UNIVERSIDADE ESTADUAL PAULISTA
"Júlio de Mesquita Filho"
Pós-Graduação em Ciência da Computação

Juliano Farias da Nóbrega

Técnicas de otimização em alinhamentos múltiplos de sequência via Cadeias de Markov

São José do Rio Preto
2016

Juliano Farias da Nóbrega

Técnicas de otimização em alinhamentos múltiplos de sequência via Cadeias de Markov

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Orientador: Prof. Dr. Geraldo Francisco Donegá Zafalon

São José do Rio Preto
2016

Nóbrega, Juliano Farias da.
Técnicas de otimização em alinhamentos múltiplos de sequência via Cadeias de Markov / Juliano Farias da Nóbrega. -- São José do Rio Preto, 2016
114 f. : il., tabs.

Orientador: Geraldo Francisco Donegá Zafalon
Dissertação (mestrado) – Universidade Estadual Paulista “Júlio de Mesquita Filho”, Instituto de Biociências, Letras e Ciências Exatas

1. Bioinformática. 2. Markov, Processos de. 3. Alinhamento de sequências. I. Zafalon, Geraldo Francisco Donegá. II. Universidade Estadual Paulista "Júlio de Mesquita Filho". Instituto de Biociências, Letras e Ciências Exatas. III. Título.
CDU – 574:518.72

Ficha catalográfica elaborada pela Biblioteca do IBILCE
UNESP - Câmpus de São José do Rio Preto

Juliano Farias da Nóbrega

Técnicas de otimização em alinhamentos múltiplos de sequência via Cadeias de Markov

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

COMISSÃO EXAMINADORA

Prof. Dr. Geraldo Francisco Donegá Zafalon (Orientador)
UNESP - São José do Rio Preto

Prof. Dr. Angelo Pássaro
IEAv/CTA - São José dos Campos

Prof. Dr. Adriano Mauro Cansian
UNESP - São José do Rio Preto

São José do Rio Preto
29 de fevereiro de 2016

Ao prof. José Márcio Machado (1957-2014)

Agradecimentos

Agradeço primeiramente a Deus, pela possibilidade de crescimento espiritual, pessoal e profissional que me permitiram chegar até aqui.

Ao Prof. Dr. Geraldo Zafalon, pelas orientações ao longo dessa jornada, pela amizade sincera e pelo companheirismo.

Ao Prof. Dr. José Márcio Machado, pelo seu enorme coração e exemplo de profissional, e que hoje está junto ao Pai, nos acompanhando em espírito.

Aos meus pais, Trajano e Marion, pela minha educação e presença desde os meus primeiros dias.

Aos meus irmãos, Luciano, Adriana, Daniela e Rafael, pela cumplicidade e pelos ótimos momentos.

Agradeço pela minha esposa Marilanda, pelo seu amor e companheirismo em todos os momentos de dificuldade e alegria.

Ao pequeno e adorável Joaquim, motivo da maior alegria.

Aos familiares e amigos que sempre estiveram presentes.

Ao Anderson Rici Amorim, pelas incontáveis ajudas durante o desenvolvimento do trabalho.

Aos amigos (Edson) Chang Hsun Ming e Rafael (Latino) Henrique Moretti e Márcio Ferro, pelo apoio e amizade dentro e fora da Universidade.

Aos funcionários da Pós-graduação: Rosemar, Alex, Mauro e Silvia, do laboratório dos Estudos Genômicos e tantos outros que participaram direta ou indiretamente desse período acadêmico.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

BLAST Basic Local Alignment Search Tool

Blosum Blocks of Amino Acid Substitution Matrix

bp base pairs

DNA Desoxirribonucleic Acid - Ácido Desoxirribonucléico

HMM Hidden Markov Model

MMEO Modelos de Markov de Estados Ocultos

MSA Multiple Sequence Alignment

MUMMALS Multiple Sequence Alignment Improved by Using Hidden Markov Models with Local Structural Information

MUSCLE Multiple Sequence Comparison by Log-Expectation

NJ Neighbor Joining

PAM Percent Accepted Mutation

RNA Ribonucleic Acid - Ácido Ribonucléico

SP Sum-of-pairs

SVM Support Vector Machine

TC Total Column

UPGMA Unweighted Pair Group Method with Arithmetic Mean

Resumo

Recentemente, a bioinformática tornou-se um recurso imprescindível para a análise e interpretação da grande quantidade de informação biológica gerada pela biologia molecular e pelos sequenciadores de última geração. O processo de comparação dessas biossequências é o ponto de partida para o estudo da evolução e diferenciação dos organismos vivos, além de ser uma das tarefas mais importantes na biologia computacional. Neste trabalho apresenta-se uma abordagem baseada na heurística de Cadeias de Markov para otimização de um algoritmo de alinhamento múltiplo de sequências biológicas, proporcionando resultados com mais qualidade e sem o comprometimento do desempenho da ferramenta MUSCLE, escolhida para dar suporte ao trabalho. As cadeias de Markov foram escolhidas como técnica de otimização devido sua eficiente aplicabilidade em diversos problemas, sobretudo na biologia computacional, pois sua metodologia probabilística torna a aplicação computacionalmente viável, contornando os problemas NP-difícil e apresentando resultados significamente precisos.

Palavras-chave: Bioinformática. Alinhamento Múltiplo de Sequências. Modelos de Makov.

Abstract

Recently, bioinformatics has become an indispensable tool for analyzing and interpreting large amounts of information biological generated by molecular biology and the next-generation sequencers. The comparison process these sequences is the starting point for the study of evolution and differentiation of living organisms as well as being one of the most important tasks in computational biology. This work presents an approach based on Markov chains heuristics for optimization of a multiple alignment algorithm of biological sequences, provides improved quality results and without compromising the performance of MUSCLE tool chosen to support the work.. Markov chains were chosen as optimization technique due to its efficient applicability in various other problems, especially in computational biology, as its probabilistic methodology makes applying computationally feasible, bypassing the NP-hard problems and stating significantly accurate results.

Keywords: Bionformatics. Multiple Sequence Alignment. Markov Models.

Capítulo 1

Introdução

1.1 Bioinformática: considerações iniciais

Recentemente, a ciência vêm rompendo diversos paradigmas, sobretudo no que se diz respeito ao grande avanço da computação e da biologia. Essas duas áreas, antes distintas, hoje estão combinadas em uma nova ciência, denominada bioinformática, que passa a solucionar problemas antes inimagináveis, e, lançando mão da matemática, física, química e estatística. As soluções tornam-se possíveis e de grande importância para a melhoria da condição da saúde humana, abrindo inúmeras possibilidades também na medicina, viabilizando o diagnóstico e tratamento de doenças associadas as mudanças genéticas, entre outras (Wu et al., 2012).

É importante destacar que o ano de 2000 foi marcado pelo fim do projeto Genoma, responsável por mapear completamente o código genético humano. Este, composto por 3 bilhões de elementos, representa a sequência do DNA, e contém toda a informação necessária para constituir e manter o ser humano

vivo. Esse processo, que inicialmente custou centenas de milhões de dólares, atualmente é possível ser realizado por alguns milhares de dólares, e em algumas semanas por meio das novas gerações de sequenciadores biológicos (Filho, 2009).

Interpretar e organizar essa enorme quantidade de informações passou a ser a principal tarefa da bioinformática, que vêm desenvolvendo diversas técnicas para a compreensão das principais sequências biológicas: o DNA e o RNA (ácidos desoxirribonucléico e ribonucléico, respectivamente), compostos pelos nucleotídeos, e as proteínas, compostas pelos aminoácidos (Alberts et al., 2010).

Dentre essas técnicas, destacam-se o alinhamento de sequências, que é composto por algoritmos responsáveis por ler e comparar trechos das sequências, buscando identificar regiões semelhantes entre si. Essas regiões, contendo padrões, podem oferecer informações importantes, tais como o descobrimento de um gene, regiões codificadoras ou não-codificadoras, ou mesmo mutações em determinados trechos importantes.

Computacionalmente, o processo de alinhamento de sequências é extremamente custoso ao se analisar várias biossequências, sendo necessário inclusive, o uso de computação de alto desempenho para resolver tais tarefas em alguns casos. Ainda assim, dada a alta complexidade, o enorme volume de dados e o extenso comprimento das sequências, algoritmos mais elaborados, contendo diversos refinamentos e técnicas de otimização passam a ser necessários na execução de tarefas de bioinformática.

Assim, esses métodos denominados heurísticos abordaram os problemas de bioinformática de forma estocástica, apresentando resultados com um

certo grau de precisão, que podem, dessa forma, ser ajustados conforme a técnica empregada durante o processo de otimização do algoritmo de alinhamento.

1.2 Objetivos do trabalho

Este trabalho tem por objetivo apresentar o processo de otimização de um algoritmo de alinhamento múltiplo de sequências utilizado na ferramenta MUSCLE *(Multiple Sequence Comparison by Log-Expectation)*. Para isso, implementou-se um Modelo de Markov Oculto, utilizando o acoplamento dos algoritmos Forward-Backward, Viterbi e Baum-Welch em uma das fases da ferramenta MUSCLE, que realiza a contagem de k-mers. Do ponto de vista da bioinformática, as técnicas capazes de otimizar os processos de análises de biossequências são fundamentais, visto a grande quantidade de dados a serem pesquisados, e sua contribuição junto a pesquisas na área da saúde.

A técnica de otimização implementada baseada em Cadeias de Markov teve por objetivo melhorar a qualidade biológica dos resultados, disponibilizando dessa forma, resultados mais relevantes. Além disso, quando possível, foram priorizadas também melhorias no tempo de execução da ferramenta, com otimizações de código.

1.3 Motivação

O processo de otimização de ferramentas computacionais capazes de realizar a comparação de várias biossequências é um dos grandes desafios da bioinfor-

mática. Os estudos envolvidos nessas áreas abrangem soluções baseadas em heurísticas diversas, tais como os modelos de Markov aplicados a ferramentas de alinhamento múltiplo bastante difundidas. A partir da melhoria dessas ferramentas, além da própria bioinformática, pesquisas na área de saúde são beneficiadas, pois a compreensão de certos processos biológicos são essenciais, por exemplo, para o desenvolvimento de novos fármacos. Além disso, é importante o acoplamento de novas heurísticas, de modo a refinar os resultados obtidos, principalmente almejando uma melhor significância biológica.

1.4 Organização do trabalho

Este trabalho está organizado da seguinte forma: no capítulo 1 é apresentada uma breve introdução sobre a bioinformática e algumas de suas características. O capítulo 2 é destinado à fundamentação teórica e levantamento bibliográfico necessário para a compreensão dos problemas em biologia e computação, assim como as técnicas matemáticas empregadas. No capítulo 3 é apresentado o processo de desenvolvimento do projeto, e a implantação dos Modelos de Markov para a melhoria dos algoritmos de alinhamento múltiplo em bioinformática. O capítulo 4 é destinado aos resultados obtidos por meio das execuções dos algoritmos. A conclusão do trabalho é por fim, apresentada no capítulo 5.

Capítulo 2

Fundamentação Teórica

2.1 Contexto Biológico

Nessa seção serão apresentados os conceitos básicos sobre a célula e seu funcionamento, assim como as principais macromoléculas biológicas e suas interações, dada a sua importância na bioinformática.

2.1.1 A Célula

Para a compreensão de qualquer forma de vida, é necessário que se faça um estudo minucioso de sua unidade fundamental: a célula. Esta possui em si os processos metabólicos responsáveis pela pela manutenção do organismo e sua reprodução, assim como armazena em seu núcleo o material genético, composto pela sequência do DNA (*Ácido Desoxirribonucléico*) (Alberts et al., 2010).

A grande variedade de organismos vivos presentes hoje na Terra decorre do lento e indiscutível processo evolutivo, e aos poucos estes foram sendo

classificados nos seus respectivos domínios, de acordo com características que foram mantidas, ou eliminadas. Essas características, resultantes de alterações aleatórias no DNA, que possibilitaram a diferenciação entre os organismos, denominada mutação, são responsáveis pela adaptação e propagação do organismo no meio em que ele vive, e a identificação dessas mutações é fundamental para o estudo da filogenia, tendo em vista que todos os organismos são provenientes de um ancestral comum.

As células podem ser classificadas inicialmente em dois grupos: as eucariontes, que possuem núcleo bem definido, onde fica armazenado o material genético, e as procariontes, que não possuem envoltório nuclear, e o material genético fica disperso no citoplasma (Alberts et al., 2010).

Além do núcleo, contendo o DNA, existem diversas organelas responsáveis pelo metabolismo celular, e pela síntese das macromoléculas RNA e as proteínas, nos eucariontes. Na figura 2.1 estão representados as componentes de uma célula eucarionte.

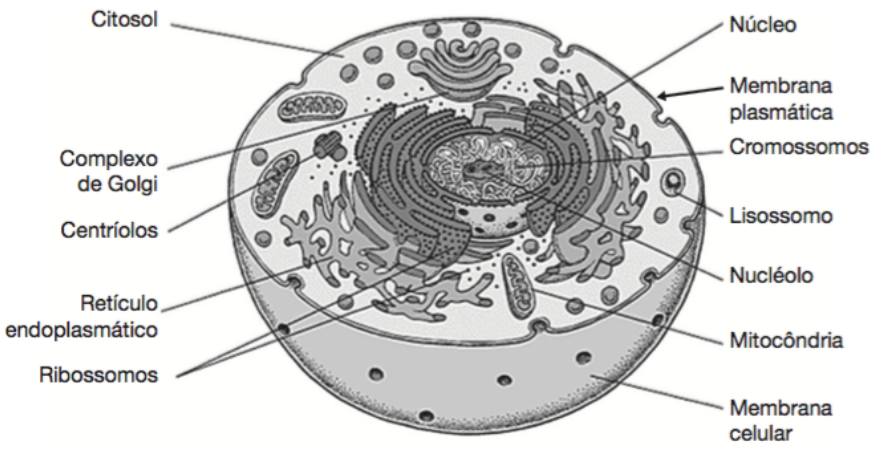

Figura 2.1: Célula eucariótica
Fonte: (Alberts et al., 2010)

É importante destacar ainda que a propriedade fundamental de uma célula está na capacidade de crescer e replicar-se, gerando células descendentes contendo cópias do seu material genético. Isso é resultado de uma série de processos metabólicos desencadeados dentro da célula, e parte destes fenômenos químicos estão relacionados à fabricação de outras moléculas simples como os açúcares e os aminoácidos, assim como outras mais elaboradas, denominadas macromoléculas biológicas, descritas a seguir.

2.1.2 As Macromoléculas Biológicas

Os seres vivos, por mais que possuam grandes diferenças entre si, são formados essencialmente pelos mesmos tipos de moléculas: proteínas, lipídeos, ácidos nucléicos e carboidratos, ou seja, moléculas nas quais baseia-se a vida, como se conhece.

As macromoléculas são denominadas polímeros, e são formadas pelo encadeamento de várias moléculas simples e semelhantes (monômeros) com comprimento variável, e dentre as moléculas mais importantes no estudo da bioinformática estão os ácidos nucléicos e as proteínas. Uma macromolécula de ácido nucléico pode ser descrita como um alfabeto de comprimento quatro (no caso do DNA e RNA), ou seja, é composta por quatro bases nitrogenadas, ou de comprimento vinte para as proteínas, que são formadas por uma combinação de 20 possíveis aminoácidos (Alberts et al., 2010).

As macromoléculas de DNA (*desoxyribonucleic acid* - ácido desoxirribonucléico) e RNA (*ribonucleic acid* - ácido ribonucléico) são compostos por quatro elementos, denominados nucleotídeos ligados por uma ponte de hidro-

gênio: Adenina (A), Timina (T), Citosina (C) e Guanina (G), sendo que no caso do RNA, a Timina (T) é substituída pela Uracila (U) (Alberts et al., 2010).

Na figura 2.2 ilustra-se a ligação dos nucleotídeos, no caso do DNA e do RNA.

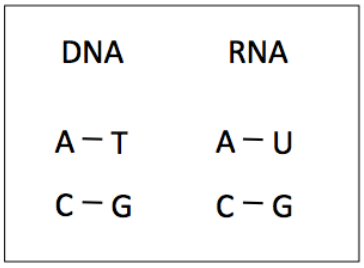

Figura 2.2: Ligação entre as moléculas de DNA e RNA

O DNA armazena todas as características genotípicas do organismo, ou seja, as responsáveis pelas informações contidas nos genes, ou trechos das cadeias de DNA. Essas informações são transcritas para o RNA, cuja sequência de nucleotídeos contém o código para a ordenação específica do aminoácido. Assim, o processo de *tradução* do RNA dá origem a uma nova molécula de proteína. Esse processo completo é denominado "Dogma Central da Biologia Molecular", conforme ilustrado na figura 2.3.

A macromolécula de DNA é composta por quatro bases nitrogenadas (devido a presença de nitrogênio em sua composição) as quais fazem a conexão entre as duas hélices (ou fitas) que compõe a forma básica do DNA, de forma que a base nitrogenada Adenina (A) ligue-se apenas com a Timina (T) ou vice-versa, assim como a Citosina (C) liga-se apenas com a Guanina (G) ou vice-versa. Na figura 2.4 está ilustrada a estrutura básica da dupla hélice do

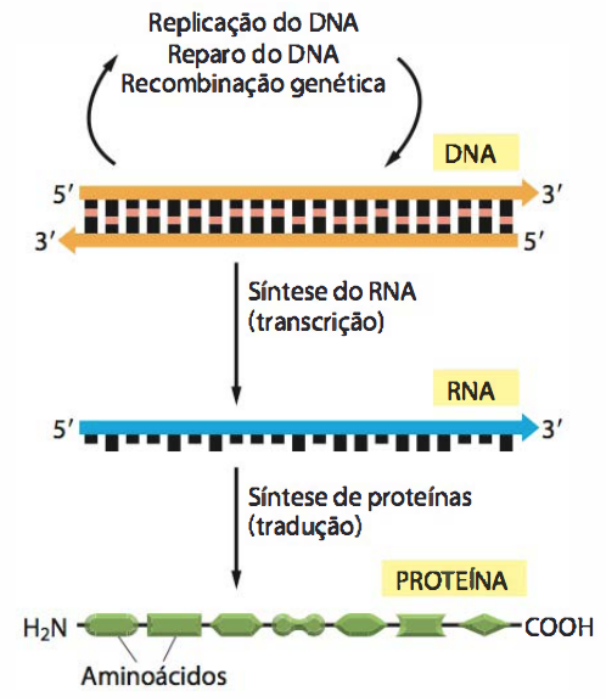

Figura 2.3: Dogma Central da Biologia Molecular
Fonte: (Alberts et al., 2010)

DNA, e suas fitas sendo utilizadas como molde para o RNA.

Por meio do processo de transcrição do DNA, o RNA é construído com a diferença de que a base nitrogenada Uracila (U) substitui a base Timina (T), e sua estrutura é constituída por uma fita simples, pois serve como molde complementar da fita do DNA (Pevzner and Shamir, 2011).Essa macromolécula é deslocada do núcleo celular para o citoplasma, com o objetivo de traduzir novas proteínas.

Normalmente, a molécula de DNA é definida pela sua sequência de bases em uma das fitas através da direção $5' \rightarrow 3'$, e seu comprimento é geralmente definido através da quantidade de pares de bases (*bp* ou *base pairs* - *bp*). Cada célula presente em um organismo possui uma cópia de todo o genoma,

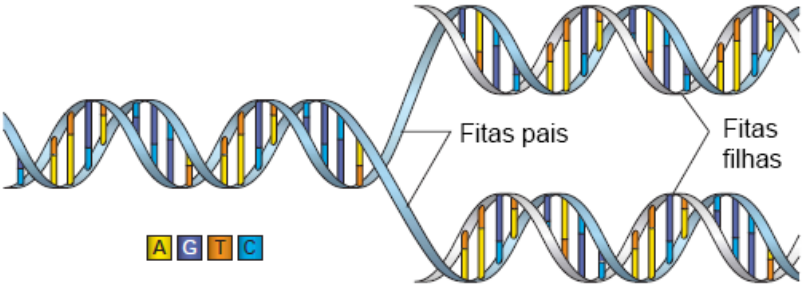

Figura 2.4: Dupla hélice do DNA
Fonte: (Alberts et al., 2010) (Adaptado)

exceto as células germinativas, e nos seres humanos, as células do sistema imune e as hemácias (Alberts et al., 2010).

Os genes são os trechos da molécula de DNA responsáveis pela codificação de uma determinada proteína. A célula reconhece onde um gene começa e termina, e com o auxílio de uma enzima chamada *polimerase*, é realizada uma cópia do gene em uma molécula de RNA, onde posteriormente, através do ribossomo, uma organela responsável pela síntese protéica, a proteína é finalmente, traduzida.

A RNA polimerase, além de realizar precisamente a cópia da sequência de nucleotídeos do DNA, possui a capacidade de identificar diversos sinais genéticos presentes no cromossomo, tais como os responsáveis pelo início e término da síntese do RNA. Esses sinais, denominados *promotores* determinam em que porção a RNA polimerase será ligada, promovendo dessa forma, a expressão e regulação do gene. Um exemplo dessa ligação pode ser encontrada em organismos procarióticos, em que os *promotores* possuem uma sequência básica (com algumas pequenas variações) TATAATG posicionada próxima à extremidade final do RNA mensageiro (RNAm). Essa sequência,

rica em AT indica alta possibilidade de separação das cadeias de DNA para inserção da RNA polimerase.

As proteínas resultantes da tradução do RNA possuem funções específicas dentro do organismo. Podem estar relacionadas à atividades hormonais (como a insulina), enzimáticas (como por exemplo, a pepsina, relacionadas ao sistema digestivo), sistemas imunológicos (imunoglobina) e até mesmo relacionadas com atividades estruturais (como a queratina e o colágeno) (Alberts et al., 2010).

A combinação de 3 nucleotídeos encadeados sequencialmente dá origem a um códon, ou aminoácido, que por sua vez dá origem a uma proteína. Na tabela 2.1 apresentam-se as respectivas posições dos nucleotídeos e o aminoácido gerado pela disposição entre eles.

Tabela 2.1: Tabela de Códon de Aminoácidos

Primeira posição	Segunda posição				Terceira posição
	G	A	C	U	
G	Gly	Glu	Ala	Val	G
	Gly	Glu	Ala	Val	A
	Gly	Asp	Ala	Val	C
	Gly	Asp	Ala	Val	U
A	Arg	Lys	Thr	Met	G
	Arg	Lys	Thr	Ile	A
	Ser	Asn	Thr	Ile	C
	Ser	Asa	Thr	Ile	U
C	Arg	Gln	Pro	Leu	G
	Arg	Gln	Pro	Leu	A
	Arg	His	Pro	Leu	C
	Arg	His	Pro	Leu	U
U	Trp	STOP	Ser	Leu	G
	STOP	STOP	Ser	Leu	A
	Cys	Tyr	Ser	Phe	C
	Cys	Tyr	Ser	Lhe	U

Uma molécula de proteína é formada por unidades conhecidas como aminoácidos. Essas unidades ligam-se linearmente, resultando em uma cadeia conhecida como polipeptídeo. Um aminoácido é composto por um carbono central (C_0), um hidrogênio (H), um grupo amino (H_2N), um grupo carboxil ($COOH$) e uma cadeia lateral (R) que distingue cada um dos 20 tipos aminoácidos diferentes na natureza. Esses aminoácidos estão conectados por uma ligação peptídica, formadas pela junção do grupo carboxil do primeiro aminoácido com o grupo amino do segundo, ou seja, liberando uma molécula de H_2O, e a cadeia resultante é composta pelos resíduos dos aminoácidos.

A sequência linear da proteína, composta pelos aminoácidos forma a estrutura primária, e essas moléculas em seguida são dobradas, e depois empacotadas, até o quarto nível, formando estrutura tridimensionais que estão diretamente relacionadas com a função bioquímica das proteínas. Na figura 2.5 ilustram-se as quatro possíveis conformações de uma proteína.

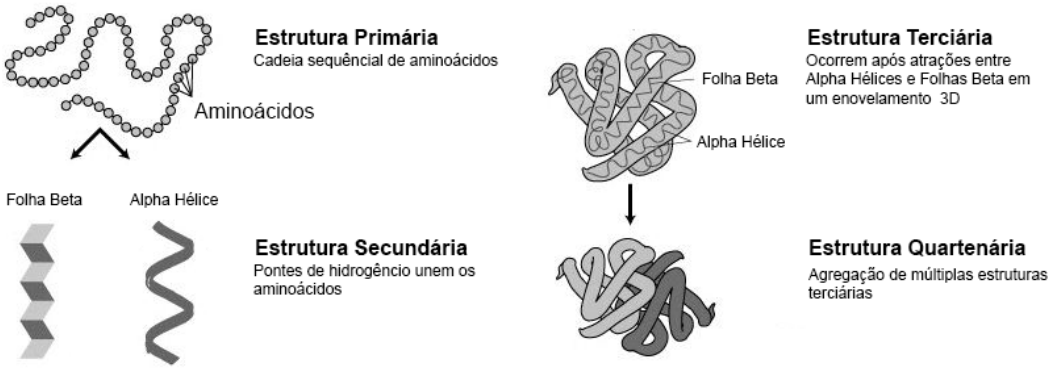

Figura 2.5: Estruturas da proteína
Fonte: (Alberts et al., 2010)

As proteínas são responsáveis por fornecer a estrutura e executar a maioria das tarefas celulares. Compõem aproximadamente 20% do peso de uma

célula, e seu comprimento varia de 100 a 5.000 aminoácidos, ou seja, a quantidade de possíveis combinações para proteínas é imensa, considerando que uma proteína possui, em média aproximadamente 800 aminoácidos, existem 20^{800} possíveis sequências diferentes de aminoácidos, por exemplo.

Cada aminoácido é formado por uma sequência de três nucleotídeos, denominadas *códons* ou *trincas*. Esses nucleotídeos, provenientes do RNA são convertidas nos aminoácidos através do processo de tradução, que gera a proteína em si, conforme ilustrado na figura 2.3.

A presença de 20 tipos diferentes de aminoácidos permite a criação de múltiplas formas irregulares e que determinam a ligação da proteína com várias outras moléculas que possuem uma forma complementar a ela. Na tabela 2.2 estão relacionais os 20 principais aminoácidos.

Existem ainda três conjuntos de aminoácidos, que são combinações de outros aminoácidos, porém com menor relevância: a Asparagina (ou Ácido Aspartâmico), a Glutamina (ou Ácido Glutâmico) e um conjunto que compõe qualquer outra combinação de aminoácidos.

2.1.3 O Gene e o projeto Genoma

Os genes são seções da cadeia de DNA que carregam determinadas informações genéticas e tornam-se disponíveis em uma célula a partir da expressão gênica. Essa expressão é altamente regulada, ou seja, em organismos multicelulares, como o ser humano, as células presentes em diferentes tecidos apresentam um conjunto de genes ativos distintos entre si, e mesmo em organismos unicelulares, como as bactérias, nem todos os genes estão ativos em

Tabela 2.2: Os 20 principais aminoácidos
Fonte: (Alberts et al., 2010)

Nome	Símbolo	Abreviação
Glicina ou Glicocola	Gly, Gli	G
Alanina	Ala	A
Leucina	Leu	L
Valina	Val	V
Isoleucina	Ile	I
Prolina	Pro	P
Fenilalanina	Phe ou Fen	F
Serina	Ser	S
Treonina	Thr, The	T
Cisteina	Cys, Cis	C
Tirosina	Tyr, Tir	Y
Asparagina	Asn	N
Glutamina	Gln	Q
Aspartato ou Ácido aspártico	Asp	D
Glutamato ou Ácido glutâmico	Glu	E
Arginina	Arg	R
Lisina	Lys, Lis	K
Histidina	His	H
Triptofano	Trp, Tri	W
Metionina	Met	M

um determinado momento (Griffiths et al., 2013).

De forma geral, um gene possui em sua estrutura uma região, denominada promotora, responsável pela sua ativação. Essa região é um segmento do DNA ao qual uma estrutura chamada DNA Polimerase é interligada, e assim, é iniciado o processo de síntese da molécula de RNA mensageiro (ou mRNA). Os promotores possuem ainda sequências de nucleotídeos comuns (conservadas) que indicam onde a polimerase deve-se ligar (Alberts et al., 2010).

Além do promotor, os genes possuem em sua estrutura, uma região codi-

ficadora e um terminador. A região codificadora é o segmento do gene que contém a informação necessária para sintetizar a proteína, e o terminador é o segmento do DNA que indica o término da síntese.

É importante destacar que o tamanho do genoma varia de acordo com a espécie. Por exemplo, a bactéria *Mycoplasma genitalium*, presente no trato genital humano possui um dos menores genomas conhecidos, com 580 mil pares de bases, enquanto o *Protopterus aethiopicus*, uma espécie de peixe pulmonado possui cerca de 130 bilhões de pares de bases (40 vezes o tamanho do genoma humano).

Ao comparar os diversos genomas, produzem-se muitos dados sobre a evolução dos seres vivos, ou seja, quanto maior a semelhança entre o DNA de duas espécies, maior é o nível de parentesco evolutivo entre elas, pois descendem de ancestrais comuns mais próximos entre si.

A partir do estudo do genoma, duas outras grandes linhas de estudo surgiram: a proteômica e o transcriptoma. O primeiro, responsável por identificar e interpretar o conjunto de proteínas codificadas pelo genoma (Tajara et al., 2012), e o segundo, responsável pela análise dos conjuntos completos de transcritos (RNA mensageiro, ribossômico, transportador e os micro RNAs) de um dado organismo, órgão ou tecido (Wang et al., 2010). Outros projetos com denominações homólogas também estão em desenvolvimento, como por exemplo, o metaboloma, farmacogenômica e interatoma, de forma que a análise global e compartilhada desses sistemas promovem o avanço da biologia moderna (Passos and Jordan, 2000; Hall, 2012).

Os anos 90 marcaram o início do projeto genoma, que tinha como finalidade o mapeamento e o registro do genoma (conjunto de genes) humano,

para que, posteriormente, essas informações pudessem ser analisadas e melhor compreendidas. Após 13 anos, e mais de US$ 3 bilhões (Collins et al., 1998), foram obtidos os 3,2 bilhões de pares de nucleotídeos, e mais de 32 mil genes (que compreendem menos de 10% de todo o genoma), sendo todos ainda desconhecidos, gerando assim a necessidade de se analisar toda essa informação e relacioná-la com os processos de regulação do organismo.

2.1.4 Análise filogenética

O processo de evolução dos organismos parte do pressuposto de que todos derivam de um ancestral comum (Verli et al., 2014), e permite que o estudo sobre homologias (ou similaridade) entre esses organismos possa construir um mapa, chamado árvore filogenética, que organiza e classifica sistematicamente a evolução dessas espécies.

Com o avanço da bioinformática, a comparação entre organismos, que antes era feita através de observações de características físicas (ou fenotípicas) passou a ser realizada através da comparação de informações genéticas (ou genotípicas). Assim, a sequência de DNA passou a ser o objeto de comparação entre os organismos, dependendo agora de ferramentas e técnicas computacionais capazes de identificar características que possam relacionar diversos organismos, sobretudo pela enorme quantidade de genomas e genes disponíveis nas bases de dados biológicas.

Na figura 2.6 representam-se os nós referentes a um conjunto de 4 sequências da árvore filogenética, e a relação dos nós na escala evolutiva.

Para a construção da árvore filogenética, é necessário primeiramente a

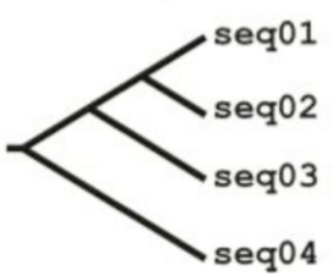

Figura 2.6: Representação da árvore filogenética

realização do alinhamento com precisão, garantindo maior confiabilidade nas análises posteriores. A partir do alinhamento inicial, os métodos utilizados para estimar a filogenia baseiam-se essencialmente em dois grupos: métodos qualitativos e quantitativos. Os métodos qualitativos são construídos através de algoritmos que escolhem a melhor opção entre todas as filogenias possíveis, ou seja, de acordo com um critério, o algoritmo escolhe a melhor representação da relação evolutiva entre as folhas da árvore. Nesta categoria, são utilizados tradicionalmente três métodos de otimização na análise de fologenia: Inferência Bayesiana, Máxima Parcimônia e Máxima Verossimilhança. Os métodos quantitativos realizam uma conversão do resultado do alinhamento já realizado em matrizes de distância contendo as distâncias entre todos os pares de sequências. A partir dessa matriz, duas técnicas distintas são aplicados para a obtenção da árvore: O algoritmo UPGMA (*Unweighted Pair Group Method with Arithmetic Mean*) e o método de Aproximação por vizinhos (Verli et al., 2014; Pevzner and Shamir, 2011; Cohen, 2001).

2.1.5 Padrões em Biossequências

O estudo de padrões em biossequências é fundamental para a compreensão de diversas funções conservadas ao longo da evolução, pois ao considerar que características comuns, ou padrões estão presentes em diversas biossequências, então além de serem importantes na função ou estrutura da molécula, estas passam a representar classes distintas, possibilitando a criação de classificadores baseados nesses padrões (Lemos et al., 2003).

Alguns padrões, denominados padrões classificadores, têm a função de indicar a qual família uma determinada proteína pertence, se e somente se, essa proteína possuir os mesmo padrões determinados pela família em questão.

Além da criação de métodos para classificação da família de proteínas, os padrões também são importantes para identificação de elementos funcionais ou estruturais relevantes na proteína através de classes criadas para essa finalidade, baseando-se na frequência da ocorrência de um determinado padrão.

Biologicamente, é importante destacar que a presença de padrões longos, mesmo que com pequenas ocorrência tem maior relevância do que padrões curtos que ocorrem com maior frequência dentro de uma sequência (Zafalon, 2009).

O uso de padrões em biossequências também é importante para a compreensão do processo de enovelamento da proteína, identificando trechos responsáveis pelas estruturas secundárias ou terciárias, além de uma melhor interpretação das atividades dos organismos estudados.

2.2 Alinhamento de Sequências

O processo de comparação de diferentes organismos a nível genético, diferentemente da análise morfológica, possibilitou aos biólogos identificar os mecanismos de evolução desses organismos através dos padrões encontrados nas biossequências. Esses padrões, responsáveis pela estrutura e funções das proteínas essenciais, são mais bem conservadas que outros trechos, devido sua importância na evolução da espécie (Lemos et al., 2003).

O avanço nas técnicas bioquímicas e o barateamento de equipamentos que realizam o sequenciamento do DNA possibilitou a disponibilização de uma grande quantidade de genomas nos bancos de dados públicos, e consequentemente exigiu o aumento da capacidade computacional, tanto para armazenamento, quanto no desenvolvimento de técnicas de análise direcionadas para a interpretação desses dados. Dentre essas técnicas, os alinhamentos de sequências passaram a ser fundamentais na bioinformática (Verli et al., 2014; Lemos et al., 2003; Souza, 2010; Almeida, 2013).

De forma simplificada, o alinhamento de biossequências (DNA, RNA ou aminoácidos) é o processo de comparação de duas (alinhamento par-a-par) ou mais que duas sequências (alinhamento múltiplo), em que são analisados conjuntos de características individuais, ou padrões que estão na mesma ordem dessas sequências pelos algoritmos computacionais (Almeida, 2013; Ortuño et al., 2013). A similaridade encontrada entre as sequências são chamadas de *identidade*. A *conservação* refere-se a mudanças em uma posição específica de uma sequência de aminoácidos que preserva as característica físico-químicas do resíduo original. A *homologia* refere-se a similaridade atribuída a partir

de um ancestral comum, e quando trechos não correspondentes são encontrados no alinhamento, denomina-se *pontos de mutação*. Os espaços vazios na sequência são considerados como *deleções*.

Durante o processo de alinhamento, as sequências são dispostas em linhas e posicionadas uma sob a outra, de forma que seus elementos componham as colunas do alinhamento. A partir disso, algoritmos computacionais buscam identificar e realizar a melhor correspondência para os elementos das sequências analisadas através da inserção de espaços entre esses elementos. As técnicas de alinhamento, dessa forma, minimizam as diferenças entre as sequências, igualando o seu comprimento. A identidade da sequência alinhada é mensurada através do percentual de elementos idênticos entre as sequências, e não pode ser confundida com a homologia, que diz respeito a descendência comum, herdadas de um ancestral comum (Simossis et al., 2003).

Na figura 2.7 verifica-se um conjunto de 4 sequências de comprimentos diferentes desalinhadas (a), e o mesmo conjunto após o alinhamento (b), com os mesmos comprimentos.

As similaridades encontradas entre as sequências de aminoácidos indicam o grau de conservação entre elas e a conservação de pares de bases de DNAs e RNAs podem indicar regras funcionais e estruturais similares, ou seja, trechos bem conservados durante a evolução indicam que uma determinada sequência é essencial para o metabolismo do organismo. O alinhamento de sequências também contempla a análise filogenética (Rech and Pilatti, 2004), uma vez que ela analisa trechos conservados de espécies diferentes, mas de famílias próximas.

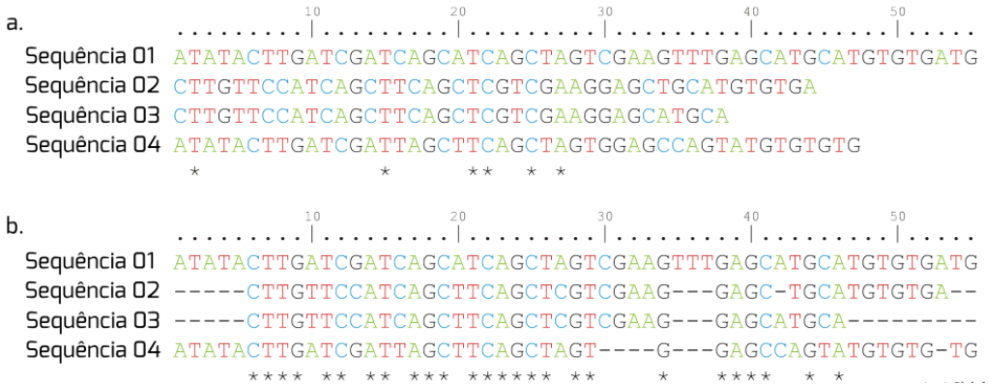

Figura 2.7: Alinhamento de sequências - Conjunto dado e o alinhado
Fonte: (Verli et al., 2014)

2.2.1 Alinhamento par-a-par

Realizar o alinhamento de duas sequências, ou dois grupos de sequências, consiste em considerar as possíveis formas de alinhamento desses pares e obter sistematicamente o melhor alinhamento entre eles. O alinhamento ótimo é obtido através da combinação das maiores similaridades e as menores divergências entre as sequências. Esse resultado apresenta com maior clareza as possíveis mudanças ocorridas durante o processo de evolução (Ye, 2008).

Dessa forma, o alinhamento ótimo pode ser obtido através de um sistema de pontuação, em que cada alinhamento recebe um *score* baseado na quantidade de penalidades, e o maior score indica o alinhamento ótimo.

Os algoritmos de programação dinâmica constroem basicamente uma matriz de comparação de resíduos das duas sequências a serem alinhadas de forma a penalizar as diferenças e valorizar as similaridades entre elas. Ao final do alinhamento, o processo de inserção de espaços, ou *gaps* faz com que as sequências fiquem com o mesmo comprimento, segundo Ye (2008).

De forma a ilustrar o algoritmo de programação dinâmica, duas sequên-

cias (Sequência 1 e Sequência 2) são alinhadas, e um alinhamento possível é apresentado na figura 2.8, em que um exemplo de alinhamento é obtido a partir de duas sequências dadas, de comprimentos diferentes..

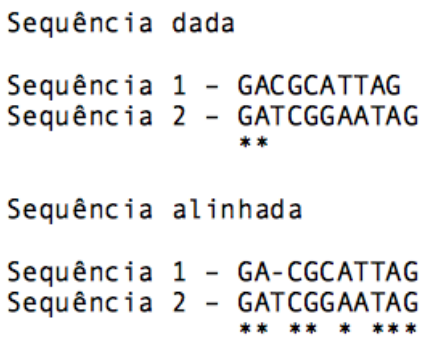

Figura 2.8: Sequências de DNA dadas e alinhadas

Durante o alinhamento, o *match* ocorre quando existe a coincidência entre as bases que estão na mesma posição. Da mesma forma, quando não há similaridade nas posições, ou quando existe a presença de um *gap*, ocorre o *mismatch*. Uma forma simples de pontuação seria atribuir valores positivos para os *matches* e negativos para o *mismatches* e os *gaps*. No caso do alinhamento apresentado na figura 2.8, considerando uma pontuação (+2) para os *matches* e (-2) para o *mismatches* e os *gaps*, a pontuação do alinhamento é baseada em 8 *matches* e 3 *mismatches*, totalizando em uma pontuação total de 8*(+2) + 3*(-2) = 10. É importante destacar que são várias as possibilidades de alinhamento, mas somente a que atinge o maior valor de pontuação é considerado o alinhamento ótimo.

Geralmente, o sistema de pontuação dos alinhamentos procura evitar a inserção excessiva de *gaps* penalizando as inserções das lacunas, pois embora os genomas sejam moldados por pressões seletivas de forma imprevisível,

esses eventos inviabilizam a funcionalidade de uma determinada proteína, e, consequentemente, de sua função no organismo. Além disso, a inserção de lacunas dificulta o processo do alinhamento e requer interpretações mais ponderadas.

As penalidades por inserções de lacunas (*gap penalties*, ou *PL*) ocorrem de acordo com um conjunto de parâmetros, de forma que a abrangência da lacuna é pontuada de acordo com a quantidade de *indels* (*in*serções ou *del*eções) presentes no alinhamento, e a equação 2.1 descreve a regra para esse cálculo.

$$PL = g + e(L - 1) \tag{2.1}$$

em que L é o comprimento da lacuna, g é a penalidade pela abertura das lacunas, e e é o valor da penalidade concedida a cada *indel*, evitando assim, a abertura desnecessária de grandes lacunas.

No caso do alinhamento de duas sequências, existem algoritmos determinísticos baseados em programação dinâmica, em que a solução ótima é sempre encontrada. Os algoritmos de Needleman e Wunsh (Needleman and Wunsh, 1970) e Smith e Waterman (Smith and Waterman, 1981) testam todas as possibilidades de alinhamento, e embora este problema consista em uma elevada complexidade computacional, a execução em um conjunto reduzido de dados torna sua execução viável (Marucci, 2009).

O processo de alinhamento pode ainda ser feito de forma global ou local. A forma de alinhamento global permite analisar a sequência como um todo, procurando identificar um máximo de similaridade entre essas sequências,

descartando trechos ou pontos específicos dessas sequências.

Ao realizar o alinhamento local de trechos de sequências, o objetivo passa a ser a identificação de pontos específicos dentro desses trechos, e que podem fornecer informações úteis, sobretudo na análise de determinados genes em uma sequência. Esses dados importantes, como por exemplo, pontos de mutações, possuem alta relevância biológica, e passam a ser chamados de *hot spots* (Zafalon, 2009).

2.2.2 Alinhamento de Aminoácidos

Ao realizar o alinhamento de nucleotídeos, o algoritmo de programação dinâmica é o recurso mais indicado, de forma que consegue obter o alinhamento ótimo em um intervalo de tempo aceitável através das pontuações dos *matches*, *mismatches* e *gaps*. No entanto, ao realizar o alinhamento de aminoácidos, faz-se necessário considerar dados evolutivos que relacionam esses aminoácidos em grupos específicos, de acordo com Cohen (2001); Ye (2008).

Na tabela 2.3 são apresentadas as cinco categorias dos aminoácidos que possuem entre si características evolucionárias semelhantes, assim como perfis em comum, e que são utilizados para a montagem das matrizes de substituição (figura 2.9 e 2.10).

Tabela 2.3: Aminoácidos e suas categorias

Categoria	Aminoácido
Ácidos e Amino	Asp(D), Glu(E), Asn(N), Gln(Q)
Básico	His(H), Lys(K), Arg(R)
Aromático	Phe(F), Tyr(Y), Trp(W)
Hidrofílico	Ala(A), Cys(C), Gly(G), Pro(P), Ser(S), Thr(T)
Hidrofóbico	Ile(I), Leu(L), Met(M), Val(V)

Com o objetivo de fornecer pesos diferentes na comparação de aminoácidos, as matrizes de substituição BLOSUM (*Blocks of Amino Acid Substitution Matrix*) e PAM (*Percent Accepted Mutation*) foram desenvolvidas por biólogos, de acordo com informações evolucionárias presentes nos 20 aminoácidos essenciais. Algumas variações dessas tabelas, tais como PAM1, PAM70, BLOSUM80, ou BLOSUM62 referem-se a variações na distribuição de pesos pelas matrizes (Rouchka, 2006), e que permitem análises distintas entre os alinhamentos.

A matriz BLOSUM é utilizada sobretudo para pontuar alinhamentos locais de sequências protéicas de natureza divergente, através da procura por regiões mais conservadas de famílias de proteínas. Os números mais elevados que acompanham as matrizes, como por exemplo, BLOSUM80, são indicadas para comparar sequências mais intimamente relacionadas, ou menos divergentes, enquanto as que possuem os menores números são designadas para comparar sequências mais distantemente relacionadas, ou mais divergentes, de acordo com os trabalhos de Henikoff and Henikoff (1992).

As pontuações, ou *scores* em uma matriz BLOSUM referem-se ao logaritmo das razões de chance que medem, em um alinhamento, a razão entre a probabilidade de dois aminoácidos possuírem uma relação biológica, e a possibilidade desses mesmos aminoácidos surgirem ao acaso. A pontuação positiva está relacionada com substituições mais prováveis, e a pontuação negativa é atribuída as substituições menos prováveis.

Na figura 2.9 é apresentada a matriz com as pontuações referentes as substituições dos aminoácidos. Nesse caso, a BLOSUM62.

O cálculo da matriz BLOSUM é realizada por meio da equação 2.2:

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	4																			
Arg	-1	5																		
Asn	-2	0	6																	
Asp	-2	-2	1	6																
Cys	0	-3	-3	-3	9															
Gln	-1	1	0	0	-3	5														
Glu	-1	0	0	2	-4	2	5													
Gly	0	-2	0	-1	-3	-2	-2	6												
His	-2	0	1	-1	-3	0	0	-2	8											
Ile	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
Leu	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
Lys	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
Met	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
Phe	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
Pro	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
Ser	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
Thr	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
Trp	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Tyr	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
Val	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Figura 2.9: Matriz de substituição BLOSUM62
Fonte: (Henikoff and Henikoff, 1992)

$$S_{ij} = (\frac{1}{\lambda}) \log(\frac{p_{ij}}{q_i * q_j}) \tag{2.2}$$

em que p_{ij} é a probabilidade dos dois aminoácidos i e j substituírem um ao outro em sequências relacionadas, e q_i e q_j são as probabilidades de encontrar os aminoácidos i e j em qualquer sequência de proteínas aleatórias. O fator λ indica a escala para cálculo de valores internos.

É importante destacar que a matriz de substituição BLOSUM62 é utilizada como padrão pelos algoritmos da ferramenta BLAST (*Basic Local Alignment Search Tool*). Ele é adaptado para a comparação de proteínas moderadamente distantes.

As matrizes da familia PAM são baseadas nas Cadeias de Markov de mutações em proteínas, de forma que a matriz PAM1 possui 1 ponto de mutação

em cada 100 aminoácidos e, dessa forma, é mais apropriada para um sistema de pontuação de sequências que possuem alto grau de similaridade entre si. No caso da comparação de sequências com baixo índice de similaridade, a matriz PAM1 é multiplicada n vezes por ela mesma. No caso da PAM250, por exemplo, ocorrem 250 substituições em cada conjunto de 100 aminoácidos.

Na figura 2.10 são apresentadas as pontuações das substituições dos aminoácidos, de acordo com o modelo PAM250.

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
Ala A	13	6	9	9	5	8	9	12	6	8	6	7	7	4	11	11	11	2	4	9
Arg R	3	17	4	3	2	5	3	2	6	3	2	9	4	1	4	4	3	7	2	2
Asn N	4	4	6	7	2	5	6	4	6	3	2	5	3	2	4	5	4	2	3	3
Asp D	5	4	8	11	1	7	10	5	6	3	2	5	3	1	4	5	5	1	2	3
Cys C	2	1	1	1	52	1	1	2	2	2	1	1	1	1	2	3	2	1	4	2
Gln Q	3	5	5	6	1	10	7	3	7	2	3	5	3	1	4	3	3	1	2	3
Glu E	5	4	7	11	1	9	12	5	6	3	2	5	3	1	4	5	5	1	2	3
Gly G	12	5	10	10	4	7	9	27	5	5	4	6	5	3	8	11	9	2	3	7
His H	2	5	5	4	2	7	4	2	15	2	2	3	2	2	3	3	2	2	3	2
Ile I	3	2	2	2	2	2	2	2	2	10	6	2	6	5	2	3	4	1	3	9
Leu L	6	4	4	3	2	6	4	3	5	15	34	4	20	13	5	4	6	6	7	13
Lys K	6	18	10	8	2	10	8	5	8	5	4	24	9	2	6	8	8	4	3	5
Met M	1	1	1	1	0	1	1	1	1	2	3	2	6	2	1	1	1	1	1	2
Phe F	2	1	2	1	1	1	1	1	3	5	6	1	4	32	1	2	2	4	20	3
Pro P	7	5	5	4	3	5	4	5	5	3	3	4	3	2	20	6	5	1	2	4
Ser S	9	6	8	7	7	6	7	9	6	5	4	7	5	3	9	10	9	4	4	6
Thr T	8	5	6	6	4	5	5	6	4	6	4	6	5	3	6	8	11	2	3	6
Trp W	0	2	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	55	1	0
Tyr Y	1	1	2	1	3	1	1	1	3	2	2	1	2	15	1	2	2	3	31	2
Val V	7	4	4	4	4	4	4	4	5	4	15	10	4	10	5	5	5	72	4	17

Figura 2.10: Matriz de substituição PAM250
Fonte: (Henikoff and Henikoff, 1992)

2.2.3 Algoritmos para análise de sequências

Nos anos 70, a Lei de Moore estabeleceu a previsão para um grande crescimento na quantidade de transistores, e a computação, como um todo evoluiu, e possibilitou que outras áreas da ciência pudessem evoluir a partir dela (Schatz et al., 2010). Novas técnicas de armazenamento e recuperação de dados, processamento distribuídos e uso de GPUs (*Graphics Processing Unit*, ou Unidade de Processamento Gráfico) passaram a fazer parte da enorme gama de possibilidades frente à bioinformática. As bases de dados que con-

tém biossequências passaram a crescer de forma exponencial nos últimos 15 anos (Zaha et al., 2014), assim como as pesquisas realizadas.

Conforme visto na seção 2.2.1, a análise e comparação das sequências através de algoritmos baseados em programação dinâmica passam a ser inviáveis devido à grande quantidade de sequências, tornando tal análise um problema NP-Completo. Assim, os cientistas passaram a buscar novas formas de otimizar e acelerar o processo de alinhamento de sequências, explorando tanto o hardware quanto o software disponível.

Os algoritmos da família BLAST (*Basic Local Alignment Search Tool*) (Altschul, 1990) foram os primeiros a utilizar métodos heurísticos para o alinhamento de sequências, e possui variantes como o BLASTN e BLASTP para comparação de ácidos nucléicos e proteínas, respectivamente (Simossis et al., 2003). Essencialmente, esse algoritmo detecta regiões de similaridade local entre as sequências. A abordagem então compara a sequência de nucleotídeos ou de proteínas com as sequências armazenadas no banco de dados e realiza o cálculo da significância estatística dos resultados. O BLAST pode ainda ser usado para inferir relações funcionais e evolutivas entre as sequências, assim como ajuda a identificar os membros de famílias de genes (Boratyn et al., 2013).

A família de algoritmos FAST (Lipman and Pearson, 1985; Almeida, 2013) baseia-se na busca rápida de proteínas e nucleotídeos focando-se em um grupo de identidade entre as sequências comparadas. Variações do algoritmo passaram a ser utilizadas, tais como a FASTP, para a comparação de proteínas, e a FASTN para nucleotídeos.

De acordo com os trabalhos de Ortuño et al. (2013), a escolha pelos

melhores algoritmos de alinhamento múltiplo baseia-se em características biológicas muito particulares, de forma que os programas atuais não cobrem 100% das necessidades, ou mesmo de um conjunto em particular de sequências. No entanto, o referido trabalho, é proposta uma técnica de aprendizado de máquina baseada em SVM (*Support Vector Machine*), em que, a partir de uma série de características das sequências, são utilizadas como treinamento do vetor para a indicação do algoritmo mais adequado para uma determinada situação. Essa técnica utiliza o conjunto de sequências de referência do BaliBASE (Thompson et al., 1999; Bahr et al., 2001; Thompson et al., 2005).

2.2.4 Alinhamento de várias sequências

Obter o alinhamento ótimo de duas sequências é por si só importante, não somente pelo desenvolvimento de algoritmo de programação dinâmica, mas pela sua conexão direta com a bioquímica, biologia computacional e construção da árvore filogenética. No entanto, na maioria da situações, é exigido a comparação de várias sequências simultaneamente, dada a disponibilidade em abundância de dados biológicos, e a necessidade em se evidenciar características comuns entre essas sequências. De acordo com Almeida (2013), o alinhamento de várias sequências, passa a ser uma generalização do conceito do alinhamento par-a-par, com complexidade $O(n^2)$.

O alinhamento de várias sequências, ou alinhamento múltiplo (MSA, do inglês *Multiple Sequence Alignment*), tem por objetivo da mesma forma que o alinhamento de duas sequências, a busca pela maior pontuação, ou seja, o maior número de coincidências entre as sequências comparadas, de forma

a melhor representar o cenário evolutivo entre elas. Trata-se de um estudo para a hipótese de homologia entre as bases ou nucleotídeos que constituem os genes do organismo, podendo dessa forma, inferir também sobre sua filogenia.

O desafio de alinhar uma quantidade finita n de sequências através do algoritmo de programação dinâmica passa a ser do tipo NP-Completo, ou seja, não possui uma solução conhecida que apresente uma resposta em tempo polinomial, ou menor com relação a entrada de dados, ou seja, passa a ser computacionalmente inviável (Almeida, 2013; Zafalon, 2012; Marucci, 2009).

De acordo com os trabalhos de Pais et al. (2014), atualmente, existem diversas abordagens para o alinhamento múltiplo, que derivam em sua maioria dos algoritmos de programação dinâmica, tais como a progressiva, iterativa, baseada em consenso, consistência, blocos ou modelos, e cada uma com uma particularidade, sendo mais adequada para um determinado tipo de análise.

Além das técnicas utilizadas para a realização do alinhamento múltiplo de sequências, diversas heurísticas foram desenvolvidas para otimização, com o objetivo de reduzir o tempo de processamento, apresentando resultados com alto grau de precisão e preservando o sentido biológico. Entre essas técnicas destacam-se algumas, tais como o uso de colônia de abelhas por Largo et al. (2016), em que são utilizadas duas funções objetivo (multiobjetivos) para preservar a qualidade e consistência do alinhamento: a soma de pares ponderada (ou WSP - *weighted sum-of-pairs*) e a pontuação do número total de colunas conservadas (TC - *totally conserved*). Técnicas metaheurísticas (Blum and Roli, 2003) baseadas em processos da natureza também foram apresentados por Zafalon (2009), em que características presentes em colônia de formigas podem ser aplicadas na otimização do problema de alinhamento,

sendo que nesse caso, os caminhos mais percorridos pelas formigas, em decorrência do reforço de feromônio, passa a ser decisivo na escolha entre os possíveis caminhos entre a fonte de alimento e a colônia de formigas.

Os trabalhos apresentados por Zafalon et al. (2015) apresentam uma abordagem baseada na paralelização da função objetivo COFFEE, em que a principal característica é a construção de uma biblioteca de referência do alinhamento, cuja avaliação é realizada por posições. Para cada posição de alinhamento estimado, uma matriz de pontuação é construída, contendo os pesos atribuídos a cada alinhamento par a par disponível na biblioteca. Na posição (coluna) analisada, cada célula da matriz corresponde ao alinhamento entre dois resíduos de posição. Se o alinhamento entre os dois resíduos é encontrado na biblioteca, o peso é atribuído à célula, caso contrário, é atribuído o valor 0. A pontuação da posição é dada pela soma de todos os valores na matriz de pontuação dividido pela soma dos pesos dos alinhamentos envolvidos. Amorim et al. (2015) propõe ainda que o alinhamento múltiplo pode ser otimizado através da função objetivo baseada na soma de pares ponderada, ao invés do uso da função COFFEE.

Além dessas técnicas de otimização, se destacam os algoritmos genéticos (Ogata, 2007), otimização dialética (Souza, 2014), simulated annealing (Garcia and Araiza, 2012), algoritmos evolucionários (Olazar, 2007), busca tabu (Riaz et al., 2001), e até mesmo técnicas para implementações em grid computacional, conforme trabalho de Zafalon (2012).

Estudos utilizando Modelos de Markov (Sharma, 2009), (Koski, 2001), (Sun et al., 2014) para otimização do alinhamento múltiplo também têm se mostrado promissores frente sua aplicabilidade estatística, os quais serão

apresentados na seção 2.3.

2.2.5 Alinhamento Progressivo

Os algoritmos do alinhamento múltiplo progressivo utilizam as relações filogenéticas das sequências para gerar o resultado do alinhamento, ou seja, considera sua relação evolutiva (Souza, 2010). O processo divide-se em três etapas: determinar a distância entre as sequências que serão alinhadas através do alinhamento par-a-par de todos os possíveis pares, em seguida é construída a árvore-guia (geralmente através do método *neighbor-joining* (Naruya and Nei, 1987), a partir das distâncias computadas), e por fim, a realização do alinhamento múltiplo propriamente dito, construído progressivamente, de acordo com a relação entre as sequências.

O problema principal do alinhamento progressivo é a dependência do alinhamento múltiplo de sequências finais em relação aos alinhamentos iniciais dos pares de sequências, ou seja, quanto mais distantes forem esses pares, mais erros (que serão propagados ao alinhamento múltiplo) serão cometidos. Outro problema é a escolha apropriada da matriz de pontuação, assim como as penalidades para as lacunas inseridas no processo de alinhamento.

Trata-se de um método rápido de alinhamento e amplamente utilizado por programas da família CLUSTALW, CLUSTALX e CLUSTAL Omega (Almeida, 2013; Cohen, 2001; Sievers and Higgins, 2013).

2.2.6 Alinhamento Iterativo

Os algoritmos baseados em processos iterativos estão presentes na maioria das ferramentas de alinhamento múltiplo. Tal fato deve-se a sua característica de refinamento dos resultados e pela simplicidade de uso, tanto na codificação quanto na complexidade temporal e espacial (Almeida, 2013; Pais et al., 2014).

As estratégias dos algoritmos iterativos baseiam-se essencialmente em extrair das sequências iniciais um perfil que contenha os dados dos elementos alinhados das sequências em cada posição, sendo que dessa forma, ao se obter um resultado mais relevante, o perfil pode então ser atualizado, ocasionando assim no aumento da pontuação do alinhamento, ou mesmo mantendo-se a mesma pontuação em determinada posição.

É importante destacar que outras sequências podem ser escolhidas e realinhadas até que o alinhamento não seja mais alterado, fazendo com que ocorra a convergência da função objetivo para um local de máxima pontuação.

O processo de refinamento ocorre até que não seja mais possível melhorar o resultado do alinhamento, ou até que uma certa quantidade de ciclos seja atingida.

Um dos algoritmos que utiliza os métodos iterativos é o MUMMALS (*Multiple Sequence Alignment Improved by Using Hidden Markov Models with Local Structural Information*) (Pei and Grishin, 2006), e o MUSCLE (*Multiple Sequence Comparison by Log-Expectation*) (Edgar, 2004b).

2.2.7 Heurísticas

Atualmente, a busca pela otimização de qualquer processo por si só se justifica, tendo em vista que é possível aproveitar melhor os recursos disponíveis, acelerando, ou tornando mais eficiente uma ou mais etapas desse processo. Na computação, por exemplo, é possível que um determinado software execute rotinas de forma mais rápida, necessitando de uma menor carga de processamento, com menor consumo de memória, aumentando consideravelmente o seu desempenho de modo geral. Tal raciocínio aplica-se a diversas áreas, e a bioinformática, com grande parte de seus desafios baseando-se em algoritmo de buscas, beneficia-se dessas características.

Ao otimizar um processo de alinhamento múltiplo de sequências, possibilita-se analisar uma quantidade maior de dados e, consequentemente, realizar mais inferências e hipóteses. Diferentes heurísticas podem ser aplicadas no mesmo problema, de forma a combinar melhores estratégias para a solução do alinhamento.

De forma simplificada, a heurística é um método dedicado ao auxílio da solução de um determinado problema de forma mais rápida que a habitual, resultando em uma resposta com um determinado grau de precisão. Computacionalmente busca-se o menor custo de processamento, com a maior eficiência na obtenção dessa resposta.

Ainda, de acordo com Combs et al. (2005), os resultados obtidos pela otimização baseiam-se no valor da função objetivo ou na função do custo. A idéia básica é identificar o valor ideal da função objetivo para os casos em que ela é aplicada, sendo ora maximizar o valor da função objetivo e obter

o valor máximo, ora obter os valores de mínimo, minimizando o valor da função objetivo.

No caso da bioinformática, busca-se na maioria das vezes a melhor pontuação do alinhamento (maior quantidade de coincidências de resíduos) através da maximização da função objetivo.

Uma dos pontos negativos das heurísticas é que não há garantias de se obter a melhor solução, ou solução ótima (Zafalon, 2009; Blum and Roli, 2003). Tal fato deve se ao caráter estatístico da técnica, que ao não atingir um determinado limiar pré-determinado, não oferece uma resposta dentro dos parâmetros de confiança estabelecidos.

2.3 Cadeias de Markov

Muitos processos envolvendo sistemas e variáveis reais são excessivamente complicados de serem resolvidos e mesmo que houvesse uma forma prática e analítica de serem modelados, em muitos casos, é mais indicado o uso de técnicas estatísticas, lançando mão das variáveis estocásticas para a solução de determinados problemas.

De acordo com os trabalhos de Ewens and Grant (2005), dentre os processos estocásticos, destacam-se os processos de Poisson, processos Gaussianos, processos Markovianos e os modelos de processos Ocultos de Markov. Esse último, objeto deste trabalho, apresenta diversos nomes para formalizar os Modelos Ocultos de Markov (ou *HMM - Hidden Markov Model*): Processos Ocultos de Markov, Fontes Markovianas, Cadeias de Markov Ocultas, Funções Probabilísticas de Cadeias de Markov. Sendo assim, a título de padronização será utilizado neste trabalho o termo MMEO (Modelos de Markov de Estados Ocultos).

As cadeias de Markov, desenvolvidas no início do século XX por Andrei Markov têm por objetivo modelar processos que ocorrem na prática, e que podem ser observados como fontes que geram sinais segundo determinadas regras. Essas fontes, ao produzirem os sinais, geram uma sequência de símbolos sobre um determinado alfabeto, de tal forma que essa sequência pode ser recuperada, ou obtida apenas por meio da verificação do estado da fonte em períodos regulares de tempo (Sharma, 2009). Pode-se classificar essas fontes como produtoras de sinais discretos ao emitirem uma sequência que podem ser representados por elementos de um conjunto discreto, e caso contrário, a

fonte produz um sinal contínuo. As cadeias de Markov compreendem tanto as fontes que emitem sinais contínuos quanto discretos.

Os sinais observados são denominados sequências de observações (ou simplesmente, observáveis) da fonte emissora, e uma forma mais intuitiva de se obter as sequências de observações a partir da fonte é simplesmente a observando em intervalos de tempo pré-definidos.

De acordo com os trabalhos de Sergio (2008); Rabiner (1989); Bell et al. (1990), o processo de examinar uma fonte que pode gerar sinais pode ser aplicado a diversas situações práticas, tais como o Reconhecimento de Voz, Compressão de Dados, Análise Climatológica, Mercado de Finanças e a própria Bioinformática.

Uma cadeia de Markov realiza a modelagem de sinais ou sistemas em que cada observação corresponde a um estado desse sistema e em que cada estado dependa apenas do estado anterior, considerando a dinâmica do sistema, em que seu estado pode mudar a cada intervalo da observação, sendo possível ainda que o estado se mantenha entre duas observações. Cada possível par de observações consecutivas que o modelo pode gerar é denominado *transição*. A seguir será apresentado a definição formal de uma cadeia de Markov.

Definição 2.1. (Cadeias de Markov): Uma *cadeia de Markov* é uma trinca $\lambda=(Q, a, \pi)$, onde Q é um conjunto finito chamado *conjunto de estados da cadeia* λ, a é uma matriz de números reais não-negativos indexada por $Q \times Q$ de forma que para $(i, j) \in Q \times Q$, a_{ij} denota a probabilidade de a cadeia estar no estado j em um instante, dado que a cadeia estava no estado i no instante anterior, e π é um vetor de número reais não-negativos indexado por Q de forma que $\pi(i)$ é a probabilidade de a cadeia estar no estado i no

início do processo de observação. A matriz a é chamada *matriz de transição* de λ e o vetor π é chamado de *vetor de probabilidades iniciais* de λ.

É usual representar uma cadeia de Markov por meio de um grafo dirigido em que o conjunto de vértices é Q, ou seja, em que cada vértice corresponde um estado da cadeia e em que um arco $(i, j) \in Q \times Q$ do grafo corresponde a uma transição da cadeia, com probabilidade da transição dada pela matriz a. É importante ressaltar que uma cadeia de Markov admite transições de um estado para ele mesmo, permitindo assim, arcos da forma (i, i), para qualquer $i \in Q$.

Na figura 2.11 está ilustrada uma cadeia de Markov com três estados (S_1, S_2 e S_3) e as possíveis transições entre eles.

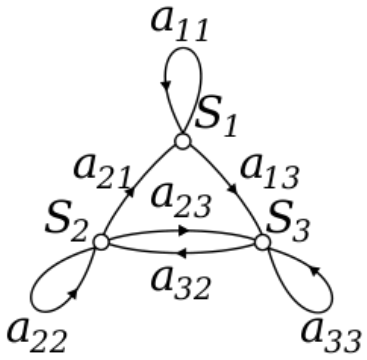

Figura 2.11: Grafo da cadeia de Markov

Um fato importante acerca das cadeias de Markov é que dada uma sequência de observações $q = q_1...q_n$ com $q_i \in Q$ para $i = 1, ..., n$ esta determina um percurso orientado no grafo de transições.

A partir de uma sequência de observações $q = q_1...q_n$, a probabilidade $Pr(q)$ de q ter ocorrido de acordo com a cadeia de Markov λ é dada por:

$Pr(q_1, ..., q_n) = Pr(q_n|q_1, ..., q_{n-1})Pr(q_1, ..., q_{n-1})$

$Pr(q_1, ..., q_n) = Pr(q_n|q_1, ..., q_{n-1})Pr(q_{n-1}|q_1, ..., q_{n-2})Pr(q_1, ..., q_{n-2})$

$Pr(q_1, ..., q_n) = Pr(q_n|q_1, ..., q_{n-1})Pr(q_{n-1}|q_1, ..., q_{n-2})...Pr(q_2|q_1)Pr(q_1)$

$Pr(q_1, ..., q_n) = Pr(q_1)\prod_{i=1}^{n-1} Pr(q_{i+1}|q_1, ..., q_i)$

em que, para cada igualdade segue de aplicações sucessivas da definição de probabilidade condicional. Como a sequência q é gerada por uma cadeia de Markov, cada símbolo observado depende apenas do símbolo anterior, e, portanto, a probabilidade da cadeia λ gerar a observação q é:

$$Pr(q) = Pr(q_1)\prod_{i=1}^{n-1} Pr(q_{i+1}|q_i) = \pi(q_1)\prod_{i=1}^{n-1} a_{q_i q_{i+1}} \tag{2.3}$$

2.3.1 Modelos de Markov de Estados Ocultos

Os Modelos de Markov de Estados Ocultos (ou MMEOs) são uma generalização das cadeias de Markov, em que o símbolo produzido por um estado, não é necessariamente único, diferentemente das cadeias de Markov. Ou seja, cada estado de um Modelo de Markov de Estado Oculto gera como observação um símbolo que pertence a um alfabeto Σ. As observações produzidas por um estado são geradas aleatoriamente de acordo com uma distribuição de probabilidades sobre esse alfabeto Σ.

Dessa forma, o fato de não existir uma relação biunívoca entre o estado que produz uma observação e o símbolo por ela produzido esclarece a nomenclatura concedida aos MMEOs (Pardoux, 2008).

Definição 2.2 (Cadeias de Markov de Estados Ocultos): Uma *Cadeia de Markov de Estados Ocultos* é uma quíntupla $\lambda=(Q, \Sigma, a, e, \pi)$, em que Q é um conjunto finito chamado *conjunto de estados da cadeia* λ; Σ é um conjunto finito, chamado *alfabeto* de λ; a é uma matriz de números reais não-negativos indexada por $Q \times Q$ de forma que para $a_{i,j}$ denota, para cada par $(i, j) \in Q \times Q$ a probabilidade de transição para o estado j dado que o estado anterior do modelo era i; e é uma matriz de números reais não-negativos indexada por $Q \times \Sigma$ tal que, para cada par $(i, \sigma) \in Q \times \Sigma$, $e_{i,\sigma}$ denota a probabilidade do símbolo σ ser gerado no estado i; π é um vetor indexado por Q de forma que $\pi(i)$ denota a probabilidade λ estar no estado $i \in Q$ no início da geração de uma sequência de observações.

De acordo com a definição das MMEOs, no geral, os estados que geram as sequências de observações são desconhecidos e a respeito dessas sequências de observações, dois questionamentos podem ser realizados inicialmente. O primeiro é como se determina a probabilidade de um determinado modelo dado ele ter gerado as observações. O segundo questionamento é identificar qual foi a sequência de estados que gerou as observações. No entanto, como no caso dos MMOEOs mais de uma sequência de estados pode ter gerado essas observações, e dessa forma, consequentemente, passa-se a admitir respostas que melhor expliquem como uma determinada sequência de observações pôde ser obtida.

Uma vez que as observações sejam conhecidas, surge outro problema: como realizar sua modelagem prática. Esse problema divide-se em duas outras partes. A primeira se refere a escolha do conjunto de estados Q e o

alfabeto Σ do modelo a ser construído. A outra parte é como se determinam os parâmetros do modelo, sendo conhecidos o conjunto de estados Q e o alfabeto Σ.

Os três problemas citados anteriormente são parte da modelagem dos sistemas de Modelos de Markov de Estados Ocultos, e seguem formalizados:

Problema 1 (Problema da Avaliação). *Dado um* MMEO $\lambda=(Q, \Sigma, a, e, \pi)$ *e uma sequência de observações* $s = s_1...s_n$, *calcular a probabilidade* $\Pr(s|\lambda)$ *de a sequência ter sido gerada pelo modelo.*

Problema 2 (Problema da Decodificação). *Dado um* MMEO $\lambda=(Q, \Sigma, a, e, \pi)$ *e uma sequência de observações* $s = s_1...s_n$, *encontrar uma sequência de estados* $q^* = q_1^*...q_n^*$ *que melhor explique, segundo um critério, a geração de s.*

Problema 3 (Problema do Treinamento). *Dado um* MMEO *em que apenas o conjunto de estados* Q *e o alfabeto* Σ *sejam conhecidos e dada uma sequência de observações s, estimar os parâmetros a, e e* π *do modelo a partir de s.*

2.3.2 Solução dos problemas canônicos

A solução dos problemas fundamentais na modelagem de um MMEO baseiam-se em duas etapas principais: a identificação dos parâmetros do modelo, e o seus ajustes, de acordo com os *Problemas-controle*.

De acordo com os 3 problemas distintos descritos na seção anterior, as soluções específicas para cada um deles são apresentadas a seguir:

Solução do Problema 1

No problema 1 procura-se identificar a forma mais adequada para se calcular a probabilidade da sequência ser gerada pelo modelo, $P(O|\lambda)$.

Para isso, deve-se considerar os seguintes parâmetros do modelo e sua respectiva sequência de estados observáveis:

$$\lambda = (\hat{A}, \hat{B}, \pi); \tag{2.4}$$

$$O = O_1, O_2, O_3 \cdots O_T; \tag{2.5}$$

De forma a otimizar o processo, considere que cada transição entre os possíveis estados q_{t-1} e q_t gere um observável O_t, e o modelo prevê transições possíveis entre quaisquer pares de estados, ou seja, $a_{q_{t-1}qt} > 0, \forall\ t$.

Assim, pode-se supor que a observação O tenha sido gerada pela seguinte sequência de estados:

$$Q = q_0, q_1, q_2 \cdots q_T, \tag{2.6}$$

sendo o índice $0 \leq t \leq T$ representando um instante no tempo, de forma que q_0 representa o estado de Markov no instante $t = 0$, ou seja, o estado inicial. Tem-se então que a probabilidade de Q pode ser dada por:

$$P(Q|\lambda) = \pi_{q_0} a_{q_0 q_1} a_{q_1 q_2} \cdots a_{q_{T-1} q_T} \tag{2.7}$$

Assume-se então que as observações são independentes entre sim, conforme segue:

$$P(O|Q,\lambda) = \prod_{t=1}^{T} P(O_t|q_{t-1}, q_t, \lambda) \tag{2.8}$$

de onde segue:

$$P(O|Q,\lambda) = b_{q_0 q_1}(O_1).b_{q_1 q_2}(O_2)\cdots b_{q_{T-1} q_T}(O_T) \tag{2.9}$$

Assim, das equações (2.8) e (2.9), pode-se escrever a probabilidade combinada de O e Q, conforme segue:

$$P(O,Q|\lambda) = P(O|Q,\lambda)P(Q,\lambda) \tag{2.10}$$

Ao aplicar o somatório de (2.10) no conjunto das sequências de estados Q, tem-se que:

$$P(O|\lambda) = \sum_{Q} P(O|Q,\lambda)P(Q|\lambda) = \sum_{Q} \pi_{q_0} \prod_{t=1}^{T} a_{q_{(t-1)} q_t} b_{q_{(t-1)} q_t}(O_t) \tag{2.11}$$

$$= \sum_{q_0 q_1 \cdots q_T} \pi_{q_0} a_{q_0 q_1} b_{q_0 q_1}(O_1) a_{q_1 q_2} b_{q_1 q_2}(O_2)\cdots a_{q_{(T-1)} q_T} b_{q_{(T-1)} q_T}(O_T) \tag{2.12}$$

Ou seja, para uma melhor compreensão da equação (2.12), considere uma sequência de estados Q, e a probabilidade de Markov que possibilita preencher um dos N possíveis estado no tempo $t = 0$ é definida por π_{q_0}. Tem-se então, que dessa forma, em $t = 1$, o sistema passa por uma transição do estado q_0 para q_1, criando o observável O_1, sendo que esse processo se repete até o

tempo $t = T$. Após identificado a probabilidade para uma dada sequência Q, o mesmo ocorre para as demais sequências restantes, sendo que a soma sobre todas as sequências fornece a probabilidade que o modelo tem de formar a sequência O de observáveis.

Dessa forma, a partir da equação (2.12), é possível observar que existem N^T sequências Q de T posições obtidas a partir de N estados, ou seja, existem N^T termos presente no conjunto de somatório, o que resulta em $N^T - 1$ adições. Da mesma forma, são T operações em que os termos $a_{q_{t-1}q_t}.b_{q_{t-1}q_t}(O_t)$ são multiplicados, de forma que $1 \leq t \leq T$, e $T - 1$ são as multiplicações entre esse conjunto de termos e seus correspondentes, desde $a_{q_0q_1}.b_{q_0q_1}(O_1)$ até $a_{q_{T-1}q_T}.b_{q_{T-1}q_T}(O_T)$, resultando em $(2T - 1)$ multiplicações em cada termo do somatório, somando no total $(2T - 1).N^T$ multiplicações, ou seja, computacionalmente inviável, tomando como exemplo um sistema composto de 5 estados, e uma sequência de 100 observáveis, e de acordo com a equação (2.12), seriam realizados $2.200.4^200 \approx 10^{72}$ operações. No entanto, conforme observado por (Rabiner, 1989; Pardoux, 2008), um procedimento mais eficiente, denominado *forward-backward* é capaz de obter a solução para o problema da avaliação de uma forma muito mais eficiente, sendo que para esse caso, apenas a parte *forward* será necessária.

O algoritmo Forward-Backward

Primeiramente, considere a variável denominada *forward* definida a seguir, como sendo a probabilidade da observação parcial da sequência de elementos observáveis, partindo do elemento O_1 até O_t.

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t, q_t = S_i|\lambda) \tag{2.13}$$

Essa sequência está representada em conjunto com a probabilidade de ocupação do estado S_i da cadeia de Markov no instante t, ou seja, em função do tempo. Esse aspecto indica o uso de conjuntos ordenados de eventos, em que pode-se assumir que $\alpha_t(i)$ é válido para qualquer $0 \leq t \leq T$. Dessa forma, é possível solucionar o Problema 1 por meio do seguinte procedimento:

1. Inicialização

$$\alpha_0(i) = \pi_i, onde 1 \leq i \leq N \tag{2.14}$$

2. Indução

$$\alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i) a_{ij} b_{ij}(O_{t+1}) \tag{2.15}$$

$$\text{com } 0 \leq t \leq T-1 \text{ e } 1 \leq j \leq N$$

3. Finalização

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{2.16}$$

É importante destacar que o processo de indução é o mais relevante nesse procedimento e será mais detalhado a seguir.

O termo $\alpha_t(i)$ indica a probabilidade conjunta da observação parcial $O = O_1 O_2 \cdots O_t$ e da ocupação do estado $q_t = S_i$, e ao multiplicar os termos a_{ij} por $b_{ij}(O_{t+1})$ é calculada a probabilidade conjunta da transição do estado

$q_t = S_i$ para $q_{t+1} = S_j$, assim como a emissão do observável O_{t+1} como resultado da transição a_{ij}.

Dessa forma, ao multiplicar os termos $\alpha_t(i)$, a_{ij} e $b_{ij}(O_{t+1})$, e ao realizar a soma sobre todos os estados $1 \leq i \leq N$ chegamos à probabilidade conjunta da observação parcial $O = O_1 O_2 \cdots O_t$, da ocupação do estado $q_{t+1} = S_j$, e da emissão do elemento observável O_{t+1} como resultado de todas as transições, que é o valor da equação (2.15), ou seja, $\alpha_{t+1}(j)$.

Ao analisar a definição da variável *forward* no instante de observação T, faz-se necessário realizar o somatório de $\alpha_T(i)$ sobre os estados $1 \leq i \leq N$, da seguinte forma:

$$\alpha_T(i) = P(O_1 O_2 \cdots O_T, q_T = S_i|\lambda) \tag{2.17}$$

Assim, é possível observar que todo o processo requer $2N^2T$ multiplicações, mais as $(N-1)NT$ adições, totalizando em $(3N-1)NT$ operações aritméticas nesse etapa. Ao utilizar o mesmo exemplo citado anteriormente, contendo uma sequência de $T = 100$ observáveis e um espaço de estado $N = 5$, seriam realizadas 7000 operações, em comparação as 10^{72} necessárias pelo sistema de força bruta, ou seja, uma diferença de ordem 10^{69}, demonstrando a superioridade da técnica *forward-backward* na solução do problema 1.

Ao considerar a variável independente *backward*, definida por:

$$\beta_t(i) = P(O_{t+1} O_{t+2} \cdots O_T|q_t = S_i, \lambda) \tag{2.18}$$

Significa que a probabilidade conjunta do modelo de Markov estar no

estado S_i em t com uma probabilidade da observação parcial $O_{t+1}O_{t+2}\cdots O_T$, nos momentos seguintes a t. O passo *backward* do procedimento é semelhante ao *forward*, conforme segue:

1. Inicialização

$$\beta_T(i) = 1, onde 1 \leq i \leq N \tag{2.19}$$

2. Indução

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_{ij}(O_{t+1}) \beta_{t+1}(j) \tag{2.20}$$

$$t = T-1, T-2, \cdots, 0$$

$$\text{com } 1 \leq i \leq N$$

Considerando o estado inicial $q_0 = S_i$, procura-se identificar β_0, ou seja, a probabilidade da sequência completa de observações. Daí, tem-se que:

$$\beta_0(i) = P(O_1 O_2 \cdots O_T | q_0 = S_i, \lambda) \tag{2.21}$$

Assim, de forma a compreender o procedimento, utiliza-se como inicialização os seguintes termos iniciais:

$$\beta_{T-1}(j) = \sum_{k=1}^{N} a_{jk} b_{jk}(O_T) \beta_T(k) = \sum_{k=1}^{N} a_{jk} b_{jk}(O_T) \tag{2.22}$$

$$\beta_{T-2}(i) = \sum_{j=1}^{N} a_{jk} b_{jk}(O_{T-1}) \beta_{T-1}(j) \tag{2.23}$$

$$= \sum_{j=1}^{N} \left[a_{ij} b_{ij}(O_{T-1}) \sum_{k=1}^{N} a_{jk} b_{jk}(O_T) \right] \tag{2.24}$$

Ao declarar que as observações são independentes, é possível concluir que a sequência está sendo desenvolvida de trás para frente, conforme a equação (2.25) a seguir:

$$\beta_{T-2}(i) = \left[\sum_{j=1}^{N} a_{jk} b_{jk}(O_{T-1}) \right] . \left[\sum_{j=1}^{N} \sum_{k=1}^{N} a_{jk} b_{jk}(O_T) \right] \tag{2.25}$$

$$= P(O_{T-1} O_T | q_{T-2} = S_i) \tag{2.26}$$

Solução do Problema 2

A solução do problema 2 consiste basicamente em identificar a sequência ótima de estados que estão associados a uma sequência de observáveis. O algoritmo de Viterbi (Forney, 1973) é utilizado por Jelinek (1998), e (Rabiner, 1989) ilustra a dificuldade na escolha do critério de otimização, adotando a idéia de que a cada instante t é possível identificar o estado mais provável, considerando assim, a seguinte definição:

$$\gamma_t(i) = P(q_t = S_i | O, \lambda) \tag{2.27}$$

em que $\gamma_t(i)$ representa a probabilidade de um dado modelo $\lambda = (\hat{A}, \hat{B}, \pi)$ e uma sequência de observáveis $O_1 O_2 \cdots O_T$, no instante de tempo t, o sistema tenha ocupado o estado S_i, que em termos das variáveis *forward-backward* teríamos:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{2.28}$$

e, conforme citado por Rabiner (1989), o fator $P(O|\lambda) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i)$ faz de $\gamma_t(i)$ uma medida de probabilidade, de forma que:

$$\sum_{i=1}^{N} \gamma_t(i) = 1 \tag{2.29}$$

Dessa forma, o algoritmo de Viterbi contempla apenas as possíveis transições, conforme segue.

O algoritmo de Viterbi

Forney (1973) propõe uma maneira recursiva para questão da estimativa de uma sequência de estados para um processo Markoviano de estado finito e tempo discreto, ou seja, identificar a melhor, ou mais provável sequência completa de estados $Q = q_1 q_2 \cdots q_T$, dada a sequência de observáveis $O = O_1 O_2 \cdots O_T$, em outras palavras, a busca pela maximização de $P(Q|O, \lambda)$, em que o resultado é a sequência de estados mais prováveis.

Sendo assim, segue a definição da probabilidade do caminho mais provável que leva ao estado S_j em t, criando os primeiros observáveis t.

$$\delta_t(j) = max_{q_1 q_2 \cdots q_{t-1}} P[q_1 q_2 \cdots q_t = S_j, O_1 O_2 \cdots O_t | q_0 = S_i, \lambda] \tag{2.30}$$

E por indução, tem-se que:

$$\delta_{t+1}(k) = max_j[\delta_t(j)a_{jk}b_{jk}(O_{t+1})] \tag{2.31}$$

Assim, de forma a armazenar a sequência de estados, é possível utilizar um vetor auxiliar $\psi_t(k)$, que armazena, para cada t, o índice j do estado $q_{t-1} = S_j$ que maximiza a sequência até o estado $q_t = S_k$, cujo processo é descrito a seguir.

1. Inicialização

$$\delta_i(j) = a_{ij}b_{ij}(O_1) \tag{2.32}$$

$$\text{com } 1 \leq j \leq N$$

$$\psi_1(j) = 0 \tag{2.33}$$

2. Indução

$$\delta_t(k) = max_{1 \leq j \leq N}[\delta_{t-1}(j)a_{jk}b_{jk}(O_t)], \tag{2.34}$$

$$\text{com } 2 \leq t \leq T \text{ e } 1 \leq k \leq N$$

$$\psi_t(k) = argmax_{1 \leq j \leq N}[\delta_{t-1}(j)a_{jk}], \tag{2.35}$$

$$\text{com } 2 \leq t \leq T \text{ e } 1 \leq k \leq N$$

3. Finalização

$$P^* = max_{1 \leq k \leq N}[\delta_T(k)], \tag{2.36}$$

$$q_T^* = argmax_{1 \leq k \leq N}[\delta_T(k)], \tag{2.37}$$

4. Recriação do Caminho

$$q_T^* = \left\{\psi_{t+1}, \left\{q_{t+1}^*\right\}\right\}, \tag{2.38}$$

$$\text{para } t = T-1, T-2, \cdots, 1$$

Assim, de acordo com os passos indicados pelo algoritmo de Viterbi, a escolha do melhor caminho é feita arbitrariamente com uma entre as demais sequências com a mesma probabilidade.

Solução do Problema 3

De acordo com Rabiner (1989), o problema do treinamento, ou da maximização da probabilidade de uma sequência de observáveis é de fato, o mais complexo de ser resolvido, pois não existe uma técnica analítica conhecida que possibilite identificar sistematicamente os parâmetros do modelo $\lambda = (\hat{A}, \hat{B}, \pi)$ de forma que estes maximizem a probabilidade do modelo gerar uma sequência completa de observáveis $P(O|\lambda)$, ou seja, buscamos a solução da equação:

$$\overline{\lambda} = argmax P(O|\lambda) \tag{2.39}$$

Entretanto, existe uma técnica denominada Algoritmo Baum-Welch, que

possui a capacidade de maximizar a probabilidade local. Esse algoritmo, citado por (Jelinek, 1998) é o método adequado para a resolução desse problema.

O algoritmo de Baum-Welch

Considere a seguinte definição da equação cuja variável ξ_t pode ser expressa em termos das variáveis *forward* e *backward*:

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda); \tag{2.40}$$

sendo $\xi_t(i,j)$ a probabilidade conjunta de estar no estado S_i no instante t, e no estado S_j no instante $t+1$. Tomando como suporte as equações (2.13) e (2.18) dos algoritmos *forward* e *backward*, respectivamente, tem-se que:

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda) = \frac{P(q_t = S_i, q_{t+1} = S_j, O|\lambda}{P(O|\lambda)} \tag{2.41}$$

$$= \frac{\alpha_t(i) a_{ij} b_{ij}(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} \tag{2.42}$$

$$= \frac{\alpha_t(i) a_{ij} b_{ij}(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_{ij}(O_{t+1}) \beta_{t+1}(j)} \tag{2.43}$$

E, a partir das equações (2.43), o somatório sobre o índice j, com $1 \leq j \leq N$ fica:

$$\sum_{j=1}^{N} \xi_t(i,j) = \sum_{j=1}^{N} \frac{\alpha_t(i) a_{ij} b_{ij}(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} \tag{2.44}$$

$$= \frac{\alpha_t(i)[\sum_{j=1}^{N} a_{ij} b_{ij}(O_{t+1}) \beta_{t+1}(j)]}{P(O|\lambda)} \tag{2.45}$$

$$= \frac{\alpha_t(i)\beta_t(j)}{P(O|\lambda)} \tag{2.46}$$

Portanto, tendo que as equações (2.46) se iguala a equação (2.20), referente à variável *backward* em t, tem-se a equivalência entre (2.46) e (2.28), da seguinte forma:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{2.47}$$

Assim, a partir de (2.47), é possível obter a estimativa da quantidade de vezes que que o estado S_i é visitado no tempo de observação T por meio do somatório de $\xi_t(i)$, e da mesma forma, para identificar a quantidade de transições a partir de S_i, basta induzir o somatório até $T-1$. Da mesma forma, ao realizar o somatório de $\xi_t(i,j)$ para $T-1$, obtem-se a estimativa da quantidade de transições entre os estados $q_{t-1} = S_i$ e $q_t = S_j$, conforme segue:

$$\sum_{t=0}^{T-1} \gamma(i) = \text{Quantidade estimada de transições a partir de } S_i \tag{2.48}$$

$$\sum_{t=0}^{T-1} \xi_t(i,j) = \text{Quantidade estimada de transições de } S_i \text{ para } S_j \tag{2.49}$$

A partir de (2.48) e (2.49), é possível refazer a estimativa dos parâmetros do modelo da seguinte forma:

Tem-se que:

$$\bar{\pi}_i = \text{quantidade esperada de vezes no estado } q_0 = S_i = \gamma_1(i) \tag{2.50}$$

$$\bar{a}_{ij} = \frac{\text{Quantidade estimada de transições de } S_i \text{ para } S_j}{\text{Quantidade estimada de transições a partir de } S_i} \tag{2.51}$$

$$= \frac{\sum_{t=0}^{T-1} \xi_t(i,j)}{\sum_{t=0}^{T} \gamma_t(j)} \tag{2.52}$$

$$\bar{b}_{ij}(k) = \frac{\text{Quantidade esperada de transições entre os estados } (i,j) \text{ e observações de } y_k}{\text{Quantidade esperada de transições entre os estados } (i,j)} \tag{2.53}$$

$$= \frac{\sum_{t=0}^{T} \gamma_t(j)}{\sum_{t=0}^{T} \gamma_t(j)} \tag{2.54}$$

Portanto, se definir o modelo em questão como $\lambda = (\hat{A}, \hat{B}, \pi)$ e utilizar os parâmetros de (2.50), (2.52) e (2.54), é possível estabelecer os novos parâmetros do modelo $\bar{\lambda} = (\bar{\hat{A}}, \bar{\hat{B}}, \bar{\pi})$, sendo que:

1. ou $\bar{\lambda} = \lambda$, ou seja, o modelo baseado em λ maximiza a sequência de observação;

2. ou $\bar{\lambda}$ tem maior probabilidade que o modelo λ, pois $P(O|\bar{\lambda}) > P(O|\lambda)$, ou seja, é um modelo mais adequado, cuja probabilidade de que a sequência de observação O tenha sido gerada é maior.

Esse processo é então, de maneira iterativa executado até que $\bar{\lambda} = \lambda$.

2.3.3 Modelos Ocultos de Markov aplicados à Bioinformática

Os Modelos Ocultos de Markov (ou MMEO - Modelos de Markov de Estados Ocultos) ocupam uma posição de destaque em diversos problemas, conforme informado na seção 2.3. A Bioinformática passou a utilizar os MMEOs a partir dos anos 80, inicialmente em problemas de reconhecimento de padrões (Liew et al., 2001), para identificação de famílias de genes por meio de *microarrays*; detecção de proteínas homólogas (Söding, 2005), predição da estrutura protéica (Karplus, 2009) e, mais recentemente, como uma técnica para alinhamento múltiplo de sequências (Mimouni et al., 2004; Sun et al., 2012, 2014; Mulia et al., 2012).

No caso da modelagem de famílias de proteínas, os Modelos Ocultos de Markov adotam uma arquitetura *left-right*, em que as observações são as próprias sequências de aminoácidos, que compõem a estrutura primária da proteína. Dessa forma, é considerado que um bom modelo para uma família de proteínas aquele que atribui uma probabilidade maior à sequências que pertencem a família modelada, e uma probabilidade menor para aquelas que não pertencem à família. Como a arquitetura de um MMEO é altamente dependente do problema, o aspecto linear e sequencial de uma proteína é modelado adotando-se os estados m_j representando cada coluna da sequência da proteína, e a cada estado é associado uma probabilidade de emissão $B = \{b_{m_j}(n)\}$ de acordo com a composição da família de proteínas na coluna

correspondente. Considerando a possibilidade de inserção (representado por i) de *gaps* e *deleções* (representado por d) causados pelas mutações, um grafo é representado por meio da figura 2.12.

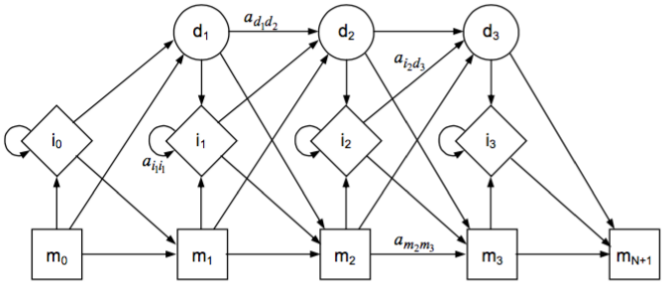

Figura 2.12: Modelagem de uma MMEO

Nascimento (2009) descreve ainda a aplicação das Cadeias de Markov para melhoramento genético por meio do método de simulação Monte Carlo, e por meio dos algoritmos Matropolis-Hastings (para obtenção das estimativas de frequências de recombinação entre pares de marcadores), *simulated annealing* (aplicado no estabelecimento da melhor ordem de ligação na constrição de mapas genéticos) e amostrador de Gibbs (para a obtenção das estimativas de parâmetros de adaptabilidade e estabilidade). Neste trabalho, as cadeias de Markov são utilizadas para contornar os problemas de alta dimensionalidade, tais como a integração numérica, em que os resultados são imprecisos quando d (ou deleções apresentadas na figura 2.12) é muito alto.

Um estudo sistemático para a identificação de genes foi apresentado por Kashiwabara (2011), onde as cadeias ocultas generalizadas de Markov foram utilizadas para otimizar o modelo para predição de genes por meio da integração de sensores e seus parâmetros arbitrários. Nesse trabalho, é observado que muitos preditores possuem uma arquitetura baseada nos moldes

dos MMEOs, em que cada modelo de gene é formado por um conjunto de estados, e cada estado possui uma distribuição de duração específica.

2.4 Ferramenta MUSCLE

A ferramenta MUSCLE (*Multiple Sequence Comparison by Log-Expectation*) é uma abordagem computacional voltada para o alinhamento múltiplo de sequências biológicas, com ênfase na qualidade do alinhamento e redução do tempo de execução (Edgar, 2004b). Nas seções 2.4.1, 2.4.2 e 2.4.3 são apresentados os detalhamentos desta ferramenta.

2.4.1 Metodologia básica

A metodologia básica desse algoritmo é baseada em 3 estágios, e envolve técnicas combinadas de alinhamentos progressivos e iterativos. O primeiro estágio consiste na obtenção das medidas de similaridade entre todos os possíveis pares de sequências, através da contagem dos *k-mers*, ou tuplas de comprimento K presentes nas sequências. Os resultados são computados em uma matriz de distância, que por sua vez fornece as informações para a construção da árvore-guia através do algoritmo UPGMA (*Unweighted Pair Group Method with Arithmetic Mean*) ou via *Neighbor-Joining* (NJ). O alinhamento progressivo é então realizado seguindo a ordem da árvore-guia, até sua raiz (Edgar, 2004b).

O segundo estágio consiste no aperfeiçoamento dos processos ocorridos no primeiro estágio: As medidas de similaridade são obtidas dessa vez pelo cálculo da identidade fracional das sequências alinhadas pela *distância de Kimura* (Kimura and Ohta, 1972), e a árvore-guia é obtida também pelo algoritmo UPGMA. As árvores do primeiro e do segundo estágio são então comparadas, de forma a identificar quais conjunto de nós das ramificações

foram alterados para um novo alinhamento. Essa etapa pode ser realizada novamente até a convergência da árvore, ocasionando o fim das iterações.

O terceiro estágio recebe os resultados do estágio anterior e é destinado ao refinamento iterativo através de uma variante do *Particionamento restrito dependente da árvore*. Essa técnica realiza a segmentação da árvore em subconjuntos distintos através da eliminação das arestas. Essas arestas são percorridas em ordem decrescente de distância da raiz. A partir da segmentação, o perfil de cada subconjunto é extraído pelo alinhamento múltiplo, e as colunas que não possuem resíduos são eliminadas. Os dois perfis obtidos na etapa anterior são então realinhados um com o outro pelo alinhamento perfil-perfil e a pontuação SP (*sum-of-pairs*, ou soma de pares) desse alinhamento é calculada. Caso a pontuação seja maior, o alinhamento é mantido, e caso contrário, é descartado.

O algoritmo é encerrado quando todas as arestas percorridas mantiverem as alterações ou quando um valor máximo de iterações definidas pelo usuário forem atingidas, caso contrário, o terceiro estágio é realizado novamente.

O alinhamento final obtido é utilizado como entrada para a função de pontuação objetivo, que indica a qualidade final do alinhamento.

2.4.2 Função Objetivo

A função objetivo é a responsável pela medida da qualidade do alinhamento final obtido e, nesse caso, tem como entrada um determinado alinhamento e a sua pontuação como saída. O sistema de pontuação utilizado pelo MUSCLE é a de soma de pares, em que a pontuação objetiva final é obtida

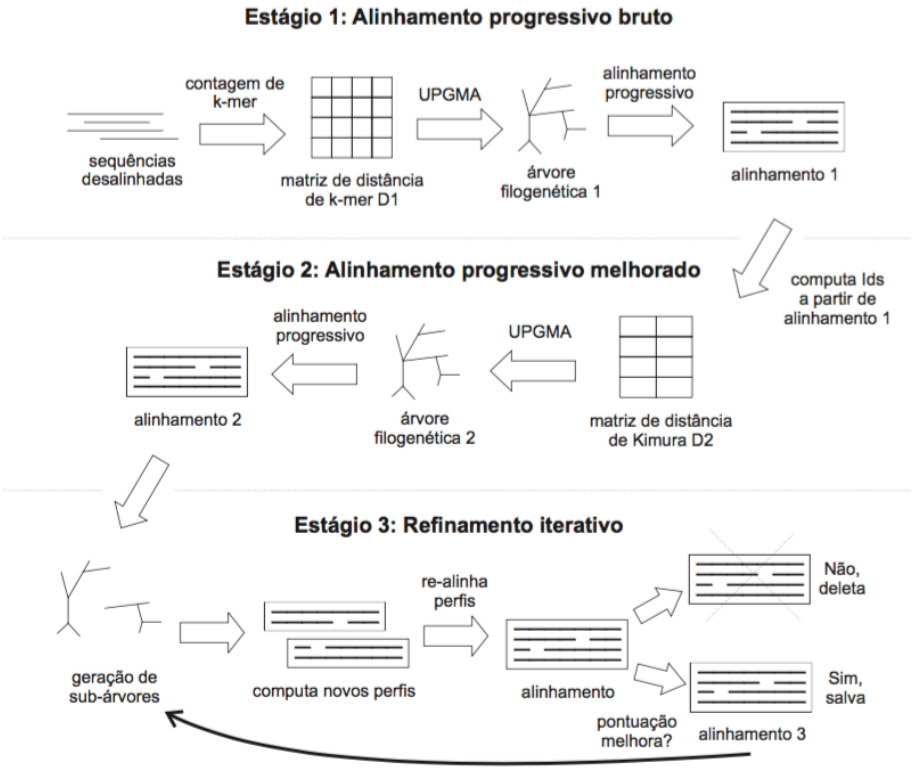

Figura 2.13: Etapas da ferramenta de alinhamento MUSCLE
Fonte: (Marucci, 2009) *Adaptado*

através da soma das pontuações de todas as inserções, remoções e substituições que ocorreram entre todos os pares possíveis de sequências alinhadas. Uma matriz de substituição e penalidades pelo uso de *gaps* é utilizada como base para essa pontuação.

As penalidades decorrentes para os *gaps* ocorre através do descarte da(s) coluna(s) em que ambas as sequências possuem *indels* (inserções ou deleções de elementos). Dessa forma, é aplicada a penalidade $g + \lambda e$ para cada *gap* observado, sendo que g é a penalidade por *gap*, e λ é o comprimento desse *gap*, e e é a penalidade de extensão.

Na fase de refinamento (no terceiro estágio) a pontuação objetiva é uti-

lizada, de forma que sempre que um novo alinhamento é realizado, este é comparado com o antecessor, sendo que prevalece sempre o alinhamento que possui a pontuação maior.

2.4.3 Contagem de K-mer

A contagem dos *k-mer* é uma etapa fundamental do primeiro estágio da ferramenta MUSCLE, o que a difere dos outros métodos de Alinhamento Múltiplo que utilizam outras técnicas para a obtenção da árvore-guia (Edgar, 2004b)(Vinga and Almeida, 2003). Conhecidas como *palavras* ou *k-tuplas*, o *k-mer* é uma subsequência contínua de comprimento k, cuja frequência esperada é mais comum em sequências relacionadas. O uso da contagem dessas *palavras* é motivada sobretudo pelo aumento de desempenho do processo, uma vez que não é necessário o uso do alinhamento par-a-par para a obtenção da matriz de distâncias.

A fórmula para o cálculo do índice de similaridade F através da contagem de *k-mer* é dada pela equação:

$$F = \sum_{\tau} min[n_x(\tau), n_y(\tau)]/[min(L_x, L_y) - k + 1] \tag{2.55}$$

em que τ representa um *k-mer*, L_x e L_y são os comprimentos das sequências, $n_x(\tau)$ e $n_y(\tau)$ é a quantidade de vezes que o *k-mer* τ aparece nas sequências X e Y, respectivamente, sendo que $(1-F)$ é considerada uma boa estimativa de distância . É importante destacar ainda que o uso de alfabetos diferentes produzem estimativas diferentes, de acordo com o comprimento o *k-mer*.

Um alfabeto comprimido C é uma subdivisão do alfabeto padrão A com-

posto pelas letras que representam os 20 aminoácidos em N classes disjuntas contendo grupos similares de aminoácidos. Diversos métodos para a construção desses alfabetos foram propostos (Li et al., 2003) de forma a observar as similaridades presentes na matriz de transição BLOSUM62.

Na tabela 2.4.3 estão destacados os alfabetos comprimidos usuais, de acordo com Edgar (2004a) e foram construídos de acordo com os estudos realizados por Dayhoff et al. (1983) ao relacionar aminoácidos próximos entre si, e que possuem relevância na estrutura e função da proteína.

Tabela 2.4: Alfabetos Comprimidos

Alfabeto	Classes
Dayhoff(6)	AGPST, C, DENQ, FWY, HKR, ILMV
SE-B(6)	AST, CP, DEHKNQR, FWY, G, ILMV
SE-B(8)	AST, C, DHN, EKQR, FWY, G, ILMV, P
Li-A(10)	AC, DE, FWY, G, HN, IV, KQR, LM, P, ST
Li-B(10)	AST, C, DEQ, FWY, G, HN, IV, KR, LM, P
Murphy(10)	A, C, DENQ, FWY, G, H, ILMV, KR, P, ST
SE-B(10)	AST, C, DN, EQ, FY, G, HW, ILMV, KR, P
SE-V(10)	AST, C, DEN, FY, G, H, ILMV, KQR, P, W
Solis-D(10)	AM, C, DNS, EKQR, F, GP, HT, IV, LY, W
Solis-G(10)	AEFIKLMQRVW, C, D, G, H, N, P, S, T, Y
SE-B(14)	A, C, D, EQ, FY, G, H, IV, KR, LM, N, P, ST, W

Os métodos para identificação de similaridade local através do uso de alfabetos comprimidos têm se mostrado promissores, de acordo com Pevzner and Shamir (2011); Edgar (2004a) sobretudo pela redução de tempo e da complexidade do alinhamento par-a-par, reduzindo a ordem da complexidade de $O(L^2)$ para $O(L)$, em sequências de comprimento L.

Capítulo 3

Desenvolvimento do Trabalho

3.1 Considerações Iniciais

Este capítulo apresenta as estratégias utilizadas para a implementação da proposta deste trabalho, cobrindo desde os algoritmos adotados para a otimização da ferramenta MUSCLE e seu acoplamento até a comparação dos resultados decorrentes dos alinhamentos através da nova ferramenta, e a avaliação dos resultados através do *benchmark* adequado.

3.2 Escopo e requisitos da ferramenta

Basicamente, a ferramenta MUSCLE é uma aplicação para ambientes *Windows*, *Linux* e *Machintosh* cuja finalidade é a produção de alinhamentos múltiplos de sequências biológicas, compostas por cadeias de aminoácidos ou nucleotídeos, oferencendo ao usuário um resultado com alta significância biológica. A manipulação dessa ferramenta é baseada no *prompt de comando*,

terminal ou *shell*, com os parâmetros padrões ou variações dos mesmos, de acordo com o propósito do alinhamento. Na figura 3.2 identificam-se os comandos da ferramenta no terminal do sistema operacional.

Através dos comandos é possível carregar um arquivo contendo as sequências desalinhadas no formato FASTA, e obter um novo arquivo em diversos formatos com o alinhamento realizado.

De acordo com o escopo da ferramenta, foram observados os seguintes requisitos funcionais:

1. Início do alinhamento: As sequências de entrada com os aminoácidos ou nucleotídeos devem ser fornecidas pelo usuário através da indicação da localização do arquivo-texto com as sequências obedecendo o padrão FASTA.
2. Ajuste dos parâmetros: O usuário tem a possibilidade de realizar as alterações necessárias, de acordo com os parâmetros ajustáveis na ferramenta: Localização das diagonais; Limitar a quantidade de iterações; Limitar a quantidade de horas de processamento; Formatar a saída para o padrão *html*, GCG ou ClustalW, entre outros.
3. Geração dos alinhamentos: O resultado do alinhamento gerado e disponibilizado ao usuário, que pode optar em gravar os dados nos formatos disponíveis pela ferramenta. O formato padrão é o FASTA.
4. Impressão e gravação do resultado: É possível imprimir os resultados contendo as sequências alinhadas, a pontuação obtida e o tempo de execução total.

Foram utilizados para a avaliação e comparação da ferramenta padrão MUSCLE e do algoritmo proposto por este trabalho o grupo de sequências

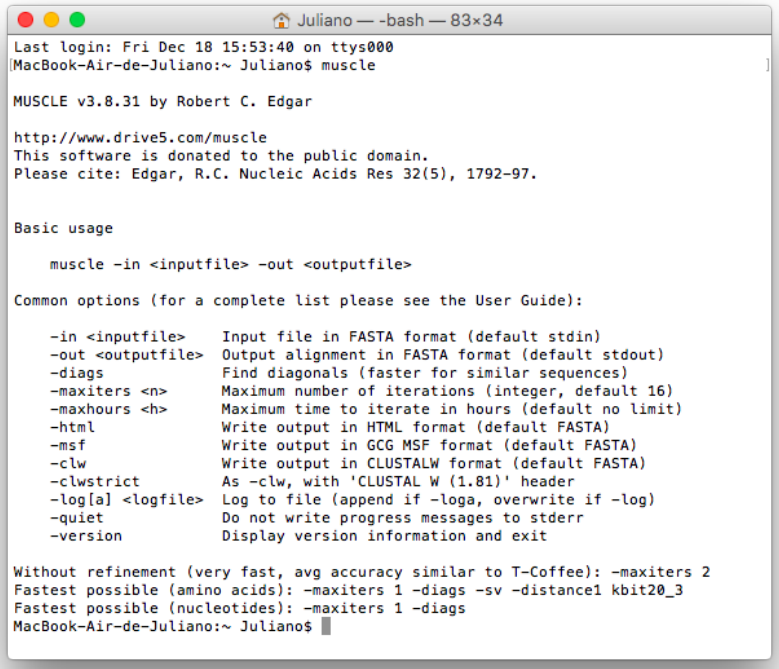

Figura 3.1: Principais comandos da ferramenta apresentadas no *shell*

fornecidos pelo *benchmark* BAliBASE que compreende basicamente 3 conjuntos baseados nos índices de similaridade, subdivididos em três categorias de tamanho: Sequências curtas, médias e longas.

Conjunto 1: Conjunto de referência com índice se similaridade menor que 25%.

Conjunto 2: Conjunto de referência com índice se similaridade entre 20% e 40%.

Conjunto 3: Conjunto de referência com índice se similaridade maior que 35%.

As categorias de tamanho das sequências, são divididas em três grupos,

onde as sequências pequenas possuem, em média 70 aminoácidos, as sequências médias possuem 250 aminoácidos e as sequências de comprimento longo possuem em média 400 aminoácidos.

3.3 Implementação do Algoritmo

A estratégia adotada neste trabalho consiste em utilizar uma técnica baseada nos modelos ocultos de Markov para otimizar a estimativa da distância entre os pares de sequências através da contagem de *k-mers* ou *tuplas* que são comuns entre todos os pares de sequências. Esse passo ocorre na segunda etapa do estágio 1, durante o alinhamento progressivo bruto.

A ferramenta MUSCLE obtém a medida de distância entre os pares de sequência através de técnicas determinísticas, realizando a contagem dos *k-mers* comuns, de acordo com a equação (2.55). A contagem de *k-mers* é realizada por meio de duas etapas, em que a primeira identifica a quantidade de palavras, ou *tuplas* comuns entre todos os pares de sequências, e a partir daí, obtém-se o índice de similaridade entre as sequências. Em seguida, por meio de transformadas específicas, é obtido a distância entre essas sequências, a partir dos dados obtidos na primeira etapa. É importante destacar que nessa primeira etapa, todos os caracteres, de todas as sequências são analisados, resultando em grande consumo de memória e processamento.

A modelagem e resolução desse problema através dos modelos de Markov é feita de acordo com a sistematização dos problemas canônicos apresentados na seção 2.3.2, em que três problemas fundamentais são apresentados, tendo em vista que inicialmente é necessário a definição dos parâmetros do modelo.

O problema 1, chamado de problema da avaliação, ou pontuação, que trata em obter a probabilidade de uma sequência de observáveis ter sido gerada por um determinado modelo tem sua solução através do passo *Forward* do algoritmo *Backward-Forward*.

O problema 2, onde se determina o caminho mais provável para a geração de uma sequência dada é solucionado por meio do algoritmo de Viterbi.

O problema 3, responsável pelo treinamento do modelo, onde ocorre a maximização da probabilidade de uma sequência de observáveis é solucionado pelo algoritmo de Baum-Welch.

De acordo com a figura 3.2, a contagem de *k-mer* na ferramenta MUSCLE ocorre no primeiro estágio, e recebe como entrada os pares de sequências e o tamanho k das tuplas a serem obtidas. A partir daí, o algoritmo realiza a leitura da sequência e constrói um vetor contendo todos os k-mers e sua respectiva frequência. A árvore filogenética é então obtida por meio da matriz de distâncias.

Na figura 3.3 ilustra-se o processo de organização dos k-mers de comprimento $k = 4$ e o processo de contagem, para posteriormente obtenção da matriz de distância.

A modelagem do problema é iniciada resolvendo-se inicialmente o algoritmo *Forward*, considerando o modelo $\lambda = (A, B, \pi)$, sendo A a distribuição da probabilidade de transição entre os estados, B é a distribuição da probabilidade de observações, e nesse caso, ao se utilizar o alfabeto padrão de 20 aminoácidos, é de 1/20 para cada probabilidade, e π, como distribuição de probabilidade inicial de $\pi = [1, 0, ..., 0]$. Verifica-se que é possível obter a probabilidade por meio dos passos de inicialização (2.14), indução (2.15)

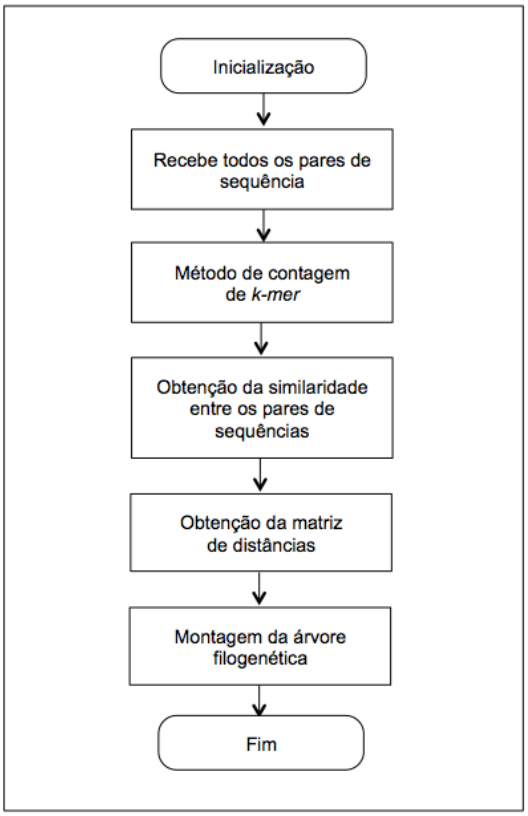

Figura 3.2: Fluxograma do método de contagem de k-mers

e finalização (2.16) através do pseudo-algoritmo 1, 2, 3 e 4, recebendo como parâmetros de entrada os observáveis, ou seja, o par de sequências.

O algoritmo 1, ilustra a aplicação da solução do problema 1:

A resolução do problema 1 identifica a probabilidade de se obter uma sequência dada, ou seja, os *k-mers* presentes no conjunto. No entanto, essa avaliação não fornece nenhum dado sobre a parte oculta do modelo, ou seja, a sequência geradoras dos *k-mers*. Assim, o problema 2 , cuja solução recorre ao algoritmo de Viterbi é resolvido através do algoritmo 3:

O algoritmo de Viterbi retorna a sequência ótima de estados associado à sequência de observáveis, restando apenas realizar o treinamento do algo-

Algorithm 1 Implementação do Algoritmo Forward

Require: $W = w_0, ..., w_{T-1}$
Require: k-mers $\{1, ..., M\} \rightarrow P(\{1, ..., N\})$
  $\alpha(0, 0) := 1.0$
  for $(1 \leq i \leq T - 1)$ **do**
    for ($t_i \in$k-mer(w_i)) **do**
      for ($t_{i-1} \in$k-mer(w_{i-1})) **do**
        $\alpha(t_i, i) := \alpha(t_i, i) + \alpha(t_{i-1}, i - 1).p_{tt}(t_i|t_{i-1}.p_{tw}(w_i|t_i))$
      end for
    end for
  end for
  $s := \alpha(0, T - 1)$
  return s

Algorithm 2 Implementação do Algoritmo Forward - Primeira Otimização

Require: $W = w_0, ..., w_{T-1}$
Require: k-mers $\{1, ..., M\} \rightarrow P(\{1, ..., N\})$
  $\alpha(0, 0) := 1.0$
  for $(1 \leq i \leq T - 1)$ **do**
    for ($t_i \in$k-mer(w_i)) **do**
      for ($t_{i-1} \in$k-mer(w_{i-1})) **do**
        $\alpha(t_i, i) := \alpha(t_i, i) + \alpha(t_{i-1}, i - 1).p_{tt}(t_i|t_{i-1})$
      end for
      $\alpha(t_i, i) := \alpha(t_i, i).p_{tw}(w_i|t_i)$
    end for
  end for
  $s := \alpha(0, T - 1)$
  return s

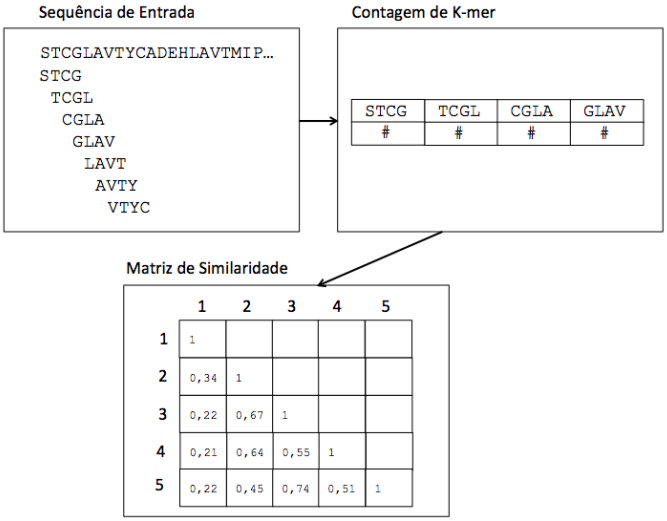

Figura 3.3: Contagem de k-mers e matriz de similaridade

ritmo, tarefa realizada pelo algoritmo Baum-Welch (representada pelo algoritmo 4, responsável pela maximização da probabilidade do modelo λ gerar a sequência completa de observáveis, ou seja,

$$\bar{\lambda} = argmax P(O|\lambda) \tag{3.1}$$

De acordo com as equações (2.54), o modelo converge se $\bar{\lambda} = \lambda$, e o processo é apresentado pelo algoritmo 4, (Baum-Welch).

A partir da aplicação do algoritmo Baum-Welch, o sistema passa a ser treinado pelas sequências de entrada, de acordo com o *k-mer* em questão.

O algoritmo padrão da ferramenta MUSCLE está disponível em linguagem nativa C, a qual foi mantida durante a implementação da heurística.

Algorithm 3 Implementação do Algoritmo de Viterbi

Entrada: Sequências de tamanho T
Saída: Caminho mais provável
Criação da matriz de probabilidade $[N+2, T]$
Criação do caminho de retorno $[N+2, T]$
for estados s de 1 a N **do**
  $forward[s,1] \leftarrow a_{0,s} x b_s(o_1)$
  ponto de retorno $[s,1] \leftarrow 0$
end for
for intervalo de tempo t, de 2 até T **do**
  for estado s de 1 a N **do**
    $viterbi[s,t] \leftarrow max_{s'=1}^{N} viterbi[s', t-1] \times a_{s',s} \times b_s(0_t)$
    ponto de retorno$[s,t] \leftarrow argmax_{s'=1}^{N} viterbi[s', t-1] \times a_{s',s}$
  end for
end for
$viterbi[q_F, T] \leftarrow max_{s=1}^{N}\ viterbi[s,T] \times a_{s,qF}$
ponto de retorno $[q_F, T] \leftarrow argmax_{s=1}^{N} viterbi[s,T] \times a_{s,qF}$

3.4 Técnicas de Benchmark

A necessidade da criação de técnicas que possam realizar a análise das diversas ferramentas de alinhamento múltiplo de sequências passa a ser necessária tendo em vista a grande variedade de cenários possíveis nas análises de bioinformática, e suas consequentes inferências.

Considerando os méritos e deficiências das diversas ferramentas de MSA (Multiple Sequence Alignment, ou Alinhamento Múltiplo de Sequências), a falta de um parâmetro para comparação de resultados de alinhamentos poderia gerar conclusões tendenciosas (Thompson et al., 1999), pois as bases de dados não forneciam informações classificatórias e estruturais para uma análise sistemática dos programas de alinhamento múltiplo.

Em decorrência desse problema, surgiram diversas ferramentas para medir a qualidade dos alinhamentos realizados pelos programas de MSA, gerando

Algorithm 4 Implementação do Algoritmo Baum-Welch

Entrada: $W = w_0, ..., w_{T-1}$
Entrada: k-mers $\{1, ..., M\} \rightarrow P(\{1, ..., N\})$
$s := \alpha(W, k - mer)$
$\beta(0, T-1) := 1.0$
for $(T - 1 \geq i \geq 0)$ **do**
  for $(t_i \in$ k-mer $(w_i))$ **do**
    $\delta'(t_i, w_i) := \delta'(t_i, w_i) + \frac{1}{s}.\alpha(i, t_i).\beta(i, t_i)$
    $\delta''(t_i) := \delta''(t_i) + \delta'(t_i, w_i)$
    for $(t_{i-1} \in$ k-mer $(w_{i-1}))$ **do**
      $p := p_{tt}(t_i|t_{i-1}).p_{tw}(w_i|t_i)$
      $\beta(t_{i-1}, i-1) := \beta(t_{i-1}, i-1) + p.\beta(t_i, i)$
      $\xi'(t_{i-1}, t_i) := \xi'(t_{i-1}, t_i) + \frac{1}{s}.\alpha(t_{i-1}, i-1).p.\beta(t_i, i)$
      $\xi''(t_{i-1}) := \xi''(t_{i-1}) + \xi'(t_{i-1}, t_i)$
    end for
  end for
end for
for $(t_i \in \{1, ..., N\})$ **do**
  for $(w \in \{1, ..., M\})$ **do**
    $p_{tw}(w|t_i) := \frac{\delta'(t_i, w)}{\delta''(t_i)}$
  end for
  for $(t_{i+1} \in \{1, ..., N\})$ **do**
    $p_{tt}(t_{i+1}|t_i) := \frac{\xi'(t_i, t_{i+1})}{\xi''(t_i)}$
  end for
end for

assim, um *benchmark*, dentre os quais, destacam-se o BAliBase (Bahr et al., 2001; Thompson et al., 1999, 2005), HOMSTRAD (Mizuguchi et al., 1998) e PREFAB (Edgar, 2004b).

Dentre os *benchmarks* mais utilizados para alinhamento de cadeias de proteínas, destaca-se o BAliBASE, que se encontra na sua terceira versão, e tem se tornado uma das medidas mais utilizadas para comparação (Essoussi et al., 2008; Hang, 2008). Nessa última versão, os alinhamentos foram obtidos computacionalmente, e depois foram refinados de forma manual.

Os conjuntos de referência do BAliBASE estão divididos em 6 grupos, representando diversas situações o qual um programa de MSA pode ser submetido, e cujas principais características estão apresentadas na tabela 3.1.

É importante destacar ainda que os alinhamentos do BAliBASE estão disponíveis de duas formas: compreendendo as sequências integrais ou truncados apenas com as regiões homólogas, sendo que as regiões denominadas *core blocks* possuem trechos altamente confiáveis para os critérios de avaliação que possuem anotações específicas.

Existem ainda duas métricas utilizadas para uma análise quantitativa dos alinhamentos gerados pelas ferramentas de alinhamento múltiplo de sequência em relação aos alinhamentos de referência do BAliBASE:

TC score (Total Columns): Indica o percentual de colunas do alinhamento de teste perfeitamente alinhados em relação ao alinhamento referência.

SP score (Sum of Pairs): Indica o percentual de pares de resíduos alinhados corretamente no alinhamento de teste quando comparado ao mesmo par de resíduos no alinhamento referência, em relação ao número total de pares de resíduos existentes no alinhamento referência.

Tabela 3.1: Grupos de referência do BAliBASE, versão 3.0

Referência	Descrição	Número de alinhamentos	Quantidade de Sequências
RV11	Sequências equidistantes com menos de 20% de identidade entre si e sem grandes inserções (> 35 resíduos)	38	265
RV12	Sequências equidistantes que compartilham entre 20 e 40% de identidade entre si, sem grandes inserções	45	411
RV20	Família de sequências que possuem mais de 40% de identidade, porém com uma sequência "orfã" com menos de 20% de identidade com qualquer outra sequência	41	1896
RV30	Alinhamento de sub-famílias, onde as sequências de uma mesma sub-família compartilham mais de 40% de identidade entre si, mas menos de 20% de identidade com qualquer sequência de outra sub-família	30	1882
RV40	Sequências que possuem mais de 20% de identidade entre si, porém com grandes extensões nas terminações N/C	48	1317
RV50	Sequências que possuem mais de 20% de identidade entre si, com grandes inserções	16	483

Capítulo 4

Testes e Resultados Obtidos

4.1 Considerações iniciais

Neste capítulo são apresentados os resultados obtidos de acordo com a proposta inicial e seus parâmetros, assim como a metodologia adequada para aferição dos alinhamentos obtidos, e o dispositivo utilizado nos testes de alinhamento de sequências. É apresentado também o conjunto de sequências escolhidas para os casos de testes.

4.2 Dispositivo de testes

Para a realização dos testes com os conjuntos de sequências do BaliBASE foi utilizado um Notebook Apple MacBook Air, com processador Corei5 Dual-Core, com Clock de 1,7 GHz, 4 GB de memória RAM com frequência 1333 MHz, disco rígido SSD sob o Sistema Operacional OSX 10.11.2. Este equipamento mostrou-se adequado para a execução das tarefas de alinhamento

que geralmente demandam por sistemas computacionais com bom poder de processamento. Os testes foram realizados no Laboratório de Bioinformática, localizado no Centro de Estudos Genômicos da Unesp, no Instituto de Biociências, Letras e Ciências Exatas de São José do Rio Preto.

4.3 Conjunto de testes

Com o objetivo de analisar a implementação proposta por este trabalho, foram utilizados os conjuntos de sequências de testes disponíveis do BaliBASE (Thompson et al., 2005; Bahr et al., 2001; Thompson et al., 1999). Esta ferramenta de *Benchmark* possui diversos grupos de sequências, divididos em categorias, com os respectivos alinhamentos de referência para comparação com os alinhamentos realizados através dos diversos algoritmos disponíveis de MSA. A comparação dos alinhamentos de referência e dos alinhamentos produzidos é realizado pelo BaliSCORE (Błazewicz et al., 2009), cuja pontuação obtida por essa ferramenta indica o nível se significância biológica desse alinhamento produzido. Esse índice varia de 0, que indica o pior alinhamento, e 1, o melhor alinhamento possível.

Nas tabelas 4.1, 4.2 e 4.3 estão apresentados os conjuntos de testes utilizados nesse trabalho, de acordo com o percentual de similaridade entre as sequências presentes no conjunto de testes da Referência 1, do BaliBASE. Cada grupo contém conjuntos aleatórios de sequências de tamanhos diferentes (curtos, médios e longos) e que representam funções distintas no organismo. Para este trabalho, foram organizados para cada conjunto, três sub-grupos de sequências, separadas por tamanho, totalizando 36 conjuntos.

Tabela 4.1: Grupo de sequências com similaridade menor que 25%

Conjunto 1 - Similaridade menor que 25%		
Conjunto	Função	Tamanho
16r9	repressor	curto
1ubi	ubiquitin	curto
1wit	twitchin	curto
2trx	thioredoxin	curto
1sbp	sulfate binding protein	médio
1uky	uridylate kinase	médio
2pia	phtalate reductase	médio
3grs	glutathione reductase	médio
1cpt	cytochrome p450	longo
1ped	alcohol dehydrogenase	longo
2myr	myrosinase	longo
gal4	gal4	longo

Tabela 4.2: Grupo de sequências com similaridade entre 20% e 40%

Conjunto 2 - Similaridade entre 20% e 40%		
Conjunto	Função	Tamanho
1aab	high mobility group protein	curto
1hpi	high-potential iron-sulfur protein	curto
1pfc	immunoglobulin PFc fragment	curto
3cyr	cytochrome c3	curto
1mrj	alpha tricosanthin	médio
1pii	anthranilate isomerase	médio
1ton	tonin	médio
2cba	anhydrase	médio
1bgl	b-galactoxidase	longo
1dlc	endotoxin	longo
1pkm	pyruvate kinase	longo
glg	glutamyl-trna synthetase	longo

Tabela 4.3: Grupo de sequências com similaridade maior que 35%

Conjunto 3 - Similaridade maior que 35%		
Conjunto	Função	Tamanho
1csp	cold shock protein	curto
1krn	serine protease	curto
2fxb	ferredoxin	curto
9rnt	ribonuclease	curto
1amk	triose phosphate isomerase	médio
1led	lectin	médio
1thm	serine protease	médio
1zin	adenylate kinase	médio
1gpb	glycogen phosphorylase b	longo
1lcf	lactoferrin	longo
1taq	taq DNA polymerase	longo
3pmg	phosphoglucomutase	longo

Como um dos objetivos deste trabalho é a implementação de uma técnica heurística para otimizar a ferramenta de alinhamento múltiplo de sequências através dos modelos de Markov, são avaliados, além da pontuação apresentada pela ferramenta BaliSCORE, que representa a significância biológica do resultado, o tempo de execução de cada alinhamento obtido, e compará-lo com a ferramenta padrão.

4.4 Testes de qualidade

Para os testes de qualidade e avaliação do algoritmo MUSCLE alterado com as instruções baseadas nos modelos de Markov e o algoritmo MUSCLE padrão foram utilizados os conjuntos de sequência descritos na seção 4.3, observando-se as respectivas similaridades (menor que 25%, entre 20 e 40% e maior que 35%).

É sabido que os algoritmos estocásticos para alinhamento múltiplo de sequências produzem resultados diferentes a cada execução, e de forma a obter resultados estatísticos relevantes e não-tendenciosos, foram executados cinco vezes cada conjunto de testes, sendo que o desvio-padrão dessas medições apresentaram convergência. A avaliação de um conjunto escolhido aleatoriamente: *2fxb* é apresentada mais detalhadamente, confirmando que o resultado converge ao se executar o algoritmo por cinco vezes consecutivas.

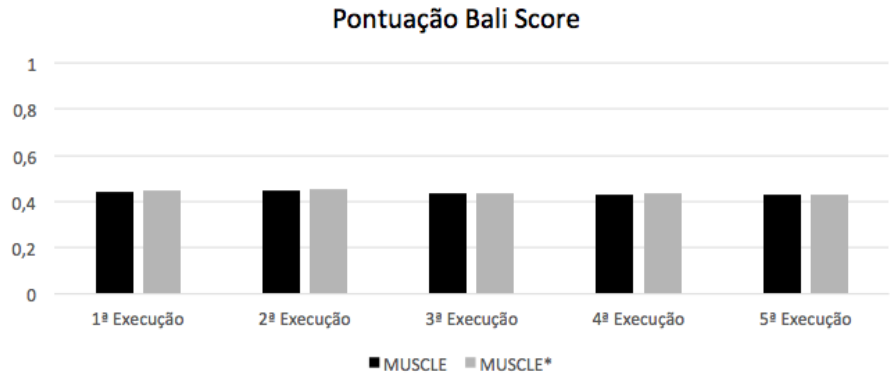

Figura 4.1: Pontuação BaliScore do conjunto *2fxb*

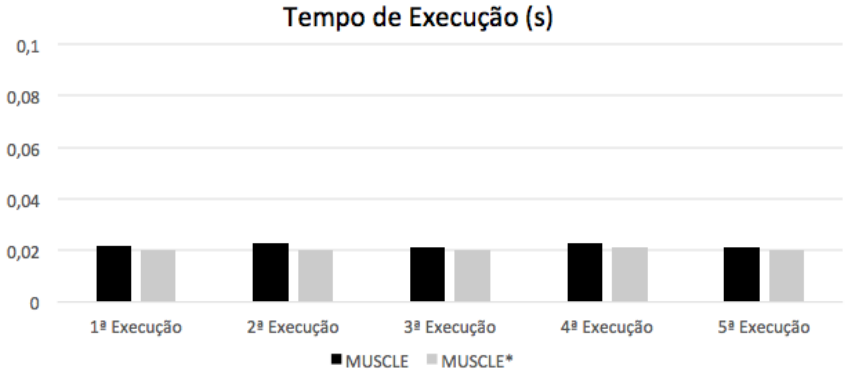

Figura 4.2: Tempo de execução do conjunto *2fxb*

A partir dos dados obtidos nos gráficos presentes nas figuras 4.1 e 4.2 obtemos um desvio-padrão de 0,00608 para a pontuação Bali Score do algoritmo padrão MUSCLE, e de 0.0092 para o algoritmo alterado. Da mesma forma, obtivemos um desvio-padrão de 0,0008 referente ao tempo de execução da ferramenta MUSCLE padrão, e de 0,00032 para o algoritmo alterado.

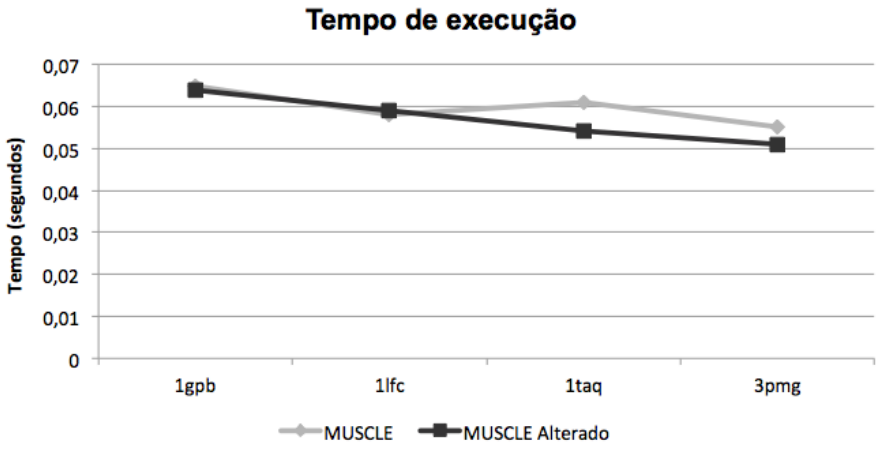

Figura 4.3: Gráfico comparativo - Tempo de execução (s)

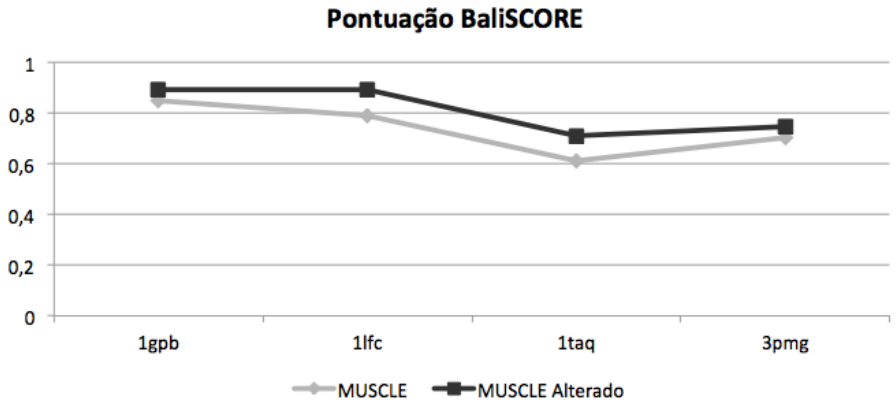

Figura 4.4: Gráfico comparativo - Pontuação BaliSCORE

Na figura 4.3 verificam-se as diferenças no tempo de execução do conjunto de sequências curtas com similaridade maior que 35%, e houve uma melhora significativa no tempo de execução do algoritmo alterado em até 11,5%, assim como na pontuação BaliSCORE em até 16,4%, de acordo com a figura 4.4. É importante ressaltar que o algoritmo alterado apresenta melhores resultados com sequências longas e com maior índice de similaridade.

A seguir, são apresentados os resultados obtidos com as execuções dos algoritmos, conforme as tabelas 4.4, 4.5 e 4.6.

Tabela 4.4: Execução do Conjunto 1 - Similaridade < 25%

Sequência	Comprimento	MUSCLE		MUSCLE*	
		Tempo (s)	Score (Bali)	Tempo (s)	Score (Bali)
1r69	curto	0,025	0,03531	0,022	0,1455
1ubi	curto	0,029	0,04535	0,021	0,03211
1wit	curto	0,031	0,3441	0,025	0,1881
2trx	curto	0,024	0,2114	0,024	0,1215
1sbp	médio	0,038	0,551	0,037	0,55
1uky	médio	0,037	0,555	0,035	0,551
2pia	médio	0,038	0,591	0,039	0,532
3grs	médio	0,036	0,495	0,035	0,505
1cpt	longo	0,065	0,501	0,066	0,499
1ped	longo	0,069	0,502	0,067	0,412
2myr	longo	0,061	0,491	0,061	0,380
gal4	longo	0,066	0,502	0,065	0,401

Tabela 4.5: Execução do Conjunto 2 - Similaridade entre 20% e 40%

Sequência	Comprimento	MUSCLE		MUSCLE*	
		Tempo (s)	Score (Bali)	Tempo (s)	Score (Bali)
1aab	curto	0,022	0,781	0,021	0,311
1hpi	curto	0,03	0,644	0,028	0,185
1pfc	curto	0,028	0,714	0,029	0,246
3cyr	curto	0,025	0,599	0,021	0,351
1mrj	médio	0,035	0,655	0,033	0,544
1pii	médio	0,033	0,641	0,033	0,651
1ton	médio	0,038	0,851	0,035	0,417
2cba	médio	0,032	0,501	0,035	0,495
1bgl	longo	0,061	0,785	0,051	0,810
1dlc	longo	0,065	0,841	0,051	0,799
1pkm	longo	0,066	0,745	0,049	0,821
glg	longo	0,064	0,791	0,054	0,851

Tabela 4.6: Execução do Conjunto 3 - Similaridade > 35%

Sequência	Comprimento	MUSCLE		MUSCLE*	
		Tempo (s)	Score (Bali)	Tempo (s)	Score (Bali)
1csp	curto	0,031	0,81	0,028	0,35
1krn	curto	0,038	0,74	0,031	0,284
2fxb	curto	0,028	0,43	0,02	0,44
9rnt	curto	0,025	0,841	0,028	0,321
1amk	médio	0,038	0,741	0,035	0,551
1led	médio	0,041	0,832	0,044	0,745
1thm	médio	0,033	0,733	0,041	0,781
1zin	médio	0,037	0,891	0,034	0,851
1gpb	longo	0,065	0,851	0,064	0,891
1lfc	longo	0,058	0,788	0,054	0,893
1taq	longo	0,061	0,611	0,054	0,711
3pmg	longo	0,055	0,705	0,051	0,748

São apresentados através das figuras 4.5, 4.6, 4.8, 4.7, 4.9 e 4.10 os gráficos de desempenho baseado na pontuação BaliSCORE e tempo de processamento.

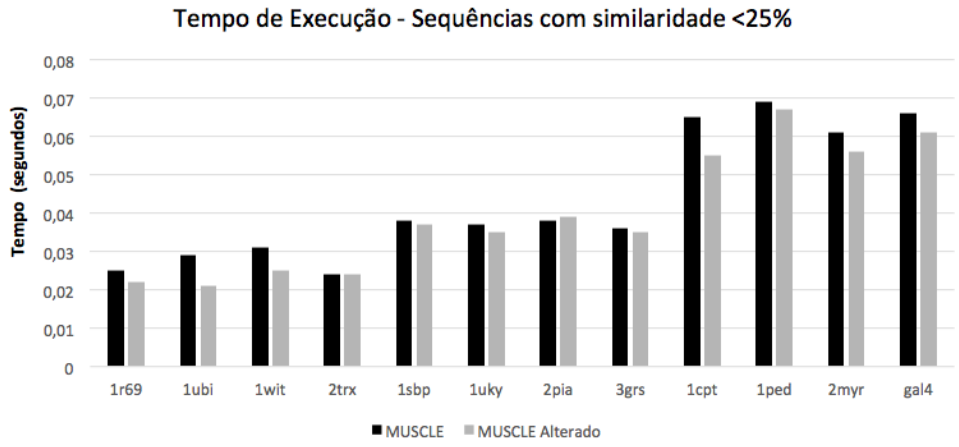

Figura 4.5: Tempo de execução - Similaridade < 25%

É possível identificar que através da figura 4.5, os tempos de execução foram menores em 83% dos casos para o algoritmo Alterado, com maior concentração nas sequências de maior comprimento.

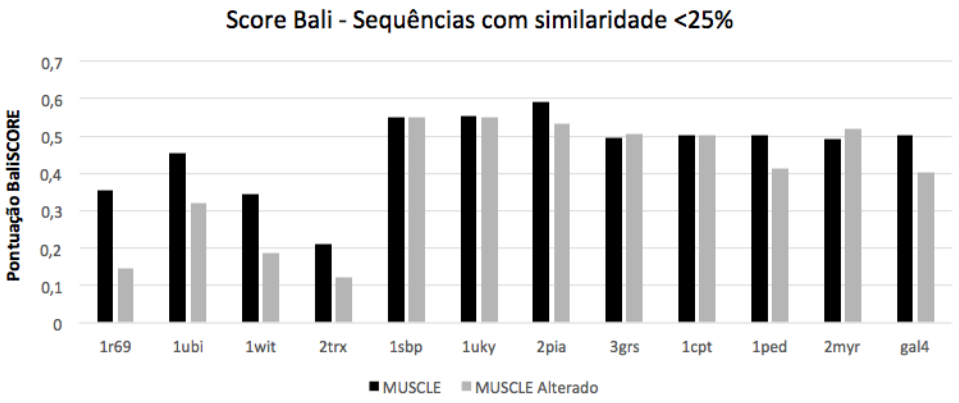

Figura 4.6: Pontuação Bali - Similaridade <25%

As sequências de comprimento mais curto apresentam resultados inferiores na média de 35%, de acordo com a figura 4.6, em decorrência do treinamento insuficiente do modelo, o que não ocorre com as sequências de

comprimento médio e longo, onde o algoritmo padrão e alterado diferem em uma média de 7%.

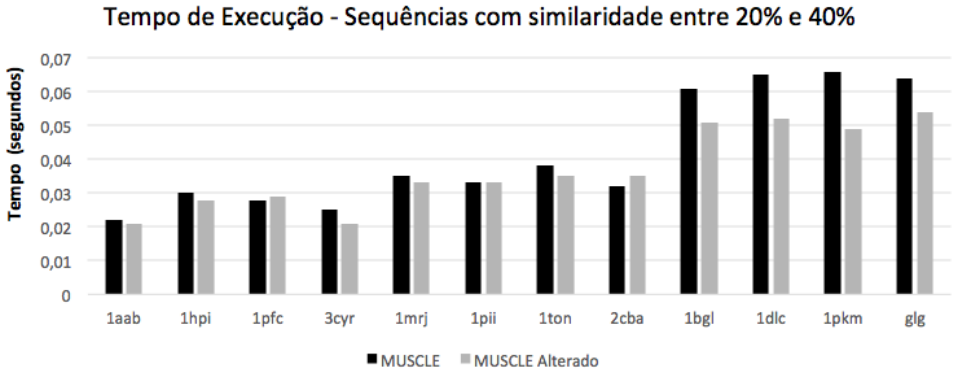

Figura 4.7: Tempo de execução - Similaridade entre 20% e 40%

O tempo de execução foi menor em 91% dos casos do conjunto de testes presentes na figura 4.7.

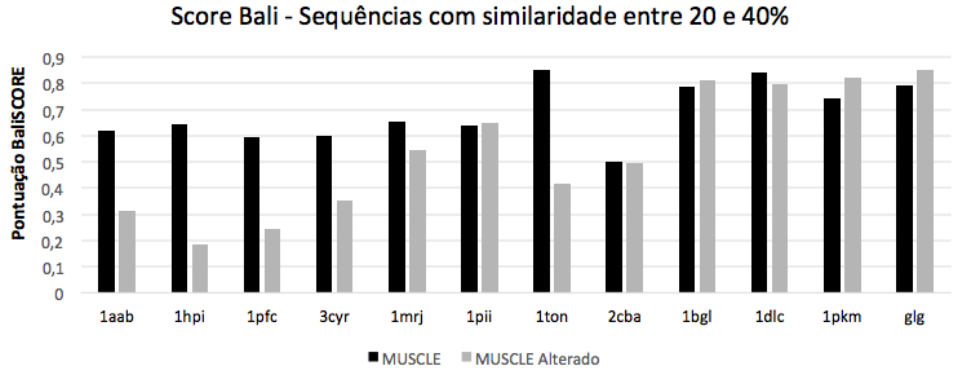

Figura 4.8: Pontuação Bali - Similaridade entre 20% e 40%

A pontuação BaliSCORE foi superior em relação ao algoritmo padrão em 75% dos casos nos conjuntos de sequências longas, reforçando que o treinamento do modelo é mais eficiente nessas condições, segundo a figura 4.8.

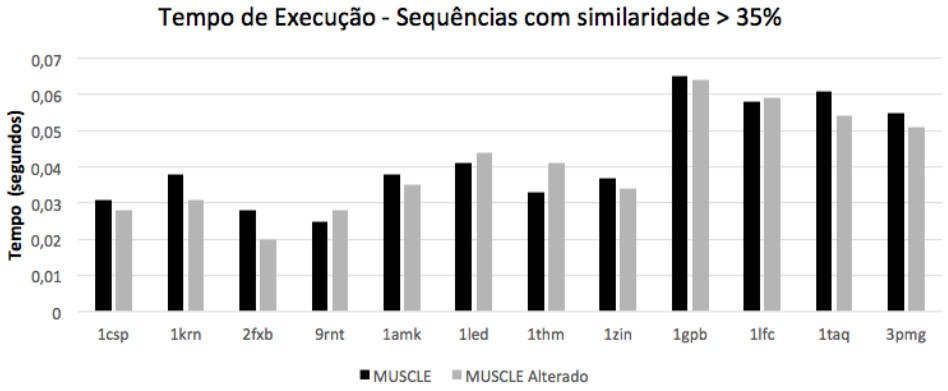

Figura 4.9: Tempo de execução - Similaridade >35%

Os tempos de execução foram inferiores em 50% dos casos de testes do conjunto com similaridade maior que 35%, com maior concentração nas sequências de menor e médio comprimento, de acordo com a figura 4.9.

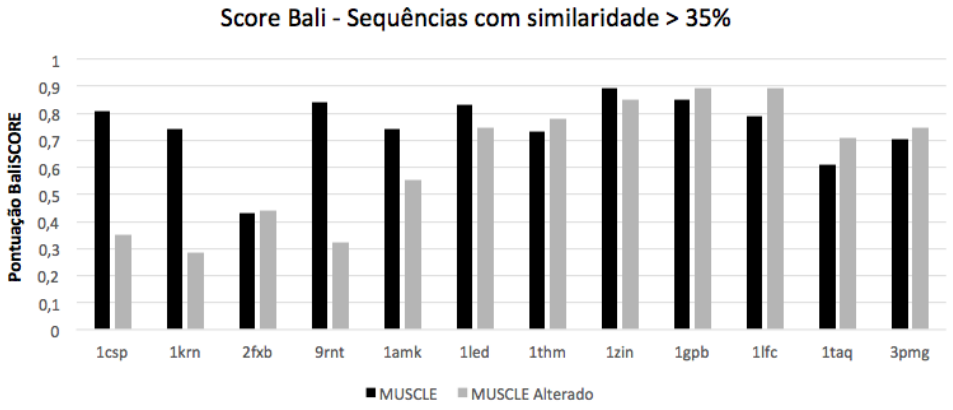

Figura 4.10: Pontuação Bali - Similaridade >35%

A pontuação BaliSCORE foi superior ao se utilizar o algoritmo alterado em 50% dos casos, com maior concentração nas sequências de comprimento médio e longo, de acordo com a figura 4.10, reforçando o fato de que sequên-

cias de comprimento maiores refletem em um treinamento mais preciso, e apresentando resultados superiores tanto em tempo de execução quanto no aspecto biológico através do score adequado para esse fim.

4.5 Considerações finais

Por meio da análise dos dados obtidos a partir da execução dos conjuntos de sequências, observa-se a capacidade de implementação dos modelos de Markov no algoritmos de alinhamento múltiplo de sequências MUSCLE. Nota-se, através dos resultados obtidos, que conforme aumenta-se a similaridade dos conjuntos de sequências, assim como o comprimento da mesma, o algoritmo apresenta melhores resultados tanto em tempo de execução quanto no contexto biológico, representado pela pontuação BaliSCORE. É importante ressaltar que dessa forma, o uso do algoritmo alterado não é adequado ao se alinhar sequências de comprimento curto, com baixa similaridade entre si.

Capítulo 5

Conclusões

5.1 Conclusões gerais

Neste trabalho foram apresentados os aspectos relevantes ao que se entende por bioinfomática, compreendendo sobretudo os conceitos elementares de biologia e genética, assim como as técnicas computacionais utilizadas como ferramentas de alinhamento de sequências. Foram descritos também os Modelos de Markov e suas aplicações na bioinformática e a ferramenta MUSCLE.

A utilização dos modelos de Markov para a otimização do processo de alinhamento mostrou-se efetiva e aplicável na ferramenta MUSCLE, otimizando o seu processo, e conforme apresentado na seção 4, com resultados significativos tanto em tempo de execução quanto na qualidade final do resultado tendo como parâmetro de qualidade a ferramenta BaliSCORE, descrita na seção 4.3. Nesse sentido, é importante destacar que as sequências consideradas curtas demais não oferecem as condições adequadas para um melhor treinamento do modelo, gerando resultados biologicamente incoerentes, e portanto,

passíveis de descarte. O fato que ocorre com as sequências curtas se deve a elas guardarem, em geral, uma quantidade menos relevante de informações, mas ao se considerar os processos estocásticos, podem haver exceções. No entanto, sequências médias e longas apresentaram resultados relevantes biologicamente, e com tempo de processamento inferior ao da ferramenta padrão, demonstrando a vantagem da utilização da nova técnica. Assim, conclui-se que os objetivos estabelecidos para o presente projeto foram cumpridos.

5.2 Trabalhos futuros

Como um dos trabalhos futuros, pretende-se investigar a utilização de alfabetos de aminoácidos alternativos, e quais deles passam a ser mais adequados para determinados grupos de sequências a serem alinhadas. Aliado a variabilidade de alfabetos, a aplicabilidade dos modelos de Markov em outros algoritmos de alinhamentos, sobretudo da família Clustal pode contribuir para a otimização da mesma.

Além disso, o desenvolvimento de uma interface multiplataforma como a linguagem JAVA, promovendo assim uma utilização mais simples e intuitiva pode contribuir muito para o biólogo, que ao invés de utilizar o modo texto dos alinhadores, passará a contar com essa facilidade, estimulando o seu uso.

Destaca-se ainda a possibilidade da aplicação de técnicas de paralelismo para uma melhoria de desempenho do algoritmo, através de metodologias híbridas que podem ser de grande utilidade em alinhamentos de sequências muito longas.

Referências Bibliográficas

Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., and Walter, P. (2010). *Biologia Molecular da Célula*. Artmed, 5 edition.

Almeida, A. A. M. (2013). *Novas abordagens para o problema do alinhamento múltiplo de sequências*. PhD thesis, Universidade Estadual de Campinas.

Altschul, S. F. (1990). A basic local alignment search tool. *Journal of Moleculare Biology*.

Amorim, A. R., Zafalon, G. F. D., Neves, L. A., Pinto, A. R., Valencio, C. R., and Machado, J. M. (2015). Improvements in the sensibility of msa-ga tool using coffee objective function. *Journal of Physics*.

Bahr, A., Thompson, J. D., Thierry, J. C., and Poch, O. (2001). Balibase (benchmark alignment database): enhancements for repeats, transmembrane sequences and circular permutations. *Nucleic Acids Research*, 29.

Bell, T. C., Cleary, J. G., and H., W. I. (1990). *Text compression*. Prentice Hall.

Błazewicz, J., Formanowicz, P., and Wojciechowski, P. (2009). Some remarks on evaluating the quality of the multiple sequence alignment based on the

balibase benchmark. *International Journal of Applied Mathematics and Computer Science*, 19(4).

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3).

Boratyn, G. M., Camacho, C., Cooper, P. S., Coulouris, G., Fong, A., Ma, N., Madden, T. L., Matten, W. T., McGinnis, S. D., Merezhuk, Y., et al. (2013). Blast: a more efficient report with usability improvements. *Nucleic Acids Research*.

Cohen, J. (2001). Bioinformatics - an introduction for computer scientists. *ACM Computer Surveys (CSUR)*, 36(2).

Collins, F. S., Patrinos, A., Jordan, E., Chakravarti, A., Gesteland, R., and Walters, L. (1998). New goals for the u.s. human genome - project: 1998–2003. *Science Magazine*, 282.

Combs, W., Hawkins, R., Pore, T., Schechet, A., Wahls, T., and Ziantz, L. (2005). The course scheduling problem as a source of student projects. *SIGCSE technical symposium on Computer Science Education*, 37(1).

Dayhoff, M. O., Barker, W. C., and Hunt, L. C. (1983). Establishing homologies in protein sequences. *Methods in Enzymology*, 91.

Edgar, R. C. (2004a). Local homology recognition and distance measures in linear time using compressed amino acid alphabets. *Nucleic Acids Research*, 32(1).

Edgar, R. C. (2004b). Muscle: A multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*.

Essoussi, N., Boujenfa, K., and Limam, M. (2008). A comparison of msa tools. *Bioinformation*, 2.

Ewens, W. J. and Grant, G. R. (2005). *Statistical Methods in Bioinformatics: An Introduction*. Springer.

Filho, J. S. R. (2009). Next-generation sequencing. *Breast Cancer Research*, 11.

Forney, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, 61(3).

Garcia, E. L. and Araiza, L. M. G. (2012). Simulated annealing with previous solutions applied to dna sequence alignment. *ISRN Artificial Intelligence*.

Griffiths, A. J. F., Wessler, S. R., Lewontin, R. C., and Carroll, S. B. (2013). *Introdução à Genética*. Guanabara Koogan.

Hall, B. K. (2012). *Evolutionary Developmental Biology*. Springer Science and Business Media.

Hang, N. T. (2008). *Comparison of multiple sequence alignment programs in practise*. PhD thesis, University of Århus.

Henikoff, S. and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22).

Jelinek, F. (1998). Statistical methods for speech recognition. *The MIT Press*.

Karplus, K. (2009). Sam-t08, hmm-based protein structure prediction. *Nucleic acids research*, 37.

Kashiwabara, A. Y. (2011). Myop/tops/sgeval: Um ambiente computacional para estudo sistemático de predição de genes. Master's thesis, Universidade de São Paulo.

Kimura, M. and Ohta, T. (1972). On the stochastic model for estimation of mutational distance between homologus proteins. *Journal of Molecular Evolution*, 2.

Koski, T. (2001). *Hidden Markov Models for Bioinformatics*. Kluwer Academic Publishers.

Largo, A. R., Rodríguez, M. A. V., and Álvarez, D. L. G. (2016). Hybrid multiobjective artificial bee colony for multiple sequence alignment. *Applied Soft Computing*, 41.

Lemos, M., Aragao, M. V. S. P., and Casanova, M. A. (2003). Padrões em biossequências. *PUC-Rio*.

Li, T., Fan, K., Wang, J., and Wang, W. (2003). Reduction of protein sequence complexity by residue grouping. *Protein Engineering*, 16.

Liew, A. W. C., Yan, H., and Yang, M. (2001). Pattern recognition techniques for the emerging field of bioinformatics: A review. *Pattern Recognition*, 38(11).

Lipman, D. J. and Pearson, W. R. (1985). Rapid and sensitive protein similarity search. *Science*.

Marucci, E. A. (2009). Paralelização da ferramenta de alinhamento de sequências muscle para um ambiente distribuído. Master's thesis, Universidade Estadual Paulista "Júlio de Mesquita Filho- Unesp.

Mimouni, N., Lunter, G., and Deane, C. (2004). Hidden markov models for protein sequence alignment. *University of Oxford*.

Mizuguchi, K., Deane, C. M., Blundell, T. L., and Overington, J. P. (1998). Homstrad: a database of protein structure alignments for homologous families. *Protein science: a publication of the Protein Society*, 7.

Mulia, S., Mishra, D., and Jena, T. (2012). Profile hmm based multiple sequence alignment for dna sequences. *Procedia Engineering*, 38.

Naruya, S. and Nei, M. (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425.

Nascimento, M. (2009). O uso de simulação de monte carlo via cadeias de markov no melhoramento genético. Master's thesis, Universidade Federal de Viçosa.

Needleman, S. and Wunsh, C. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Moleculare Biology*, 48(3):443–53.

Ogata, A. K. O. (2007). Multialinhamento de sequências biológicas utilizando algoritmos genéticos. Master's thesis, USP - Universidade de São Paulo.

Olazar, M. R. R. (2007). Algoritmos evolucionários multiobjetivo para alinhamento múltiplo de sequências biológicas. Master's thesis, Universidade Federal do Rio de Janeiro.

Ortuño, F., Venezuela, O., Pomares, H., and Rojas, I. (2013). Determining the most suitable multiple sequence alignment methodology by using a set of heterogeneous biological features. *Progressive - International Work-Conference on Bioinformatics and Biomedical Engineering*, 2.

Pais, F. S. M., Ruy, P. C., Oliveira, G., and Coimbra, R. S. (2014). Assessing the efficiency of multiple sequence alignment programs. *Algorithms for Molecular Biology*.

Pardoux, E. (2008). *Markov Processes and Applications*. Wiley.

Passos, G. A. S. and Jordan, C. N. B. (2000). Projeto transcriptoma: Análise da expressão gênica em larga escala usando dna - arrays. Biotecnologia, Ciência e Desenvolvimento.

Pei, J. and Grishin, N. V. (2006). Mummals: multiple sequence alignment improved by using hidden markov models with local structural information. *Nucleic Acids Research*, 34.

Pevzner, P. and Shamir, R. (2011). *Bioinformatics for Biologists*. Cambridge.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2).

Rech, D. H. and Pilatti, R. (2004). 992align - uma ferramenta para alinhamento múltiplo de sequências de dna e proteínas. Master's thesis, Universidade Federal de Santa Catarina.

Riaz, T., Y., W., and Li, K. B. (2001). Multiple sequence alignment using tabu search. *APBC'04 Proceedings of the second conference on Asia-Pacific bioinformatics*, 29.

Rouchka, E. C. (2006). Aligning dna sequences using dynamic programming. *ACM Student Magazine*, 3(1).

Schatz, M. C., Langmead, B., and Salzberg, S. L. (2010). Cloud computing and the dna data race. *Nature biotechnology*, 28.

Sergio, D. (2008). *Análise e compressão de sequências genómicas*. PhD thesis, Instituto Politécnico de Bragança.

Sharma, K. R. (2009). *Sequence Alignment and Markov Models*. McGrawHill.

Sievers, F. and Higgins, D. G. (2013). Clustal omega, accurate alignment of very large numbers of sequences. *Methods in Molecular Biology*.

Simossis, V., Kleinjung, J., and Heringa, J. (2003). An overview of multiple sequence alignment. *Current Protocols in Bioinformatics*.

Smith, T. and Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Moleculare Biology*, 147:195–197.

Söding, J. (2005). Protein homology detection by hmm-hmm comparison. *Bioinformatics*, 21(7).

Souza, A. L. (2010). Alinhamento múltiplo de sequência de proteínas. Master's thesis, Universidade Estadual de Campinas.

Souza, R. G. (2014). Alinhamento múltiplo de sequências utilizando otimização dialética. Master's thesis, UFP - Universidade Federal de Pernambuco.

Sun, J., Palade, V., Wu, X., and Fang, W. (2014). Multiple sequence alignment with hidden markov models learned by random drift particle swarm optimization. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 11(1).

Sun, J., Wu, X., Fang, W., Ding, Y., Long, H., and Xu, W. (2012). Multiple sequence alignment using the hidden markov model trained by an improved quantum-behaved particle swarm optimization. *Information Sciences*, 182(1).

Tajara, E. H., Barbosa, E. B., Vidotto, A., Polachini, G. M., Henrique, T., and T., M. A. B. (2012). Proteômica: metodologias e aplicações no estudo de doenças humanas. *Elsevier*.

Thompson, J. D., Koehl, P., Ripp, R., and Poch, O. (2005). Balibase 3.0: Latest developments of the multiple sequence alignment benchmark. *PROTEINS: Structure, Function, and Bioinformatics*, 61.

Thompson, J. D., Plewniak, F., and Poch, O. (1999). Balibase: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, 15.

Verli, H. et al. (2014). Bioinformática: Da biologia à flexibilidade molecular. *Porto Alegre, Brasil*, 1.

Vinga, S. and Almeida, J. (2003). Alignment-free sequence comparison - a review. *Bioinformatics*, 19(4).

Wang, Z., Gerstein, M., and Snyder, M. (2010). Rna-seq: a revolutionary tool for transcriptomics. *Nat Rev Genet*, 10.

Wu, D., Rice, C. M., and Wang, X. (2012). Cancer bioinformatics: A new approach to systems clinical medicine. *BMC Bioinformatics*, 13(71).

Ye, S. Q. (2008). *Bioinformatics - A practical approach*. Chapman and Hall / CRC.

Zafalon, G. F. D. (2009). Algoritmos de alinhamento múltiplo e técnicas de otimização para esses algoritmos utilizando ant colony. Master's thesis, Universidade Estadual Paulista "Júlio de Mesquita Filho- Unesp.

Zafalon, G. F. D. (2012). *Implementação de algoritmos de alinhamemto múltiplo de sequências em ambientes Grid*. PhD thesis, Escola Politécnica da Universidade de São Paulo.

Zafalon, G. F. D., Visotaky, J. M. V., Amorim, A. R., Valencio, C. R., Neves, L. A., Souza, R. C. G., and Machado, J. M. (2015). A parallel approach of coffee objective function to multiple sequence alignment. *Mathematical Modeling in Physical Sciences*.

Zaha, A., Ferreira, H. B., and P., P. L. M. (2014). *Biologia Molecular Básica*. Artmed.