhamento de genomas [35, 38, 39], busca por *motifs* [17, 70, 197] ou melhor alinhamento local [238]. Todas estas são áreas muito ativas de pesquisa.

A.5 BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark

Este trabalho [216] apresenta a versão 3.0 do BAliBASE. Tal ferramenta é muito utilizada para avaliação de sistemas destinados a alinhamento múltiplo de seqüências de proteína [5, 62, 98, 107, 109, 120, 122, 129, 141, 160, 232]. Além da qualidade de seus alinhamentos, o BAliBASE provê conjuntos de referência, o que permite avaliar o desempenho do alinhador em diversas situações e assim rapidamente detectar onde estão suas principais deficiências. Através de seu *web site* [1] é possível obter os conjuntos de referência e as ferramentas para a execução automática dos testes.

Diversos são os complicadores que um alinhador deve enfrentar na realização de sua tarefa de forma adequada, dentre eles podemos citar: grandes proteínas multidomínio, em especial de organismos eucariotos; seqüências heterogêneas, que podem induzir diversos erros de alinhamento; além de erros durante o processo de seqüenciamento, que inevitavelmente levam a seqüências incorretas. Fica cada vez mais evidente que soluções que utilizem um único método tendem a não ser capazes de lidar bem com todas estas situações, gerando assim resultado de qualidade inferior ao de soluções que empreguem estratégias cooperativas.

Diversas soluções têm surgido usando esta abordagem. Uma delas, que parece interessante, é combinar algoritmos globais e locais para produzir um único alinhamento [54, 133, 151, 165, 220]. Outro recurso que tem melhorado significativamente o desempenho do alinhadores é o uso de métodos para identificação rápida de regiões de alta similaridade nas seqüências [62, 122]. Os *benchmarks* têm um papel de grande importância. Eles permitem uma avaliação precisa das soluções que se apresentam mais efetivas na solução do problema.

Neste contexto, BAliBASE não está sozinho. Recentemente diversos outros *benchmarks* para alinhamento múltiplo de seqüências de proteína têm sido propostos. Dentre eles podemos citar: OXBench [180], PREFAB [63] e SABmark [229]. Todos com grande conjunto de teste gerado por métodos automáticos de superposição de estrutura. Outra opção ainda é o HOMSTRAD [149], que apesar de não ter sido concebido com tal finalidade, tem sido utilizado.

A.5.1 Visão geral

BAliBASE provê alinhamentos de referência de alta qualidade baseado na combinação de superposição de estruturas 3D com passo manual de validação e refinamento dos alinhamentos. Nesta nova versão são incluídos novos casos de teste mais desafiadores e que correspondem a problemas reais, encontrados quando alinhando grandes conjuntos de seqüências complexas. Foi incluído também um novo protocolo semi-automático de atualização, que facilita a incorporação de novas famílias de proteína e conjuntos de referência, permitindo assim atualizações mais freqüentes do *benchmark*. O processo é semi-automático devido ao passo manual de verificação e refinamento, essencial para a manutenção da qualidade na base de dados do *benchmark*.

No BAliBASE 3.0 há cobertura da maior parte dos tipos de dobra presentes nas proteínas. São 6.255 seqüências em sua nova base, contra 1.444 da anterior. São 217 alinhamentos, contra 141. Há agora, em todos os casos de teste, seqüências *full-length*, que tornam a tarefa do alinhador, seja global ou local, ainda mais difícil. A variabilidade das seqüências

nos conjuntos de teste também foi aumentada, devido a exclusão dos alinhamentos mais conservados.

Os alinhamentos são organizados em conjuntos de referência projetados para representar problemas reais em alinhamento múltiplo, são eles:

Referência 1: contém alinhamentos de seqüências equidistantes. É subdividido em dois subconjuntos, de acordo com os níveis de variabilidade das seqüências. Até a versão anterior eram três, mas o terceiro foi excluído porque não apresentava grande dificuldade para os alinhadores e assim não agregava informação útil;

Referência 2: contém famílias alinhadas com uma ou mais seqüências "orfãs" altamente divergentes;

Referência 3: contém subfamílias divergentes;

Referência 4: contém seqüências com grandes extensões de terminais N/C;

Referência 5: contém seqüências com grandes inserções internas;

Referência 6: contém seqüências com regiões de transmembranas;

Referência 7: contém seqüências com repetições; e

Referência 8: contém seqüências com domínios invertidos.

O protocolo de atualização inclui um definição automática de regiões "*core block*" em cada alinhamento. Tais regiões são aquelas que podem ser confiavelmente alinhadas, ou seja, desconsideram-se regiões ambigüas, que não podem ser estruturalmente sobrepostas.

A.5.2 Construção das referências

Em sua segunda versão, o BAliBASE possui um total de 141 alinhamentos múltiplos. Destes, há 82 alinhamentos de famílias de proteína na Referência 1 com um total de 532 seqüências extraídas da base PDB [33]. Para a geração dos conjuntos de referência da versão 3.0 foram usadas estas seqüências juntamente com famílias da classe multidomínio do SCOP (do inglês, *Structural Classification of Proteins*) [155].

Para cada família de proteína, realiza-se uma etapa de superposição de cada um dos seus membros com estruturas 3D conhecidas. Esta etapa, que resulta em um alinhamento de estrutura primária da família, é realizada pelo seguinte protocolo: (a) realiza-se uma busca no PDB, usando PSI-BLAST [11, 190], para cada membro da família. No caso de famílias que não estavam incluídas na versão anterior realiza-se a busca com a única seqüência selecionada do SCOP. Filtram-se apenas as seqüências PDB com $E < 10^{-3}$. (b) Visto que em estudo anterior [218] os alinhadores tinham facilidade em gerar resultados de boa qualidade quando as seqüências possuíam mais de 40% identidade de resíduos, remove-se as seqüências com identidade $> 40\%$. Neste passo utiliza-se o MAFFT [122] para determinar tais seqüências. (c) Então constrói-se um alinhamento de referência com as seqüências resultantes. Utiliza-se neste passo o programa SAP [211] para superposição 3D. Tal seleção foi feita com base em critérios de confiabilidade e funcionalidade. (d) O alinhamento de estrutura gerado é então manualmente verificado e refinado.

No passo anterior, constrói-se um alinhamento de estrutura primária para cada família. Em tais alinhamentos temos pequenos conjuntos de seqüência PDB divergentes, que pode

ser alinhado com precisão através de superposição de estruturas 3D. Visando a criação de conjuntos de teste maiores, foram incorporadas seqüências de estrutura desconhecida oriundas do Uniprot [47]. Agora inclui-se a cada grupo as seqüências Uniprot com alto grau de similaridade a pelo menos uma das seqüências no alinhamento de seqüências primárias, pois só assim poderemos realizar alinhamentos precisos. O procedimento é o seguinte: (a) realiza-se uma busca BlastP [11] para cada uma das seqüências no alinhamento de seqüências primárias. Filtram-se as seqüências com $E < 10^{-3}$. (b) Removem-se as seqüências com mais de 80% de identidade de resíduos. (c) Por não dispôr de informações estruturais acerca das seqüências Uniprot, filtram-se aquelas com mais de 40% de identidade com pelo menos umas das seqüências PDB. (d) Uma vez alinhadas as seqüências resultantes, realiza-se a verificação e refinamento manual. O resultado desta etapa constitui o alinhamento múltiplo primário.

A partir dos alinhamentos múltiplos primários, constrói-se automaticamente os conjuntos de referência de 1 a 5. Para o primeiro conjunto são selecionadas seqüências eqüidistantes que não possuam grandes extensões internas (mais que 35 resíduos). Em V1 são alocadas as famílias com, pelo menos, quatro seqüências onde quaisquer dois membros possuem menos de 20% de identidade de resíduos. Em V2 são alocadas as famílias com, pelo menos, quatro seqüências onde quaisquer dois membros possuem identidade de resíduos entre 20 e 40%.

Para o segundo conjunto são selecionadas famílias cujas seqüências possuem mais de 40% de identidade de resíduos e que, pelo menos, uma das seqüências possua a estrutura 3D conhecida. Adicionam-se orfãs ($< 20\%$ de identidade) à cada família. Tais orfãs só podem ser seqüências PDB, pois só assim haverá garantia de um alinhamento preciso.

Para o terceiro conjunto são selecionadas subfamílias tais que as seqüências de uma mesma subfamília possuam $> 40\%$ de identidade, mas quaisquer duas seqüências de subfamílias distintas possuam $< 20\%$ de identidade. Cada subfamília deve possuir, pelo menos, uma seqüência PDB. Para as referências 1-3, o percentual de identidade é calculado sobre a região homóloga apenas. Nestes conjuntos não há seqüências com grandes inserções internas.

Para as referências 4 e 5 são selecionadas seqüências com $> 20\%$ de identidade com, pelo menos, uma outra seqüência da família. No quarto conjunto são incluídas as seqüências com grandes extensões de terminais N/C. No quinto conjunto adicionam-se as seqüências com grandes inserções internas.

A seguir definem-se os "*core blocks*", regiões confiavelmente alinhadas, para cada alinhamento. Para sua definição é utilizado um método automático baseado na combinação de superposição de estrutura secundária e conservação de seqüência. NorMD [219] é utilizado para mensuração da conservação das seqüências.

Há ainda um último passo onde são adicionadas informações acerca de estrutura e funcionalidade das seqüências. Para cada seqüência PDB utiliza-se DSSP [118] para calcular elementos da estrutura secundária. Para as seqüências Uniprot são extraídas entradas da FT (do inglês, *Feature Table*), além de realizar buscas por domínios da família de proteína no Pfam [219].

A.5.3 Conclusão

Comparações dos métodos no BAliBASE (ainda em sua versão 1.0) mostraram que a maioria das soluções alinham com sucesso seqüências que possuam mais que 40% de identidade de

resíduos, mas há uma grande perda de precisão quando esta identidade cai para menos de 20% (*twilight zone*) [218]. Tal estudo também mostrou que métodos globais, em geral, funcionam melhor quando as seqüências possuem tamanho similar, mas os métodos locais são superiores para o caso de identificação de *motifs* mais conservados em seqüências com grandes extensões e inserções.

O uso de *benchmarks* para alinhamento múltiplo de seqüências de proteínas é de grande importância na avaliação da efetividade das ferramentas e na comparação de seus desempenhos. Os diferenciais do BAliBASE neste contexto ficam por conta de sua alta qualidade, garantida pela verificação e refinamento manual dos alinhamentos por especialistas, e por sua organização dos conjuntos de teste, que permite avaliar o alinhador em diversas situações específicas.

A.6 A comprehensive comparison of multiple sequence alignment programs

Este trabalho [218] apresenta o primeiro estudo sistemático dos programas de alinhamento múltiplo mais comumente utilizados até então (1999). Para a realização de tal estudo utilizou-se, como casos de teste, alinhamentos providos pela ferramenta de *benchmark* BAliBASE 1.0 [217].

Tem se tornado uma prática comum o alinhamento múltiplo de grande número de seqüências de proteína, onde neste estão inseridas seqüências muito divergentes e proteínas multi-domínio freqüentemente com grandes extensões de terminais N/C ou inserções internas. Outra prática que tem se tornado comum é o alinhamento de uma única seqüência divergente (tipicamente de um eucarioto) com um grande grupo de seqüências fortemente relacionadas (tipicamente de procariotos). O desenvolvimento de programas de alinhamento múltiplo confiáveis e precisos capazes de tratar estas situações é então de grande importância.

Há poucas comparações disponíveis acerca das performances relativas e confiabilidade das ferramentas para MSA. Doze alinhadores progressivos, dentre globais e locais, foram comparados usando alinhamentos de quatro domínios de proteínas diferentes como casos de teste [142]. Em geral métodos globais obtiveram resultados melhores que métodos locais nos testes, mas o desempenho de todos os programas foram afetados pelo número e o grau de identidade das seqüências, assim como pelo número de inserções/ deleções no alinhamento.

Sete servidores de alinhamento múltiplo na web, cobrindo vários métodos globais e locais, também foram comparados para avaliar suas habilidades em identificar regiões confiáveis no alinhamento [36]. Entretanto não há estudos e comparações abrangentes dos diversos novos algoritmos. A falta de um conjunto padronizado de alinhamentos de referência tem impedido uma avaliação apropriada dos programas existentes, assim como ainda não foi possível mensurar com precisão os ganhos obtidos com os novos métodos iterativos.

Recentemente, um banco de dados de alinhamentos de referência para *benchmark*, chamado BAliBASE [217], foi desenvolvido especificamente para este propósito. Os 142 alinhamentos de teste validados de proteínas reais baseado em superposição tridimensional são organizados em conjuntos de referência, que representam alguns dos problemas mais comuns em MSA. Para cada alinhamento são definidos *core blocks*, que definem aquelas regiões que podem ser confiavelmente alinhadas.

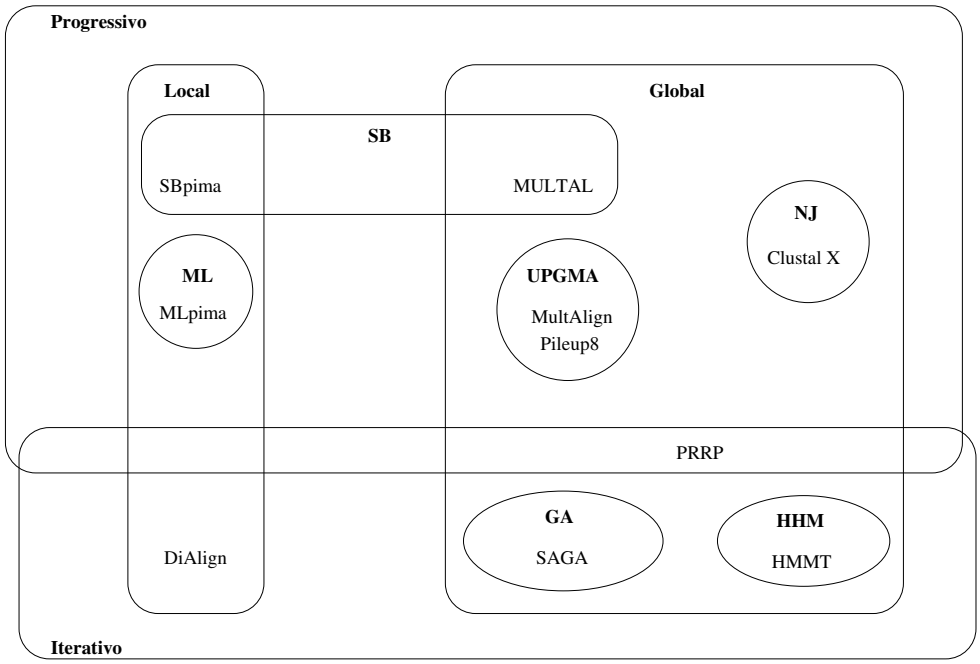

Figura 1: Apresentação esquemática da relação entre os diferentes alinhadores e métodos. Os nomes dos métodos utilizados estão em negrito e seu escopo é delimitado por uma caixa, círculo ou elipse. Dentre eles temos: progressivo, iterativo, local, global, ML (do inglês, *Maximum Linkage*), SB (do inglês, *Sequential Branching*), UPGMA, NJ (do inglês, *Neighbor-Joining*), GA (do inglês, *Genetic Algorithm*) e HMM (do inglês, *Hidden Markov Model*). A partir da figura é possível extrair que PRRP é um alinhador que utiliza as abordagem progressiva e iterativa para construir um MSA global, assim como, SBpima é um alinhador múltiplo local que constrói o alinhamento progressivamente por *sequential branching*.

Neste trabalho apresenta-se uma análise e comparação sistemática dos principais programas de alinhamento em uso atualmente (enumerados na Figura 1), usando os alinhamentos de referência do BAliBASE como casos de teste.

A.6.1 Material e Método

Todos os programas foram instalados em um computador DEC Alpha 6100 executando OSF Unix e cada programa foi testado usando os parâmetros padrões (exceto para o PRRP, onde utilizou-se a opção -b para indicar que as seqüências de entrada não foram pré-alinhadas).

O BAliBASE contém 142 alinhamentos de referência, divididos em cinco conjuntos de referência hierárquicos onde cada um possui pelo menos 12 alinhamentos representativos. Definem-se regiões, chamadas *core blocks*, onde pode-se alinhar confiavelmente. Tais regiões, que representam 58% dos resíduos nos alinhamentos, excluem trechos ambíguos ou não superpostos tridimensionalmente.

A primeira referência consiste de alinhamentos de pequeno número de seqüências eqüidistantes e de tamanho similar. Não possuem grandes extensões ou inserções. Este teste avalia o efeito do tamanho da seqüência e do percentual de identidade no desempenho do alinhador e provê a base para os outros testes.

A segunda referência contém alinhamentos de uma família (seqüências fortemente relacionadas com mais que 25% de identidade) com a adição de até três seqüências "órfãs" (membros distantes da família com menos que 20% de identidade, que compartilham uma dobra em comum). Foi projetado para avaliar a precisão do programa de acordo com dois critérios: a estabilidade do alinhamento da família quando órfãs são introduzidas no conjunto de seqüências e a qualidade do alinhamento das seqüências órfãs. O programa MULTAL foi removido deste teste pois freqüentemente excluiu as seqüências divergentes indicando tratarem-se de seqüências não relacionadas ou não alinháveis.

A terceira referência demonstra a habilidade do programa de alinhar corretamente famílias aproximadamente eqüidistantemente divergentes em um único alinhamento. Os alinhamentos de referência consistem de até quatro famílias, com menos de 25% de identidade entre quaisquer duas seqüências de famílias diferentes. MULTAL também não foi incluído nos testes com a terceira referência devido a um problema semelhante ao da segunda referência.

A quarta e quinta referências contêm seqüências com grandes extensões de terminais N/C e inserções internas, respectivamente. Para avaliar a habilidade do programa de identificar a presença de inserções, os *core blocks* no BAliBASE definem apenas os mais conservados *motifs* delimitando extensões e inserções. Estes testes não foram projetados para julgar a qualidade de um alinhamento completo. Mais uma vez MULTAL teve de ser removido. Desta vez HMMT também foi excluído, pois muitos dos alinhamentos contêm apenas um pequeno número de seqüências.

Para calcular o desempenho dos programas nestes estudo, estabeleceu-se duas pontuações distintas, que estimam a qualidade de uma alinhamento comparando os resultados aos alinhamentos de referência do BAliBASE. A pontuação SPS (do inglês, *sum-of-pairs score*) é calculada de forma que tenha seu valor acrescido de acordo com o número de seqüências alinhadas corretamente. É usada para determinar a extensão dos alinhamentos entre pares de seqüências em que se obteve sucesso. A pontuação CS (do inglês, *column score*) é uma pontuação binária que avalia a habilidade dos programas em alinhar todas as seqüências corretamente.

Suponha que tenhamos um alinhamento de teste com N seqüências e M colunas. Seja $A_{i1}, A_{i2}, \ldots, A_{iN}$ a i-ésima coluna do alinhamento. Para cada par de resíduos A_{ij} e A_{ik} define-se p_{ijk} tal que $p_{ijk} = 1$ se A_{ij} e A_{ik} estão alinhados no alinhamento de referência e $p_{ijk} = 0$, caso contrário. A pontuação S_i para a i-ésima coluna é definida como:

$$S_i = \sum_{j=1,k\neq j}^{N} \sum_{k=1}^{N} p_{ijk}.$$

A pontuação SPS é definida por:

$$SPS = \frac{\sum_{i=1}^{M} S_i}{\sum_{i=1}^{M_r} S_{ri}},$$

onde M_r é o número de colunas no alinhamento de referência e S_{ri} é a pontuação S_i para a i-ésima coluna do alinhamento de referência.

Define-se $C_i = 1$ se todos os resíduos na i-ésima coluna do alinhamento são os mesmos no alinhamento de referência e $C_i = 0$ caso contrário. A pontuação CS é dada por:

$$CS = \frac{\sum_{i=1}^{M} C_i}{M}$$

Para cada um dos testes de referência selecionou-se a função de pontuação mais adequada de acordo com a natureza do teste.

Em cada referência, BAliBASE provê um número de alinhamentos representativos que são usados como uma amostra em análises estatísticas. Para cada alinhamento de referência calcula-se uma pontuação estimando a precisão do alinhamento produzido por todos os programas. Como é esperado que a distribuição das pontuações não seja normal ou simétrica, utiliza-se a mediana como uma medida de localização e o intervalo interquartil como uma medida de dispersão. O primeiro e o terceiro quartis indicam a forma da distribuição.

Foi utilizado o teste de Friedman [69] para estimar se havia um padrão sistemático na forma como os programas eram classificados pela pontuação em todos os alinhamentos. Em outras palavras, verificar, por exemplo, se algum dos programas tende a executar significantemente melhor que outros nos alinhamentos de referância.

Os testes de Wilcoxon [235] foram utilizados para determinar se uma alteração nas condições de um alinhamento, tal como a adição de órfãos (referência 2) ou um incremento no número de famílias (referência 3), interferem de forma significativa na pontuação. HMMT foi removido deste teste por gerar soluções distintas para uma mesma entrada, o que impede uma comparação confiável.

A.6.2 Resultados

Referência 1: pequeno número de seqüências eqüidistantes. Regiões ambíguas representam 42% dos resíduos no BAliBASE e somam, em média, 32, 22 e 11% das pontuações *full-length* calculadas nas categorias V1 ($< 25\%$ ID), V2 ($20 - 40\%$ ID) e V3 ($> 35\%$ ID), respectivamente.

Construiu-se um gráfico com os programas no eixo X, a precisão no eixo Y e com três curvas, respresentando V1, V2 e V3. A precisão no caso refere-se a mediana das pontuações para *core blocks*. A característica mais marcante do gráfico é uma nítida perda de precisão a medida que o grau de identidade entre as seqüências diminui. A queda na precisão é mais marcante ainda na categoria V1, que refere-se a chamada *twilight zone*. Mesmo assim, é possível ainda realizar algum alinhamento abaixo da *twilight zone*.

O melhor alinhamento em V1 foi obtido pelo PRRP, com 72% do total de resíduos alinhados corretamente. Os programas que melhor pontuaram (PRRP, Clustal X e SAGA), alinharam corretamente, em média, 61% dos resíduos (ou 42% das colunas) nos *core blocks* e 47% do total de resíduos (ou 26% do total de colunas) em V1. Em contraste, para V2 92% dos resíduos (ou 87% das colunas) em *core blocks* e 82% dos resíduos (ou 72% das colunas) do total de resíduos foram alinhados corretamente.

As maiores diferenças nas pontuações dos programas são sempre observadas na categoria V1. De acordo com o teste de Friedman, PRRP classifica-se significantemente melhor que

os outros programas. Clustal X e SAGA classificam-se em seguida. Observa-se também que, no geral, programas de alinhamento global executam melhor que os métodos locais.

Ao construir um gráfico onde as três curvas representam as seqüências curtas (< 100 resíduos), médias ($200 - 300$) e longas (> 400) de V1, pode-se observar que para seqüências longas os programas locais obtêm uma qualidade de alinhamento similar aos globais. A única exceção ficou por conta do PRRP, que obtém resultados significantemente melhores que os outros programas de acordo com o teste de Friedman. No geral, *core blocks* são melhor alinhados em seqüências médias e longas que em seqüências curtas. A exceção fica por conta do Clustal X, que funciona melhor para seqüência médias e curtas.

Já uma análise das pontuações *full-length* revela que: (a) há uma queda nas pontuações *full-length* comparadas às pontuações de *core blocks*; (b) a precisão dos resultados de programas globais cai com o aumento no tamanho das seqüências; (c) já a pontuação de programas locais mantém a tendência de crescer a medida que o tamanho da seqüência cresce.

Observou-se que os programas tendem a alinhar melhor *core blocks* em seqüências longas que em seqüências curtas. Tal fato surpreendeu. Isto pode ser provocado pelo tamanho da seqüência em si ou por uma diferença na natureza dos *core blocks* em cada categoria de tamanho. Para investigar a causa, comparou-se os alinhamentos de tamanhos médios na categoria V1 com uma nova população de seqüências curtas criadas artificialmente pela divisão de seqüências de tamanho médio em duas com metade do tamanho. É importante salientar que algumas divisões foram deslocadas para não dividir *core blocks*. Utilizando-se o teste de Wilcoxon observou-se que, desta vez, os *core blocks* alinharam melhor em seqüências curtas que nas seqüências médias. Deduziu-se que a redução da precisão observada em seqüências curtas na referência 1 não se deve simplesmente ao tamanho das seqüências.

Referência 2: família e seqüências órfãs. Para este teste utilizam-se famílias pequenas com 4 seqüências e famílias maiores com um número de seqüências variando de 14 a 22. Inicialmente as famílias foram alinhadas sem as seqüências órfãs. Posteriormente foram construídos os alinhamentos com uma, duas e finalmente três órfãs (com identidade de 10 a 20% sempre que comparada com um membro da família). Para cada caso, calculou-se a pontuação SPS para o programa.

Surpreendentemente um teste de Wilcoxon indicou a ausência de uma redução significativa nas pontuações dos alinhamentos. Há, entretanto, um pequeno número de casos onde observou-se perda de qualidade de até $6,9\%$ para famílias grandes e perda de $23,6\%$ em famílias pequenas.

Avaliando as pontuações SPS para alinhamentos com uma única seqüência órfã observou-se que programas de alinhamento global executam melhor que os locais. Entretanto, é importante salientar que agora Clustal X e SAGA obtêm melhores resultados que PRRP. Um teste de Wilcoxon realizado indica que há uma melhora significativa no alinhamento quando as famílias são maiores.

A habilidade de alinhar uma seqüência órfã, para todos os programas, é também afetada pela presença de outras órfãs no conjunto de seqüências. Entretanto, uma exata correlação ainda não está clara. Percebe-se apenas que, dependendo do grau de relacionamento entre as órfãs e do relacionamento entre as órfãs e a família, o alinhamento por ser melhorado ou deteriorado.

Referência 3: diversas famílias em um único alinhamento. Para este teste foram

utilizados pequenos números de famílias divergentes. A pontuação CS é usada neste teste, pois ela é uma melhor estimativa para alinhamento de famílias.

Utilizando um teste de Fredman, chegou-se a conclusão de que as estratégias iterativas do PRRP e do SAGA executaram melhor neste teste que os métodos de alinhamento progressivo tradicionais. O Clustal X, apesar de executar pior que PRRP e SAGA, executa melhor que os outros métodos progressivos. No geral, métodos globais apresentaram melhores resultados que os locais.

Foi ainda realizado um teste para verificar se o desempenho dos programas era melhor no alinhamento de famílias eqüidistantes ou no alinhamento de seqüências eqüidistantes. Para realizar tal avaliação foi construído um novo conjunto de testes pela seleção de uma seqüência de cada família nos alinhamentos da referência 3. Uma comparação nas pontuações obtidas mostrou que, no geral, famílias são melhor alinhadas que seqüências individualmente. A exceção aqui ficou por conta do MLpima e SBpima.

Referência 4: extensões de terminais N/C. Todos os testes anteriores foram realizados com seqüências de tamanhos similares. Agora são utilizadas seqüências com grandes extensões de terminais N/C para investigar se os programas são capazes de alinhar os *core blocks* nesta situação. É importante salientar entretanto que ainda não há grandes inserções internas.

Observou-se que, utilizando um teste de Fridman e pontuação CS, três programas que implementam uma estratégia de alinhamento local (SBpima, DiAlign e MLpima) se sobressaem perante os demais. Pileup 8 é o único método baseado em alinhamento global que chega próximo aos locais neste teste. As estratégias iterativas do PRRP e do SAGA falham, devido devido a dificuldades inerentes aos métodos globais diante de tal situação.

Referência 5: inserções internas. Este teste também possui seqüências de tamanhos distintos, mas desta vez as inserções são internas aos domínios homólogos. Utilizando-se a pontuação CS neste teste, observou-se que, assim como na referência 4, o DiAlign sobressaiu-se, entretando MLpima e SBpima já não obtiveram tanto sucesso desta vez e apresentaram os piores resultados, ficando atrás dos métodos globais.

A.6.3 Discussão

Um dos objetivos deste estudo foi estabelecer um sistema de avaliação (*benchmarking*) objetivo, que possa ser utilizado para comparar, avaliar e melhorar programas de alinhamento múltiplo.

Os alinhamentos do BAliBASE fornecem casos de teste reais contendo proteínas ou módulos cuja estrutura tridimensional já tenha sido determinada. Os alinhamentos são validados para assegurar o alinhamento correto de resíduos catalíticos e outros resíduos conservados. Apenas regiões que podem ser confiavelmente alinhadas, chamadas *core blocks*, estão anotadas na base e permitem uma avaliação mais precisa dos programas.

Os resultados deste trabalho deixam clara a necessidade de comparar alinhamentos utilizando tanto pontuações baseadas em seqüências completas como aquelas restritas aos *core blocks*.

Durante a realização deste estudo, observou-se que todos os programas analisados são capazes de alinhar, em média, 80% dos resíduos corretamente em um alinhamento de seqüências com mais que 20% de identidade. Assim, comparações entre os programas neste nível de identidade de seqüências tornam-se pouco significativos.

Observa-se, entretanto, uma importante perda na precisão quando alinhamos seqüências com identidade entre 10 e 20% (referência 1, V1). Neste caso, os melhores programas alinham corretamente, em média, apenas 47% dos resíduos. A chamada *twilight zone* [56] constitui uma barreira real para todos os programas analisados. Abaixo da *twilight zone*, os alinhamentos produzidos são freqüentemente não confiáveis e com grande dispersão nas pontuações. Para seqüências longas de mais de 400 resíduos em V1, os programas globais e locais podem não ser distintos no que se refere aos resultados.

Fica claro que os esforços para melhorar a qualidade dos alinhadores deve se concentrar no alinhamento de seqüências abaixo de 20 ou 25% de identidade de resíduos. Todavia, é importante salientar resultados notáveis, com seqüências abaixo da *twilight zone*, do PRRP, que é capaz de alinhar corretamente 27 a 72% dos resíduos.

Uma avaliação das melhorias introduzidas pelos novos métodos iterativos é inconclusiva. Quatro programas destacaram-se como os mais bem sucedidos sob as condições distintas de alinhamento testadas, foram eles: PRRP, SAGA, Clustal X e DiAlign. Note que, com exceção apenas do Clustal X, todos utilizam uma estratégia iterativa para refinar o alinhamento.

Clustal X tem melhorado claramente os programas tradicionais de alinhamento progressivo, embora os parâmetros padrões possam não ser ótimos para seqüências longas.

Os novos algoritmos iterativos freqüentemente fornecem alinhamentos com melhores precisões, além de ter a capacidade de adiquirir conhecimento sobre as seqüências a alinhar e melhorar o alinhamento, se houver informações suficientes no conjunto de dados.

É importante, entretanto, salientar que o processo iterativo pode algumas vezes ser instável na presença de ambiguidade no conjunto de seqüências, tal como uma única seqüência órfã. Tal situação pode levar a iteração a divergir do alinhamento correto.

Dentre os alinhadores locais, o DiAlign, que iterativamente usa um algoritmo de alinhamento de segmento local, é o mais bem sucedido.

Uma grande desvantagem das técnicas iterativas correntes é que requerem muito tempo de processamento. Como uma exemplo, para 89 seqüências de histonas consistindo de 66 a 92 resíduos, o tempo de processamento requerido para o alinhamento é de $161s$ para o Clustal X, $13.649s$ para o DiAlign e $13.209s$ para o PRRP.

Em geral, desenvolveu-se duas classes básicas de programas de alinhamento. Programas de alinhamento global tentam alinhar seqüências completas. Já os alinhadores locais buscam apenas pelos *motifs* mais conservados. Dependendo da natureza das seqüências, uma classe sobrepõe-se sobre a outra.

Algoritmos globais produzem resultados mais precisos e confiáveis em situações que envolvam seqüência eqüidistantes, famílias divergentes de seqüências e o alinhamento de uma seqüência órfã com uma família de seqüências. Resultados semelhantes aos do trabalho de McClure e colegas [142].

Entretanto, na presença de grandes extensões de terminais N/C e inserções internas, DiAlign, que implementa uma algoritmo local de alinhamento de segmentos sem *gaps*, é a solução mais bem sucedida na tarefa de encontrar os *core blocks* altamente conservados. Os métodos globais, que tendem a gerar alinhamentos co-lineares no tamanho completo das seqüências, são menos eficientes, produzindo freqüentemente um alinhamento totalmente incorreto das seqüências.

Os resultados destes testes sugerem possíveis caminhos para melhorias na precisão dos

programas para famílias e seqüências divergentes.

O alinhamento de seqüências órfãs com uma família é mais bem sucedido se a família contém mais seqüências. Entretanto, o efeito de alinhar muitas seqüências órfãs simultaneamente é imprevisível, depende do grau de relacionamento entre as seqüências. O alinhamento de órfãs pode também ser melhorado se uma pequena sub-família puder ser criada para a órfã.

Parece ser uma boa estratégia incluir o número máximo possível de seqüências para obter os melhores resultados. Mesmo que altamente relacionadas, seqüências podem prover informações úteis adicionais.

Testes sugerem que alinhamento progressivos podem ser melhorados pela reconstrução da árvore guia durante o processo de alinhamento.

Neste estudo pôde-se observar que a escolha de um programa de alinhamento depende do conjunto de seqüências a alinhar e que não há um único melhor programa. Fatores como a repartição das identidades das seqüências, o tamanho e as extensões de terminais N/C afetam a precisão e a confiabilidade dos programas. Nenhum dos alinhadores incluídos neste estudo foram capazes de produzir alinhamento bons e confiáveis em todas estas situações.

Trabalhos futuros visando a melhoria dos resultados de alinhadores deve concentrar esforços em problemas como grandes inserções, extensões e fragmentos de seqüência. O alinhamento de seqüências de tamanho similar é relativamente bem sucedido, mesmo quando há uma baixa identidade entre elas. Outra importante área de interesse que está se tornando cada vez mais freqüente é o alinhamento de famílias de seqüências. Freqüentemente, ao incrementar o número de seqüências na entrada, a qualidade do alinhamento de seqüências divergentes melhora significantemente.

Os resultados deste estudo podem ser utilizados para indicar os programas mais adeqüados para problemas particulares de alinhamento. Deve-se agora ser possível pré-determinar a natureza do conjunto de seqüências, em especial a repartição de identidade de seqüências e a presença de seqüências de tamanhos distintos. Um servidor de alinhamento deveria ser capaz de automaticamente selecionar o programa que muito provavelmente gerem os melhores resultados.

É importante salientar que neste estudo foram utilizados os parâmetros padrões dos programas e que o foco estava em avaliar a qualidade dos resultados. Questões importantes para a escolha de um alinhador, como facilidade de uso, portabilidade e requisitos de tempo e memória, foram desprezados.

A.7 The alignment of sets of sequences and the construction of phyletic trees: an integrated method

Este trabalho [105] foi a primeira descrição do método progressivo para alinhamento múltiplo de seqüências. Nele é defendido que alinhamento de seqüências e construção de árvores filogenéticas não podem ser tratados separadamente, uma vez que o conceito de "bom alinhamento" perde sentido se não houver referência a uma árvore filogenética e a construção de árvores filogenéticas pressupõem alinhamento de seqüências. Desta forma propõe-se um método integrado que gera ao mesmo tempo um alinhamento e uma árvore filogenética a partir de um conjunto de seqüências. É um método iterativo que parte de uma árvore hipotética e iterativamente ajusta a árvore através de alinhamentos sucessivos de pares de

seqüências guiados pela árvore corrente.

Há algoritmos eficientes para alinhamento de pares de seqüências [158, 194, 199, 234]. Estes utilizam alguma medida de similaridade como critério para otimização e adicionam *gaps* nas seqüências durante o procedimento de busca pelo alinhamento de máxima pontuação. Desta forma, buscam maximar o número de bases idênticas pareadas. Tais algoritmos ainda utilizam uma penalização para inibir formações freqüentes de *gaps*. Alguns autores desconsideram o tamanho de tais *gaps* apenas considerando o número de ocorrências, como Needleman e Wunsch [158]. Já outros consideram seu tamanho, como Sellars [194].

Um dos problemas nestes algoritmos para alinhamento de pares de seqüência está no fato de produzir muitos possíveis alinhamentos com mesma pontuação. Por exemplo, podem haver diversos alinhamentos com distribuições de *gaps* distintas e mesma pontuação. O posicionamento dos *gaps* não é determinado unicamente e variações no pareamento de bases podem ocorrer em trechos onde há pequena similaridade. Uma grande preocupação aqui deve estar em selecionar, dentre aqueles com mesma pontuação, o alinhamento com maior relevância biológica. Há possibilidade de considerar a estrutura secundária das seqüências nos alinhamentos, mas esta deve ser utilizada com cautela, uma vez que pode levar a uma super valorização da estrutura ao invés da ancestralidade. O mesmo método utilizado para alinhamento de par de seqüências é, a priori, extensível para mais de duas seqüências, mas neste caso os requisitos de tempo e espaço tornam-se impraticáveis a medida que aumenta-se o número de seqüências.

A maioria dos algoritmos para construção de árvores filogenéticas tentam de alguma forma minimizar o custo mutacional ao longo dos ramos de uma árvore. Desta forma, uma das abordagens é fazer uma busca exaustiva, gerando todas as topologias possíveis e escolhendo a melhor. Não é difícil de perceber que este método torna-se rapidamente impraticável a medida que aumenta-se o número de seqüências e, assim, temos de utilizar heurísticas. Podemos classificar tais métodos em: baseados em matriz de similaridade e baseados em caracter. Métodos baseados em matriz de similaridade são mais simples e são mais freqüentemente utilizados na prática. A grande vantagem dos métodos baseados em matriz de similaridade está em precisar apenas de alinhamentos de pares de seqüências, diferentemente dos baseados em caracter, que precisam do alinhamento global.

A.7.1 O método

A idéia é alinhar as seqüências através de uma série de sucessivos alinhamentos de pares, que seguem os ramos da árvore inicial [68]. Então passa-se a otimizar a árvore e o alinhamento iterativamente usando a árvore corrente para construir o próximo alinhamento (árvore). Segue uma descrição mais detalhada do procedimento.

1. Alinha-se todos os pares de seqüências e constrói-se uma matriz $(N \times N)$ com estes valores de similaridade.
2. Constrói-se uma árvore filogenética inicial a partir da matriz.
3. Sucessivamente alinha-se pares, seguindo os ramos da árvore. Inicia-se com as duas seqüências que são mais similares e segue-se adicionando sempre a seqüência mais próxima. Parte-se da premissa de que é mais confiável alinhar seqüências similares. Dar-se prosseguimento utilizando o seguinte critério na construção dos nós internos:

(a) se todas as m posições de uma dada coluna i possuem a mesma base x, x é assumida como a base da coluna i do alinhamento; (b) caso contrário, a decisão sobre a base a posicionar na coluna i do alinhamento é postergada. Durante o cálculo do alinhamento, nas posições ainda indefinidas utilizamos a base que maximiza a pontuação do alinhamento.

4. O número de mutações ao longo dos ramos é calculado e assume-se estes comprimentos de ramos de árvore não alterando a topologia.
5. Calcula-se a freqüência mutacional de cada posição do alinhamento da seguinte forma:

$$W_i = \frac{N_i}{M_i + 1},$$

onde W_i é o peso da i-ésima posição, N_i é o número de nucleotídeos diferentes na posição e M_i é o número de mutações que ocorrem naquela posição. Essas freqüências podem ser usadas como pesos de caracteres.

6. As seqüências alinhadas são usadas na próxima execução do procedimento iterativo. Uma nova matriz de similaridade é então calculada.
7. O processo é repetido a partir do passo 2 até que haja convergência ou até que uma regra de parada é satisfeita.

A.7.2 Conclusão

Neste trabalho defende-se que a construção de árvores filogenéticas e alinhamento múltiplo de seqüências são tarefas que não têm sentido se pensadas isoladamente. Sugere-se um método que ao mesmo tempo constrói árvores filogenéticas e alinhamentos múltiplos. No decorrer do tempo, tal método mostrou-se bastante eficiente e é hoje em dia o método mais empregado para alinhamento múltiplo, sendo utilizado em pacotes como o Clustal W [215].

Os dois principais problemas apontados neste método dizem respeito a convergência e parcimônia. O primeiro corresponde à dificuldade em se detectar ciclos nas árvores geradas, ou seja, dificuldade em descobrir quando os alinhamentos e árvores não irão mais ganhar em qualidade. O segundo surge pelo uso de um método não exato, que impossibilita qualquer garantia de que a árvore e o alinhamento obtidos são os melhores ou mais adequados.

A.8 CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment

Este trabalho [215] detalha o Clustal W, que é o *software* mais utilizado para alinhamento multiplo de seqüências. Esta é a terceira versão do *software*. Anteriormente foram implementadas as versões Clustal [103] e Clustal V [102].

Alinhamento múltiplo de seqüências (MSA, do inglês *Multiple Sequence Alignment*) tornou-se uma ferramenta essencial em biologia molecular, como podemos ver nas suas diversas aplicações: encontrar padrões para caracterizar famílias de proteínas, detectar homologia entre novas seqüências e famílias existentes, dar suporte a predição de estruturas secundárias e terciárias de novas seqüências, sugerir *primers* para PCR (do inglês, *Polymer Chain Reaction*) ou para análise evolucionária molecular.

Quando alinha-se apenas um par de seqüências é comum utilizar o algoritmo de Needleman e Wunsch [158], que através de uma abordagem de programação dinâmica garante um alinhamento matematicamente ótimo, dada uma tabela de pesos (com pontuações para *matches* e *mismatches*) e penalização para *gaps* de diferentes tamanhos. Uma primeira abordagem para MSA é estender o algoritmo de Needleman e Wunsch, mas desta forma ficamos limitados a um pequeno número de seqüências [136]. Isso ocorre porque a medida que o número e/ou o tamanho das seqüências cresce, o problema torna-se impraticável devido aos elevados requisitos de memória ou tempo de computação. Assim, qualquer método capaz de tratar um grande número de seqüências em um tempo razoável faz uso de heurísticas. A grande maioria dos algoritmos para MSA usa a abordagem progressiva de Feng e Doolittle [67], que iremos descrever a seguir. Tal abordagem apresenta bons resultados quando as seqüências são similares, mas quando todas as seqüências são altamente divergentes (menos que 25-30% de identidade entre qualquer par de seqüências), torna-se muito menos confiável.

A abordagem progressiva apresenta dois problemas principais: mínimo local e escolha dos parâmetros de alinhamento. O problema do mínimo local surge pela natureza gulosa da estratégia de alinhamento. Qualquer mal alinhamento realizado no início do processo não pode ser mais corrigido apenas com o uso da abordagem progressiva. A única forma de corrigir este problema é através do uso de um procedimento iterativo ou de amostragem estocástica [79,131,139]. Já o problema na escolha dos parâmetros surge do fato das escolhas das pontuações serem mais adequadas para seqüências similares ou divergentes. Assim como surgem pelo fato das pontuações para *gaps* também serem sensíveis às seqüências e às diferentes posições/estruturas na seqüência. *Gaps* são mais freqüentes em determinadas posições, que em outras.

Neste artigo são descritas melhorias para MSA progressivo, que aumenta a sensibilidade sem sacrifícios em termos de velocidade e eficiência. Tais melhorias focam na escolha dos parâmetros.

A.8.1 O método básico de alinhamento

O algoritmo progressivo básico consiste de três estágios:

1. Geração de uma matriz de distâncias a partir de alinhamentos de pares.
2. Geração de uma árvore guia a partir da matriz de distâncias.
3. Alinhamento progressivo das seqüência de acordo com a árvore guia.

Para os alinhamentos do primeiro estágio, é possível optar pelo método aproximado descrito por Bashford e colegas [26] ou pelo algoritmo de Needleman e Wunsch [158]. O primeiro consome menos tempo e o segundo oferece resultados precisos. Uma vez alinhadas duas seqüências, calcula-se sua pontuação dividindo o número de identidades pelo número de resíduos comparados, excluindo-se os *gaps*. O valor obtido é subtraído de 1 para gerar a distância, que é registrada na matriz.

Uma vez gerada a matriz de distâncias, construímos a árvore guia utilizando o método *Neighbor-Joining* [186]. É importante salientar que versões anteriores do Clustal utilizavam o método UPGMA [201] para a construção da árvore guia. Tal método gera uma árvore onde as seqüências são suas folhas e a cada aresta é associado um valor diretamente relacionado

aos valores da matriz. A partir de tais valores das arestas são calculados pesos para cada uma das seqüências. Os pesos são normalizados de forma que o maior deles é redefinido para 1 e os outros são ajustados na mesma proporção do maior. A grosso modo, seqüências com pesos maiores estão mais distantes das outras seqüências.

Seguindo as ramificações da árvore guia e usando uma série de alinhamentos, alinha-se grupos cada vez maiores de seqüências. Inicia-se por alinhar as folhas mais próximas (de menor peso, que estão em ramos diretamente ligados). Segue-se alinhando, sempre buscando os ramos ligados que possuam menor peso. A cada passo utiliza-se o algoritimo Needleman-Wunsch. *Gaps* não são removidos. Para calcular a pontuação de uma posição de uma seqüência ou alinhamento e uma de outro, é calculada a média das pontuações de cada par formado por uma seqüência de um grupo com uma seqüência do outro. Por exemplo, se alinharmos 2 alinhamentos com 2 e 3 seqüências, a pontuação de cada posição é a média de 6 comparações. As pontuações são multiplicadas pelos pesos das seqüências. Comparações envolvendo *gaps* são pontuadas com 0. Os valores para *matches* e *mismatches* são sempre positivos.

A.8.2 Melhorias

A primeira parte das melhorias é a atribuição de pesos às seqüências. Seqüências altamente relacionadas recebem pesos menores, pois contém muita informação duplicada. Já seqüências altamente divergentes recebem pesos maiores. A segunda parte das melhorias fica por conta dos *gaps*. São utilizadas duas penalizações para os *gaps*: abertura (GOP, do inglês *gap opening penalty*) e extensão (GEP, do inglês *gap extension penalty*). O GOP indica o custo de abrir um *gap*. Já o GEP indica o custo de cada item do *gap*, ou ainda, de estender um *gap*. Dados valores iniciais para o GOP e GEP, que podem ser definidos pelo usuário, o algoritmo tenta escolher valores apropriados para cada alinhamento de seqüência.

Para a definição do GOP ele usa os seguintes parâmetros: tabela de *mismatches*, similaridade das seqüências e tamanho das seqüências. A tabela de *mismatches* é utilizada para variar as penalizações para *gaps* de acordo com a matriz de pesos, que já comprovou ser uma boa forma de aumentar a precisão do alinhamento [140, 200]. Aumenta-se o custo dos *gaps* em seqüências relacionadas e diminui-se tal custo em seqüências divergentes. Além disso o GOP é incrementado pelo logarítmo do tamanho da menor seqüência.

Para a definição do GEP apenas utiliza-se a diferença entre os tamanhos da seqüências como parâmetro. Se uma seqüência é muito menor que a outra, aumenta-se o custo do GEP para inibir a criação de muitos *gaps* longos na seqüência menor.

Antes de qualquer par de seqüências (ou grupos de seqüências alinhadas) ser alinhado, são recalculadas as penalizações para cada posição nos dois conjuntos de seqüências. Se já há *gap* na posição, GOP e GEP são reduzidos e as outras regras não se aplicam. Se não há *gaps* na posição então o GOP é aumentado caso esteja a no máximo 8 resíduos de distância de um *gap*. Esta regra inibe *gaps* muito próximo. Para qualquer posição em uma cadeia de resíduos hidrofílicos, que geralmente indicam laços na proteína, a penalização é reduzida. São geralmente considerados resíduos hidrofílicos: D, E, G, K, N, P, Q, R e S. Se a posição não pertence a uma cadeia de resíduos hidrofílicos a penalização é modificada de acordo com os resíduos [172]. Caso a posição possua uma variedade de resíduos, o fator a utilizar é dado pela média das contribuições de cada seqüência.

As duas principais séries de matrizes de pesos são: PAM [50] e BLOSUM [95]. Sendo a

segunda a mais comumente utilizada. Cada série provê matrizes que se aplicam às seqüências de acordo com o grau de similaridade. Elas oferecem matrizes mais restritivas para o caso de seqüências similares e mais flexíveis para seqüências divergentes.

Para tornar possível o alinhamento de seqüências compridas utiliza-se o algoritmo de Myers e Miller [156]. Tal algoritmo sacrifica um pouco o tempo de processamento, mas reduz sensivelmente o uso de memória. É importante salientar que este algoritmo inicialmente não permitia adotar pontuações distintas para abertura e extenção de *gap*. Thompson modificou tal algoritmo para permitir o uso do GOP e GEP [213].

A.8.3 Conclusão

É possível corrigir regiões mal alinhadas pela repetição do procedimento apenas no trecho problemático. Na repetição pode-se redefinir os parâmetros. Os autores mostram um caso onde conseguiram melhorar uma região mal alinhada com apenas mais uma execução e iguais parâmetros. Os autores citam dois outros trabalhos [127,198] onde também alteram-se os parâmetros de acordo com as posições. Defende-se que a abordagem utilizada no Clustal W faz o mesmo só que utilizando apenas as seqüências a alinhar. Desta forma, não cria dependências e torna-se mais flexível.

As modificações descritas neste trabalho aumentam a sensibilidade do MSA progressivo sem promover sacrifícios na velocidade e na eficiência. Mas os autores ainda citam áreas onde é possível melhorar. As matrizes de peso e as penalizações dos *gaps* podem se tornar mais precisas a medida que mais dados tornam-se disponível e que são criadas medidas para tipos específicos de seqüência. Pode-se melhorar também através do uso de métodos de otimização iterativos ou estocásticos. Outro ponto de grande relevância para estudo é a função objetivo.

A.9 Multiple DNA and protein sequence alignment based on segment-to-segment comparison

Este trabalho [152] apresenta o primeiro método para alinhamento que não requer qualquer definição de penalidade para *gap*.

Em 1970, Needleman e Wunsch [158] publicaram um algoritmo para alinhamento de duas seqüências. Tal algoritmo alinha seqüências por maximizar uma pontuação que é atribuída de acordo com *matches* e adição de *gaps*. Desde então este algoritmo foi estendido e a grande maioria dos algoritmos hoje em dia são baseados nesta idéia. Tais algoritmos apresentam bons resultados se as seqüências são altamente relacionadas, mas apresenta grande dificuldade em detectar curtas regiões similares em seqüências longas de baixa similaridade. Há problema também no fato de seus resultados serem altamente sensíveis ao conjunto de parâmetros definidos pelo usuário, especialmente a penalidade para *gap*. Outro problema é que, embora facilmente extensível a mais que duas seqüências, a complexidade do algoritmos cresce exponencialmente com o número de seqüências [154,158].

Neste trabalho é proposto um novo conceito de alinhamento de seqüências onde segmentos de tamanhos variados são alinhados sem uma definição explícita de *gap*.

A.9.1 Algoritmo básico para alinhamento de duas seqüências

É requerido que os segmentos a alinhar possuam o mesmo tamanho e não é permitido qualquer *gap* nos segmentos. Os alinhamentos são também chamados de diagonais, uma vez que o par de seqüências é uma diagonal na matriz de pontos. O método é chamado de DiAlign.

Um alinhamento é definido como uma relação consistente de eqüivalência no conjunto de todas as posições de todas as seqüências envolvidas. Consistente significa que a ordem geral das posições em cada seqüência é respeitada. Se apenas alinhamos duas seqüências, há consistência se para quaisquer duas diagonais D_1 e D_2 pertencentes ao alinhamento ou $D_1 \ll D_2$ ou $D_2 \ll D_1$, onde "$D_1 \ll D_2$" significa que as posições alinhadas em D_1 precedem aquelas alinhadas em D_2.

Define-se uma medida para avaliar a relevância de uma dada diagonal. Seja D uma diagonal, l o tamanho de D e m o número de *matches* em D. Denota-se por $P(l, m)$ a probabilidade de qualquer diagonal de tamanho l conter, pelo menos, m *matches*. Segue que:

$$P(l,m) = \sum_{i=m}^{l} \binom{l}{i} p^i (1-p)^{l-i}, \tag{1}$$

onde p é a probabilidade de um ponto da matriz de pontos representar um *match*. A título de simplificação, podemos assumir uma distribuição uniforme ($p = 0.25$ para ácido nucléico e $p = 0.05$ para proteína).

Define-se também peso de uma diagonal D por:

$$w(D) = \begin{cases} E(l,m), & se\ E(l,m) > T \\ 0 & se\ E(l,m) \leq T, \end{cases} \tag{2}$$

onde T é um limiar definido pelo usuário e

$$E(l,m) = -\ln P(l,m). \tag{3}$$

Se as diagonais D_1, D_2, ..., D_c constituem um conjunto consistente de diagonais, a pontuação deste conjunto é definida como $\sum_{i=1}^{c} w(D_i)$. Um conjunto consistente de diagonais com pontuação máxima é chamado alinhamento máximo.

O método utiliza programação dinâmica e recursão para calcular um alinhamento máximo. Seja $X = (X_1, \ldots, X_{L_1})$ e $Y = (Y_1, \ldots, Y_{L_2})$ duas seqüências. O algoritmo inicialmente determina, para todo par de posições (i, j) com $1 \leq i \leq L_1$ e $1 \leq j \leq L_2$, todos os inteiros $k \geq 0$, com $k \leq min(i-1, j-1)$, para os quais a diagonal $(X_{i-k}, Y_{j-k}; X_i, Y_j)$ de $(i-k, j-k)$ até (i, j) tem peso positivo. Depois, para cada um dos pares (i, j), define-se o valor $pontuacao(i, j)$, que é a pontuação de um alinhamento máximo dos prefixos $(X_1, \ldots, X_i)$ e $(Y_1, \ldots, Y_j)$. Sejam $D_1, D_2, \ldots, D_c$ as diagonais do alinhamento máximo dos prefixos $(X_1, \ldots, X_i)$ e $(Y_1, \ldots, Y_j)$ e $D_1 \ll D_2 \ll \ldots \ll D_c$. Define-se $prec(i, j) = D_c$.

Para cada diagonal $D = (X_{i-k}, Y_{j-k}; X_i, Y_j)$ com peso positivo, denota-se por $\sigma(D)$ a soma máxima dos pesos acumulados e incluindo D. Denota-se por $\pi(D)$ a diagonal que precede D. Note que as relações apresentadas nas Equações 4 e 5 são válidas.

$$\sigma(D) = pontuacao(i-k-1, j-k-1) + w(D) \tag{4}$$

$$\pi(D) = prec(i-k-1, j-k-1) \tag{5}$$

Assim, pode-se computar recursivamente o valor da pontuação por

$$pontuacao(i,j) = max\{pontuacao(i,j-1), pontuacao(i-1,j), \sigma(D_{i,j})\}$$

onde $D_{i,j}$ é qualquer diagonal que termina no ponto (i,j) e satisfaz

$$\sigma(D_{i,j}) = max\{\sigma(D) : \ D \ termina \ no \ ponto \ (i,j)\},$$

enquanto que *prec* pode ser definido por

$$prec(i,j) = \begin{cases} prec(i,j-1) & se \ pontuacao(i,j) = pontuacao(i,j-1), \\ prec(i-1,j) & se \ pontuacao(i,j-1) < pontuacao(i,j) = pontuacao(i-1,j), \\ D_{i,j} & se \ pontuacao(i,j-1) < pontuacao(i,j) = \sigma(D_{i,j}) \\ & e \ pontuacao(i-1,j) < pontuacao(i,j) = \sigma(D_{i,j}). \end{cases}$$

Finalmente, define-se $D_1 = prec(L_1, L_2)$ e $D_{i+1} = \pi(D_i)$. Agora utilizando-se programação dinâmica, calcula-se todos estes valores, que então permite-nos encontrar o alinhamento máximo.

A.9.2 Alinhamento múltiplo

Uma possível abordagem para estender o algoritmo básico de duas seqüências para n seqüências, onde $n \geq 3$, seria utilizar diagonais N-dimensionais, mas tal abordagem levaria a um grande crescimento na complexidade computacional do algoritmo. Decidiu-se então adotar uma abordagem onde constrói-se alinhamentos múltiplos a partir de todas as diagonais em duas dimensões de todos os $\frac{n}{2}(n-1)$ pares de seqüência.

Uma vez calculadas as diagonais de todos os pares de seqüências, construímos um conjunto com todas as diagonais. Denotamos tal conjunto por $\mathcal{D}$. Para reduzir o número total de diagonais em $\mathcal{D}$ e, assim, ganhar eficiência, decidiu-se por apenas adicionar a $\mathcal{D}$ aquelas diagonais que pertençam ao alinhamento máximo de um dos pares de seqüências.

O passo seguinte é ordenar $\mathcal{D}$ de acordo com o peso das diagonais. Valendo-se de uma estratégia gulosa adicionamos uma a uma as diagonais começando pela de maior peso. Antes de adicionar uma nova diagonal ao alinhamento múltiplo é necessário verificar se a consistência do alinhamento é mantida. Caso contrário ela é descartada.

A.9.3 Conclusão

O método é capaz de encontrar regiões limitadas de similaridade, ou seja, encontrar alinhamentos locais assim como definido por Smith e Waterman [199], mas não apenas o melhor alinhamento local. Ele é capaz de encontrar as diversas regiões similares cujo peso ultrapasse o limiar.

Apesar de construir alinhamentos múltiplos usando alinhamentos iterativos de pares assim como outros autores [10,67,79], no método DiAlign a ordem de execução dos alinhamentos dos pares não são cruciais para os resultados.

Uma grande vantagem do método é a pequena dependência nos parâmetros definidos pelos usuários. Tal vantagem origina-se em não tratar *gaps* explicitamente e assim não requer a definição de penalidades para *gaps*.

Em testes apresentados no trabalho fica claro o potencial do método. O método apresenta resultados próximos dos melhores métodos com a vantagem de ser independente quanto a definição de parâmetros.

A.10 ProbCons: Probabilistic consistency-based multiple sequence alignment

Este trabalho [54] introduz o conceito de consistência probabilística, assim como apresenta o alinhador ProbCons. Tal ferramenta realiza alinhamento múltiplo de seqüências de proteína através de um algoritmo progressivo baseado em consistência probabilística. De acordo com avaliações dos autores através de *benchmarks* tais como BAliBASE [217], SABmark [229] e PREFAB [63], o ProbCons alcança resultados significativamente mais precisos quando comparado com diversas ferramentas largamente empregadas para alinhamento múltiplo, ao mesmo tempo que mantém uma velocidade praticável de execução. Código fonte e executáveis estão disponíveis em domínio público [55].

Modelos ocultos de Markov (HMM, do inglês *hidden Markov models*) é um dos métodos que podem ser empregados na solução do problema do alinhamento múltiplo de seqüências. Nesta abordagem parâmetros do modelo obtêm uma interpretação probabilística intuitiva e podem ser treinados em dados reais usando métodos probabilísticos padrões, supervisionados ou não. O algoritmo de Viterbi [226], por exemplo, computa a maior probabilidade de alinhamento de duas seqüências de acordo com um alinhamento HMM. No HMM padrão para alinhamento (com três-estados), o algoritmo de Viterbi pode ser visto como uma instanciação do algoritmo de Needleman e Wunsch [158] no qual os parâmetros de alinhamento são determinados por uma transformação do esquema de pontuação do modelo HMM [58].

Para alinhamento múltiplo de seqüências, a estratégia heurística de alinhamento progressivo [67] é, de longe, a mais popular nas ferramentas disponíveis. Neste esquema, o alinhamento final é construído a partir de diversos passos de alinhamento de pares. Como em qualquer abordagem hierárquica, erros cometidos em estágios iniciais do processo são propagados até o final, assim como podem levar a geração de novos erros devido a sinais incorretos de conservação. Para amenizar este tipo de problema são adicionados passos de pós-processamento tais como refinamento iterativo [82].

Esquemas baseados em consistência possuem uma visão alternativa da solução. Eles adotam a prática da "prevenção é o melhor remédio". Observe que para qualquer

alinhamento múltiplo, os alinhamentos induzidos dos pares são necessariamente consistentes, ou seja, dado um alinhamento múltiplo contendo três seqüências x, y e z, se a posição x_i alinha com a posição z_k e a posição z_k alinha com y_j nos alinhamentos projetados $x - z$ e $z - y$, então x_i deve alinhar com y_j no alinhamento projetado $x - y$. Técnicas baseadas em consistência aplicam este princípio de forma invertida para guiar os alinhamentos de pares durante os passos de um alinhamento progressivo. Ao ajustar a pontuação do pareamento dos resíduos $x_i \sim y_j$ de acordo com o suporte de uma posição z_k que alinha a ambos x_i e y_j nas respectivas comparações $x - z$ e $z - y$, funções objetivo baseadas em consistência incorporam informação de múltiplas seqüências na pontuação dos alinhamentos de pares.

Consistência já foi utilizada para identificar pontos âncora [78], que reduziam o espaço de busca de um alinhamento múltiplo. Consistência já foi reformulada para ser tratada por multiplicação de matrizes booleanas, que deu origem ao *software* MALI [223]. Uma formulação alternativa deu origem ao *software* DiAlign [152]. Recentemente foi introduzida uma nova função objetivo baseada em consistência COFFEE [166]. Baseado nesta função, foi implementada a ferramenta T-COFFEE [165], que apresenta resultados superiores a métodos como Clustal W [215], DiAlign e PRRP [82] de acordo com comparações realizadas no BAliBASE. Sendo que o Clustal W é o *software* mais utilizado para alinhamento múltiplo.

A.10.1 O algoritmo

ProbCons é um algoritmo de alinhamento progressivo baseado em HMM para pares de seqüências. Ele utiliza o algoritmo de *máxima precisão esperada* ao invés do algoritmo de Viterbi, que é tipicamente utilizado, além de fazer uso de *transformações de consistência probabilística* para incorporar informações acerca de conservação de seqüências durante alinhamento de pares. ProbCons utiliza o modelo HMM apresentado na Figura 2 para especificar a probabilidade de distribuição sobre todos os alinhamentos entre um par de seqüências. Probabilidades de emissão correspondem aos valores da matriz de substituição BLOSUM62 [95]. Probabilidades de transição, que correspondem a penalidades para *gaps*, são treinadas com *maximização de expectativa* não supervisionada (EM, do inglês *expectation maximization*).

Mantendo as probabilidades de emissão fixas (BLOSUM62), o modelo HMM do ProbCons é completamente especificado por apenas três parâmetros: probabilidade de iniciar o alinhamento com a inserção de um *gap* π_{insert}, probabilidade de abrir um *gap* δ e a probabilidade de estender um *gap* ϵ. Para treinar o ProbCons via EM, foram aplicadas 20 iterações do algoritmo Baum-Welch [29] nas seqüências não alinhadas do BAliBASE. Os parâmetros foram inicializados com valores aleatórios. Os parâmetros resultantes ($\pi_{insert} = 0,19598$, $\delta = 0,019931$ e $\epsilon = 0,79433$) são usados como padrão pelo programa.

O modelo HMM do ProbCons é extremamente simples se comparado com outros como o de Durbin e colegas [58], que com uma abordagem de construção de perfis-HMM é mais rico, porém possui grande número de parâmetros e conseqüentemente

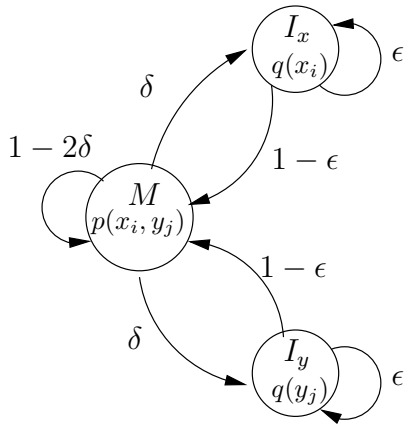

Figura 2: Modelo oculto de Markov três-estados, utilizado pelo ProbCons, para o alinhamento entre duas seqüências, x e y. O estado M emite duas letras, uma de cada seqüência. Corresponde ao alinhamento delas. O estado I_x emite uma letra da seqüência x e corresponde ao alinhamento da letra de x com um *gap*. De forma semelhante, I_y emite uma letra da seqüência y. Os símbolos δ e ϵ denotam, respectivamente, probabilidades de abrir e estender um *gap*. A função p denota a probabilidade de emissão de M e tem seus valores calculados a partir da matriz de substituição. A função q denota a probabilidade de emissão para I_x e I_y.

torna a tarefa de treinamento mais complexa.

Dadas m seqüências $S = \{s^{(1)}, \ldots, s^{(m)}\}$, o algoritmo realiza o alinhamento da seguinte forma.

Passo 1: Cálculo das matrizes de probabilidade posterior. Para todo par de seqüências $x, y \in S$ e todo $i \in \{1, \ldots, |x|\}$, $j \in \{1, \ldots, |y|\}$, calcule a matriz P_{xy}, onde $P_{xy}(i, j) = P(x_i \sim y_j \in a^*|x, y)$ é a probabilidade que letras x_i e y_j são pareadas em a^*, um alinhamento de x e y gerado pelo modelo.

Passo 2: Cálculo da precisão esperada. Defina a precisão esperada de um alinhamento a entre x e y como sendo o número esperado de pares de letras corretamente alinhados dividido pelo tamanho da seqüência mais curta:

$$E_{a^*}(accuracy(a, a^*)|x, y) = \frac{1}{\min\{|x|, |y|\}} \sum_{x_i \sim y_j \in a} P(x_i \sim y_j \in a^*|x, y).$$

Para cada par de seqüências $x, y \in S$, compute o alinhamento a que maximiza a precisão esperada através de programação dinâmica e defina o respectivo valor $E(x, y) = E_{a^*}(accuracy(a, a^*)|x, y)$.

Passo 3: Cálculo das transformações de consistência probabilística. Redefina as pontuações $P(x_i \sim y_j \in a^*|x, y)$ pela aplicação de transformações de consistência probabilística, que incorporam a similaridade de x e y a outras seqüências de S no alinhamento $x - y$:

$$P'(x_i \sim y_j \in a^*|x,y) \leftarrow \frac{1}{|S|}\sum_{z\in S}\sum_{z_k} P(x_i \sim z_k \in a^*|x,z)P(z_k \sim y_j \in a^*|z,y).$$

Na forma de matriz, isso pode ser reescrito da seguinte forma:

$$P'_{xy} \leftarrow \frac{1}{|S|}\sum_{z\in S} P_{xz}P_{zy}.$$

Algo importante aqui é que a maioria dos valores nas matrizes P_{xz} e P_{zy} são próximos a zero. Assim a transformação pode ser computada eficientemente usando algoritmos para multiplicação de matrizes esparsas ao ignorar entradas menores que um limiar ω (por padrão $\omega = 0,01$). Este passo pode ser repetido tantas vezes quanto desejado. Com o uso de algoritmos para multiplicação de matrizes esparsas, este passo cai de uma complexidade $O(L^3)$ para $O(c^2L)$, onde L corresponde ao tamanho das seqüências e c é o número médio de elementos diferentes de zero por linha (tipicamente $1 \leq c \leq 5$).

Foram feitos testes para validar a implementação do limiar, que gera matrizes esparsas e assim permite um melhor desempenho do sistema. Os resultados mostraram que as execuções com (algoritmo $O(c^2L)$) e sem uso (algoritmo $O(L^3)$) de limiar apresentam qualidade semelhantes. Tal qualidade é praticamente idêntida quando se realizam 2 passos de refinamento iterativo.

Passo 4: Geração da árvore guia. Contrua a árvore guia para S através de agrupamento hierárquico. Como uma medida de similaridade entre duas seqüências x e y use $E(x,y)$, calculado no segundo passo. Defina a similaridade entre dois grupos por uma média ponderada das similaridades de pares entre seqüências dos dois grupos.

A maioria dos alinhadores usam distância evolucionária estimada de alinhamentos de pares ou estatísticas de k-mer para construir uma árvore evolucionária aproximada via algoritmo "neighbor joining" [186] ou UPGMA [201]. No ProbCons não há uma tentativa de construção de uma árvore evolucionária correta, mas sim uma árvore com alta confiabilidade esperada de alinhamento através de um método guloso derivado do UPGMA.

Passo 5: Alinhamento progressivo. Alinhe os grupos de seqüências iterativamente de acordo com a ordem especificada pela árvore guia. Alinhamentos são pontuados usando uma função de soma-de-pares na qual resíduos alinhados são associados a pontuações transformadas $P'(x_i \sim y_j \in a^*|x,y)$ e penalidades para *gaps* são definidas como zero.

Passo 6: Refinamento iterativo. Particione o alinhamento múltiplo aleatoriamente em dois grupos de seqüências e realinhe. Este passo também pode ser repetido tantas vezes quanto desejado.

A.10.2 Testes

A performance do ProbCons foi testada usando três diferentes pacotes de *benchmark* para alinhamento múltiplo. Foram eles BAliBASE 2.01 [216], PREFAB 3.0 [63] e SABmark 1.63 [229].

Os resultados do ProbCons foram comparados com os resultados de seis outros sistemas de alinhamento múltiplo: Clustal W 1.83 [215] (o método mais popular), DiAlign 2.2.1 [152] (alinhador local que usa homologia baseada em segmento), T-COFFEE 1.37 [165] (alinhador baseado em consistência que combina alinhamentos globais e locais), MAFFT 3.88 [122] (um conjunto de seis *scripts* com uma variedade de técnicas de refinamento iterativo), MUSCLE 3.3 [62] (alinhador recente que apresenta os melhores resultados no BAliBASE até então) e Align-m 1.0 [230] (utiliza um método baseado em consistência). Todos os sistemas foram avaliados de acordo com os respectivos valores padrões de parâmetros.

O modelo HMM do ProbCons foi treinado usando a base de dados do BAliBASE, sendo assim, o PREFAB e o SABmark foram utilizados, também, com o objetivo de prover uma validação externa. Os resultados mostraram que o ProbCons traz contribuições de grande relevância na qualidade dos alinhamentos. Mostra também que o ProbCons exige um tempo de execução elevado. O tempo requisitado é maior que a exigida pela maioria dos sistemas testados, entretando tal demanda não chega a tornar proibitiva sua utilização.

A.10.3 Conclusão

ProbCons utiliza um modelo simples de similaridade de seqüências (HMM de três estados) e não faz tentativas de incorporar conhecimento biológico tal como pontuação de *gap* para posição específica, construção de árvores evolucionárias rigorosas e outras características usadas por alinhadores tais como Clustal W. ProbCons não usa informações além das extraídas das matrizes de substituição BLOSUM. Realizando a substituíção desta por valores equivalentes para nucleotídeos é possível gerar alinhamentos múltiplos para DNA.

Como todo treinamento para o programa foi feito automaticamente em seqüências não alinhadas usando EM sem intervenções humanas, é possível retreinar ProbCons para tipos específicos de seqüência. E assim obter parâmetros mais apropriados para tarefas particulares de alinhamento.

Teste realizados com diferentes variações do ProbCons indicam que as características determinantes para sua precisão são o uso de máxima precisão esperada como função objetivo e a aplicação de transformações de consistência probabilística.

A abordagem de consistência probabilística pode ser interessante para busca comparativa de genes e predição de RNA ou estruturas de proteínas.

A.11 SAGA: sequence alignment by genetic algorithm

Este trabalho [164] apresenta a primeira tentiva de aplicar algoritmos genéticos a alinhamento de seqüências.

No contexto de alinhamento múltiplo de seqüências, a abordagem progressiva de Feng e Doolittle [67] ou variações [25, 209, 215] é a que mais se adota nas soluções existentes. Tal abordagem apresenta velocidade e simplicidade aliadas a uma boa sensibilidade, mas enfrenta problemas com mínimo local. Uma das principais alternativas à abordagem progressiva é o uso de funções objetivo, que quantificam a qualidade de um dado alinhamento. Os métodos que fazem uso de tais funções buscam o alinhamento de maior qualidade. Uma outra alternativa são modelos ocultos de Markov (HMM, do inglês *Hidden Markov Models*) [127]. Em ambos os casos enfrenta-se problemas com o crescimento no número de seqüências a alinhar.

Visando solucionar este problema com o uso de funções objetivo, pode-se utilizar o pacote MSA ou métodos estocásticos. O pacote MSA [90, 136] tenta reduzir o espaço de solução para uma área relativamente pequena onde parece estar a solução. Ele encontra o alinhamento ótimo global ou um muito próximo dele começando com um conjunto de seqüências sem qualquer alinhamento. Mesmo com esta redução no espaço de solução, o método continua com grandes limitações quanto ao número de seqüências. Os métodos estocásticos podem ser implementados de diversas formas. Uma destas formas é o *simulated annealing* [2], que é bastante utilizado [104,111,125], mas parece apenas funcionar bem no refinamento de alinhamentos pré-construídos. Outra forma é o *Gibbs Sampling* [131], que mostrou bons resultados na busca por melhor bloco de alinhamento múltiplo local sem *gaps*. E outra forma ainda são os algoritmos genéticos (GA, do inglês *genetic algorithm*) [106]. Até então havia um único trabalho conhecido na área que fazia uso de GA, mas este utilizava um esquema híbrido de programação dinâmica/GA [111].

No restante desta seção é descrita a estratégia usada pelo método. Tal estratégia deu origem a um sistema chamado SAGA, cujos fontes é possível obter com os autores.

A.11.1 Método

O método basicamente usa uma medida para qualidade de alinhamento múltiplo, que é a chamada função objetivo (OF, do inglês *objective function*), e algoritmo genético para fazer a otimização. A OF utilizada é uma soma de pares com peso e penalidades afim para *gaps*. A cada par de seqüências é associado um peso que indica o grau de similaridade em relação às outras seqüências e é uma tentativa de minimizar informação redundante. Observe que a OF pode ser variada de diversas formas: podemos usar diferentes conjuntos de pesos, diferente conjuntos de custos para substituições (matrizes PAM [50] ou tabelas BLOSUM [95]) ou ainda diferentes esquemas para pontuação de *gaps*. Segue o custo total de um alinhamento múltiplo (A).

$$Custo(A) = \sum_{i=2}^{n} \sum_{j=1}^{i-1} W_{i,j} * Custo(A_i, A_j)$$

onde $Custo(A_i, A_j)$ indica a pontuação do alinhamento entre as seqüências A_i e A_j e $W_{i,j}$ é o peso das seqüências. Note que $Custo(A_i, A_j)$ inclui penalizações para abertura e extensão de *gaps*.

A estratégia utilizada deriva diretamente de algoritmo genético descrito por Goldberg [73]. Ou seja, há uma população inicial de soluções que evolui por meio de seleção natural. Em nosso caso, cada indivíduo na população é um alinhamento. Inicialmente, é gerada aleatoriamente uma geração zero (G_0) com 100 indivíduos (o tamanho da população é mantido constante). Para ir para a próxima geração, filhos são gerados de pais selecionados por uma espécie de seleção natural, com base no grau de afinidade dos pais mensurado pela OF. Uma vez selecionados os pais, para gerar o filho ainda é necessário selecionar um dos operadores. Para a nova geração é mantida uma porção de indivíduos da população corrente. Estes passos são iterativamente repetidos até que chegamos a uma situação de estabilidade na população. No Algoritmo 1 é apresentado um pseudo-código com os passos.

Algoritmo 1: Versão simplificada do algoritmo utilizado pelo SAGA.

```
Cria G_0
while população não estabilizar do
    Seleciona indivíduos para substituição
    Avalia a prole esperada
    while população G_{n+1} incompleta do
        Seleciona os pais dentro da população corrente
        Seleciona o operador
        Gera um novo filho
        Mantém ou descarta o novo filho em G_{n+1}
    end
    n = n + 1
end
```

Um operador é um pequeno programa que modifica um alinhamento. No SAGA foram definidos 22 operadores, que possuem probabilidades específicas de serem utilizados. Os operadores podem pertencer a uma de duas classes: cruzamento e mutação. Os operadores de cruzamento têm por característica combinar o conteúdo de dois indivíduos para gerar um terceiro. Já os operadores de mutação simplesmente gera um indivíduo a partir de modificações em um outro indivíduo.

Durante a execução do algoritmo é utilizado um escalonador dinâmico para a seleção do operador [49]. O escalonador atribui inicialmente probabilidades iguais de seleção para cada operador e altera tais probabilidades de acordo com a evolução

da população. A probabilidade de uma dada operação ser utilizada é incrementada em função de sua eficiência nas últimas gerações. Quanto a escolha dos locais para a realização das mutações, foi criado um mecanismo que privilegia a seleção de pontos onde há maior concentração de *gaps*.

Foram feitos diversos testes com o SAGA comparando seu desempenho com o MSA e com o Clustal W. O SAGA obteve resultados tão bons ou, em alguns casos, melhores que os outros.

A.11.2 Conclusão

O uso de algoritmos genéticos para alinhamento múltiplo de seqüências parece ser uma boa alternativa como heurística para o problema, mas certamente ainda há muito a melhorar. Um grande problema encontra-se em seu tempo de resposta, que possivelmente poderá ser resolvido com a utilização de um método híbrido usando abordagem progressiva / algoritmo genético. Desta forma, tentando unir a velocidade da abordagem progressiva com a precisão do algoritmo genético. Outro trabalho futuro seria avaliar a possibilidade de tornar o SAGA uma ferramenta para refinamento de alinhamentos. Para tanto poderia utilizar como entrada uma primeira geração composta por resultados do Clustal W.

Os autores creditam os bons resultados do SAGA ao grande número de operadores e ao escalonador. Eles também destacam como grande característica do *software* a flexibilidade para substituir a OF e poder mudar sensivelmente os resultados, podendo assim rapidamente testar novas OFs.

A.12 Further improvement in methods of group-to-group sequence alignment with generalized profile operations

Este trabalho [80] foi a primeira tentativa de sistematicamente melhorar a precisão do método MSA pela uso de alinhamento estrutural.

É conhecido que é possível estender o algoritmo que faz uso de programação dinâmica para o caso de alinhamento entre dois grupos de seqüências. Porém sua grande desvantagem encontra-se no fato do tempo computacional crescer na proporção que o produto do número de seqüências nos dois grupos ($M \times N$) cresce. Neste trabalho é apresentado um método que possui precisão semelhante nos resultados, mas complexidade de tempo menos dependente no tamanho dos grupos. É importante salientar que tal desempenho só é alcançado quando há um número grande de seqüências sendo alinhadas, por exemplo: testes realizados mostraram que ele executa ~ 10 vezes mais rápido quando $M \times N \approx 200$ e mais que 100 vezes mais rápido quando $M \times N > 2500$.

Alinhamento de grupos de seqüências tem mostrado sua importância no contexto de MSA. Um bom exemplo disto é o fato de geralmente ser melhor alinhar grupos distantes de seqüências homólogas que fazer alinhamentos individuais [13]. Um outro

bom exemplo é o fato de ser um dos passos na maioria dos métodos iterativos para MSA [44].

Anteriormente Gotoh havia apresentado algoritmos para alinhamento entre dois grupos de seqüências [79], mas o mais preciso destes tinha sua complexidade de tempo proporcional a $M \times N$. Com o rápido crescimento no número de seqüências de aminoácidos e nucleotídeos disponíveis, este algoritmo tem se tornado de uso impraticável. Já naquele trabalho Gotoh havia alertado que é possível reduzir os requisitos de processamento pelo uso de vetores de perfis [87], mas a complexidade de tempo continuava $O(M \times N)$, pois não havia forma de avaliar o custo de abertura de *gap* em tempo menor que $O(M \times N)$.

Neste trabalho, foi apresentado um novo algoritmo que permite uma avaliação rápida e precisa dos custos de *gap*. O ponto chave aqui é separar a avaliação dos *gaps* estáticos (pré-existentes) dos *gaps* dinâmicos, que são inseridos no alinhamento. Os *gaps* estáticos passam a ser avaliados apenas uma vez antes do procedimento principal de alinhamento. Desta forma são eliminadas operações redundantes e o tempo de computação do algoritmo deixa de ser diretamente dependente do número de seqüências e passa a depender da distribuição de *gaps*. Baseado nos resultados deste trabalho o método corrente de alinhamento escolhe o algoritmo mais apropriado a utilizar, dentre os quatro desenvolvidos no trabalho anterior (alp, aln, rrp ou rrn) [79].

A.12.1 Algoritmo

Seja A um grupo de seqüências de comprimento I e composto de M seqüências pré-alinhadas. Seja X um conjunto de símbolos. Cada elemento de A, $a_{m,i} \in X$ ($1 \leq m \leq M$, $1 \leq i \leq I$), é um caracter representando um aminoácido, um nucleotídeo ou um *null*, que indica uma remoção. Desta forma X é composto por cinco (para nucleotídeos) ou 21 (para aminoácidos) elementos. Nesta seção quando fala-se em remoção refere-se a uma única posição e quando fala-se em *gap* refere-se a uma seqüência de remoções consecutivas em uma seqüência. Cada remoção estática é representada por $\triangle$. A variável booleana (0 ou 1) $q^A_{m,i} \equiv (a_{m,i} = \triangle)$ indica se $a_{m,i}$ representa uma remoção ou um resíduo. A variável $Q^A_{m,i}$ representa o estado de *gap* da posição, ou seja, o número de *nulls* consecutivos incluindo e imediatamente anterior ao elemento $a_{m,i}$ na linha m. Representamos por $f^A_{x,i}$ o número de ocorrências do elemento $x \in X$ na coluna i de A. O vetor f^A_i, de dimensão 5 ou 21, é chamado vetor de freqüência. O perfil da posição i é representado pelo vetor p^A_i de mesma dimensão que f^A_i:

$$p^A_{x,i} \equiv \sum_{y \in X} d(x,y) * f^A_{y,i},$$

onde $d(x,y)$ denota a dissimilaridade entre x e y.

Seja B um outro grupo de seqüências com mesmo comprimento I e composto de N seqüências pré-alinhadas. Calcula-se a pontuação SP (do inglês, *Sum of Pairs*) entre A e B da seguinte forma.

$$SP(A,B) = \sum_{i=1}^{I} \sum_{m=1}^{M} \sum_{n=1}^{N} (d(a_{m,i}, b_{n,i}) + v * g_{m,n,i}), \tag{6}$$

onde v é o custo de abertura de *gap* e

$$g_{m,n,i} = (1 - q^A_{m,i}) q^A_{n,i} (Q^A_{m,i-1} \geq Q^B_{n,i-1}) + q^A_{m,i} (1 - q^B_{n,i}) (Q^A_{m,i-1} \leq Q^B_{n,i-1}).$$

Observe que com o uso dos vetores de perfil e freqüência podemos reescrever o somatório do primeiro termo na Equação 6 de diversas formas

$$\sum_{m=1}^{M} \sum_{n=1}^{N} d(a_{m,i}, b_{n,i}) = \sum_{n=1}^{N} p^A_{b_{n,i},i} = \sum_{m=1}^{M} p^B_{a_{m,i},i}, \tag{7}$$

assim como

$$\sum_{m=1}^{M} \sum_{n=1}^{N} d(a_{m,i}, b_{n,i}) = \sum_{x \in X} p^A_{x,i} * f^B_{x,i} = \sum_{x \in X} f^A_{x,i} * p^B_{x,i}. \tag{8}$$

Dependendo dos valores de M e N podemos escolher a expressão que possa ser avaliada de forma mais econômica. De forma análoga há também formas eficientes de calcular o segundo termo da Equação 6 através de estruturas que identificam valores semelhantes para $Q_{m,i}$.

No trabalho anterior, Gotoh [79] propôs quatro algoritmos, (A,B,C,D), para obter um alinhamento entre dois grupos de seqüências. Os algoritmos mais simples, (A) e (B), não são capazes de avaliar precisamente o custo de abertura de *gaps*, ao contrário dos algoritmos (C) e (D). O algoritmo (C) segue o procedimento de programação dinâmica padrão [158]. Já o algoritmo (D) adota o paradigma da lista de candidatos [147], que permite otimizações rigorosas em casos que o algoritmo (C) não consegue.

No trabalho é apresentado como adaptar os algoritmos (C) e (D) para fazer uso dos perfis. O autor também mostra que é possível aplicar a estratégia iterativa, proposta por Berger e Munson [32], para realizar o alinhamento de grupos de seqüências.

A.12.2 Conclusão

A maior contribuição do trabalho foi reduzir o tempo para computação de alinhamento de grupos de seqüências sem perda de precisão. O novo método chega a executar $\sim$ 100 vezes mais rápido que o método anterior. Outra contribuição foi uma redução significante nos requisitos de memória. É sugerido que a inclusão de informações baseadas em estrutura seja algo a melhorar o desempenho de alinhamento múltiplo de grupos.

A.13 DbClustal: rapid and reliable global MSAs of protein detected by database searches

Este trabalho [220] apresenta o DbClustal, pacote que produz alinhamentos globais de alta qualidade para seqüências com boas pontuações em resultados de buscas com BLAST [9]. Seu procedimento é automático e o tempo de execução permite seu uso para análise de genomas em larga escala. O DbClustal combina as vantagens dos algoritmos de alinhamento global e local na tradicional abordagem progressiva.

Uma importante aplicação de alinhamentos múltiplos é o alinhamento de conjuntos de seqüências detectadas por busca de homologia em bases de dados. Entretanto, com o crescimento exponencial das bases, os requisitos de tempo têm se tornado o principal fator limitante dos métodos automáticos para este tipo de tarefa. Em particular, a análise comparativa de genomas completos necessita de alto *throughput* no processamento automático das milhares de buscas por homologia.

Existem diversos pacotes para a construção de alinhamento múltiplo local das seqüências com boas pontuações identificadas nas buscas em bases de dados [126, 159, 187, 210]. Entretanto só há alguns pacotes (semi-)automáticos para alinhamento múltiplo global de seqüências detectadas por buscas em bases de dados, que requerem uma certa intervenção manual por um especialista. Gracy e Argos [84] usam um método progressivo com correção manual para classificação automática de seqüências de proteína. Jiang e Jacob no EbEST [113] construíram alinhamentos de ESTs (do inglês *Expressed Sequence Tags*). Baxevanis e Landsman [30] construíram alinhamentos manuais de *motifs* de dobra de histonas e então usaram Clustal W [215] para alinhar o restante das seqüências. Srinivasarao e colegas [203] também usaram Clustal W para gerar alinhamentos múltiplos no banco de dados PIR-ALN, seguido por correção manual.

Trabalhos recentes [37, 40] têm combinado alinhamentos de diversas fontes, incluindo tanto algoritmos locais como globais, com o objetivo de tratar padrões complexos induzidos por proteínas modulares altamente variáveis. Embora produzam alinhamentos precisos para entradas muito complexas, os requisitos computacionais são muito elevados.

A.13.1 O método

O DbClustal foi basicamente construído a partir de modificações no Clustal W 1.81 [215], que passou por um processo de modularização para permitir evoluções mais facilmente. Além disso, agora passa a receber na entrada informações produzidas pelo Ballast (pacote de pós-processamento do BLAST) [179], que gera âncoras entre pares de seqüências. É importante salientar que, da forma que foi implementado, é possível substituir o Ballast por outro pacote desde que gere uma saída no formato do Ballast. Pesos são associados às âncoras e definidos de forma a encorajar a manutenção de *motifs* conservados no alinhamento. Clustal W e DbClustal foram implemantados em `ANSI C` e têm seus fontes disponíveis em `ftp://ftp-igbmc.`

u-strasbg.fr/pub/DbClustal/. Ballast também foi impletado em ANSI C e tem seus fontes disponíveis em ftp://ftp-igbmc.u-strasbg.fr/pub/Ballast/.

Ballast funciona da seguinte forma. Empilha pares de segmentos sem *gaps* extraídos de uma busca do BlastP para construir um perfil de conservação ao longo da seqüência de consulta. São descartados do processamento *hits* com *e-value* maiores que $0,1$. Então usa-se este perfil para predizer segmentos de máximo local (LMS, do inglês *Local Maximum Segments*) na seqüência de consulta. Segmentos de alinhamento (BLAST) que sobrepõem LMSs são chamados pares de segmentos de máximo local (LMSPs, do inglês *Local Maximum Segment Pairs*). A cada LMSP é atribuido uma pontuação dependendo de parâmetros do perfil e da similaridade entre as seqüências do par. Para maiores detalhes, verificar o trabalho de Plewniak e colegas [179]. LMSPs assim definem âncoras entre a seqüência de consulta e as seqüências da base de dados com um peso igual à pontuação do LMSP dividida pela maior pontuação possível na busca corrente.

Clustal W utiliza o algoritmo global de programação dinâmica proposto por Needleman e Wunsch [158] para construir um alinhamento múltiplo. Inicialmente são alinhadas as duas seqüências mais relacionadas e então progressivamente alinha-se grupos cada vez maiores de seqüências até que todas estejam alinhadas. O algoritmo para duas seqüências X e Y de tamanhos N e M, respectivamente, requer uma pontuação para alinhar qualquer dois resíduos X_i e Y_j, assim como penalidades para abertura e extensão de *gaps* no alinhamento. O algoritmo recursivo pode ser sumarizado como:

$$H_{i,j} = max \begin{cases} H_{i-1,j-1} + S_{i,j} \\ max_{1 \leq k < i}\{H_{i-k,j} - (g + hk)\} \\ max_{1 \leq l < j}\{H_{i,j-l} - (g + hl)\} \end{cases},$$

onde $S_{i,j}$ é a pontuação para o alinhamento dos resíduos X_i e Y_j, g e h são as penalidades para abertura e extensão de *gap*, respectivamente, e k e l indicam o tamanho do *gap* aberto para cada seqüência. No Clustal W a pontuação $S_{i,j}$ é simplesmente igual a pontuação $C_{i,j}$ da matriz de comparação de resíduos. O alinhamento de dois grupos de seqüências é uma simples extensão do algoritmo, onde a pontuação para alinhar dois resíduos é substituída pela pontuação para alinhar duas colunas nos respectivos grupos.

O sistema de pontuação no DbClustal foi modificado para incorporar informações sobre conservação local, oriundas do Ballast. Durante o alinhamento progressivo, uma matriz $M \times N$ posição-específica de âncoras é calculada para cada par de seqüências (grupos) a serem alinhadas. Para a coluna i no primeiro grupo e coluna j no segundo grupo, a pontuação $ANCHOR_{i,j}$ é dada por:

$$ANCHOR_{i,j} = max(0, max_{1 \leq k \leq L}(W_k)),$$

para todas as âncoras contendo qualquer par de resíduos nas colunas i, j, onde W_k é o peso definido pelo Ballast e L é o número de âncoras. A pontuação para alinhamento dos resíduos A_i e B_j passa a ser definido por

$$C_{i,j} + ANCHOR_{i,j},$$

onde $C_{i,j}$ é a pontuação para o alinhamento dos resíduos A_i e B_j segundo a matriz de comparação. Quanto às penalizações para *gap*, foi mantido tal como no Clustal W.

A.13.2 Conclusão

Com seu método para manipulação de âncoras, o DbClustal conseguiu avançar na solução de problemas tipicamente encontrados em pacotes para alinhamento múltiplo, tais como alinhamento de seqüências divergentes e seqüências com grandes inserções e extensões. DbClustal apresenta bons resultados mesmo com seqüências de baixa homologia (*e-value* ≥ 0.1) e pelos testes também fica evidenciado o quão rápido ele é, permitindo, assim, seu emprego em cenários que exijam algoritmos automáticos e de alto *throughput* para alinhamento múltiplo, tal como em análise e anotação automática de genomas.

Neste trabalho também pode-se destacar a modularização do Clustal W, que permite a implementação ou modificação de funcionalidades de uma forma mais simples.

A.14 M-COFFEE: combining MSA methods with T-COFFEE

Este trabalho [228] apresenta um meta-método para construção de alinhamento múltiplo de seqüências pela combinação da saída de diversos outros algoritmos em um único alinhamento. O M-COFFEE é capaz realiza combinação dos alinhamentos rapidamente, gastando menos tempo nesta tarefa que a execução de um algoritmo de alinhamento múltiplo. Sua grande vantagem em relação aos outros métodos está em sua precisão. M-COFFEE é parte do pacote do T-COFFEE, que foi escrito em `Perl` e `C` e executa em plataformas baseadas em `UNIX`. Ele é um *freeware* de código aberto sob a lincença `GNU` e disponível em `http://www.tcoffee.org`.

No decorrer da evolução dos algoritmos para MSA, um importante passo foi o desenvolvimento de métodos baseados em consistência, onde o propósito é gerar um alinhamento consistente com um conjunto de alinhamentos de pares. O uso de consistência foi primeiro descrito por Gotoh [78] e Kececioglu [124] e re-formalizado por Vingron e Argos [224] como um procedimento de multiplicação de matrizes de pontos. Posteriormente, consistência foi re-descoberta por Morgenstern e colegas [152] culminando no método DiAlign. Então, em 2000, Notredame e colegas [165] definiram o T-COFFEE, um método que combina as idéias do DiAlign com uma estratégia de alinhamento progressivo. Tal método resultou em uma significante melhoria na precisão dos resultados em MSA. Desde então, função objetivo baseada em consistência tem sido utilizada em vários dos novos pacotes, tais como: POA [133], MAFFT 5 [121], MUSCLE 6 [62], ProbCons [54] e PCMA [177].

A maneira que tem sido adotada para a avaliação dos diversos métodos para MSA é através de avaliações empíricas em alinhamentos de seqüência baseados em estrutura. Há diversas coleções alternativas de *benchmark* disponíveis, dentre elas: BAliBASE [218], PREFAB [63] e HOMSTRAD [149]. É razoável considerar que métodos com melhores performances médias são melhores. Entretanto é importante salientar que tal escolha não é garantia de sucesso, pois o método escolhido pode não ser o mais preciso em algum banco de dados específico. Dentre os principais problemas desta abordagem está o fato de gerar alinhamentos estruturalmente ao invés de evolucionariamente corretos e o fato de que a montagem de MSAs baseados em estrutura é uma tarefa difícil.

Em meta-métodos, uma etapa importante e, no caso de MSA, complicada é o procedimento de combinação das entrada. No M-COFFEE este problema é resolvido simple e elegantemente através de funções objetivas baseadas em consistência que geram consenso entre os elementos da entrada. Dada uma coleção de alinhamentos alternativos, tais funções definem um alinhamento ótimo como sendo aquele de maior nível de consistência com a coleção, em outras palavras, consenso. Esta abordagem foi inicialmente descrita por Bucka-Lassen e colegas [40] e é núcleo do algoritmo do T-COFFEE, onde é definido o conceito de bibliotecas. Ele não alinha seqüências explicitamente, o que faz é compilar bibliotecas baseando-se em alinhamentos produzidos externamente. Durante o processo, as bibliotecas são combinada em um MSA final. Apesar de usar o Clustal W e Lalign, o T-COFFEE permite que qualquer pacote de alinhamento, múltiplo ou de par, seja utilizado. No M-COFFEE, o que é feito é utilizar exatamente esta possibilidade e explorar as alternativas. São analisados 15 algoritmos e é sugerido um subconjunto destes que apresentam resultados melhores que os apresentados até então.

A.14.1 O Método

No estudo, foram selecionados 15 programas para MSA largamente utilizados de um total de 8 diferentes laboratórios. O objetivo foi explorar uma grande variedade de algoritmos para alinhamento de seqüências de proteína.

Clustal W 1.83 [46, 215]: é o programa mais utilizado para MSA. Utiliza uma abordagem progressiva.

T-COFFEE 2.03 [165]: usa função objetivo baseada em consistência [166] otimizada através de um alinhamento progressivo.

ProbCons 1.09 [54]: é também um método baseado em consistência, além de modelos ocultos de Markov. Até então era considerado o método mais preciso, conforme atestado pelo HOMSTRAD [149].

PCMA 2.0 [177]: utiliza função objetivo baseada em consistência para alinhar seqüências de menor homologia e um algoritmo semelhante ao do Clustal W para seqüências de maior homologia.

MUSCLE [62] :

3.52: usa um algoritmo de alinhamento progressivo.

6.0: usa uma função objetivo tal como o ProbCons para refinar o alinhamento do MUSCLE 3.52.

DiAlign2 2.2.1 [150]: através de melhorias no algoritmo do DiAlign [152] realiza alinhamentos múltiplos locais.

DiAlign-T 0.1.3 [207]: nova versão do DiAlign, que insere a função objetivo do DiAlign em um algoritmo de alinhamento progressivo.

MAFFT 5.531 [121] :

FFT-NS1: alinhamento progressivo que usa transformada rápida de Fourier para calcular a árvore guia.

FFT-NS2: Semelhante ao anterior, mas com a diferença de re-calcular a árvore guia depois do primeiro alinhamento e re-alinhar.

FFT-NSI: Semelhante ao FFT-NS1, mas inclui um passo iterativo de refinamento.

F-INSI: incorpora informação sobre alinhamento local de pares.

G-INSI: incorpora informação sobre alinhamento global de pares.

POA 2.0 [85]: usa grafos de ordem parcial para construir os MSAs. É possível optar pelo tipo de algoritmo, dentre eles:

Local: uma versão que realiza alinhamento local.

Global: uma versão que realiza alinhamento global.

A fim de exibir visualmente o nível de similaridade entre os vários métodos, assim como de calcular pesos para os métodos, foi calculada uma árvore. O primeiro passo foi computar uma matriz de distâncias onde cada entrada indica uma medida de diferença média entre dois métodos no conjunto de dados completo do HOMSTRAD. Então aplicou-se o algoritmo UPGMA na matriz.

O M-COFFEE foi implementado a partir do T-COFFEE. Bibliotecas são geradas dos alinhamentos criados pelos diferentes pacotes. Todas as bibliotecas são então dadas como entrada para o T-COFFEE. O peso padrão usado na biblioteca é o percentual de identidade das seqüências pais. No M-COFFEE este esquema de peso muda um pouco. O peso original é multiplicado pelo peso do método (da biblioteca em questão). Foram definidos quatro esquemas, sendo que dois deles utilizam a árvore de métodos descrita acima e os outros dois não.

A primeira tarefa na construção do M-COFFEE foi determinar como os 15 métodos utilizados deveriam ser considerados no consenso de alinhamento. A primeira tentativa foi usar um procedimento guloso a fim de determinar um subconjunto de métodos. Os métodos foram classificados de acordo com sua precisão no HOMSTRAD. Em seguida foram realizados testes, inicialmente utilizando apenas o melhor método (ProbCons), em seguida usando os dois melhores (ProbCons + MUSCLE

6.0), e assim sucessivamente. Observou-se que, a partir do teste com os três melhores métodos, os resultados são mais precisos, o que estabelece a eficiência da combinação. Observou-se também que com a inserção de métodos muito similares há uma degradação na precisão. Observe que desta forma haverá a repetição dos mesmos erros em diversos dos alinhamentos, o que resulta na inserção do erro no consenso. O maior pico na precisão ocorreu quando foi realizado um teste com os seis melhores algoritmos (ProbCons, MUSCLE 6.0, T-COFFEE, MUSCLE 3.52, F-INSI e PCMA).

Em uma etapa posterior, para evitar problemas com algoritmos similares, decidiu-se utilizar apenas um dos algoritmos (o melhor) de cada laboratório. Os oito algoritmos selecionados foram: POA-global, DiAlign-T, Clustal W, PCMA, F-INSI, T-COFFEE, MUSCLE 6.0 e ProbCons. Comparando o resultado do M-COFFEE com cada um dos 8 métodos isoladamente, o M-COFFEE sempre ganha. O M-COFFEE é mais preciso inclusive que o ProbCons, mesmo sem incluir os resultados deste. Esta versão do algoritmo que combina oito métodos foi então selecionada como padrão do M-COFFEE, mas note que o conjunto de alinhadores individuais pode ser facilmente substituido.

A.14.2 Conclusão

Os resultados dos testes mostraram que o M-COFFEE ganha em quase o dobro das vezes do melhor método individual no conjunto completo de dados do HOMSTRAD, PREFAB e BAliBASE. Sempre comparando o M-COFFEE com o melhor método individual no conjunto em questão. Em termos de tempo de CPU, o M-COFFEE é muito similar ao T-COFFEE padrão, sendo inclusive um pouco menor por não requerer a geração de uma biblioteca de pares. É importante salientar que não foi levado em consideração o tempo de computação dos alinhamentos pelos métodos individuais.

Apesar de mostrar ser significativamente mais preciso, o M-COFFEE requer grande tempo para execução se levado em consideração todo o tempo requerido, desde as computações dos alinhamentos por métodos individuais até a combinação dos alinhamentos num alinhamento consenso.

A.15 Multiple alignment of biological sequences with gap flexibility

Este trabalho [145] apresenta uma abordagem heurística para obter alinhamento múltiplo de seqüências biológicas. Faz uso de grafos direcionados acíclicos (DAGs, do inglês, *directed acyclic graph*) para representar alinhamentos. Tais grafos conseguem codificar diversos alinhamentos ótimos com uma estrutura compacta. A entrada para o algoritmo de alinhamento múltiplo, nesta abordagem, é um único DAG representando todas as seqüências e cabe a ele gerar o alinhamento a partir delas, essencialmente associando colunas aos caracteres e, conseqüentemente, posicionando os *gaps*. Os autores sugerem a aplicação desta abordagem para montagem

de fragmentos de DNA.

Na maioria das vezes, alinhamentos múltiplos são construídos a partir de alinhamentos de pares. Um problema conhecido, porém comumente ignorado pelas abordagens para MSA é o fato de que não existe, necessariamente, um único alinhamento ótimo de pares. Podem existir diversos. Outro ponto importante é que alinhamentos próximos do ótimo também podem ser interessantes, uma vez que no processo de seqüenciamento ou posterior manipulação das seqüências podem existir erros, tais como remoção ou inserção de bases nucléicas (ou aminoácidos dependendo do tipo de seqüência), ou ainda a substituição de uma destas bases (aminoácidos) por outra devido a um erro na leitura do fragmento.

Na abordagem apresentada neste trabalho é feita uma tentativa de manter um conjunto de bons alinhamentos entre duas seqüências e deixar a decisão do alinhamento a utilizar (dentre os bons) para passos posteriores, onde se pode tomar uma decisão mais acertada. Os diversos alinhamentos entre duas seqüências são armazenados em um DAG, chamado pelos autores de TLG (do inglês, *trace layout graph*), onde nós representam os caracteres das seqüências e as arestas ligam caracteres consecutivos. A cada aresta é acrescido um rótulo que indica as seqüências a que está relacionada. Como exemplo de sua estrutura e poder de síntese, tome como exemplo o TLG apresentado na Figura 3. Ele representa os diversos alinhamentos entre as seqüências `TTTAGC` e `TTTAAAC`, enumerados nas Tabelas 7, 8 e 9. Referenciaremos tais seqüências no TLG como 1 e 2, respectivamente. Observe que os TLGs não fazem qualquer menção a *gaps* ou *mismatches*.

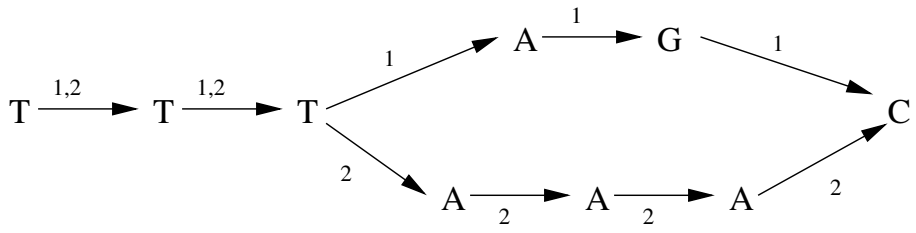

Figura 3: TLG representando alinhamentos entre as seqüências `TTTAGC` e `TTTAAAC`.

T	T	T	A	G	-	C
T	T	T	A	A	A	C

Tabela 7: Primeira possibilidade de alinhamento entre as seqüências `TTTAGC` e `TTTAAAC`.

T	T	T	A	-	G	C
T	T	T	A	A	A	C

Tabela 8: Segunda possibilidade de alinhamento entre as seqüências `TTTAGC` e `TTTAAAC`.

T	T	T	-	A	G	C
T	T	T	A	A	A	C

Tabela 9: Terceira possibilidade de alinhamento entre as seqüências `TTTAGC` e `TTTAAAC`.

A.15.1 TLGs: criação, junção e reparos

A abordagem utilizada por este trabalho deriva da noção de *trace* introduzida por Sankoff e Kruskal [188], posteriormente generalizada para múltiplas seqüências por Kececioglu [123].

Utilizando-se um algoritmo de programação dinâmica padrão para alinhamento global e, tomando-se os alinhamentos ótimos *upmost* e *downmost*, constrói-se o TLG. Os *matches* que ocorreram em ambos os alinhamentos definem nós de junção. Os nós de junção são aqueles onde as seqüências se fundem, acrescido dos nós onde as seqüências começam e terminam.

Uma vez gerados os TLGs para todos os pares de seqüências, precisamos uní-los em um único TLG, uma vez que a entrada para o algoritmo de alinhamento múltiplo é apenas um grande TLG. Para tanto realizamos junções no conjunto de TLGs de pares e então submetemos o resultado para uma etapa de pós-processamento, onde são feitos reparos e otimizações no grafo visando, dentre outras coisas, minimizar o grafo. Observe que nesta etapa de pós-processamento deve ser garantido que o grafo é um DAG, ou seja, ciclos devem ser eliminados.

A junção de duas TLGs é realizada através das partes comuns entre elas. Para isso é necessário navegar em ambas as estruturas, atualizando rótulos quando necessário e possivelmente estabelecendo novos nós de junção ou criando novos nós onde as seqüências diferem. Para detalhes na implementação desta etapa, consultar a tese de doutorado de João Meidanis [144]. Como exemplo de junção, tome os TLGs das Figuras 3 e 4. A Figura 5 apresenta a junção deles.

O passo de pós-processamento (reparos) é responsável por eliminir possíveis ciclos e minimizar o grafo quando possível. Para entender melhor o porque é possível minimizar o grafo, observe que quando temos três seqüências A, B e C, por exemplo, e seus TLGs de pares AB, AC e BC, uma das possibilidades é juntar AB e BC formando uma TLG com as três seqüências. Note, entretanto, que na junção não havia qualquer informação sobre como A e C estão relacionados. É possível que ao juntar AB com BC tenha sido gerado um TLG onde ainda é possível colapsar nós entre A e C.

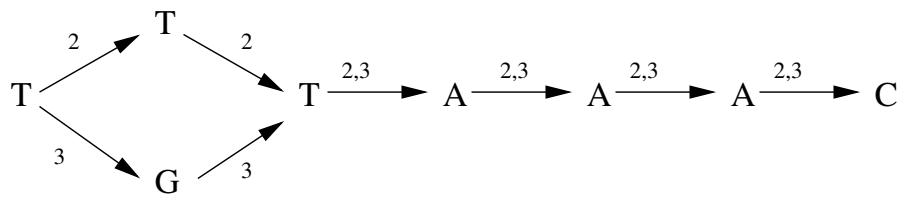

Figura 4: TLG representando alinhamentos entre as seqüências `TTTAAAC` e `TGTAAAC`.

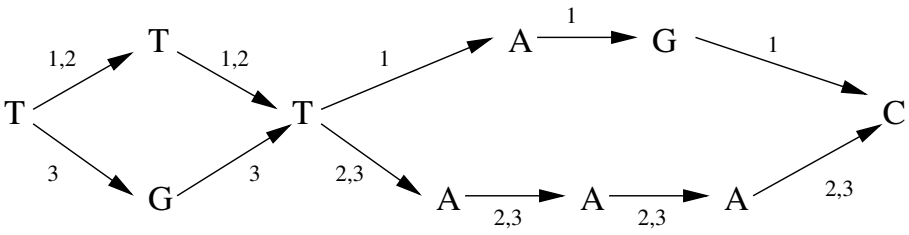

Figura 5: TLG representando a junção dos TLGs apresentados nas Figuras 3 e 4.

Este problema de reduzir o número total de nós tanto quanto possível, mantendo o significado original do TLG, é NP-Difícil [144]. Contudo, existem algoritmos eficientes (não exatos) com esta finalidade que resolvem o problema de forma satisfatória. Meidanis e Setubal apresentam um algoritmo para esta etapa que não garante um grafo mínimo, mas é eficiente e de implementação prática. Assumindo que a entrada seja um TLG, utiliza-se uma abordagem gulosa. O grafo é atravessado em ordem topológica e verificado se qualquer porção do caminho corrente alinha perfeitamente (de forma exata) com qualquer outro caminho percorrido anteriormente. É importante definir um limite inferior para o tamanho destes alinhamentos, caso contrário as alterações na estrutura podem não ser as desejadas. Os autores sugerem cinco caracteres como limite inferior.

Uma vez que se encontra um alinhamento perfeito de tamanho suficiente entre dois caminhos no grafo, é necessário verificar se estes são paralelos antes de realizar qualquer alteração na estrutura. Isso pode ser feito pela verificação das posições relativas dos pontos extremos dos caminhos. Sejam u,v e x,y os nós extremos dos caminhos. Se há um caminho que ligue v e x ou y e u, então os caminhos não são paralelos e, assim, não é possível reduzir a estrutura.

A.15.2 Algoritmo para alinhamento múltiplo

O algoritmo de alinhamento recebe como entrada um TLG, aquele resultante da etapa de pós-processamento. O algoritmo então introduzirá *gaps* no alinhamento quando caminhos ramificados no TLG não possuem o mesmo tamanho. Em outras

palavras, o problema de alinhar TLG consiste em associar número de coluna para cada nó. Uma vez que tenhamos tais números, basta preencher uma matriz inserindo, em cada linha, os caracteres da seqüência correspondente nas colunas apropriadas. Deseja-se que haja espaço suficiente na matriz para os ramos mais longos, mas que não seja alocado mais espaço que o necessário.

Inicialmente calcula-se uma ordenação topológica para o TLG. Escolhe-se uma das fontes e assoca-se o valor 0 a ela. A partir deste momento inicia-se uma sucessão de propagações *forward* e *backward* alternadamente até que não são encontrados novos nós extremos. Mantém-se duas variáveis auxiliares para cada nó, referentes a limites inferior (*lower bound*) e superior (*upper bound*). Atualizam-se os limites inferiores nos passos *forward* e os limites superiores nos passos *backward*. Tais variáveis são utilizadas para propagar valores.

Além desta variáveis ainda há uma fila para cada passo (`fwdqueue` e `backqueue`). É importante observar que estas filas seguem prioridades definidas pela ordenação topológica.

As principais unidades do algoritmo apresentado neste trabalho são exibidas nos Algoritmos 2, 3 e 4. Denota-se por $u \rightarrow v$ que o nó u alcança v considerando-se a orientação das arestas. As funções $GetMin$ e $GetMax$ retornam e removem o menor e o maior elemento da lista, respectivamente. A função $length(u, v)$ indica o número de arestas do menor caminho entre u e v.

Algoritmo 2: Rotina principal do algoritmo MSAGF.

Input: TLG
Inicializa todos os nós com $lb = -\infty$ e $ub = +\infty$
Calcula a ordenação topológica e armazena os índices dos nós
Seleciona um fonte, s, aleatoriamente
$lb(s) = ub(s) = 0$
$fwdqueue = \emptyset$
$backqueue = \emptyset$
$Insere(fwdqueue, s)$
$fwdflag = true$
repeat
  if $fwdflag$ **then**
    $ForwardPass(fwdqueue, backqueue)$
  else
    $BackwardPass(backqueue, fwdqueue)$
  end
  $fwdflag = \mathbf{NOT}(fwdflag)$
until $Empty(fwdqueue)$ ***AND*** $Empty(backqueue)$

Uma vez concluído o processamento, utiliza-se o valor do limite inferior para definir a coluna do nó. A Tabela 10 apresenta o resultado do processamento do algoritmo $MSAGF$ tendo como entrada o TLG da Figura 5.

Algoritmo 3: ForwardPass

```
Input: fwdqueue, backqueue
while NOT(Empty(fwdqueue)) do
    node = GetMin(fwdqueue)
    if node é um terminal then
        if ub(node) = +∞ then
            ub(node) = lb(node)
            Insert(backqueue, node)
        end
    else
        for todo v tal que node → v do
            if lb(node) + length(node, v) > lb(v) then
                lb(v) = lb(node) + length(node, v)
                Insert(fwdqueue, v)
            end
        end
    end
end
```

Algoritmo 4: BackwardPass

```
Input: backqueue, fwdqueue
while NOT(Empty(backqueue)) do
    node = GetMax(backqueue)
    if node é uma fonte then
        if lb(node) = −∞ then
            lb(node) = ub(node)
            Insert(fwdqueue, node)
        end
    else
        for todo u tal que u → node do
            if ub(node) − length(u, node) < ub(u) then
                ub(u) = ub(node) − length(u, node)
                Insert(backqueue, u)
            end
        end
    end
end
```

T	T	T	A	G	-	C
T	T	T	A	A	A	C
T	G	T	A	A	A	C

Tabela 10: Resultado do alinhamento do TLG da Figura 5.

Seja n o número de nós no TLG e m o número de arestas. A complexidade de tempo do algoritmo apresentado neste trabalho é $O(m + n \log n)$. Quanto a espaço, a complexidade fica em $O(n)$.

A.15.3 Conclusão

As principais contribuições deste trabalho são a introdução de uma estrutura de dados (os TLGs) para entrada em algoritmo de alinhamento múltiplo e os algoritmos para a construção do alinhamento múltiplo baseado nelas. A flexibilidade de *gap* é outro ponto forte na solução.

O método parece ser interessante, mas foi muito pouco avaliado e otimizado. Não há qualquer referência de evolução deste a publicação deste trabalho em 1995.

A.16 Alinhamento Múltiplo de Proteínas via Algoritmo Genético Baseado em Tipos Abstratos de Dados

Este trabalho [189] apresenta uma solução para alinhamento múltiplo de proteínas, que faz uso de algoritmos genéticos. Para ser mais preciso, neste trabalho é apresentada uma instanciação de um algoritmo genético baseados em tipos abstratos de dados para MSA, de acordo com as definições apresentadas por Roberta Vieira em sua tese de doutorado [222].

Nesta abordagem, há a definição de conceitos como cromossomos, bases e genes. Um cromossomo representa um indivíduo do algoritmo genético e indica uma possível solução. Uma base é um par composto por um aminoácido (ou um *gap*) e um número natural. Bases são agrupadas em genes, que formam as características dos indivíduos. Em outras palavras, genes são as colunas de um alinhamento.

Além destes, tem-se ainda o conceito de bloco gênico e população. Um bloco gênico é composto por genes consecutivos e é utilizado na definição de algumas das operações. População é um conjunto de cromossomos.

Na montagem de um cromossomo cada seqüência é alocada em uma linha e são inseridos *gaps*, representados pelo símbolo "-", para deixar todas as linhas com o mesmo comprimento. A *gaps* sempre há o natural 0 associado. A aminoácidos são associados números naturais maiores que 0, que indicam a posição do aminoácido na seqüência original (sem *gaps*). Tal número é utilizado para não permitir que a ordem dos aminoácidos em uma seqüência seja alterada.

Como grau de adaptação de um gene foi utilizado um método baseado na pontuação de soma dos pares e, assim, a pontuação de um gene (coluna) é dada pela

soma das pontuações de cada par de aminoácidos adicionado da soma das pontuações dos *gaps*. Por exemplo, uma forma simplista seria atribuir 1 para *match* entre aminoácidos, -1 para *mismatch* e -2 para cada *gap*. Outros exemplos de pontuação para pares de aminoácidos seria utilizar uma das tabelas de séries como PAM ou BLOSUM. Para *gaps* também é possível torná-la mais complexa levando em consideração sua posição ou a adjacência de outros *gaps*.

O grau de adaptação de um cromossomo é dado pela soma dos graus de seus genes. Adaptação média de uma população é dada pela média aritmética dos graus de adaptação de seus cromossomos.

Quanto às operações, são distribuídas em duas categorias: cruzamento e mutação. Cruzamento é uma operação onde dois cromossomos são combinados dando origem a um novo cromossomo. Mutações são operações que alteram um dado cromossomo na tentativa de fazê-lo melhorar quanto ao grau de adaptação.

Na categoria de cruzamento definiu-se apenas uma operação que recebeu o mesmo nome da categoria. Tal operação faz a combinação de forma que o cromossomo resultante seja ainda válido, ou seja, composto por seqüências que caso removidos os *gaps* sejam iguais às seqüências da entrada. O cromossomo resultante só sobreviverá para a próxima geração caso possua um grau de adaptação maior ou igual a adaptação média da população atual.

Na categoria de mutação, há as seguintes operações: inserção, supressão e mutação. Na inserção, uma coluna de *gaps* é inserida no cromossomo. Na supressão uma coluna de *gaps* é removida do cromossomo. É importante salientar que apesar de uma coluna de *gaps* não ter um significado biológico, definiu-se a operação de inserção com o objetivo de possibilitar uma maior variabilidade nos cromossomos. No resultado final do algoritmo deve-se remover todas as colunas de *gaps*. A mutação promove a mudança de posição de *gaps* em uma dada linha do cromossomo.

A população inicial é gerada aleatoriamente. Para cada cromossomo, adicionam-se q_{gaps} *gaps* na maior das seqüências da entrada em posições aleatórias. O objetivo disto é permitir o deslocamento das bases desta maior seqüência durante as aplicações do operador de mutação. Seja s_m a maior seqüência da entrada. Definiu-se que $q_{gaps} = length(s_m)/25$. Depois disso adicionam-se *gaps*, também em posições aleatórias, no restante das seqüências até que todas elas possuam o mesmo tamanho. Convencionou-se que o tamanho da população inicial é de 100 cromossomos.

O processo evolutivo começa com a definição do critério de preservação sobre a população atual. O critério utilizado foi um ponto de corte quanto ao grau de adaptação. Selecionam-se os cromossomos cujo grau seja maior ou igual ao grau de adaptação médio da população atual. Tal subconjunto será utilizado para os cruzamentos. Para as mutações são selecionados todos aqueles cromossomos abaixo do ponto de corte. Os cromossomos que melhorarem o grau com a mutação, vão para a próxima geração. A população, que começa com tamanho 100, pode variar de tamanho de uma geração para outra, mas cresce no máximo até 200. Tal decisão deveu-se a restrições de *hardware*. A iteração é interrompida quando há uma convergência nos

resultados.

Avaliou-se a solução com BAliBASE [217], que, além de permitir uma comparação simples de novas soluções com as existentes, provê diversos conjuntos de entrada divididos por tamanho e grau de similaridade entre as seqüências.

Diversos testes foram realizados a fim de determinar o tamanho máximo da população. Decidiu-se por 200 pelo fato de ter apresentado uma boa relação entre tempo de execução e qualidade dos resultados. Outro fator importante nesta decisão foi a grande limitação de memória no *hardware* disponível para os testes.

Testes também foram realizados para comparar as diversas matrizes de substituição: BLOSUM62, Dayhoff, PAM40, PAM80, PAM120 e PAM250. Neste caso, o melhor resultado apresentado foi quando utilizou-se a matriz Dayhoff. Esta obteve uma pontuação no BAliBASE mais elevada para as seqüências utilizadas nos testes (conjunto `1aab`).

É importante salientar que este resultado no máximo indica que entradas da mesma família (conjuntos de seqüências semelhantes ao avaliado) devem ter resultados minimamente satisfaztórias. Para outras famílias, porém, deve-se verificar qual a melhor matriz a se utilizar.

Outros testes foram feitos para avaliar se deveria-se utilizar reprodução sexuada ou assexuada, ou seja, se os pais utilizados em cruzamentos deveriam ou não ser divididos em duas partições (machos e fêmeas).

Os testes apontaram que o melhor era fazer uma reprodução sexuada, dividindo os pais em dois conjuntos sem intersecção e permitir apenas o cruzamento entre cromossomos de conjuntos distintos. Isso porque reduz o número de combinações, o que afeta diretamente no tempo de execução, e por promover uma maior diversidade genética da população.

Uma vez especificados os parâmetros para o algoritmo genético, foram feitos os alinhamentos múltiplos de quatro outros conjuntos de entradas semelhantes extraídos da mesma família que `1aab`, foram eles: `1fj1A`, `1hpi`, `1csy` e `1tgxA`.

Calculadas as pontuações BAliBASE para estes alinhamentos, pôde-se comparar seu desempenho com uma série de soluções existentes: PRRP, Clustal X, SAGA, DiAlign, SBpima, MLpima, MultiAlign, Pileup, MULTAL e HMMT.

A.16.1 Conclusão

Os resultados indicam que a solução tem potencial, mas precisa ainda de uma série de evoluções e avaliações. Comparando diretamente as pontuações BAliBASE, com as outras soluções, a solução vence apenas do HMMT e do SAGA (de forma pouco significativa, neste caso) para `1aab`, HMMT para `1fjlA`, HMMT e MULTAL (muito pouco significativo) para `1tgxA`.

Foram apenas 5 "vitórias" em 50 comparações. Fora que venceu apenas uma vez do SAGA e de forma pouco significativa. É importante salientar que o SAGA é a maior referência no que se refere a solução por algoritmos genéticos para MSA. Para

a solução começar a ganhar credibilidade precisará apresentar resultados significativamente melhores que o SAGA.

São necessários mais testes da solução. Por exemplo, o BAliBASE é composto por 142 alinhamentos de referência divididos em diversas categorias. O objetivo de cada categoria é avaliar a solução em uma determinada situação. No caso dos testes realizados, até então, foram alinhados apenas 5 dos 142 conjuntos de que dispõe o BAliBASE e, além disso, todos os conjuntos de entrada pertenciam a mesma categoria. Para um teste mais preciso é necessário avaliar a solução nestas diversas soluções, pois é comum que uma dada solução seja muito boa em algumas categorias e apresente maus resultados para outras.

A.17 A Graph-Based Genetic Algorithm for the Multiple Sequence Alignment Problem

Este trabalho [138] apresenta uma solução para alinhamento múltiplo, que faz uso de algoritmos genéticos e descreve uma nova forma de representar um alinhamento através de grafos orientados multidimensionais. Desta forma consegue reduzir dramaticamente a complexidade de armazenamento.

A partir das n seqüências de entrada, constrói-se uma matriz n-dimensional, na qual cada dimensão corresponde a uma seqüência. Indivíduos no GA representam um caminho na matriz, que é assim um grafo orientado multimensional. Um caminho válido sempre inicia na célula correspondente ao primeiro aminoácido de todas as seqüências e acaba na célula correspondente ao último aminoácido de todas as seqüências.

Para representar o grafo usa-se um cromossomo de tamanho variável. Este cromossomo é composto por m genes, cujo número será no mínimo eqüivalente ao tamanho da maior das seqüências e no máximo a soma dos tamanhos da seqüências decrescido de um. Cada gene é composto de dígitos binários, que indicam a direção da próxima célula e que desta forma indica como os aminoácidos, associados as células de origem e destino, são alinhados. Por exemplo, para o alinhamento de duas seqüências, um gene pode receber três valores: 01, 10, 11. Cada um deles associado com um deslocamento: célula a esquerda, célula abaixo ou célula na diagonal. O número de dígitos será diretamente proporcional ao número de seqüências.

Para avaliar uma solução candidata, decodifica-se o cromossomo em um MSA. Em seguida calcula-se a pontuação SP para o alinhamento e normaliza-se o valor para poder comparar com outros alinhamentos.

Três são os operadores genéticos nesta solução, são eles: cruzamento, mutação aleatória e mutação dinâmica. Todos eles foram projetados de forma a preservar a integridade e validade do caminho (solução).

A operação de cruzamento recebe como entrada dois cromossomos e seleciona um gene aleatoriamente. A partir deste momento, passa a combinar os cromossomos de entrada, copiando a parte inicial do primeiro e a parte final do segundo. Depois

disso, muito provavelmente, ainda será necessário completar o caminho do cromossomo resultante. Para isso é utilizado o mesmo procedimento aleatório que gerou a população inicial.

A operação de mutação aleatória tem como objetivo aumentar a diversidade genética da população. Ela recebe como entrada um cromossomo e escolhe aleatoriamente dois genes e uma dimensão. A seguir inverte os *bits* correspondentes àquela dimensão e então realiza os ajustes necessários para manter a validade do caminho. Tal ajuste é necessário caso os *bits* não sejam complementares.

O operador de mutação dinâmica também tem o objetivo de aumentar a diversidade genética da população. Ele recebe como entrada um cromossomo e escolhe aleatoriamente um segmento do alinhamento (colunas consecutivas). O tamanho deste segmento é fixado por um parâmetro do algoritmo. A seguir é realizado um alinhamento dinâmico progressivo nas colunas correspondentes ao segmento selecionado.

A.17.1 Conclusão

A solução apresentada resulta em uma grande redução na complexidade de memória quando comparada com a solução exata obtida pela extensão direta do algoritmo de Needleman e Wunsch [158]. A solução proposta tem complexidade $O(N \times L)$ enquanto Needleman e Wunsch tem $O(L^N)$, para N seqüências e L tamanho médio das seqüências.

As avaliações realizadas ficaram restritas a uma comparação de desempenho com o Clustal utilizando o BAliBASE. Os resultados foram desfavoráveis, mas ainda mais crítico que isso é ausência de comparação com a solução MSA de referência usando GA, o SAGA.