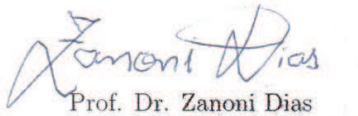
Alinhamento Múltiplo Progressivo de Sequências de Proteínas

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Maria Angélica Lopes de Souza e aprovada pela Banca Examinadora.

Campinas, 23 de Setembro de 2010.



Prof. Dr. Zanoni Dias
Instituto de Computação - UNICAMP
(Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Maria Fabiana Bezerra Müller – CRB8 / 6162

Souza, Maria Angélica Lopes de
So89a Alinhamento múltiplo progressivo de sequências de proteínas/Maria Angélica Lopes de Souza-- Campinas, [S.P. : s.n.], 2010.

Orientador : Zanoni Dias.
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1.Bioinformática. 2.Alinhamento progressivo. 3.Alinhamento múltiplo de sequências . I. Dias, Zanoni. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Título em inglês: Progressive multiple alignment of protein sequences

Palavras-chave em inglês (Keywords): 1. Bio-informatics. 2. Progressive alignment.
3. Multiple sequence alignments.

Área de concentração: Bioinformática

Titulação: Mestre em Ciência da Computação

Banca examinadora: Prof. Dr. Zanoni Dias (IC – UNICAMP)
Prof. Dr. Guilherme Pimentel Telles (IC – UNICAMP)
Prof. Dr. Nalvo Franco Almeida Junior (Faculdade de Computação-UFMS)

Data da defesa: 23/07/2010

Programa de Pós-Graduação: Mestrado em Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 23 de julho de 2010, pela Banca examinadora composta pelos Professores Doutores:

Prof. Dr. Nalvo Franco de Almeida Junior
Faculdade de Computação / UFMS

Prof. Dr. Guilherme Pimentel Telles
IC / UNICAMP

Prof. Dr. Zanoni Dias
IC / UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

Alinhamento Múltiplo Progressivo de Sequências de Proteínas

Maria Angélica Lopes de Souza[1]

Setembro de 2010

Banca Examinadora:

- Prof. Dr. Zanoni Dias (Orientador)
- Prof. Dr. Nalvo Franco Almeida Junior
- Prof. Dr. Guilherme Pimentel Telles
- Prof. Dr. João Meidanis (Suplente)
- Prof. Dra. Maria Emilia Machado Telles Walter (Suplente)

[1]Bolsa do CNPq

Resumo

O alinhamento múltiplo de sequências é uma tarefa de grande relevância em Bioinformática. Através dele é possível estudar eventos evolucionários e restrições estruturais ou funcionais, sejam de sequências de proteína, DNA ou RNA, tornando possível entender a estrutura, função e evolução dos genes que compõem um organismo.

O objetivo do alinhamento múltiplo é a melhor representação do cenário de evolução das sequências ao longo do tempo, considerando a possibilidade de ocorrerem diferentes eventos de mutação.

Encontrar um alinhamento múltiplo de sequências ótimo é um problema NP-Difícil. Desta forma, diversas abordagens têm sido desenvolvidas no intuito de encontrar uma solução heurística que represente da melhor maneira possível o cenário de evolução real, dentre elas está a abordagem progressiva.

O alinhamento progressivo é uma das maneiras mais simples de se realizar o alinhamento múltiplo, pois utiliza pouco tempo e memória computacional. Ele é realizado em três etapas principais: determinar a distância entre as sequências que serão alinhadas, construir uma árvore guia a partir das distâncias e finalmente construir o alinhamento múltiplo.

Este trabalho foi desenvolvido a partir do estudo de diferentes métodos para realizar cada etapa de um alinhamento progressivo. Foram construídos 342 alinhadores resultantes da combinação dos métodos estudados. Os parâmetros de entrada adequados para a maioria dos alinhadores foram determinados por estudos empíricos.

Após a definição dos parâmetros adequados para cada tipo de alinhador, foram realizados testes com dois subconjuntos de referência do BAliBASE. Com esses testes observamos que os melhores alinhadores foram aqueles que utilizam o agrupamento de perfil para gerar o alinhamento múltiplo, com destaque para os que utilizam pontuação afim para penalizar buracos. Observamos também, que dentre os alinhadores de agrupamento por consenso, os que utilizam função logarítmica para penalizar buracos demonstraram melhores desempenhos.

Abstract

The multiple sequence alignment is a relevant task in Bioinformatics. Using this technique is possible to study evolutionary events and also structural or functional restrictions of protein, DNA, or RNA sequences. This study helps the understanding of the structure, function, and evolution of the genes that make up an organism.

The multiple sequence alignment tries to achieve the best representation of a sequence evolution scenario, considering different mutation events occurrence.

Finding an optimal multiple sequence alignment is a NP-Hard problem. Thus, several approaches have been developed in order to find an heuristic solution that represents the real evolution cenario, such as the progressive approach.

The progressive alignment is a simple way to perform the multiple alignment, because its low memory usage and computational time. It is performed in three main stages: (i) determining the distance between the sequences to be aligned, (ii) constructing a guide tree from the distances and finally (iii) building the multiple alignment guided by the tree.

This work studied different methods for performing each step of progressive alignment and 342 aligners were built combining these methods. The input parameters suitable for most aligners were determined by empirical studies.

After the parameters definition for each type of aligner, which where tested against two reference subsets of BAliBASE. The test results showed that the best aligners were those using the profile alignment to generate the multiple alignment, especially those using affine gap penalty function. In addition, this work shows that among the aligners of grouping by consensus, those that use the logarithmic gap penalty function presented better performance.

Agradecimentos

Gostaria de agradecer a Deus, pois sem ele certamente a caminhada para estar aqui hoje teria sido muito mais árdua. Agradecer imensamente aos meus pais, Artur e Maria do Carmo, e à minha irmã, Vanessa, por sempre me apoiarem não só no desenvolvimento deste trabalho, mas em todas as escolhas que me fizeram chegar até aqui.

Agradeço ao meu orientador, professor Zanoni Dias, pela paciência durante a orientação e por sempre encontrar um tempinho, em meio aos seus inúmeros afazeres, para dedicar-se a este trabalho. Ao André Atanasio pela amizade e momentos de estudo divididos. Ao Felipe Rodrigues da Silva pelo suporte com os conceitos de Biologia.

Não poderia deixar de agradecer aos amigos com quem convivi nesses anos de mestrado, principalmente os amigos *bubbles*: Cristianno, Douglas, Faveri, Guilherme, Isaura, Marcos, Thiago, Tripodi e Willian. Agradecer especialmente aos amigos Daniel Vecchiato, ao Fábio Pessoa, à Vanessa Baptista e ao Robson Peixoto por sempre estarem por perto para ouvirem meus desabafos e pelas dicas com a escrita deste texto. E aos colegas de trabalho nesses últimos meses na Scylla Bioinformática, em especial ao professor João Meidanis, pela confiança em mim depositada.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

No século XIX Mendel descreveu os princípios básicos da herança genética. No início do século XX Thomas Morgan comprovou que os cromossomos são compostos por genes, unidades fundamentais para a hereditariedade. Durante todo o século XX diversos avanços foram feitos nas áreas da genômica e proteômica. Esses avanços estão relacionados ao aumento na capacidade de processamento dos computadores, pois um volume cada vez maior de dados biológicos tem sido produzido.

Grande parte dos dados biológicos é produzida por projetos de sequenciamento. Sequenciamento consiste em determinar as sequências de nucleotídeos (formados por um base nitrogenada, um grupo fosfato e uma pentose) que fazem parte de um trecho de DNA. Na década de 1970 foram desenvolvidos os primeiros métodos de sequenciamento, o mais tradicional deles é o da Terminação de Cadeia [62], ele produz sequências de até 1000 bases. Na última década novos métodos de sequenciamento tem se destacado como alternativos ao da Terminação de Cadeia, como o pirossequenciamento [57–59, 65], que produz uma quantidade maior de sequências que o método tradicionalmente usado. No entanto o tamanho das sequências produzidas é menor, aproximadamente 300 bases.

De posse das sequências de bases que compõem o DNA é possível determinar os genes que fazem parte do genoma e consequentemente as proteínas que são expressas por estes genes. O entendimento da estrutura, função e evolução dos genes que fazem parte de um organismo é um dos objetivos dos projetos de sequenciamento. O alinhamento de sequências de RNA, DNA e proteínas é tarefa fundamental para este entendimento, pois através do alinhamento é possível estudar como se deu a evolução das sequências alinhadas e também as restrições estruturais das mesmas.

Métodos para alinhar duas sequências, pelo alinhamento de pares, ou conjuntos de sequências, pelo Alinhamento Múltiplo de Sequências (MSA, do inglês *Multiple Sequence Alignment*), têm sido desenvolvidos por pesquisadores de Bioinformática. A Bioinformática objetiva gerar ferramentas que possam auxiliar na organização, processamento e visu-

alização dos dados gerados pelas pesquisas biológicas, combinando diferentes áreas do conhecimento como a Biologia e a Computação.

Alinhar duas sequências consiste na inserção de espaços nas sequências de modo que elas fiquem do mesmo tamanho, e que seja possível sobrepô-las para a comparação de bases [63]. Em 1970, Needleman e Wunsch [48] desenvolveram o primeiro algoritmo para alinhar pares de sequências. Este algoritmo utiliza programação dinâmica para alinhar as sequências de forma global, ou seja, as duas sequências por completo. No ano de 1981, Smith e Waterman [64] descreveram um algoritmo para alinhar localmente pares de sequências. No alinhamento local alinham-se subsequências que correspondem a regiões conservadas das sequências de entrada.

Quando o objetivo é observar as características de famílias de proteínas, o alinhamento de pares não é suficiente. É preciso realizar o alinhamento de várias sequências simultaneamente através do MSA. A generalização do algoritmo de Needleman e Wunsch [48] fica limitada a um número pequeno de sequências e de tamanho reduzido devido ao alto custo de tempo e espaço necessário para que o algoritmo seja executado.

Algoritmos que realizam o alinhamento múltiplo de sequências e que buscam representar o processo de evolução das sequências a serem alinhadas, em especial, de proteínas, envolvem o uso de diversas heurísticas. Abordagens distintas como: progressiva, iterativa, baseada em consistência, baseada em modelos e em blocos podem ser utilizadas no MSA.

O foco deste trabalho está nos alinhadores progressivos. Seu principal objetivo é estudar os métodos que podem ser utilizados em cada etapa do alinhamento progressivo, assim como o desempenho de cada combinação possível de métodos.

O alinhamento progressivo é realizado em três etapas principais: determinação da matriz de distância para cada par de sequências, construção de uma árvore filogenética a partir da matriz de distâncias e a construção do alinhamento múltiplo.

Podemos citar como principais contribuições desta dissertação:

- Estudo de métodos para realização de cada etapa do alinhamento progressivo.
- Implementação de algoritmos para a construção de um alinhamento múltiplo.
- Estudos e determinação dos parâmetros a serem utilizados por um conjunto de alinhadores.
- Avaliação e comparação dos métodos propostos entre si.

O restante deste trabalho está organizado da seguinte forma. No Capítulo 2 são apresentados os conceitos de Biologia necessários para auxiliar no entendimento do trabalho. No Capítulo 3 são descritos os métodos básicos para o alinhamento de sequências. A abordagem progressiva e cada um dos métodos utilizados na construção dos alinhadores

são descritos no Capítulo 4. O Capítulo 5 mostra os alinhadores implementados e como realizou-se o estudo para determinar os parâmetros dos mesmos. Os resultados do trabalho são descritos no Capítulo 6. Finalmente, no Capítulo 7 são apresentadas as conclusões do trabalho, assim como propostas de trabalhos futuros.

Capítulo 2

Conceitos Básicos

Neste capítulo serão apresentados alguns conceitos usados em Bioinformática que são importantes para o entendimento deste trabalho.

2.1 Genômica

Genômica é a área da biologia que estuda o genoma de diferentes organismos. Todos os organismos vivos apresentam informações que constroem seu material genético, estas constituem o genótipo do organismo e são transmitidas entre diferentes gerações por meio da reprodução. O genótipo de um organismo aliado ao meio em que vive determina o seu fenótipo.

2.1.1 Mendel

O mistério sobre a aparência, fenótipo, da maioria dos organismos foi desvendado por Gregor Mendel no ano de 1865. Naquele ano ele descreveu os princípios da herança genética [42].

As unidades fundamentais da hereditariedade são chamadas de genes e podem assumir várias formas diferentes. Cada forma assumida por um gene é denominada alelo.

Em seus experimentos Mendel utilizou ervilhas (por serem de fácil cultivo, curto ciclo de vida e possuir características em pares bem definidas) e notou que apesar de um pé de ervilha mostrar apenas um fenótipo, ele possuía dois alelos para cada gene. Os dois alelos de um gene podem ser iguais para uma característica, neste caso o gene é denominado homozigoto para essa característica. Quando os alelos de um gene são diferentes para uma determinada característica, ele é denominado heterozigoto para tal característica.

Um gene homozigoto pode expressar apenas um fenótipo para a característica que representa. Enquanto que um gene heterozigoto pode expressar um dos dois fenótipos

representados por seus alelos.

Mendel observou a existência de fenótipos dominantes em relação a outros quando realizou o cruzamento de pés de ervilhas homozigotos de diferentes fenótipos. As plantas que nasceram deste cruzamento eram heterozigotas e possuíam o mesmo fenótipo de um dos pais, o dominante em relação ao outro. Esta análise fez com que Mendel concluísse que fenótipos recessivos aparecem apenas quando o gene é homozigoto e que genes dominantes precisam de apenas um alelo, no par de alelos, para que o fenótipo que ele representa se manifeste.

Através desse estudo Mendel definiu duas leis da Genética, conhecidas como Leis de Mendel. A primeira delas diz que os alelos segregam de forma aleatória, isto é, que um filho tem a mesma chance de herdar um (e apenas um) dos dois alelos de cada um dos pais. A segunda Lei de Mendel, também conhecida como da segregação independente, diz que a herança dos alelos dos genes ocorre de maneira independente, isto é, que diferenças de uma característica são herdadas independentemente de outras características. Para observar a independência entre as características, Mendel escolheu genes que não estavam no mesmo cromossomo.

Cromossomos são compostos por DNA (ácido desoxirribonucléico) e proteína, neles estão armazenadas as informações genéticas de um organismo. No século XX, estabeleceu-se a ligação entre os estudos de Mendel e os cromossomos. Cromossomos homólogos carregam os alelos de um gene, um em cada cromossomo, e na meiose (produção de gametas durante a divisão celular) estes alelos são transmitidos para as próximas gerações. Alguns vírus possuem as informações genéticas armazenadas no RNA, ácido ribonucléico.

2.2 DNA e RNA

O DNA é um polinucleotídeo composto por quatro nucleotídeos (um nucleotídeo é formado por uma pentose, uma base nitrogenada e um grupo fosfato) diferenciados por suas bases nitrogenadas: adenina, guanina (as púricas), citosina e timina (as pirimídicas), simbolizadas por A, G, C e T, respectivamente. Auxiliados por um grupo hidroxil (OH^-), dois nucleotídeos são unidos pelo terceiro carbono da pentose do primeiro nucleotídeo com o fosfato do segundo.

A estrutura do DNA é conhecida como de dupla hélice, pois ela é composta por duas fitas de polinucleotídeos em forma helicoidal ligadas por pontes de hidrogênio. As bases A e T e as bases C e G se complementam em fitas opostas. A Figura 2.1a mostra as bases complementares e suas ligações por pontes de hidrogênio. O fato de pares de bases serem compostos por uma base púrica e outra pirimídica pode ser explicado pela geometria e tamanho das mesmas, pois bases púricas possuem três pontes de hidrogênio, já as pirimídicas possuem duas. Se ambas fossem púricas o par ficaria grande para a hélice

e se fossem pirimídicas seria pequeno, desta forma mantém-se o equilíbrio da estrutura de hélice.

As ligações entre os nucleotídeos fazem com que cada fita de DNA possua duas extremidades livres, a 3' (grupo hidroxil livre) e a 5' (grupo fosfato livre), de forma que a extremidade 3' de uma fita é correspondente a 5' da outra. Na Figura 2.1b podem ser vistas as extremidades de uma molécula de DNA. Por convenção a escrita (e leitura) de moléculas de DNA é feita em apenas uma das fitas na direção 5'→ 3', obtendo-se a outra fita pela inversão da ordem das bases e complementando cada uma delas, o que denomina-se complemento reverso. Também na Figura 2.1b é possível observar a estrutura de dupla hélice do DNA, assim como a ligação das bases das fitas complementares em sentidos opostos.

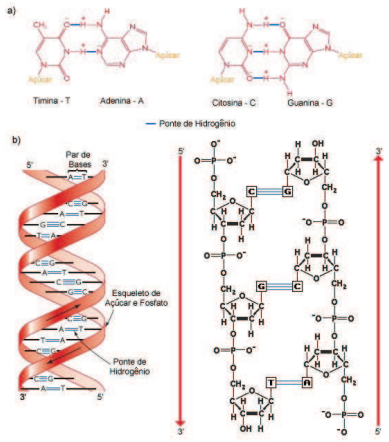


Figura 2.1: a) Bases que compõem o DNA e ligações de bases complementares por pontes de hidrogênio. b) Estrutura de dupla hélice do DNA com terminações 5' (fosfato livre) e 3' (hidroxil livre) [4].

O RNA é um polinucleotídeo como o DNA, mas possui apenas uma fita. Além disso, o RNA difere-se do DNA pela presença da base nitrogenada U (uracila) no lugar da base T, e da pentose ribose no lugar da desoxirribose.

A genética é a área da biologia que realiza o estudo dos genes, eles possuem informações armazenadas em trechos de DNA que codificam uma proteína ou RNA [7], através de reações bioquímicas denominadas de expressão gênica.

2.3 Expressão Gênica

O processo de expressão gênica, dividido em transcrição e tradução, transforma informações genéticas em um produto funcional, como as proteínas [32].

No processo de transcrição do gene é produzido o RNA mensageiro, mRNA, como uma cópia de uma das fitas do DNA, trocando T por U (podem ainda ser produzidos a partir de genes os RNAs transportador, tRNA, e ribossômico, rRNA). O início da transcrição ocorre quando a enzima RNA polimerase, que ativa a reação, liga-se a uma sequência de bases do DNA que marcam o início do processo. Neste momento as fitas de DNA se separam e a polimerase percorre a fita de DNA molde transcrevendo-a até que encontre uma sequência de bases que indicam o término da transcrição. Quando isso ocorre o RNA não é mais transcrito e a fita molde se une com a fita com a qual faz par.

O processo de tradução utiliza o mRNA produzido na transcrição para síntese de proteínas, que são formadas por diferentes aminoácidos. As bases do mRNA são lidas como códons (triplas de bases que especificam um aminoácido). Existem 64 códons relacionados a 20 aminoácidos, com exceção de três códons (TAA, TAG, TGA) que são utilizados para indicar o final da tradução e por isso não se relacionam com nenhum aminoácido. Na Tabela 2.1 pode-se observar o *Standard Genetic Code* (Código Genético Padrão), usados pela maioria dos organismos [21].

A tradução do mRNA em proteína é realizada no ribossomo (uma organela que percorre o mRNA) com auxílio de tRNAs. Os tRNAs transportam de um lado aminoácidos e de outro anticódons (códons com os nucleotídeos complementares) que se acoplam aos códons dos mRNAs e liberam os aminoácidos a serem unidos com a cadeia de aminoácidos que formará a proteína.

Na Figura 2.2 pode ser visto um exemplo dos processos de transcrição e tradução dos aminoácidos Metionina, Alanina, Serina e Valina, observe que o códon TAA não codifica nenhum aminoácido por ser um stop-códon. A transcrição é feita no sentido 5'→ 3' inciando no códon ATG, o start-códon.

Em seres mais simples como bactérias (procariotos, sem membrana nuclear), em geral, não há necessidade de pré-processamento para o processo de tradução, enquanto que em seres mais complexos (como eucariotos, que possuem material genético separado do restante da célula por uma membrana nuclear) ele deve ser realizado. No pré-processamento devem ser removidos os *introns* do mRNA, trechos dos genes que não codificam proteínas, permanecendo apenas os trechos que de fato produzem proteínas, conhecidos como *exons*.

Cromossomos que compõem a célula de um organismo denominam-se genoma [7], estes possuem uma cópia completa do material genético produzido a cada divisão celular. O processo de divisão celular em que cada célula produz duas células idênticas é chamado

Aminoácido	Cod. 3	Cód. 1	códon
Alanina	ALA	A	CCA, GCC, GCG, GCU
Arginina	ARG	R	AGA, AGG, CGA, CGC, CGG, CGT
Asparagina	ASN	N	AAC, AAF
Ácido Asparmático	ASP	D	GAC,GAT
Cysteína	CYS	C	TGC, TGT
Ácido Glutêmico	GLU	E	GAA, GAC
Glutamina	GLN	Q	CAA, CAG
Glycina	GLY	G	GCA, GGC, GGG, GCT, CAG
Histidina	HIS	H	CAG, CAT
Isoleucina	ILE	I	ATA, ATT, ATC
Leucina	LEU	L	CTA, CTC, CTG, CTT, TTA, TTG
Lysina	LYS	K	AAA, AAG
Metionina	MET	M,	ATG (start-codon)
Phenilalanina	PHE	F	TTC, TTT
Prolina	PRO	P	CCT, CCC, CCA, CCG
Serina	SER	S	AGT, TCA, TCC, TCT, TCG
Theonina	THR	T	ACA, ACC, ACG, ACT
Tryptopan	TRP	W	TGG
Tyrosina	TYR	Y	TAC, TAT
Valina	VAL	V	GTA, GTC, GTG, GTT
Stop-Codon	-	*	TAA, TAG, TGA

Tabela 2.1: Código Genético utilizado pela maioria dos organismos. A primeira coluna traz o nome dos aminoácidos, a segunda a abreviação do nome em três letras, a terceira coluna traz as abreviações com uma letra, finalmente, a quarta tem os códons correspondentes a cada aminoácido.

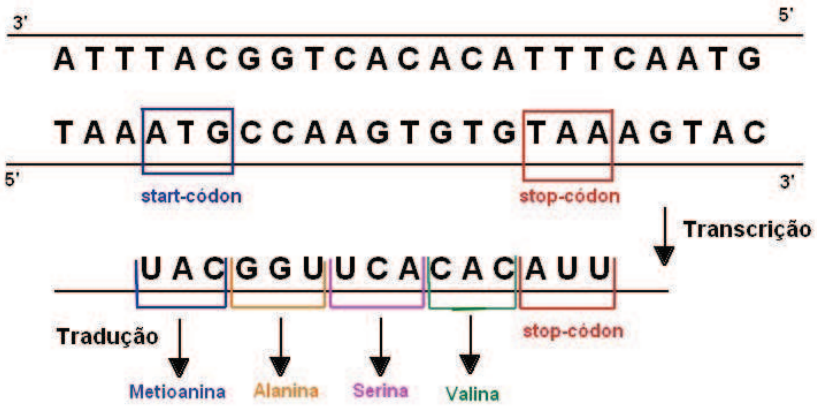


Figura 2.2: Processos de transcrição e tradução dos aminoácidos Metionina, Alanina, Serina e Valina, a partir de uma molécula de DNA.

de mitose. Já o processo em que a divisão celular produz quatro células filhas com a metade dos cromossomos é chamado de meiose. Se o processo de divisão não for perfeito podem ocorrer mutações. Mutações podem fazer com que uma célula ou um organismo morra. Entretanto, caso ele sobreviva, as características que mutaram provavelmente serão herdadas pelos descendentes do organismo.

2.4 Proteínas

Um dos principais problemas da biologia é entender a evolução das espécies que existem hoje, pois é comum encontrar proteínas exercendo a mesma função em diferentes organismos [63].

Proteínas possuem quatro tipos de estrutura: primária, secundária, terciária e quaternária. A estrutura primária constitui da sequência de aminoácidos que formam a proteína e das pontes de dissulfureto. Esta sequência determina a estrutura de níveis mais altos da molécula, pois a mudança de um aminoácido pode levar a profundas mudanças na estrutura das proteínas [9].

Na estrutura secundária, as proteínas dobram-se de diferentes formas. Uma possível é a alfa hélice, na qual o peptídeo é enrolado em torno de um cilindro imaginário. Outra forma possível é a beta, folhas planas, na qual os aminoácidos adotam a conformação de uma folha de papel. Ainda existentes algumas partes da estrutura que não são muito

estáveis e podem assumir formação de bobina aleatória.

A terciária é composta pela formação tridimensional completa, incluindo as pontes de hidrogênio e ligações entre resíduos. Aminoácidos distantes na estrutura primária podem estar próximos na terciária devido à forma como a cadeia se dobra. A estrutura quaternária é o arranjo de duas ou mais subunidades polipeptídicas das proteínas. Na Figura 2.3 podemos observar as quatro estruturas que formam uma proteína.

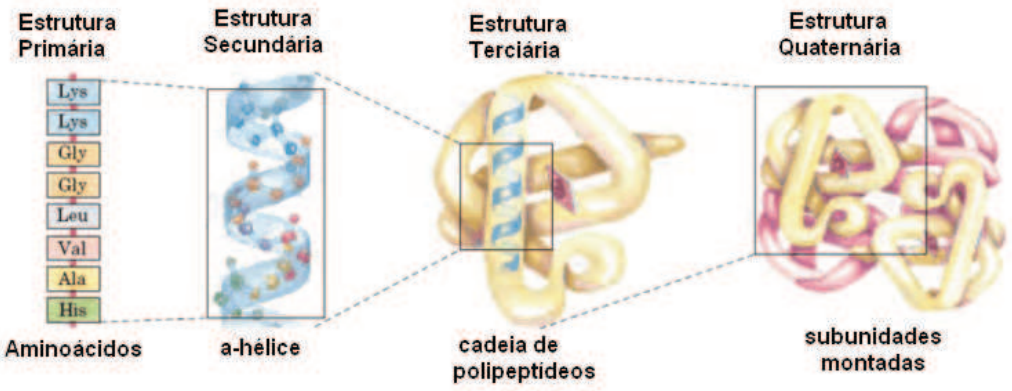


Figura 2.3: Estruturas de uma proteína [39].

Estruturas semelhantes de proteínas em seres vivos de espécies distintas podem ser homólogas, quando estas se originam de um ancestral comum. Proteínas homólogas podem apresentar diferenças em sua estrutura primária, quando estas não ocorrem em uma região a denominamos de região conservada. Regiões altamente conservadas são chamadas de *motifs*.

Algumas regiões conservadas possuem catalisadores, sequências peptídicas que possuem na sua estrutura tridimensional locais onde os reagentes se encaixam com facilidade, que agilizam a ocorrência de reações químicas.

Além da função enzimática, as proteínas ainda podem desempenhar as seguintes funções: reguladora (como as vitaminas), reserva (como sementes de plantas), transportadora (transportam nutrientes e metabólitos entre fluidos e tecidos), estrutural (matéria-prima para construção de estruturas celulares), defesa (agem contra antígenos que invadem o organismo humano), hormonal (estimulam ou inibem uma determinada atividade em certo órgão).

2.5 Sequenciamento

Sequenciar um gene significa determinar a cadeia de nucleotídeos que o compõe. Um genoma pode possuir bilhões de bases, o que faz com que seja complexo sequenciá-lo inteiramente. Diante disso, realiza-se a fragmentação do genoma em pequenas partes e posteriormente, para obter a sequência genômica completa, realiza-se a montagem dos *reads*, pequenas sequências, sequenciados.

A fragmentação é a primeira etapa do sequenciamento e pode ser feita pelo método de digestão, que utiliza enzimas de restrição para quebrar o DNA, mas é limitado por depender de sítios de restrição apropriados do DNA-alvo e também da possível necessidade de várias enzimas [63]. Uma boa estratégia para o sequenciamento é a *shotgun*, muito indicada para sequenciamento em larga escala.

A estratégia *shotgun* submete o DNA a altas taxas de vibração, que promovem a quebra da cadeia de nucleotídeos em vários fragmentos que em geral são únicos. Normalmente, selecionam-se os fragmentos que serão usados posteriormente para o reconhecimento das bases e estes são clonados para amplificar a quantidade de fitas de DNA disponíveis.

Através de clonagem é possível criar cópias idênticas de moléculas de DNA e comumente é realizada utilizando-se de um vetor (bactérias, vírus), por possuírem curto ciclo de vida. O fragmento de DNA a ser clonado é inserido no DNA do vetor e quando este se multiplica, o fragmento de DNA também se multiplica.

Após a divisão do genoma em partes menores inicia-se então o sequenciamento das bases de cada uma delas, sendo o método de terminação de cadeia, ou de Sanger [62], usado tradicionalmente desde a década de 1970.

Atualmente pesquisadores têm se esforçado para desenvolver um método de sequenciamento mais simples e robusto que o de Sanger [62], principalmente quando se trata do sequenciamento de *reads* curtos, que possuem aproximadamente 300 bases. Um método interessante e que está sendo muito utilizado é o pirossequenciamento [57–59, 65].

Sequências resultantes dos projetos de sequenciamento, podem conter artefatos. Artefatos são partes das sequências que possuem regiões de baixa qualidade, baixa complexidade ou ainda que não fazem parte do organismo sequenciado. As regiões que não fazem parte do organismo sequenciado podem ser trechos de vetores e adaptadores, por exemplo, que foram utilizados durante o sequenciamento. Em regiões de baixa qualidade são mais comuns os erros de sequenciamento.

O sequenciamento em larga escala leva à necessidade do uso de computadores e ferramentas não só durante o processo, mas também para interpretar e utilizar o grande volume de dados produzido. Identificar a presença de artefatos, assim como corrigir erros de sequenciamento são importantes tarefas a serem realizadas antes da análise das sequências produzidas pelos projetos de sequenciamento e podem ser feitas utilizando

recursos computacionais.

De posse dos resultados do sequenciamento, isto é, de se ter tomado conhecimento das bases que formam o DNA de um organismo, é possível descobrir quais dessas bases de fato codificam proteínas e quais proteínas são codificadas. Um gene pode conter o código de várias proteínas e desempenhar diversas funções.

2.6 Proteômica

Com o crescente número de genomas sendo sequenciados surgiu a necessidade de se estudar com mais cuidado e detalhes os elementos expressos pelos genes. Tendo em vista que os genes geram as proteínas após os processos de transcrição e tradução, utilizá-las como elementos para este estudo tornou-se uma idéia interessante.

A proteômica é um novo campo de estudo e tecnologia que se propõe a analisar de forma global o conjunto de proteínas expressas numa célula ou tecido, ou seja, o proteoma [11]. O proteoma de um organismo, diferentemente do genoma, altera-se de acordo com as condições as quais o organismo está exposto.

O avanço tecnológico no estudo de proteínas, aliado à crescente geração de dados genômicos, fizeram da proteômica uma ferramenta poderosa para entender a genética e fisiologia de diversos organismos [60]. A combinação de técnicas como a eletroforese bidimensional (separa as diferentes proteínas sintetizadas por uma célula), juntamente com a Bioinformática possibilitou o entendimento detalhado de uma grande quantidade de proteínas.

2.7 Bioinformática

A Bioinformática visa lidar com os resultados das iniciativas de sequenciamento de genes, que produzem uma quantidade cada vez maior de dados sobre proteínas, DNA e RNA [72]. Ela combina conhecimentos de diferentes áreas como: computação, estatística, matemática e biologia.

Desta forma, a Bioinformática pode ser definida como a área da Ciência da Computação que visa analisar dados e resolver problemas de aplicações biológicas através do desenvolvimento de ferramentas. O avanço tecnológico dos computadores possibilitou à Bioinformática o desenvolvimento de softwares para processar grande volume de dados gerados pelos projetos da biologia.

Ao sequenciar o genoma de um organismo objetiva-se entender do que ele é composto, assim como evoluiu geneticamente ao longo do tempo. A análise da evolução de um genoma inclui observar a sua estrutura, função e até as mutações que ocorreram durante

sua evolução genética [14].

Analisar manualmente as inúmeras sequências produzidas em projetos de sequenciamento é impraticável, por isso é importante o desenvolvimento de técnicas automáticas. Dentre as atividades desenvolvidas pela Bioinformática para análise de dados biológicos, o alinhamento de sequências de proteínas é o foco do trabalho proposto.

Sequências podem ser comparadas aos pares ou alinhadas simultaneamente, construindo o chamado MSA (do inglês, *Multiple Sequence Alignment*) para visualização do efeito da evolução em uma família inteira de proteínas. MSAs podem ser utilizados também na construção árvores filogenéticas e predição de estrutura secundária e terciária de proteína.

Capítulo 3

Alinhamento de Sequências

Uma das principais tarefas em bioinformática é o alinhamento de sequências, uma vez que através deste é possível estudar eventos evolucionários e restrições estruturais ou funcionais de sequências de proteína, DNA ou RNA [14]. Em outras palavras, permite entender a estrutura, função e evolução dos genes que compõem um organismo.

Alinhar duas sequências consiste na inserção de espaços nas sequências de modo que elas fiquem do mesmo tamanho, e que seja possível sobrepô-las para a comparação de letras [63].

Sejam S e T duas sequências de caracteres e espaços, sendo que $|S|$ e $|T|$ representam os tamanhos de S e T, respectivamente. Um alinhamento A entre as sequências S e T, é um mapeamento de S e T em outras sequências S' e T' com os mesmos caracteres e na mesma ordem de S e T, sendo possível inserir espaços de forma que $|S'|=|T'|$.

No alinhamento de sequências busca-se posicionar as letras de forma que os que forem iguais, ou semelhantes, fiquem em uma mesma coluna. Caso haja diferenças entre elementos de uma coluna é sinal de que ocorreu uma mutação. Para completar um alinhamento realiza-se a inserção de buracos, que correspondem aos *indels* que ocorreram no processo evolutivo, isto é, inserções ou remoções de letras. Desta forma, um alinhamento ótimo é aquele que melhor representa o cenário de evolução das sequências.

Para calcular o alinhamento é criado um sistema de pontuação que permite quantificar a qualidade de um alinhamento. Desta forma é possível comparar alinhamentos, e assim, escolher um melhor, ou seja, aquele de maior relevância biológica. Um esquema de pontuação comumente empregado é aquele que diferencia *matches*, *mismatches* e buracos. *Matches* e *mismatches* ocorrem quando há duas letras em uma mesma coluna. Para um *match* é necessário que as duas letras sejam iguais, caso contrário ocorreu *mismatch*. Já um buraco ocorre quando uma letra é posicionada ao lado de um espaço, é representado por um hífen. A pontuação do alinhamento é dada pela soma das pontuações das colunas.

3.1 Alinhamento de Pares de Sequências

O primeiro algoritmo para alinhamento de pares de sequências, desenvolvido em 1970 por Needleman e Wunsch [48], utiliza programação dinâmica para alinhar duas sequências e requer complexidade $\Omega(mn)$ para tempo e espaço, onde m e n são os comprimentos das sequências. Existem diferentes tipos de algoritmos para comparação de duas sequências: global, semi-global e local.

Na abordagem global, realiza-se a comparação das duas sequências alinhando-as completamente. O algoritmo básico de Needleman e Wunsch [48] possui duas etapas: preenchimento da matriz de pontuação, M, e recuperação do alinhamento a partir da matriz. A matriz é preenchida inicializando cada coluna j da primeira linha com o valor $j \times gap$ (onde *gap* é a penalidade individual para cada buraco, aqui representado por um hífen), e cada linha i da primeira coluna, com o valor $i \times gap$.

Seja *score(i,j)* a pontuação quando se alinham as letras i e j, da primeira e segunda sequências, respectivamente, de acordo com os valores definidos para *match*, *mismatch* e *gap*. As demais posições, $M[i,j]$, da matriz assumem o seguinte valor: $M[i,j] = max\{M[i-1,j] + gap, M[i-1,j-1] + score(i,j), M[i,j-1] + gap\}$.

Considerando que a primeira sequência do alinhamento é a disposta a esquerda da matriz de pontuação e a segunda como a disposta no topo. Cada posição i, da primeira sequência, e j, da segunda sequência, possui uma das seguintes opções para alinhamento:

- Letra na posição i da primeira sequência alinhada com um buraco quando a maior pontuação é $M[i-1,j] + gap$.
- Alinhamento entre a posição i da primeira sequência e j da segunda, quando a maior pontuação é $M[i-1,j-1] + score(i,j)$.
- Letra na posição j da segunda sequência alinhada com um buraco quando maior pontuação é $M[i,j-1] + gap$.

As etapas do alinhamento global para preenchimento da matriz e recuperação do alinhamento são descritas respectivamente nos Algoritmos 1 e 2.

A construção da matriz de pontuação, assim como a recuperação do alinhamento (caminho em negrito) para as sequências `ACTGGACACT` e `ACATGGAT`, com *match*=1, *mismatch*=-1 e *gap*=-2 podem ser vistos na Tabela 3.1. O alinhamento resultante foi o seguinte:

```
AC-TGGACACT
ACATGG--A-T
```

Algoritmo 1 Algoritmo global para alinhamento de pares de sequências

Entrada: sequence1, sequence2
Saída: M
m ← sequence1.length
n ← sequence2.length
for $i \leftarrow 0$ to m **do**
  M[i,0] ← i × gap
for $j \leftarrow 0$ to n **do**
  M[0,j] ← j × gap
for $i \leftarrow 0$ to m **do**
  for $j \leftarrow 0$ to n **do**
    M[i,j] ← max{M[i-1,j]+gap, M[i-1,j-1]+score(i,j), M[i,j-1]+gap}
return M

		A	C	A	T	G	G	A	T
	0	-2	-4	-6	-8	-10	-12	-14	-16
A	-2	**1**	-1	-3	-5	-7	-9	-11	-13
C	-4	-1	**2**	**0**	-2	-4	-6	-8	-10
T	-6	-3	0	1	**1**	-1	-3	-5	-7
G	-8	-5	-2	-1	0	**2**	0	-2	-4
G	-10	-7	-4	-3	-2	1	**3**	1	-1
A	-12	-9	-6	-3	-4	-1	**1**	4	2
C	-14	-11	-8	-5	-4	-3	**-1**	2	3
A	-16	-13	-10	-7	-6	-5	-3	**0**	1
C	-18	-15	-12	-9	-8	-7	-5	**-2**	-1
T	-20	-17	-14	-11	-8	-9	-7	-4	**-1**

Tabela 3.1: Matriz de pontuação do alinhamento global das sequências `ACTGGACACT` e `ACATGGAT`, com *match*=1, *mismatch*=-1 e *gap*=-2. Em negrito o caminho da recuperação do alinhamento.

Algoritmo 2 Algoritmo para recuperação do alinhamento de pares de sequência a partir da matriz gerada pelo Algoritmo 1

```
Entrada: sequence1, sequence2, M
Saída: alignSequence1, alignSequence2
i ← sequence1.length
j ← sequence2.length
alignSequence1 ← ""
alignSequence2 ← ""
while i >0 and j >0 do
  if M[i,j] = M[i-1,j] + gap then
    alignSequence1 ← sequence1[i] + alignSequence1
    alignSequence2 ← "-"+ alignSequence2
    i ← i - 1
  else
    if M[i,j] = M[i-1,j-1] + score(i,j) then
      alignSequence1 ← sequence1[i] + alignSequence1
      alignSequence2 ← sequence2[j] + alignSequence2
      i ← i - 1
      j ← j - 1
    else
      alignSequence1 ← "-"+ alignSequence1
      alignSequence2 ← sequence2[j] + alignSequence2
      j ← j - 1
while i > 0 or j>0 do
  if i > 0 then
    alignSequence1 ← sequence1[i] + alignSequence1
    alignSequence2 ← "-"+ alignSequence2
    i ← i - 1
  else
    alignSequence1 ← "-"+ alignSequence1
    alignSequence2 ← sequence2[j] + alignSequence2
    j ← j - 1
return (alignSequence1, alignSequence2)
```

No alinhamento semi-global não são penalizados os buracos nas extremidades das sequências, pois geralmente os erros de sequenciamento são mais concentrados nas extremidades das sequências. A fim de alcançar o melhor alinhamento entre o prefixo de uma sequência e o sufixo de outra, o alinhamento semi-global utiliza os Algoritmos 1 e 2 com algumas adaptações. Para não penalizar buracos no final da primeira sequência deve-se recuperar o alinhamento a partir da posição de maior valor da última linha da matriz. Já para não penalizar no final da segunda sequência, deve-se recuperar o alinhamento a partir da posição de maior valor da última coluna da matriz. Quando deseja-se não penalizar os buracos no início da primeira sequência preenche-se a primeira coluna da matriz com zeros e, para não penalizar no início da segunda, preenche-se a primeira linha da matriz com zeros.

A Tabela 3.2 mostra a matriz de pontução para o alinhamento semi-global das sequências `TATTGAAGCATTAGGGCG` e `GAAGCATAGTGCGTGAT`. Em negrito a recuperação alinhamento utilizando pontuação *match*=5, *mismatch*=-1 e *gap*=-3. O alinhamento resultante é:

```
TATTGAAGCATTAGGGCG----
----GAAGCA-TAGTGCGTGAT
```

A abordagem local busca encontrar o alinhamento de maior pontuação entre subsequências das duas sequências de entrada. É interessante quando se deseja encontrar as regiões melhores conservadas entre duas sequências. Em 1981, Smith e Waterman [64] descreveram um algoritmo de abordagem local, também de complexidade $O(mn)$. Este algoritmo também é semelhante ao global, diferenciando-se no preenchimento da matriz e na recuperação do alinhamento. Ao preencher a matriz, o Algoritmo 1 é modificado de forma a exigir que o valor mínimo de cada posição seja 0, isto é: $M[i,j] = max\{M[i-1,j]+gap$ $M[i-1,j-1] + score(i,j), M[i,j-1] + gap, 0\}$. Nunca haverá um elemento negativo nas células dessa matriz, pois no pior dos casos há sempre o alinhamento entre os sufixos vazios, [1,i] da primeira sequência com [1,j] da segunda sequência, que dá uma pontuação igual a zero [63]. Na recuperação do alinhamento o Algoritmo 2 é modificado para que a matriz seja percorrida a partir do seu maior valor (valor ótimo para o alinhamento), até que se encontre uma posição com valor 0 ou uma das bordas seja alcançada.

A Tabela 3.3 mostra a matriz de pontuação para o alinhamento local das sequências `TATTGAAGCATTAGGGCG` e `GAAGCATAGTGCGTGAT`, utilizando *match*=3, *mismatch*=-3 e *gap*=-5. Em negrito a recuperação do seguinte alinhamento:

```
GAAGCATTAGGGCG
GAAGCA-TAGTGCG
```

3.1.1 Alinhamento Recursivo

A abordagem local busca alinhar corretamente as partes bem conversadas das sequências. Programas como o BLAST [2] utilizam esta estratégia. Uma adaptação do algoritmo de

		G	A	A	G	C	A	T	A	G	T	G	C	G	T	G	A	T
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T	**0**	-1	-1	-1	-1	-1	-1	5	2	-1	5	2	-1	-1	5	2	-1	5
A	**0**	-1	4	4	1	-2	4	2	10	7	4	4	1	-2	2	4	7	4
T	**0**	-1	1	3	3	0	1	9	7	9	12	9	6	3	3	1	4	12
T	**0**	-1	-2	0	2	2	-1	6	8	6	14	11	8	5	8	5	2	9
G	0	**5**	2	-1	5	2	1	3	5	13	11	19	16	13	10	13	10	7
A	0	2	**10**	7	4	4	7	4	8	10	12	16	18	15	12	10	18	15
A	0	-1	7	**15**	12	9	9	6	9	7	9	13	15	17	14	11	15	17
G	0	5	4	12	**20**	17	14	11	8	14	11	14	12	20	17	19	16	14
C	0	2	4	9	17	**25**	22	19	16	13	13	11	19	17	19	16	18	15
A	0	-1	7	9	14	22	**30**	27	24	21	18	15	16	18	16	18	21	18
T	0	-1	4	6	11	19	**27**	35	32	29	26	23	20	17	23	20	18	26
T	0	-1	1	3	8	16	24	**32**	34	31	34	31	28	25	22	22	19	23
A	0	-1	4	6	5	13	21	29	**37**	34	31	33	30	27	24	21	27	24
G	0	5	2	3	11	10	18	26	34	**42**	39	36	33	35	32	29	26	26
G	0	5	4	1	8	10	15	23	31	39	**41**	44	41	38	35	37	34	31
G	0	5	4	3	6	7	12	20	28	36	38	**46**	43	46	43	40	37	34
C	0	2	4	3	3	11	9	17	25	33	35	43	**51**	48	45	42	39	36
G	0	5	2	3	8	8	10	14	22	30	32	40	48	**56**	**53**	**50**	**47**	**44**

Tabela 3.2: Matriz de pontuação do alinhamento semi-global das sequências `TATTGAAGCATTAGGGCG` e `GAAGCATAGTGCGTGAT`, com *match*=5, *mismatch*=-1 e *gap*=-3.

		G	A	A	G	C	A	T	A	G	T	G	C	G	T	G	A	T
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	3	0	0	3	0	0	0	3	0	0	3
A	0	0	3	3	0	0	3	0	6	1	0	0	0	0	0	0	3	0
T	0	0	0	0	0	0	0	6	1	3	4	0	0	0	3	0	0	6
T	**0**	0	0	0	0	0	0	3	3	0	6	1	0	0	3	0	0	3
G	0	**3**	0	0	3	0	0	0	0	6	1	9	4	3	0	6	1	0
A	0	0	**6**	3	0	0	3	0	3	1	3	4	6	1	0	1	9	4
A	0	0	3	**9**	4	0	3	0	3	0	0	0	1	3	0	0	4	6
G	0	3	0	4	**12**	7	2	0	0	6	1	3	0	4	0	3	0	1
C	0	0	0	0	7	**15**	10	5	0	1	3	0	6	1	1	0	0	0
A	0	0	3	3	2	10	**18**	13	8	3	0	0	1	3	0	0	3	0
T	0	0	0	0	0	5	**13**	21	16	11	6	1	0	0	6	1	0	6
T	0	0	0	0	0	0	8	**16**	18	13	14	9	4	0	3	3	0	3
A	0	0	3	3	0	0	3	11	**19**	15	10	11	6	1	0	0	6	1
G	0	3	0	0	6	1	0	6	14	**22**	17	13	8	9	4	3	1	3
G	0	3	0	0	3	3	0	1	9	17	**19**	20	15	11	6	7	2	0
G	0	3	0	0	3	0	0	0	4	12	14	**22**	17	18	13	9	4	0
C	0	0	0	0	0	6	1	0	0	7	9	17	**25**	20	15	10	6	1
G	0	3	0	0	3	1	3	0	0	3	4	12	20	**28**	23	18	13	8

Tabela 3.3: Matriz de pontuação do alinhamento local das sequências `TATTGAAGCATTAGGGCG` e `GAAGCATAGTGCGTGAT` com *match*=3, *mismatch*=-3 e *gap*=-5.

alinhamento local é apresentada, nela as sequências são alinhadas recursivamente.

Inicialmente são utilizadas as sequências inteiras no primeiro alinhamento local. As subsequências não alinhadas são tomadas como novas sequências e apresentadas recursivamente (à esquerda e à direita) como entrada para o alinhador local recursivo. Os alinhamentos produzidos por cada chamada recursiva são concatenados, formando uma sequência que representa o alinhamento completo. O alinhamento recursivo é executado enquanto há um alinhamento local para as sequências de entrada atual. Quando o alinhamento local não é mais possível (isto é, retorna sequências vazias), recorre-se ao alinhamento global para alinhar estas sequências não alinhadas.

O Algoritmo possui complexidade $O(n^3)$ conforme podemos observar na Recorrência: $T(n,m) = T(n1,m1) + T(n2,m2) + O(n^2) = O(n^3)$.

O alinhamento recursivo é apresentado no Algoritmo 3. Neste algoritmo, **Local** é o algoritmo de Smith e Waterman [64] para alinhamento local e **Global** é o algoritmo de Needleman e Wunsch [48] para alinhamento global.

Algoritmo 3 Pseudocódigo do alinhamento recursivo

```
Entrada: sequence1, sequence2
Saída: seq1Algn, seq2Algn
if sequence1.length>0 and sequence2.length>0 then
  (seq1Algn,seq2Algn,iniAlgnSeq1,iniAlgnSeq2,fimAlgnSeq1,fimAlgnSeq2) ← Local(sequence1,sequence2)
  if seq1Algn.length = 0 then
    (seq1Algn,seq2Algn) ← Global(sequence1,sequence2)
  else
    (seq1AlgnLeft,seq2AlignLeft) ← LocalRecursive(sequence1[1:iniAlgnSeq1-1],sequence2[1:iniAlgnSeq2-1])
    (seq1AlgnRight,seq2AlignRight) ← LocalRecursive(sequence1[fimAlgnSeq1+1:sequence1.length],sequence2[fimAlgnSeq2+1:sequence2.length])
    seq1Algn ← seq1AlgnLeft + seq1Algn + seq1AlgnRight
    seq2Algn ← seq2AlgnLeft + seq2Algn + seq2AlgnRight
else
  seq1Algn ← sequence1
  seq2Algn ← sequence2
  for i ← 1 to sequence1.length do
    seq2Algn ← "-"+ seq2Algn
  for i ← 1 to sequence2.length do
    seq1Algn ← seq1Algn + "-"
return(seq1Algn,seq2Algn)
```

Sejam as sequências `GGGAAGCATAGTGCGTGAT` e `TATTGAAGCATTAGGGCGGT`, e *match*=2,

mismatch=-3 e *gap*=-5, os valores dos parâmetros. Vamos observar o alinhamento local rescursivo.

A primeira chamada ao algoritmo gera o seguinte alinhamento local, chamado de A_1:

```
GAAGCA-TAGTGCG
GAAGCATTAGGGCG
```

O alinhamento não utiliza as sequências de entrada por completo, restam subsequências não alinhadas à direita e à esquerda da subsequência alinhada. Portanto, são realizadas chamadas recursivas ao algoritmo de alinhamento local recursivo. O primeiro para esquerda, L_1, para alinhar `GG` com `TATT`, estas sequências não são alinhadas pelo alinhador de pares local, sendo necessário utlizar do alinhador de pares global, que produz o seguinte alinamento, chamado de A_2:

```
--GG
TATT
```

Agora é preciso alinhar as subsequências `TGAT` e `GT`, não alinhadas a direita, chamando a recursão R_1. Que as alinha da seguinte forma, no alinhamento A_3:

```
T
T
```

Ainda restam subsequências de R_1 que não foram alinhadas, para que sejam alinhadas é preciso uma recursão para esquerda, L_2 que alinhe: `TGA` e `G`. O alinhamento, A_4, resultante foi:

```
G
G
```

As letras T, à esquerda (L_3) do alinhamento de L_2, e A à direita (R_2) são alinhadas com buracos.

O alinhamento final é obtido pela concatenação dos alinhamentos das chamadas recursivas realizadas. Neste exemplo concatena-se o resultado de L_1 com o A_1 e R_1. A recursão L_1 é igual ao alinhamento A_2. Por sua vez, a recursão R_1 é obtida pela concatenação L_3, A_4,R_2 e A_3. O alinhamento resultante dessas concatenações é:

```
--GGGAAGCA-TAGTGCGTGAT
TATTGAAGCATTAGGGCG-G-T
```

A Figura 3.1 mostra uma árvore com as chamadas recursivas que ocorreram no método, onde L_x é uma chamada recursiva para esquerda, R_x uma chamada recursiva para direita e A_x o alinhamento local da entrada corrente. A concatenação de todos os alinhamentos locais resultou no alinhamento final.

O alinhamento local do Algoritmo 3 permite que o alinhamento resultante do alinhamento local tenha qualquer tamanho, com isso pequenas partes das sequências podem ser

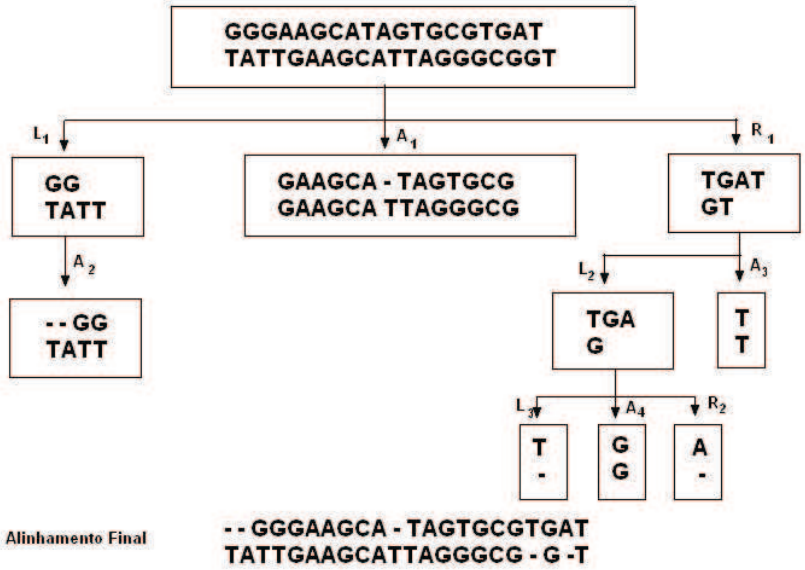


Figura 3.1: Etapas do alinhamento recursivo, Algoritmo 3

alinhadas. O alinhamento de poucas letras das sequências originais pode levar a quebra de blocos conservados. Uma alternativa para evitar, ou ao menos reduzir essa quebra é limitar um valor mínimo para o tamanho do alinhamento de pares local.

Objetivando conseguir um alinhamento local que utilize a maior pontuação da matriz de pontuação, mas que satisfaça esse limite mínimo de tamanho do alinhamento utilizamos uma matriz auxiliar a já utilizada em Smith e Waterman [64], nesta matriz auxiliar são armazenados os tamanhos dos alinhamentos. Desta forma, na recuperação do alinhamento verifica-se também, na matriz de tamanhos, se o tamanho do alinhamento a partir desta posição da matriz é maior que o mínimo determinado. Caso o tamanho seja menor que o mínimo, procura-se a segunda maior pontuação da matriz e verifica-se o tamanho do alinhamento, se ainda não satisfizer o mínimo segue em busca do terceiro valor e assim sucessivamente. Com isso, o alinhamento local resultante recuperado é o com maior pontuação que satisfaça o limite mínimo de tamanho. Se nenhum dos alinhamentos possíveis possuírem o tamanho mínimo, recorre-se ao alinhamento global para alinhar as sequências.

3.2 Pontuações para Alinhamentos de Sequências

Diferentes funções são utilizadas para expressar a penalidade de buracos. Elas podem ser aditivas ou sub-aditivas. Funções aditivas são as que possuem a seguinte característica $w(k_1 + k_2) = w(k_1) + w(k_2)$ e como exemplo podemos citar funções lineares. Funções sub-aditivas possuem a seguinte característica $w(k_1 + k_2) \leq w(k_1) + w(k_2)$ e como exemplo podemos citar funções afim e logarítmica.

3.2.1 Pontuação Linear

A pontuação linear para buracos é determinada por funções em que cada buraco é pontuado individualmente. São representadas como na Equação 3.1, na qual $w(k)$ é a função de penalidade de buracos, k é o número de buracos e *gap* é a pontuação para cada buraco.

$$w(k) = gap \times k \tag{3.1}$$

Utilizando como exemplo as seguintes sequências de nucleotídeos:

```
GCGATGACGATGACGATGACGATGACGATGACGATGACGATGACGATATATTATAGACGATGAGAGGGGATTACGATGACAG
GCGATGACTGGGATGATGACGATGACTAGAGAAGACGATATATGACGATGGGTCTC
```

Foram alinhadas com função linear para penalidade de buracos, com *match*=4, *mismatch*=-1 e *gap*=-5. Podemos observar que o alinhamento produzido, mostrado a seguir, possui 26 buracos distribuídos em 13 blocos diferentes.

```
GCGATGACGATGACGATGACGATGACGATGACGATGACGATGACGATATATTATAGACGATGAGAGGGGATTACGATGACAG
GCGATGAC--TGG-GATGA---TGACGATGACTA-GA-GAAGACGATATAT----GACGATG-G-G----T--C--T--C--
```

3.2.2 Pontuação Afim

A pontuação afim para penalizar buracos visa introduzir a idéia de que k buracos em sequência são mais prováveis que k buracos isolados. Isso é implementado pela aplicação de uma penalidade grande para cada abertura de um bloco de buracos [63]. Isto supõe que é mais provável que ocorram inserções/remoções de letras em conjunto que individualmente.

A Equação 3.2 apresenta a função $w(k)$ para penalizar buracos utilizando pontuação afim

$$w(k) = gop + gep \times k \tag{3.2}$$

onde, k é o número de buracos, *gop* (*gap opening*) é uma constante de penalidade de abertura de um bloco de buracos e *gep* (*gap extended*) é a penalidade para cada um dos k buracos consecutivos do bloco.

A seguir apresentamos o alinhamento das mesmas sequências do exemplo de pontuação linear (Seção 3.2.1) com *match*=4, *mismatch*=-1, *gop*=-17 e *gep*=-1. O alinhamento com pontuação afim para penalidade de buracos concentra os 26 buracos em apenas três blocos, pelo o fato de que estender um bloco de buracos sofre menor penalidade do que abrir um novo bloco de buracos.

```
GCGATGACGATGACGATGACGATGACGATGACGATGACGATGACGATATATTATAGACGATGAGAGGGGATTACGATGACAG
GCGATGACTGGGATGATGACGATGAC--------TAGAGAAGACGATATAT----GACGATG--------------GGTCTC
```

3.2.3 Pontuação Logarítmica

Apesar do grande uso de função afim no alinhamento de sequências, alguns pesquisadores levantaram questões sobre a justificativa de seu uso. De acordo com Gu e Li [26] se a frequência de *indels* segue uma distribuição geométrica, deve-se usar a penalidade linear, enquanto que se essa frequência segue uma lei de potência (relação polinomial que possui a propriedade de escala invariante e relaciona duas grandezas) a penalidade logarítmica para buracos é mais apropriada. A realização de estudos a este respeito mostrou que o tamanho de um *indel* é relacionado com sua frequência em uma escala logarítmica, e portanto obedecem uma lei de potência [8].

A distribuição de Zipfian é uma lei de potência que mostra que a probalidade de um *indel* ter tamanho k é $P(k|z) = k^{-z}/\zeta(z)$, onde $z > 1$ e $\zeta(z) = \sum_{n=1}^{\infty} n^{-z}$ é a função Zeta de Riemann's. Se $1 < z \leq 2$, a média da distribuição é infinita, e se $1 < z \leq 3$, a variância é infinita [8].

A função logarítmica é um exemplo de função côncava, e é definida como na Equação 3.3,

$$w(k) = gop + c \times \log(k) \tag{3.3}$$

onde $w(k)$ é a função para penaizar uma sequência de k buracos, *gop* é uma constante de penalidade de abertura de um bloco de buracos, e c é uma constante de penalidade para extensão de buracos.

Miller e Myers [43] ressaltam que a análise de *indels* consecutivos pode ser tratada como uma operação atômica e não como uma troca individual de símbolos. Ainda mostram como utilizar uma função côncava ($\Delta w_k \geq \Delta w_{k+1}$, onde $\Delta w_k = w_{k+1} - w_k$), como a logarítmica para representar um bloco de buracos.

Um algoritmo para alinhar pares de sequência com penalidade logarítmica de buracos tem complexidade de tempo $O(n^3)$ e $O(n^2)$ de espaço, onde n é o tamanho das sequências, é descrito no Algoritmo 4.

As mesmas sequências do exemplo de penalidade linear para buracos (seção 3.2.1), alinhadas utilizando pontuação logarítmica com 6, -1, -8, -2 como valores para *match*, *mismatch*, *gop* e *c*, respectivamente, possuem o seguinte alinhamento.

```
GCGATGACGATGACGATGACGATGACGATGACGATGACGATGACGATATATTATAGACGATGAGAGGGGATTACGATGACAG
GCGATGACTGGGATGATGACGATGACTA-GA-------GAAGACGATATAT----GACGATGGGT--------------CTC
```

Podemos observar que os 26 buracos foram agrupados em 4 blocos, respeitando o fato da troca de símbolos ser considerada uma operação atômica.

Tendo em vista que o tamanho de sequências de proteínas pode ser grande, um algoritmo cúbico pode ser impraticável quando se trata de tempo computacional gasto para se alinhar um conjunto de sequências protéicas. Miller e Myers [43] desenvolveram um algoritmo que possui complexidade de tempo $O(n^2 \log n)$ e $O(n)$ de espaço.

Esse algoritmo se baseia no paradigma de lista de candidatos em que, para cada linha, todas as colunas são candidatas à posição de início do bloco de buracos. Assim como, para cada coluna, todas as linhas são candidatas ao início do bloco de buracos. São mantidas $n + 2$ listas, uma para todas as linhas e uma para cada coluna, pois a cada linha pode-se sobrescrever a lista.

3.3 Modelos de Substituição

A pontuação para alinhamento de letras pode ser feita não só por valores fixos de *match* e *mismatch*, mas também por matrizes de substituição. Matrizes de substituição indicam a possibilidade de mutação de um aminoácido i por um aminoácido j, ou seja, os valores de *match* e *mismatch* para os aminoácidos. Elas são geradas a partir de diversos alinhamentos

Algoritmo 4 Algoritmo cúbico para alinhamento com função logarítmica [76]

```
Entrada: alignment1, alignment2, gop, c
Saída: matrix
length1 ← alignment1.length
length2 ← alignment2.length
matrix [length1 + 1,length2 + 1]
matrix[0,0] ← 0
for i ← 0 to length1 do
  matrix[i,0] ← gop + c × log(i)
for j ← 1 to lenght2 do
  matrix[0,j] ← gop + c × log(j)
for  i ← 1 to length1 do
  for j ← 1 to length2 do
    M ← matrix[i - 1,j - 1] + scoreAlign
    for k ← 1 to i do
      if M < matrix[k - 1,j] + gop + c × log(i-k) then
        M ← matrix[k - 1,j] + gop + c × log(i-k)
    for k ← 1 to j do
      if M < matrix[i,k - 1] + gop + c × log(j-k) then
        M ← matrix[i,k - 1] + gop + c × log(j-k)
    matrix[i,j] ← M
return matrix
```

de pares de sequências de aminoácidos. As matrizes mais utilizadas são as da família PAM [12] e as da família BLOSUM [27].

As matrizes PAM (do inglês, *Percent of Accepted Mutation*) foram desenvolvidas por Margaret Dayhoff [12] em 1978. A família PAM foi construída com base em 1.572 mutações observadas em 71 famílias de proteínas. Foram considerados alinhamentos de sequências de proteínas com pelo menos 85% de identidade. De forma geral, a família PAM é definida como na Equação 3.4

$$PAM_k(i,j) = \left\lceil 10 \times log \frac{F^k(i,j)}{freq(a_j)} \right\rceil \tag{3.4}$$

onde $F(i,j)$ é a probabilidade de um aminoácido a_i mutar para um a_j em sequências com distância 1-PAM, $freq(a_j)$ a frequência do aminoácido a_j, e k é o índice PAM.

A distância 1-PAM equivale à substituição de 1 a cada 100 aminoácidos e indica que pode haver a troca de um aminoácido por qualquer outro. Quanto maior os valores de k para as matrizes k-PAM, mais distantes são as sequências. Note que: duas proteínas que possuem distância k-PAM não são necessariamente diferentes em k% aminoácidos.

Matrizes BLOSUM (BLOcks of amino acid SUbstitution Matrix) foram desenvolvidas

matriz PAM	matriz BLOSUM
PAM_{100}	$BLOSUM_{90}$
PAM_{120}	$BLOSUM_{80}$
PAM_{160}	$BLOSUM_{60}$
PAM_{200}	$BLOSUM_{50}$
PAM_{250}	$BLOSUM_{45}$
PAM_{400}	$BLOSUM_{30}$

Tabela 3.4: Matrizes PAM, na primeira coluna, com as correspondentes BLOSUM, na segunda coluna, de acordo com a entropia

por Henikoff e Henikoff [27] em 1992, baseadas em alinhamentos locais de 2000 blocos de sequências que caracterizaram mais de 500 grupos de proteínas. A Equação 3.5 mostra a fórmula geral da construção das matrizes BLOSUM.

$$BLOSUM_k(i,j) = \left\lceil log_2 \frac{freq_k(a_i, a_j)}{freq_k(a_i) freq_k(a_j)} \right\rceil \tag{3.5}$$

Na família BLOSUM, em cada matriz $BLOSUM_k$, k indica que os alinhamentos utilizados para gerar a matriz tem no máximo $k\%$ de identidade.

Apesar das matrizes BLOSUM serem derivadas de alinhamentos de blocos e das matrizes PAM serem derivadas das taxas de mutação, elas podem ser comparadas utilizando entropia relativa [27]. Através da entropia dos aminoácidos, determinada pela Equação 3.6, é possível determinar qual das matrizes PAM é equivalente a qual matriz da família BLOSUM, como mostra a Tabela 3.4. Por exemplo, a matriz PAM_{100} é equivalente a matriz $BLOSUM_{90}$ de acordo com a entropia.

$$H = \sum_{i=1}^{20} \sum_{j=1}^{i} freq(a_i) freq(a_j) \sigma(a_i, a_j) \tag{3.6}$$

onde H é a entropia, $\sigma(a_i, a_j)$ a troca de a_i por a_j, a_x são aminoácidos e $freq(a_x)$ é a frequência de um aminoácido a_x.

Em um alinhamento caracterizado por frequências objetivo, a entropia mede a informação média disponível por posição que distingue um dado alinhamento de um conjunto de caracteres do alinhamento ao acaso desses caracteres. Um alto valor de entropia pode ser obtido através de alinhamentos mais curtos, enquanto que, valores mais baixos podem ser obtidos por alinhamentos mais longos [1]. A entropia relativa é 0 quando a distribuição observada do par de frequências é a mesma da esperada, e aumenta quando

as duas distribuições se tornam mais distinguíveis. Altschul [1] mostrou que a entropia diminui com o aumento do índice k PAM. Para matrizes BLOSUM, a entropia aumenta com o aumento da porcentagem de clusterização.

3.3.1 DCMut

Dayhoff e colegas [12] construíram matrizes de probabilidade, família PAM, dependentes do tempo para expressar o modelo de substituição de aminoácidos. No entanto, a substituição de aminoácidos pode também ser expressa em função de matrizes de IRMs (do inglês *Instantaneus Rate Matrix*).

Proposto por Kosiol e Goldman [38], o DCMut (*Directed Computation with Mutabilities*) é uma variação do modelo proposto por Dayhoff e colegas em 1978. Utilizando os mesmos dados usados por Dayhoff, com o DCMut foi possível construir diferentes matrizes IRMs a partir das matrizes de probabilidade PAM. Uma matriz IRM, aqui denominada de Q, se relaciona com o modelo de Dayhoff (matriz $P(t)$ em um tempo t) de acordo com a Equação 3.7.

$$P(t) = I + tQ + \frac{(tQ)^2}{2!} + \frac{(tQ)^3}{3!} + ... \tag{3.7}$$

onde I é a matriz identidade e $Q = (q_{ij})_{i,j=1,...,20}$.

DCMut considera apenas as trocas observadas e as mutações de aminoácidos. Neste modelo, cada elemento q_{ij} da matriz Q é determinado pela Equação 3.8, para $i \neq j$.

$$q_{ij} = \frac{m_i n_{ij}}{\sum_{k \neq i}^{20} n_{ik}} \tag{3.8}$$

onde n_{ij} corresponde ao número de ocorrências de um aminoácido i em uma sequência e um aminoácido j em outra para todos pares de sequências, $q_{ii} = -\sum_j q_{ij}$. As taxas de mutações (m_i), que medem a troca de um aminoácido i por um aminoácido j, são calculadas como na Equação 3.9.

$$m_i = \frac{\sum_{j \neq i} n_{ij}}{\sum_{k=1}^{k=20} n_{ik}} \tag{3.9}$$

3.3.2 Jones-Taylor-Thornton

O modelo de Jones-Taylor-Thornton [33] é similar ao modelo de Dayhoff PAM [12], diferenciando-se por realizar a recontagem do número de trocas observadas nos aminoácidos. A amostragem de proteínas utilizadas é maior que a de Dayhoff e a distância é medida em unidades da fração esperada de aminoácidos trocados.

A matriz que mede a taxa de substituição dos aminoácidos é construída em três etapas:

- Agrupamento das sequências em famílias homólogas.
- Cálculo das mutações observadas entre sequências altamente similares.
- Listagem das frequências de mutações que ocorreram ao acaso.

O modelo utiliza um método para inferir relações filogenéticas entre conjuntos de sequências (*pairwise present-day ancestor scheme*). Considera a distribuição em triplas de aminoácidos entre duas sequências. Se duas triplas são suficientementes idênticas entre duas sequências elas mostram-se potencialmente homólogas. Pega-se a maior sequência e constrói-se uma tabela de hash contendo as frequências com que as triplas ocorrem nesta sequência. Comparam-se então as frequências das triplas da menor sequência com a da maior, por uma pontuação S, como na Equação 3.10.

$$S = \frac{\sum_{pqr=AAA}^{VVV} min(f_a^{pqr}, f_b^{pqr})}{min(n_a, n_b) - 2} \tag{3.10}$$

onde f_a^{pqr} e f_b^{pqr} são as frequências da tripla pqr de aminoácidos nas sequências a e b, e n_a e n_b são os tamanhos das sequências.

O valor de S normalizado é usado para determinar a porcentagem de identidade, I, das sequências, como na Equação 3.11.

$$I \approx 100S^{0,3912} \tag{3.11}$$

A fim de alinhar apenas as sequências com pontuação de triplas corretas, selecionaram-se apenas as que tiverem $I \geq 45\%$ e, em seguida, excluem-se os pares de sequências cuja pontuação da identidade do alinhamento é menor ou igual a 85%. Combinando esta heurística de identidade com um modelo de programação dinâmica constróem matrizes de identidade baseadas nas matriz PAM de Dayhoff.

A matriz criada utilizando o modelo JTT, chamada de PET91, equivalente a 250-PAM, pode ser observada na Tabela 3.5. Esta matriz foi gerada com base em 16.130 proteínas da distribuição 15.0 do banco de dados de proteínas SWISS-PROT [3].

3.3.3 PMB

O modelo *Probability Matrix from Blocks* (Matriz de Probabilidade de Blocos), o PMB [71], foi desenvolvido em 2003 por Veerassamy e colaboradores. Este modelo é derivado das

A	2																		
R	-1	5																	
N	0	0	3																
D	0	-1	2	5															
C	-1	-1	-1	-3	11														
Q	-1	2	0	1	-3	5													
E	-1	0	1	4	-4	2	5												
G	1	0	0	1	-1	-1	0	5											
H	-2	2	1	0	0	2	0	-2	6										
I	0	-3	-2	-3	-2	-3	-3	-3	-3	4									
L	-1	-3	-3	-4	-3	-2	-4	-4	-2	2	5								
K	-1	4	1	0	-3	2	1	-1	1	-3	-3	5							
M	-1	-2	-2	-3	-2	-2	-3	-3	-2	3	3	-2	6						
F	-3	-4	-3	-5	0	-4	-5	-5	0	0	2	-5	0	8					
P	1	-1	-1	-2	-2	0	-2	-1	0	-2	0	-2	-2	-3	-6				
S	1	-1	1	0	1	-1	-1	1	-1	-1	-2	-1	-1	-2	1	2			
T	2	-1	1	-1	-1	-1	-1	-1	-1	1	-1	-1	0	-2	1	1	2		
W	-4	0	-5	-5	1	-3	-5	-2	-3	-4	-2	-3	-3	-1	-4	-3	-4	-15	
Y	-3	-2	-1	-2	2	-2	-4	-4	4	-2	-1	-3	-2	5	-3	-1	-3	0	9
V	1	-3	-2	-2	-2	-3	-2	-2	-3	4	2	-3	2	0	-1	-1	0	-3	-3	4
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V

Tabela 3.5: Matriz construída com o modelo JTT equivalente a 250-PAM, os valores foram multiplicados por 10 e arredondados para cima [33].

matrizes BLOSUM [27] e pode ser usado para análise evolucionária de sequências de proteínas.

PMB utiliza blocos de banco de dados de alinhamentos de proteínas e distâncias evolucionárias aditivas para aproximar as probabilidades de substituição de aminoácidos como uma função da distância evolucionária. Veerassamy e colaboradores analisaram blocos de banco de dados [28] excluindo os tendenciosos em sua composição. Ainda utilizaram o programa BLOSUM [27] para observar a frequência de cada tipo de substituição de aminoácidos, variando a porcentagem de clusterização de 30% a 100% incrementando, de 2% em 2%.

Para cada clustering com porcentagem c de clusterização é possível determinar diferentes matrizes, M_c, de mutabilidade que descrevem a frequência de substituição de um aminoácido por outro. Para cada matriz é possível determinar M_{ij} pela Equação 3.12,

$$M_{ij} = \frac{F_{ij}}{\sum_{j=1}^{20} F_{ij}} \tag{3.12}$$

onde F_{ij} é a frequência de substituição, de um aminoácido i por uma aminoácido j, obtida de uma matriz BLOSUM.

A frequência observada para cada aminoácido é definida pela Equação 3.13, onde π é o vetor de frequências. Cada entrada do vetor corresponde à frequência para um aminoácido, cujo valor é obtido pela divisão da soma das frequências F_{ij} para uma linha da matriz pela soma de todas as entradas da matriz.

$$\pi_i = \frac{\sum_{j=1}^{20} F_{ij}}{\sum_{i'=1}^{20} \sum_{j=1}^{20} F_{i'j}} \tag{3.13}$$

A matriz de mutabilidade e o vetor de frequências são usados para calcular a frequência média de substituição, $D(M)$, através da Equação 3.14.

$$D(M) = 1 - \sum_{i=1}^{20} \pi_i M_{ii} \tag{3.14}$$

Cada matriz M_c pode ser considerada uma função de P_c, sendo que P_c corresponde a distância evolucionária atual. Podemos definir a distância d_c como a distância esperada para uma distância média atual P_c, como na Equação 3.15.

$$d_c = D(M_c(P_c)) \tag{3.15}$$

Utilizando uma fórmula de correlação C, pode ser estimada uma distância evolucionária atual, P'_c, como uma função da distância d_c, como mostra a Equação 3.16.

$$P'_c = C(d_c) \tag{3.16}$$

Para cada porcentagem de clusterização, c, é possível determinar uma IRM (*Instantaneous Rate Matrix*), denotada por Q_c, assim como apresentada na Equação 3.17.

$$Q_c = \frac{ln(M_c)}{P'_c} \tag{3.17}$$

Finalmente, a uma matriz PMB(P) pode ser definida através da Equação 3.18, onde P é a distância evolucionária em unidades PAM, onde Q pode ser determinada na Equação 3.17.

$$PMB(P) = exp(0.01PQ) \tag{3.18}$$

A melhor matriz PMB encontrada foi a Q_{68} e pode ser observada na Tabela 3.6. A primeira coluna corresponde à frequência de cada aminoácido.

7.56%	A	0	0.0444	0.0216	0.0218	0.0249	0.0380	0.0544	0.1292	0.0164	0.0375	0.0766	0.0505	0.0380	0.0236	0.0502	0.2196	0.0759	0.0045	0.0210	0.1499
5.38%	R	0.0624	0	0.0530	0.0311	0.0084	0.1053	0.0831	0.0452	0.0357	0.0250	0.0559	0.2675	0.0201	0.0135	0.0274	0.0733	0.0637	0.0118	0.0264	0.0354
3.77%	N	0.0434	0.0757	0	0.1468	0.0124	0.0726	0.0735	0.1092	0.0656	0.0266	0.0336	0.0961	0.0178	0.0211	0.0342	0.1570	0.0946	0.0051	0.0296	0.0418
4.47%	D	0.0368	0.0374	0.1237	0	0.0087	0.545	0.1914	0.0737	0.0230	0.0130	0.0276	0.0574	0.0043	0.0132	0.0310	0.0992	0.0496	0.0046	0.0166	0.0228
2.85%	C	0.0660	0.0158	0.0164	0.0136	0	0.0110	0.0092	0.0284	0.0150	0.0297	0.0476	0.0168	0.0132	0.0307	0.0112	0.0545	0.0366	0.0048	0.0184	0.0540
3.39%	Q	0.0846	0.1670	0.0807	0.0718	0.0092	0	0.2250	0.0673	0.0508	0.0211	0.0739	0.1782	0.0515	0.0183	0.0391	0.1155	0.0914	0.0135	0.0335	0.0505
5.35%	E	0.0769	0.0835	0.0518	0.1600	0.0049	0.1426	0	0.0433	0.0312	0.0237	0.0398	0.1222	0.0130	0.0117	0.0525	0.0799	0.0650	0.0072	0.0190	0.0448
7.80%	G	0.1251	0.0311	0.0527	0.0422	0.0104	0.0293	0.0296	0	0.0161	0.0109	0.0291	0.0325	0.0089	0.0189	0.0261	0.0886	0.291	0.0055	0.0126	0.0201
3.00%	H	0.0421	0.0639	0.0823	0.0343	0.0142	0.0573	0.055	0.0418	0	0.0164	0.0337	0.0551	0.0112	0.0240	0.0184	0.0614	0.0618	0.0093	0.0703	0.0272
5.99%	I	0.0473	0.0224	0.0167	0.0097	0.0141	0.0120	0.0212	0.0142	0.0082	0	0.3432	0.0258	0.0804	0.0609	0.0143	0.0288	0.0590	0.0069	0.0239	0.5186
9.58%	L	0.0604	0.0314	0.0132	0.0129	0.0142	0.0261	0.0222	0.0237	0.0106	0.2145	0	0.0257	0.1206	0.1015	0.0184	0.0242	0.0418	0.0192	0.0271	0.1403
5.20%	K	0.0734	0.2766	0.0697	0.0493	0.0092	0.1162	0.1257	0.0488	0.0319	0.0298	0.0473	0	0.0218	0.0133	0.0424	0.0846	0.0724	0.0077	0.0232	0.0388
2.19%	M	0.1309	0.0494	0.0306	0.0089	0.0172	0.0797	0.0317	0.0317	0.0154	0.2197	0.5271	0.0518	0	0.0941	0.0195	0.0593	0.0852	0.0127	0.0366	0.1509
4.50%	F	0.0396	0.0161	0.0177	0.0131	0.0194	0.0138	0.0139	0.0327	0.0160	0.0810	0.2160	0.0153	0.0458	0	0.0214	0.0447	0.0327	0.0354	0.1859	0.0658
4.20%	P	0.0903	0.0350	0.0307	0.0329	0.0076	0.0316	0.0667	0.0485	0.0131	0.0203	0.0419	0.0525	0.0101	0.0229	0	0.0940	0.0589	0.0087	0.0193	0.0489
6.82%	S	0.2432	0.0578	0.0868	0.0650	0.0228	0.0574	0.0626	0.1013	0.0271	0.0253	0.0340	0.0645	0.0191	0.0295	0.0579	0	0.2760	0.0099	0.0253	0.0462
5.64%	T	0.1016	0.0608	0.0632	0.0393	0.0185	0.0549	0.0616	0.0402	0.0329	0.0626	0.0710	0.0667	0.0331	0.0261	0.0439	0.3325	0	0.0137	0.0238	0.1416
1.57%	W	0.0215	0.0404	0.0123	0.0131	0.0087	0.0292	0.0243	0.0272	0.0177	0.0263	0.1167	0.0256	0.0177	0.1012	0.0233	0.0430	0.0490	0	0.1381	0.0445
3.60%	Y	0.0441	0.0394	0.0311	0.0207	0.0146	0.0316	0.0282	0.0274	0.0587	0.0398	0.0722	0.0336	0.0223	0.2326	0.0226	0.0479	0.0373	0.0604	0	0.0435
7.15%	V	0.1585	0.0267	0.0221	0.0143	0.0215	0.0240	0.0335	0.0220	0.0114	0.4345	0.1881	0.0282	0.0463	0.0415	0.0287	0.0441	0.1118	0.0098	0.0219	0
Freq		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	
AA																							

Tabela 3.6: Matriz Q_{68} construída com o modelo PMB [71]

3.3.4 Modelo de Categorias

Certos tipos de mutações são mais frequentes que outras. Em sequências de aminoácidos, esta característica é bem marcante no sentido em que aminoácidos com carga elétrica positiva dificilmente são trocados por aminoácidos com carga elétrica negativa [56], e vice-versa.

O modelo de categorias utiliza essa idéia de que nem todos os aminoácidos podem ser trocados com a mesma frequência. Baseia-se no modelo de dois parâmetros de Kimura [36] para sequências de nucleotídeos, adaptando-o para aminoácidos.

O modelo de dois parâmetros de Kimura afirma que é mais provável que haja transições do que transversões. Transição é a troca de um nucleotídeo por outro do mesmo grupo (púricos por púricos, ou pirimídicos por pirimídicos). Transversão é a troca entre nucleotídeos de grupos diferentes (púricos por pirimídicos, ou pirimídicos por púricos). A distância de Kimura é dada pela Equação 3.19, onde s é a porcentagem de transições que ocorreram de uma sequência para outra e v a porcentagem de transversões.

$$K = -\frac{ln(1-2s-v)}{2} - \frac{ln(1-2v)}{4} \tag{3.19}$$

No modelo de categorias, as trocas entre aminoácidos de mesma categoria sempre são permitidas, já as que alteram a categoria são permitidas apenas em certas frações do tempo. Esta fração tem valor 1 quando todas as trocas são permitidas e 0 quando são quase impossíveis de acontecer [19]. Um dos padrões para classificação de aminoácidos, que podem ser utilizados para medir as proporções de transições e transversões, é o de George-Hunt-Barker [23].

Na utilização desde modelo para observar a distância entre duas sequências, é importante analisar as sequências por completo. Esta análise inclui as posições que não sofreram mutações, para que o valor da distância não tenda a ser maior do que deveria.

3.4 Alinhamento Múltiplo de Sequências

Para encontrar características conservadas, como *motifs*, de famílias de proteínas, é necessário o uso do alinhamento múltiplo de sequências (MSA, do inglês *Multiple Sequence Alignment*).

Sejam S_1, S_2, ..., S_n sequências de caracteres e espaços, sendo que $|S_i|$ representa o tamanho de S_i. Um alinhamento A entre as sequências S_1 a S_n, é um mapeamento de S_1 a S_n em outras sequências S_i' a S_n' com os mesmos caracteres e na mesma ordem de S_i a S_n, sendo possível inserir espaços de forma que $|S_1|'=|S_2|'=...=|S_n|'$.

O algoritmo de Needleman e Wunsch pode ser adaptado de forma a generalizar o alinhamento de duas para mais sequências. Porém este fica limitado a um pequeno número

de sequências e não muito longas [41]. Este algoritmo requer $\Omega(k2^k n^k)$ de tempo e $\Omega(n^k)$ de espaço, onde k é o número de sequências e n o comprimento da maior sequência, para calcular um alinhamento ótimo.

A fim de produzir um alinhamento múltiplo global ótimo, em 1989, Lipman e colaboradores [41] desenvolveram o programa MSA, que utiliza uma variante multidimensional do modelo de programação dinâmica. No entanto, a complexidade do algoritmo é exponencial. Em 1994, Lusheng Wang e Tao Jiang [75] provaram que o problema do alinhamento múltiplo de sequências é NP-Difícil.

Assim como no alinhamento de pares de sequências, no alinhamento múltiplo, busca-se alcançar aquele que melhor represente o cenário evolucionário. Desta forma devem ser considerados os diferentes eventos de mutação que podem ocorrer na evolução. Estes eventos são substituição, inserção e remoção de aminoácidos. A probabilidade de ocorrência desses eventos depende do código genético, assim como dos efeitos fenotípicos de mutação, sendo mais frequentes substituições conservativas (entre aminoácidos com propriedades bioquímicas semelhantes) que outros tipos de substituições.

O alinhamento múltiplo de sequências é baseado na escolha da função objetivo (modelo de comparação) e na otimização dessa [50]. Uma função objetivo permite determinar a qualidade de um alinhamento. No entanto, uma função objetivo matematicamente perfeita nem sempre é biologicamente perfeita, pois esta deveria incorporar informações dificilmente encontradas sobre estrutura, história e função das sequências.

Geralmente, os algoritmos utilizados no alinhamento múltiplo de sequências são heurísticas que não garantem um alinhamento ótimo como resultado. Obter um alinhamento matematicamente ótimo é uma tarefa complexa mesmo quando se assumem que as sequências e a função objetivo são biologicamente perfeitas.

Um bom alinhamento depende do modelo utilizado para penalidade de buracos: linear, afim, ou ainda modelos logarítmos. Em modelos mais realísticos considera-se um buraco como um único evento mutacional, cuja penalidade é proporcional ao seu tamanho. Diferentes abordagens são utilizadas para alinhar sequências, sejam de proteínas ou de DNA, estas diferenciam-se nos métodos usados para construção do alinhamento nos tipos de pontuações e nas heurísticas em que se baseiam.

3.4.1 Abordagens para MSA

O MSA pode ser construído por meio de abordagens distintas, tais como: a progressiva, a iterativa, e a baseada em consistência, consenso, modelos ou blocos. Neste trabalho será explorada a construção de alinhadores progressivos.

O alinhamento progressivo [20, 30] é uma das maneiras mais simples de se realizar o alinhamento múltiplo. Utiliza pouco tempo e memória computacional em relação aos

outros métodos. Sua estratégia consiste na computação da matriz de distâncias de pares de sequências, construção da árvore guia a partir das distâncias e geração do alinhamento múltiplo, guiado pela árvore, através dos alinhamentos de pares.

Alinhadores progressivos utilizam a estratégia gulosa, em que uma solução ótima local é feita esperando que ela leve à uma solução ótima global. Ou seja, a escolha entre quais alinhamentos serão agrupados é feita como a ótima localmente, esperando alcançar uma solução ótima no final de todo o processo. Isto pode ser uma problema do método, pois um erro cometido no inicio do alinhamento não é mais corrigido. São exemplos de alinhadores progressivos: Clustal W [66], MUSCLE [15, 16], Pileup [13] e MultiAlign [10]. Eles serão brevemente descritos a seguir.

O Clustal W [66] utiliza esquema de pontuação inicial para abertura e extensão de buracos e matriz de substituição (PAM, BLOSUM ou Gonnet [24]) para construir a matriz de distâncias. Para construção da árvore guia, utiliza-se a matriz de distâncias. Inicialmente coloca-se uma raiz na árvore com base no valor médio das distâncias estimadas e os demais ramos são colocados de forma proporcional. Finalmente, realiza-se o alinhamento múltiplo a partir das folhas da árvore. Nesta etapa utiliza-se um esquema de normalização de pesos das sequências. Ainda realiza-se a pontuação de abertura e extensão de buracos a partir da inicial, levando em consideração a similaridade e tamanho das sequências.

O MUSCLE (*Multiple Sequence Comparison by Log-Expectation*) [16] usa uma função de perfil para alinhamento múltiplo com pontuação logarítmica e refinamento usando uma árvore dependente de particionamento. Produz um alinhamento inicial com distância *k-mer* (subsequência contigua de tamanho k, também conhecida como palavra). Em seguida melhora este alinhamento recalculando a distância entre as sequências já alinhadas usando a distância de Kimura [37] e faz um segundo alinhamento múltiplo, em ambos os casos utiliza UPGMA [49] para construção da árvore guia. Finalmente, o segundo alinhamento é refinado, percorrendo a segunda árvore, visitando primeiro as arestas mais distantes da raiz.

O Pileup [13] usa o método tradicional de programação dinâmica para o alinhamento de pares. No alinhamento múltiplo inicialmente alinha duas sequências. Em seguida alinha a terceira ao alinhamento das duas e assim sucessivamente, até que todas as sequências sejam alinhadas. Mantém uma matriz de pontuação contendo informações de todas as sequências alinhadas até o momento. Estas, por sua vez, são incluídas no alinhamento de acordo com a proximidade com as já alinhadas, por exemplo, a terceira sequência é escolhida como a que mais se parece com a primeira ou a segunda, já alinhadas.

Também baseado em programação dinâmica para o alinhamento de pares, o MultiAlign [10] é realizado em 5 etapas: alinhamento de pares de sequência, construção de uma hierarquia de *clusters* baseada na pontuação do alinhamento de pares, construção de uma árvore para alinhamento de pares dos *clusters* alinhados, pontuação dos alinhamentos

dos *clusters*, construção de uma nova hierarquia de *clusters* de acordo com a pontuação calculada, substituição do alinhamento anterior caso o novo seja melhor. O alinhamento de pares é feito sobre uma matriz com raciocínio inverso, de forma que a posição de intersecção da primeira linha com a primeira coluna seja a pontuação final do alinhamento. Os *clusters* são unidos de acordo com a maior proximidade entre eles, os mais parecidos primeiro.

A abordagem iterativa busca refinar alinhamentos iniciais, produzidos por algum outro algoritmo em diversos ciclos até que alguma condição de parada seja atingida. É subdividida em métodos determinísticos e estocásticos. Os determinísticos são mais simples e consistem em retirar uma sequência por vez do alinhamento e realinhá-las com o restante das sequências. Os métodos estocásticos interessantes por separarem bem o processo de otimização e a função objetivo. PRRP [25], SAGA [53], PRALINE [29] e InterAlign [6] são exemplos de alinhadores iterativos.

Métodos baseados em consistência, inicialmente descritos por Kececioglu [35], definem um alinhamento ótimo como aquele que está de acordo com a maior parte dos possíveis alinhamentos de pares para um conjunto de sequências. Precisa apenas de qualquer método capaz de alinhar duas sequências por vez e não de uma matriz de substituição específica. Os alinhadores T-COFFEE [51] e SAGA (com a função objetivo COFFEE [52]) utilizam esta abordagem.

Métodos baseados em consenso realizam a combinação de diferentes alinhamentos, produzidos por diferentes métodos para MSA. O M-COFFEE [73] pode ser citado como exemplo de alinhador baseado em consenso. Ele utiliza alinhador T-COFFEE para combinar rapidamente os alinhamentos fornecidos pelos outros métodos.

Os métodos baseados em modelo surgiram como alternativa para incorporação de informação sobre as sequências, na tentativa de resolver o problema de que as sequências a serem alinhadas podem não ser homólogas. A implementação pode ser feita por extensão estrutural (identifica-se um modelo estrutural no Protein Data Bank [5] utilizando o BLAST [2]) ou por homologia (utiliza um perfil ao invés de estrutura). São alinhadores baseados em modelos: 3D-COFFEE [54], DbClustal [70], PRALINE [29].

No alinhamento baseado em blocos, utilizam-se blocos conservados para guiar o alinhamento. Os blocos são alinhamentos locais de fragmentos de sequências, considerando na maioria das vezes apenas blocos sem buracos, não dependendo tanto das penalidades adotadas para buracos. DiAlign [45] é o alinhador baseado em blocos mais conhecido.

Capítulo 4

Alinhamento Progressivo

Para realização do alinhamento múltiplo progressivo é preciso construir a matriz de distâncias para as sequências que serão alinhadas. Através dessa matriz realiza-se a construção da árvore guia e então o alinhamento progressivo, agrupando as sequências seguindo a ordem sugerida pelos nós da árvore.

Além de ser muito sensível aos parâmetros utilizados, o alinhamento múltiplo progressivo geralmente não é ótimo devido à estratégia gulosa que utiliza. Em outras palavras, decisões são tomadas por parecer a melhor no momento. Faz-se a escolha por uma solução ótima local esperando que ela leve a uma ótima global [66]. Por isso, é preciso acertar na combinação dos métodos utilizados para realizar cada etapa do alinhamento progressivo, assim como nos parâmetros escolhidos.

Neste capítulo serão descritos os diferentes métodos que podem ser usados para a construção de um alinhador múltiplo progressivo. Na Seção 4.1 serão apresentados dois métodos usados para a pontuação de um alinhamento, com esta pontuação é possível construir a matriz de distâncias. Na Seção 4.2 serão descritos os métodos utilizados na construção da árvore guia. A seleção de pares é descrita na Seção 4.3. Na Seção 4.4 serão descritos os métodos usados para atribuir o peso das sequências. Os métodos de agrupamento serão descritos na Seção 4.5. Finalmente, na Seção 4.6 será descrito um método para adequar os parâmetros dos alinhadores às sequências de entrada.

4.1 Pontuação Entre as Sequências de Proteínas

Distâncias evolucionárias são fundamentais para o estudo de proteínas e, usualmente, são determinadas pelas diferenças entre os aminoácidos que compõem suas sequências. Tais diferenças podem ser devido a substituições, remoções ou inserções de aminoácidos que ocorreram no processo evolutivo. Nas Seções 4.1.1 e 4.1.2 apresentaremos dois métodos para determinar a distância entre duas sequências através da pontuação do alinhamento.

4.1.1 Pontuação Recursiva

A pontuação recursiva entre duas sequências é calculada pela pontuação do alinhamento local [64] entre elas. A idéia utilizada é a mesma do Algoritmo Recursivo da Seção 3.1.1, mas somando a pontuação destes alinhamentos. Realiza-se um primeiro alinhamento local entre as sequências e calcula-se a pontuação do mesmo. Os aminoácidos (subsequências), à esquerda e à direita, não alinhados neste alinhamento local são tomados como novas sequências e submetidos ao algoritmo de alinhamento. Obtêm-se a pontuação da chamada recursiva, que é somada a pontuação do primeiro alinhamento. Este processo é feito recursivamente até que a subsequência não alinhada no alinhamento local seja nula, ou até que o alinhamento local não consiga alinhar as sequências. Nesses casos a pontuação retornada é zero. A cada recursão a pontuação do alinhamento final para as duas sequências tem seu valor incrementado pelo valor do alinhamento local produzido pelas chamadas recursivas.

O Algoritmo 5, onde **Local** é o algoritmo local para alinhamento de pares de sequência, mostra como se realiza o cálculo da pontuação do alinhamento recursivo.

Algoritmo 5 localScore. Pseudocódigo do cálculo da pontuação local recursiva entre duas sequências

```
Entrada: sequence1, sequence2
Saída: pontuation
pontuation ← 0
if sequence1.length>0 and sequence2.length>0 then
  (seq1Algn,seq2Algn,iniAlgnSeq1,iniAlgnSeq2,fimAlgnSeq1,fimAlgnSeq2) ←
   Local(sequence1,sequence2)
  if seq1Algn.length ≠ 0 then
    scoreLocalAlignment ← scoreAlgnLocal (seq1Algn,seq2Algn)
    scoreLeft ← localScore(sequence1[1:iniAlgnSeq1-1],sequence2[1:iniAlgnSeq2-1])
    scoreRight ← localScore(sequence1[fimAlgnSeq1+1:sequence1.length],
    sequence2[fimAlgnSeq2+1:sequence2.length])
    distance ← scoreLeft + scoreLocalAlignment + scoreRight
return pontuation
```

Sejam as sequências `GGGAAGCATAGTGCGTGAT` e `TATTGAAGCATTAGGGCGGT`, as mesmas utilizadas no exemplo da Seção 3.1.1. Observando ainda o retorno das chamadas recursivas temos que a pontuação local é a dada pela soma das pontuações dos alinhamentos. A pontuação, p, é a seguinte:

$p = pL_1 + pA_1 + pR_1 = pA_2 + pA_1 + pL_2 + pA_3 = pA_2 + pA_1 + pL_3 + pA_4 + pR_2 + pA_3 = 0 + 11 + 0 + 2 + 0 + 2 = 15$

onde pA_x é a pontuação do alinhamento A_x da Figura 3.1. Note que quando não é possível

fazer o alinhamento local das sequências a pontuação é zero, como acontece na recursão L_1.

A pontuação do alinhamento recursivo, Seção 3.1.1 se difere da pontuação recursiva. Enquanto no alinhamento recursivo deseja-se considerar a pontuação de um alinhamento por completo, a pontuação recursiva considera-se apenas a pontuação de alinhamentos que representam as regiões mais conservadas das sequências.

4.1.2 Pontuação Logarítmica

Nesta seção apresentamos no Algoritmo 6 um método que determina a pontuação entre pares de sequências tendo como base o alinhamento de pares com pontuação logarítmica para penalidade de *gaps*, descrito na Seção 3.2.3.

Algoritmo 6 Pseudocódigo do cálculo da potuação do alinhamento de duas sequências com pontuação logarítmica

```
Entrada: sequence1, sequence2, match, mismatch, gop, c
Saída: distance
score ← 0
size_gap ← 0
open_gap ← false
for i ← 1 to sequence1.length do
  if sequence1 ≠ "-"and sequence2 ≠ "-" then
    if open_gap then
      score ← score + c × log(size_gap)
      size_gap ← 0
      open_gap ← false
    if sequence1[i] = sequence2[i] then
      score ← score + match
    else
      score ← score + mismatch
  else
    if open_gap then
      size_gap ← size_gap + 1
    else
      score ← score + gop
      size_gap ← size_gap + 1
      open_gap ← true
if open_gap then
  score ← score + c × log(size_gap)
return score;
```

Ainda utilizando as mesmas sequências do exemplo do alinhamento local, mas dessa vez como entrada para o Algoritmo 6, temos o seguinte alinhamento:

```
GG--GAAGCA-TAGTGCGTGAT
TATTGAAGCATTAGGGCG--GT
```

A pontuação logarítmica, p, calculada a partir da pontuação do alinhamento, utilizando as pontuações *match=6*, *mismatch=-1*, *gop*=-8 e $c = -2$ é a seguinte:

$p = -1 - 1 + (-8 - 2 \times \log(2)) + 6 + 6 + 6 + 6 + 6 + 6 + -8 + 6 + 6 + 6 - 1 + 6 + 6 + 6 + (-8 - 2 \times \log(2)) + -1 + 6 = -2 - 10 + 36 - 8 + 18 - 1 + 18 - 10 - 1 + 6 = 46$

Determinadas as pontuações dos alinhamentos entre os pares de sequências, seja pela pontuação local recursiva, descrita na Seção 4.1.1 ou pela logarítmica, descrita na Seção 4.1.2, estas são convertidas para os valores de distâncias. Primeiro realiza-se a normalização dos valores das pontuações como na Equação 4.1, para que variem entre 0 e 1. Após a normalização, as pontuações são subtraídas de 1 para determinar as distâncias.

$$p = 1 - \frac{score - min}{max - min} \tag{4.1}$$

onde *score* é a pontuação do alinhamento de um par de sequências, *min* e *max* são respectivamente as pontuações mínima e máxima em relação a todos os alinhamentos de pares.

4.2 Árvore Guia

Uma árvore filogenética para um conjunto de objetos biológicos S é uma árvore rotacionada em que as folhas representam os objetos pertencentes a S, e os nós internos representam seus antecessores [18].

Seja S um conjunto de objetos e $\delta : S \times S \rightarrow \Re^+$ uma função. Então δ é uma métrica para S e satisfaz três probriedades:

- Para todo par $a, b \in S, \delta(a, b) = 0 \iff a = b$.
- Para todo par $a, b \in S, \delta(a, b) = \delta(b, a)$.
- Para toda trinca $a, b, c \in S, \delta(a, b) \leq \delta(a, c) + \delta(c, b)$.

Considerando uma matriz de distâncias M em que cada entrada representa uma métrica para um conjunto S. Esta matriz é chamada de aditiva se a partir dela é possível construir uma árvore T em que para toda folha $a, b, M[a, b] = {d_{ab}}^T$ [18]. Onde d é a distância entre as folhas a e b da árvore. Neste caso, a árvore construída é chamada de aditiva.

Uma árvore T é chamada de ultramétrica se a distância (soma dos pesos das arestas) de qualquer caminho da raiz a qualquer folha da árvore é sempre a mesma [18].

Para um conjunto S de objetos biológicos e $\delta : S \times S \rightarrow \Re^+$ uma métrica para S. Então δ é ultramétrica para S se satifaz a seguinte condição:

- Para toda trinca $a, b, c \in S$ ou $\delta(a, b) = \delta(c, b)$ ou $\delta(a, c) \leq \delta(a, b) = \delta(b, c)$ ou $\delta(b, c) \leq \delta(b, a) = \delta(a, c)$.

A construção de uma árvore guia é feita com base na distância entre as sequências que se deseja alinhar. O alinhador progressivo segue os ramos da árvore para selecionar e agrupar as sequências, de acordo com os métodos de seleção de pares que serão descritos na Seção 4.3.

4.2.1 UPGMA

O método UPGMA (*Unweighted Pair Group Method with Arithmetic Mean*) é indicado para a construção de filogenia molecular quando a taxa de substituição dos aminoácidos é quase constante [49].

A partir da matriz de distâncias entre todos os pares de sequências, constrói-se a árvore filogenética, cujas folhas são as próprias sequências. A árvore é formada agrupando-se as subárvores existentes, sempre em pares e usando os dois nós com menor distância corrente. O agrupamento se dá pela criação de um novo nó, cuja distância entre ele e os demais (excluindo os seus filhos) é obtida como na Equação 4.2.

$$d_{xj} = \sum_{i \in A, j \in B} \frac{d_{ij}}{n} \tag{4.2}$$

onde A e B são as subárvores, x é o novo nó ($x \in A$), d_{ij} é a distância de uma folha i de A (filha de x) a uma folha j de B, e n o número de distâncias d_{ij}.

O tamanho, L, dos ramos das árvores é dado pela Equação 4.3, onde L conecta um nó $i \in A$ e $j \in B$, por meio do novo nó x.

$$L_{xi} = L_{xj} = \frac{d_{ij}}{2}. \tag{4.3}$$

Considerando cada nó como uma árvore folha e a matriz da Tabela 4.1 como a matriz de distâncias inicial, para as sequências de 1 a 5. Podemos notar que a menor distância é entre os nós 4 e 5, estes são unidos formando o nó x da árvore apresentada na Figura 4.1.

É preciso calcular a distância entre o novo nó x e os demais nós da árvore. Utilizando a Equação 4.2, temos:

$$d_{x1} = \frac{d_{14}+d_{15}}{2} = \frac{39+41}{2} = 40$$

	1	2	3	4	5
1	-				
2	22	-			
3	39	41	-		
4	39	41	18	-	
5	41	43	20	10	-

Tabela 4.1: Matriz de distâncias inicial para construção da árvore pelo método UPGMA

	x	1	2	3
x	-			
1	40	-		
2	42	22	-	
3	19	39	41	-

Tabela 4.2: Matriz de distâncias do UPGMA adicionando-se o nó x, unindo os nós 4 e 5

$$d_{x2} = \frac{d_{24}+d_{25}}{2} = \frac{41+43}{2} = 42$$

$$d_{x3} = \frac{d_{34}+d_{35}}{2} = \frac{18+20}{2} = 19$$

E também o comprimento dos ramos que ligam o nó x ao nó 4, e o nó x ao nó 5. Utilizando a Equação 4.3 temos:

$$L_{x4} = L_{x5} = \frac{d_{45}}{2} = \frac{10}{2} = 5$$

Com as novas distâncias podemos construir a matriz de distâncias incluindo o nó x, como mostra a Tabela 4.2 (a matriz representada na tabela é uma matriz resumida, os nós 4 e 5 continuam representados na matriz real).

Agora a menor distância, da matriz da Tabela 4.2, ocorre entre os nós x e 3. Estes nós são unidos e surge o nó y. As distâncias entre o nó y e os demais nós da árvore são:

$$d_{y1} = \frac{d_{13}+d_{14}+d_{15}}{3} = \frac{39+39+41}{3} = 39,7$$

$$d_{y2} = \frac{d_{23}+d_{24}+d_{25}}{3} = \frac{41+41+43}{3} = 41,7$$

E os novos ramos da árvore, em relação ao nó y, terão tamanho:

$$L_{y3} = L_{y4} = L_{y5} = \frac{d_{x3}}{2} = \frac{19}{2} = 9,5$$

	y	1	2
y	-		
1	39,7	-	
2	41,7	22,0	-

Tabela 4.3: Matriz de distâncias do UPGMA adicionando-se o nó y, unindo os nós x e 3

$$\text{e } L_{xy} = 9,5 - 5 = 4,5$$

Com esses valores calculamos uma nova matriz de distâncias, apresentada na Tabela 4.3.

Os nós mais próximos a esta altura são nós 1 e 2, que são unidos formando o nó z. O tamanho dos novos ramos da árvore, com a inclusão de z, são:

$$L_{1z} = L_{2z} = \frac{d_{12}}{2} = \frac{22}{2} = 11$$

Resta agora unir os nós y e z, dando origem ao nó r. A distância entre y e z é dada por:

$$d_{yz} = \frac{d_{13}+d_{14}+d_{15}+d_{23}+d_{24}+d_{25}}{6} = \frac{39+39+41+41+41+43}{6} = 40,7$$

E os tamanhos dos ramos em relação ao nó r são:

$$L_{r1} = L_{r2} = L_{r3} = L_{r4} = L_{r5} = d_{xz}/2 = 20,35$$

$$L_{rz} = 20,35 - d_{1z} = 20,35 - d_{2z} = 20,35 - 11 = 9,35$$

$$L_{ry} = 20,35 - d_{3y} = 20,35 - d_{4y} = 20,35 - d_{5y} = 20,35 - 9,5 = 10,85$$

Finalmente, temos a árvore completa e o comprimento de todos os ramos, como mostra a Figura 4.1. Note que esta árvore não é ultramétrica, pois os nós 3, 4 e 5 ferem a propriedade de distância ultramétrica. A distância entre os nós 3 e 4 é 19, enquanto a distância entre os nós 4 e 5 é 10.

É importante lembrar que o método pode produzir erros topológicos quando a taxa de mutação dos ramos da árvore não for constante [46]. Neste caso não é possível construir uma árvore ultramétrica.

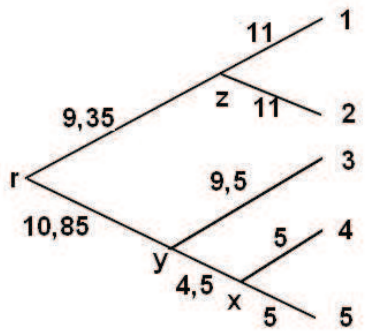


Figura 4.1: Árvore construída com o método UPGMA, com base na matriz de distâncias da Tabela 4.1

4.2.2 Neighbor Joining

O método Neighbor Joining [61] baseia-se no princípio de evolução mínima a cada união de nós vizinhos, mas não examina todas as topologias possíveis. Além da topologia da árvore fornece também o tamanho dos ramos.

A árvore é construída unindo-se a cada iteração nós que forem vizinhos. Nós vizinhos são definidos como pares de nós conectados por um único nó interior em uma bifurcação da árvore sem raiz.

Inicialmente, toma-se a árvore em formato de estrela, onde todos os nós são vizinhos. A cada união de dois vizinhos, cria-se um novo nó, pai, que será considerado vizinho dos que não sofreram união e os nós filhos não serão mais considerados vizinhos dos demais, pois agora existe mais de um nó entre eles e o nó central da árvore estrela. A cada nó criado, calcula-se a distância entre ele e os demais nós da árvore, assim como o tamanho dos novos ramos da árvore. Com a criação do novo nó é possível calcular uma nova matriz de distâncias, que o conterá.

O algoritmo pode ser resumido da seguinte forma:

- Calcula-se a soma, S_{ij}, do tamanho dos ramos para todos os pares de OTUs (*Operational Taxonomic Units*, folhas da árvore ou nós externos, que podem representar famílias, gêneros, espécies, genes) i,j, como na Equação 4.4.

$$S_{ij} = \frac{1}{2(n-2)} \sum_{k=1,k\neq i\neq j}^{n} (d_{ik} + d_{jk}) + \frac{1}{2} d_{ij} + \frac{1}{n-2} \sum_{l=1,l\neq i\neq j}^{n-1} \sum_{k=l+1,k\neq i\neq j}^{n} d_{lk} \qquad (4.4)$$

  onde d_{ij} é a distância entre os nós i,j da árvore, e n o número de nós da árvore.

- Seleciona-se como vizinhos o par i,j com menor $S_{i,j}$, unindo-os.

- Obtém-se o tamanho do ramo entre o nó pai e o nó central da árvore estrela, conforme a Equação 4.5.

$$L_{pai,central} = \frac{1}{2(n-2)} \left[\sum_{k=1,k\neq i\neq j}^{n} (d_{ik} + d_{jk}) - (n-2)d_{ij} - 2 \sum_{l=1,l\neq i\neq j}^{n-1} \sum_{k=l+1,k\neq i\neq j}^{n} d_{lk} \right] \tag{4.5}$$

  onde i e j são os nós selecionados anteriormente.

- Obtém-se o tamanho dos ramos $L_{i,pai}$ e $L_{j,pai}$ criados, como nas Equações 4.6 e 4.7.

$$L_{i,pai} = \frac{d_{ij} + \frac{1}{n-2} \sum_{k=1,k\neq i\neq j}^{n} (d_{ik} - d_{jk})}{2} \tag{4.6}$$

$$L_{j,pai} = \frac{d_{ij} + \frac{1}{n-2} \sum_{k=1,k\neq i\neq j}^{n} (d_{jk} - d_{ik})}{2} \tag{4.7}$$

- Calcula-se a nova matriz de distâncias com dimensão $(n-1) \times (n-1)$, onde n é o número de nós antes da junção de vizinhos. A cada junção retiram-se duas linhas e duas colunas, correspondentes aos nós vizinhos, e acrescenta-se uma nova coluna e uma nova linha correspondente ao nó pai criado, cuja distância é obtida como na Equação 4.8.

$$d_{xk} = \frac{d_{ik} + d_{jk}}{2} \tag{4.8}$$

  onde x é o novo nó, k representa cada nó que será mantido na matriz de distâncias, e i e j são os nós unidos pelo novo nó x.

- Os passos acima são repetidos, reduzindo-se a cada iteração o número de OTUs, enquanto houver mais que três OTUs. Quando restam apenas três OTUS temos uma árvore sem raiz.

Observe a Tabela 4.4, que representa a matriz de distâncias entre as OTUs 1 a 8. Observe também a árvore da Figura 4.2a, em forma de estrela, inicial para a construção da árvore guia pelo método de Neighbour Joining.

Aplicamos a equação 4.4, para toda OTU i,j e obtemos a matriz da Tabela 4.6. Por exemplo, para $i = 1$ e $j = 2$, temos:

$$S_{12} =$$
$$\frac{1}{2(8-2)} \times [(d_{13} + d_{23}) + (d_{14} + d_{24}) + (d_{15} + d_{25}) + (d_{16} + d_{26}) + (d_{17} + d_{27}) + (d_{18} + d_{28})] +$$

	1	2	3	4	5	6	7	8
1	-							
2	7	-						
3	8	5	-					
4	11	8	5	-				
5	13	10	7	8	-			
6	16	13	10	11	5	-		
7	13	10	7	8	6	9	-	
8	17	14	11	12	10	13	8	-

Tabela 4.4: Matriz de distâncias inicial para 8 OTUs

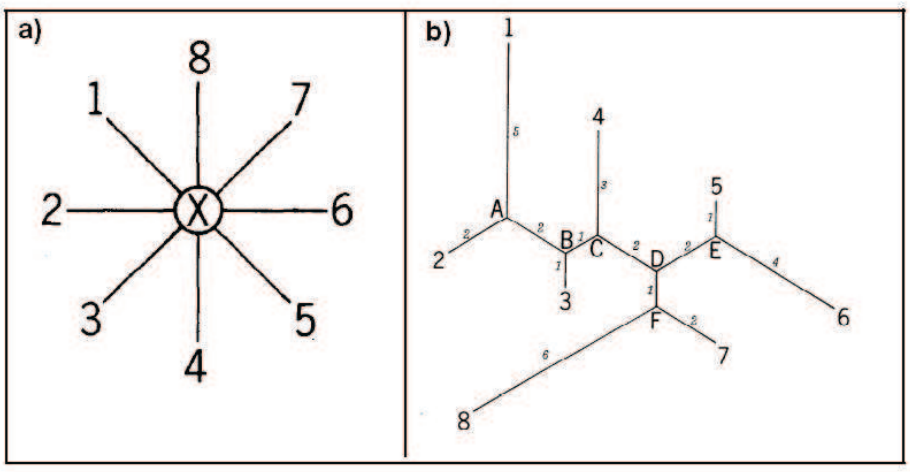


Figura 4.2: a)Árvore estrela, inicial para construção da árvore guia. b)Árvore guia construída pelo método Neighbor Joining [61]

	1	2	3	4	5	6	7	8
1	-							
2	36,67	-						
3	38,33	38,33	-					
4	39,00	39,00	38,67	-				
5	40,33	40,33	40,00	39,67	-			
6	40,33	40,33	40,00	39,67	37,00	-		
7	40,17	40,17	39,83	39,50	38,83	38,83	-	
8	40,17	40,17	39,83	39,50	38,83	38,83	37,67	-

Tabela 4.5: S_{ij} iniciais para as 8 OTUs, de acordo com as distâncias da Tabela 4.4

$$\frac{1}{2}\times d_{12}+\frac{1}{8-2}\times[(d_{34}+d_{35}+d_{36}+d_{37}+d_{38}+d_{45}+d_{46}+d_{47}+d_{48}+d_{56}+d_{57}+d_{58}+d_{67}+d_{68}+d_{78})] =$$
$$= \frac{1}{2(8-2)} \times [(8+5)+(11+8)+(13+10)+(16+13)+(13+10)+(17+14)] + \frac{1}{2} \times 7 +$$
$$\frac{1}{8-2} \times (5+7+10+7+11+8+11+8+12+5+6+10+9+13+8) =$$
$$= \frac{1}{2(8-2)} \times 138 + \frac{1}{2} \times 7 + \frac{1}{8-2} \times 130 = 36,67$$

Como mostra a Tabela 4.6, o menor valor para S_{ij} é o dos vizinhos $i = 1$ e $j = 2$. Estes são unidos dando origem a um nó pai, o nó X. A nova árvore pode ser observada na Figura 4.3a, onde o nó X é o nó criado e o nó Y o nó central.

O tamanho dos ramos entre os nós $i = 1$ e $j = 2$ e o novo nó X, são calculados de acordo com as Equações 4.6 e 4.7, respectivamente.

$$L_{1X} = \frac{d_{12}+\frac{1}{8-2}[(d_{13}-d_{23})+(d_{14}-d_{24})+(d_{15}-d_{25})+(d_{16}-d_{26})+(d_{17}-d_{27})+(d_{18}-d_{28})]}{2} =$$
$$= \frac{d_{12}+\frac{1}{6}[(8-5)+(11-8)+(13-10)+(16-13)+(13-10)+(17-14)]}{2} = \frac{7+3}{2} = 5$$
$$L_{2X} = \frac{d_{12}+\frac{1}{8-2}[(d_{23}-d_{13})+(d_{24}-d_{14})+(d_{25}-d_{15})+(d_{26}-d_{16})+(d_{27}-d_{17})+(d_{28}-d_{18})]}{2} =$$
$$= \frac{d_{12}+\frac{1}{6}[(5-8)+(8-11)+(10-13)+(13-16)+(10-13)+(14-17)]}{2} = \frac{7-3}{2} = 4$$

Em seguida são calculados os novos valores de S_{ij}, para cada i, j como na matriz da Tabela 4.6. Estes valores são calculados com base nas distâncias entre cada OTUs (excluindo 1 e 2, agora representados por X).

Nesta iteração os vizinhos 5 e 6 serão unidos, pois apresentam o menor S_{ij} (como pode ser visto na Tabela 4.6). A Figura 4.3b mostra a junção desses nós criando o nó X, tendo o nó Y como nó central. Este processo é repetido até que fiquem apenas 3 OTUs, quando temos uma árvore sem raiz. Na Figura 4.3 pode ser observada a ordem com que os vizinhos são unidos. Na Figura 4.3c une-se o nó 3 à junção dos nós 1 e 2. Na Figura 4.3d une-se o nó 4 à junção dos nós 1, 2 e 3. Na Figura 4.3e unem-se as junções dos nós 1, 2 e 3 com a dos nós 4, 5. Finalmente, unem-se as junções dos nós 1, 2, 3, 4 e 5 com os nós 7 e 8 (OTUs X e Y da Figura 4.3e), formando a árvore da Figura 4.2b.

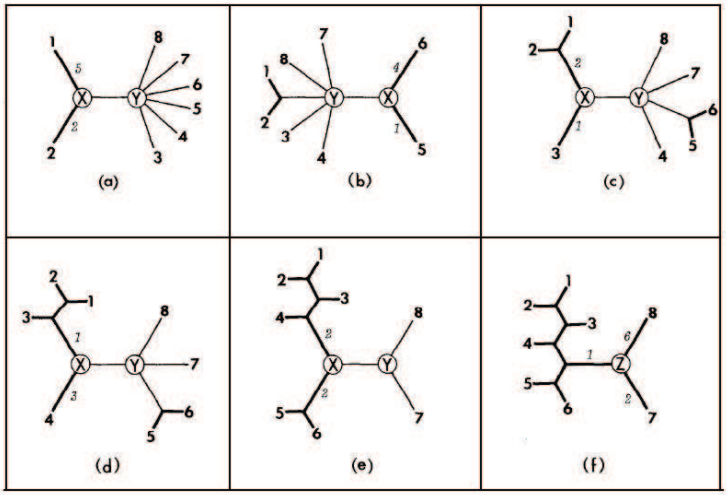


Figura 4.3: Etapas da construção da árvore guia, a partir árvore inicial da Figura 4.2a para produzir a da Figura 4.2b [61]

	1-2	3	4	5	6	7	8
1-2	31,50	-					
4	32,30	32,30	-				
5	33,90	33,90	33,70	-			
6	33,90	33,90	33,70	31,30	-		
7	33,70	33,70	33,50	33,10	33,10	-	
8	33,70	33,70	33,50	33,10	33,10	31,90	-

Tabela 4.6: S_{ij} para as 6 OTUs iniciais e o X.

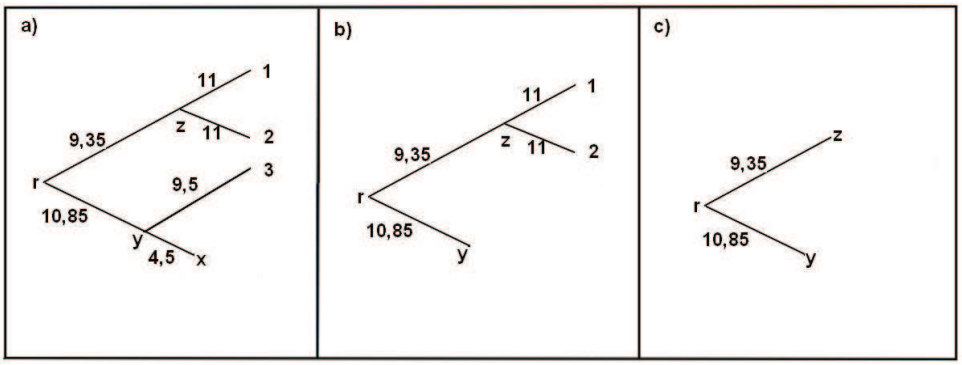


Figura 4.4: Seleção de pares mais próximos para a árvore da Figura 4.1

4.3 Seleção de Pares

A escolha de quais pares de sequências ou alinhamentos serão agrupados em cada ciclo na etapa do agrupamento do processo progressivo não pode ser arbitrária. Devem ser escolhidas as sequências com maior semelhança (menor distância) primeiro. Para isto podem ser usados os métodos de par mais próximo e bloco único, descritos nas Seções 4.3.1 e 4.3.2.

4.3.1 Par Mais Próximo

A partir da árvore guia, toma-se cada folha da árvore como um alinhamento composto por uma única sequência [69]. Seleciona-se o par de folhas mais próximo, então agrupa-se pelos métodos que serão apresentados na Seção 4.5 o alinhamento de ambas as folhas. O alinhamento formado no agrupamento é associado ao nó pai do par de folhas mais próximas. Assume-se então o nó pai como folha e as folhas escolhidas como mais próximas são removidas.

A partir da árvore da Figura 4.1, a seleção de pares pelo par mais próximo ocorre na seguinte ordem. O par mais próximo inicialmente é o 4 e 5, cuja distância entre eles é 10. Estas folhas são alinhadas e agrupadas no nó x, que passará a ser folha, como mostra a Figura 4.3.1a. Em seguida agrupam-se os nós x e 3 no nó y, como na Figura 4.3.1b. O par mais próximo agora é composto pelas folhas 1 e 2, que são agrupadas no nó z, como na Figura 4.3.1c. Finalmente, agrupam-se os alinhamentos das folhas y e z.

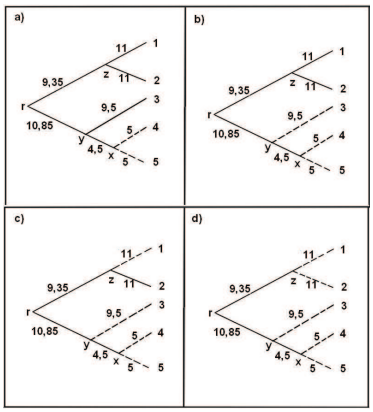


Figura 4.5: Seleção de pares por bloco único para a árvore da Figura 4.1. Linhas tracejadas indicam as sequências já incluídas no bloco.

4.3.2 Bloco Único

O algoritmo de seleção por bloco único [69] inicia assumindo uma sequência central como bloco, cuja soma das distâncias entre ela e todas as outras sequências seja a menor do conjunto. A segunda sequência a ser colocada no bloco é a de menor distância em relação à central. A partir dai adiciona-se a sequência mais próxima ao bloco (a distância entre o bloco e cada sequência fora dele é obtida somando-se a distância entre ela e cada sequência do bloco). Aquela que possuir menor soma é tida como mais próxima. A iteração ocorre até que todas elas façam parte do mesmo bloco.

Observando a distância entre cada par de sequências na Figura 4.1. Temos que a distância entre cada uma das folhas (sequências) e todas as outras são:

- 1: d_{12}+d_{13}+d_{14}+d_{15}=22+40,7+40,7+40,7=144,1
- 2: d_{21}+d_{23}+d_{24}+d_{25}=22+40,7+40,7+40,7=144,1
- 3: d_{31}+d_{32}+d_{34}+d_{35}=40,7+40,7+19+19=119,4
- 4: d_{41}+d_{42}+d_{43}+d_{45}=40,7+40,7+19+10=110,4
- 5: d_{51}+d_{52}+d_{53}+d_{54}=40,7+40,7+19+10=110,4

Como podemos observar as sequências 4 e 5 são as de menor distância em relação as demais. Assumindo a 5 com sequência central, adicionamos a sequência 4 ao bloco, por ser a mais próxima da 5, como na Figura 4.3.2a. Em seguida, calculamos a distância entre as sequências 1, 2 e 3 e o bloco formado pelas sequências 4 e 5:

- 1: d_{14}+d_{15}=40,7+40,7=81,4
- 2: d_{24}+d_{25}=40,7+40,7=81,4
- 3: d_{34}+d_{35}=19+19=38

A sequência 3 é a de menor distância para o bloco e agora também fará parte dele, como mostra a Figura 4.3.2b. Novamente calculamos as distâncias entre as sequências que ainda não fazem parte do bloco e as que já estão no bloco:

- 1: d_{13}+d_{14}+d_{15}=40,7+40,7+40,7=122,4
- 2: d_{23}+d_{24}+d_{25}=40,7+40,7+40,7=122,4

A este ponto, como o valor das distâncias para 1 e 2 é o mesmo. Adicionamos primeiro a sequência 1, como mostra a Figura 4.3.2c, e depois a sequência 2, como na Figura 4.3.2d. Com isso, temos todas as sequências como parte do alinhamento.

4.4 Peso das Sequências

A inclusão de pesos nas sequências é interessante quando estas não são equidistantes, pois visa equilibrar as informações entre as sequências. As mais divergentes recebem pesos maiores, para que sua informação diferenciada seja considerada na construção do alinhamento. Sequências divergentes em relação a um grupo tendem a ter seu alinhamento sub valorizado caso o sistema de pesos não seja usado.

4.4.1 Média das Distâncias

O peso de cada sequência corresponde à média das distâncias entre ela e todas as outras do conjunto. Os valores das distâncias utilizados são normalizados, e portanto, variam entre 0 e 1.

Considere as sequências:

```
>1
GGGAAGCATAGTGCGTGAT
>2
TATTGAAGCATTAGGGCGGT
>3
GGGAAGCATATTAGGGCGGT
```

	1	2	3	4
1	-			
2	0,50	-		
3	0,78	0,78	-	
4	0,72	0,78	0,34	-

Tabela 4.7: Distâncias normalizadas entre as quatro sequências:
`(1)GGGAAGCATAGTGCGTGAT, (2)TATTGAAGCATTAGGGCGGT, (3)GGGAAGCATATTAGGGCGGT, (4)TATTGAAGCAGTGCGTGAT`.

```
>4
TATTGAAGCAGTGCGTGAT
```

Considere também as distâncias entre elas, apresentadas na Tabela 4.7.

Podemos calcular o peso de cada sequência pela média das distâncias entre ela e as demais com as quais será alinhada. Os pesos p_1 a p_4, onde p_x corresponde ao peso da sequência x, são calculados abaixo:

$p_1 = \frac{d_{12}+d_{13}+d_{14}}{3} = \frac{0,50+0,78+0,72}{3} = 0,666$
$p_2 = \frac{d_{21}+d_{23}+d_{24}}{3} = \frac{0,50+0,78+0,78}{3} = 0,686$
$p_3 = \frac{d_{31}+d_{32}+d_{34}}{3} = \frac{0,78+0,78+0,34}{3} = 0,633$
$p_4 = \frac{d_{41}+d_{42}+d_{43}}{3} = \frac{0,72+0,78+0,34}{3} = 0,613$

4.4.2 Peso Idêntico

Todas as sequências recebem peso 1. Esse esquema corresponde a desativar o peso das sequências.

4.5 Agrupamento

Métodos de agrupamento permitem unir diferentes alinhamentos múltiplos em um só alinhamento. Neste trabalho foram utilizados dois tipos principais de algoritmos de agrupamento, o de consenso e o de perfil.

4.5.1 Agrupamento por Consenso

O agrupamento por consenso é realizado tendo como base o alinhamento dos consensos dos alinhamentos de entrada [69]. Os consensos de cada alinhamento de entrada são computados e alinhados por alinhamento global [48].

O consenso é obtido da seguinte forma: para cada coluna do alinhamento e para cada letra calcula-se a pontuação da combinação desta com todas as outras letras possíveis, ou seja, os 23 aminoácidos. A letra que tiver maior pontuação é escolhida para o consenso. Desta forma a sequência consenso terá o mesmo número de colunas que o alinhamento. Realiza-se então o alinhamento dos consensos. Em seguida, as sequências dos alinhamentos de entrada que originaram os consensos são agrupadas alinhando-se as colunas de aminoácidos dos alinhamentos de entrada quando há alinhamento de aminoácidos na coluna correspondente do consenso. No caso de haver buraco no alinhamento dos consensos, alinha-se a coluna de aminoácidos correspondente ao consenso que ficou com aminoácido na coluna do alinhamento com uma coluna de buracos nas sequências do alinhamento cujo consenso ficou com buraco em tal coluna. O Algoritmo 7 mostra como se dá a construção do consenso de um alinhamento e o Algoritmo 8 mostra como são alinhadas cada sequência de acordo com o consenso.

Algoritmo 7 Pseudocódigo para construção da sequência consenso

```
Entrada: alignment, numberOfSequences, alignmentColumns
Saída: consensus
aminoacidList ← "ABCDEFGHIKLMNPQRSTVWXYZ"
maxScore ← ∞
for i ← 1 to alignmentColumns do
  for j ← 1 to aminoacidList.length do
    aminoacid ← aminoacidList[j]
    score ← 0;
    for k ← 1 to numberOfSequences do
      curBase ← alignment[k]
      if curBase ≠ "-" then
        score ← score + substitutionMatrix(aminoacid, curBase)
    if score> maxScore then
      maxScore ← score
      maxBase ← aminoacidList[j]
  consensus[i] ← maxBase
return consensus
```

Construiremos o consenso conforme os Algoritmos 7 e 8, tomando como entrada as seguintes sequências:

```
>1
MERLSEDDPAAQALEYRHDASSVQHPAYEEGQTCLNCLLYTDASAQDWGPCSVFPGKLVSANGWCTAWVAR
> 2
SAPANAVAADNATAIALKYNQDATKSERVAAARPGLPPEEQHCADCQFMQADAAGATDEWKGCQLFPGKLINVNGWCASWTLKAG
>3
```

Algoritmo 8 Agrupamento dos alinhamentos de acordo com a sequência consenso

```
Entrada: alignment1, alignment2,numberOfSeq1,numberOfSeq2
Saída: alignmentGroup
consensus1 ← getConsensus(alignment1,numberOfSeq1,alignment1[1].length)
consensus2 ← getConsensus(alignment2,numberOfSeq2,alignment2[1].length)
alignmentLengthTotal ← numberOfSeq1 + numberOfSeq2
consensusAligned ← aligner.getAlignment(consensus1, consensus2)
index1 ← 1
index2 ← 1
for i ← 1 to alignmentLengthTotal do
  if consensusAligned1[1][i] = "-" then
    for j ← 1 to numberOfSeq1 do
      alignmentGroup[j,i] ← "-"
  else
    for j ← 1 to numberOfSeq1 do
      alignmentGroup[j,i] ← alignment1[j,index1]
    index1 ← index + 1
  if consensusAligned[2][i] = "-" then
    for j ← 1 to numberOfSeq2 do
      alignmentGroup[j+numberOfSeq1,i] ← "-"
  else
    for j ← 1 to numberOfSeq2 do
      alignmentGroup[numberOfSeq1 + j,i] ← alignment2[j,index2]
    index2 ← index2 + 1
return alignmentGroup
```

```
EPRAEDGHAHDYVNEAADASGHPRYQEGQLCENCAFWGEAVQDGWGRCTHPDFDEVLVKAEGWCSVYAPAS
>4
AAPLVAETDANAKSLGYVADTTKADKTKYPKHTKDQSCSTCALYQGKTAPQGACPLFAGKEVVAKGWCSAWAKKA
>5
EDLPHVDAATNPIAQSLHYIEDANASERNPVTKTELPGSEQFCHNCSFIQADSGAWRPCTLYPGYTVSEDGWCLSWAHKTA
>6
QDLPPLDPSAEQAQALNYVKDTAEAADHPAHQEGEQCDNCMFFQADSQGCQLFPQNSVEPAGWCQSWTAQN
```

Inicialmente o alinhador agrupa as sequências 2 e 5, posteriormente adiciona 4 e 1, compondo o seguinte alinhamento:

```
SAPANAVAADNATAIALKYNQDATKSERVAAARPGLPPEEQHCADCQFMQADAAGATDEWKGCQLFPGKLINVNGWCASWTLKAG
EDLPHVDAATNPIAQSLHYIEDANASERNPVTKTELPGSEQFCHNCSFIQADSGA----WRPCTLYPGYTVSEDGWCLSWAHKTA
AAPLVAETDAN--AKSLGYVADTTKADK---TKYPKHTKDQSCSTCALYQ----GKTAPQGACPLFAGKEVVAKGWCSAWA-KKA
MERLSED---DPAAQALEYRHDAS------SVQHPAYEEGQTCLNC-LLYTDASAQ--DWGPCSVFPGKLVSANGWCTAWVAR--
```

E formando o seguinte consenso:

```
SAPLNADAATNPTAQALHYIQDATKSERNPATKHPLPPEEQHCANCSFLQADAGGQTDDWGPCPLFPGKLVSANGWCTAWAHKTA
```

O aminoácido S foi escolhido para primeira posição consenso pois alcançou maior pontuação na primeira coluna (de acordo com a matriz BLOSUM62). Por exemplo, comparando S com E, A e M, temos as seguintes pontuações (note que essas pontuações são calculadas para cada um dos vinte aminoácidos):

$score(S) = score(S,S) + score(S,E) + score(S,A) + score(S,M) = 4 + 0 + 1 - 1 = 4$

$score(E) = score(E,S) + score(E,E) + score(E,A) + score(E,M) = 0 + 5 - 1 - 2 = 2$

$score(A) = score(A,S) + score(A,E) + score(A,A) + score(A,M) = 1 - 1 + 4 - 1 = 3$

$score(M) = score(M,S) + score(M,E) + score(M,A) + score(M,M) = -1 - 2 - 1 + 5 = 1$

Em seguida agrupa 6 e 3, formando um segundo alinhamento.

```
QDLPPLDPSAEQAQALNYVKDTAEAADHPAHQEGEQCDNCMFF-QADSQGCQL-----FPQNSVEPAGWCQSWTAQN
------EPRAEDGHAHDYVNEAADASGHPRYQEGQLCENCAFWGEAVQDGWGRCTHPDFDEVLVKAEGWCSVYAPAS
```

O que originou o seguinte consenso:

```
QDLPPLDPRAEDGHAHNYVNDTADAADHPRHQEGQQCDNCMFWGQADQDGWGRCTHPDFPQNLVEPEGWCQSWTPQN
```

O alinhamento dos consensos foi o seguinte:

```
SAPLNADAATNPTAQALHYIQDATKSERNPATKHPLPPEEQHCANCSFLQADAGGQTDDWGPC--PLFPGKLVSANGWCTAWAHKTA
QDLPPLDPRAE-DGHAHNYVNDTA-----DAADHPRHQEGQQCDNCMFW---GQADQDGWGRCTHPDFPQNLVEPEGWCQSWTPQN-
```

Finalmente, estes alinhamentos são unidos em um único alinhamento, de acordo com alinhamento dos consensos.

```
SAPANAVAADNATAIALKYNQDATKSERVAAARPGLPPEEQHCADCQFMQADAAGATDEWKGC--QLFPGKLINVNGWCASWTLKAG
EDLPHVDAATNPIAQSLHYIEDANASERNPVTKTELPGSEQFCHNCSFIQADSGA----WRPC--TLYPGYTVSEDGWCLSWAHKTA
AAPLVAETDAN--AKSLGYVADTTKADK---TKYPKHTKDQSCSTCALYQ----GKTAPQGAC--PLFAGKEVVAKGWCSAWA-KKA
MERLSED---DPAAQALEYRHDAS------SVQHPAYEEGQTCLNC-LLYTDASAQ--DWGPC--SVFPGKLVSANGWCTAWVAR--
QDLPPLDPSAE-QAQALNYVKDTA-----EAADHPAHQEGEQCDNCMFF----QADSQGCQL-----FPQNSVEPAGWCQSWTAQN-
------EPRAE-DGHAHDYVNEAA-----DASGHPRYQEGQLCENCAFW---GEAVQDGWGRCTHPDFDEVLVKAEGWCSVYAPAS-
```

Ainda foram implementadas variações do agrupamento por consenso. Tais variações

diferenciam-se pelo tipo e pontuação do alinhamento de pares utilizados para alinhar os consensos, e são listadas abaixo:

- Alinhamento global com pontuação logarítmica (definida na Seção 3.2.3).
- Alinhamento semi-global com pontuação afim (definida na Seção 3.2.2).
- Alinhamento semi-global com pontuação logarítmica (definida na Seção 3.2.3).
- Alinhamento recursivo (definido na Seção 3.1.1) com pontuação linear (definida na Seção 3.2.1).

4.5.2 Alinhamento de Perfis

O alinhamento de perfis tem como entrada dois alinhamentos múltiplos e os alinha com base na pontuação dos pares de colunas. A pontuação de um par de colunas é obtida somando-se a pontuação dos pares formados por uma letra da coluna corrente de um dos alinhamentos de entrada com uma letra da coluna corrente do outro. Pares de buraco não são pontuados. Caso alinhe-se uma coluna de um dos alinhamentos de entrada com buracos, substitui-se a coluna do outro alinhamento por buracos [50].

No alinhamento de letras descrito na Seção 3.1, a escolha do alinhamento era feita pela de maior pontuação entre alinhamento de buraco com letra, letra com letra, e letra com buraco. No alinhamento de perfil procede-se de forma semelhante, a escolha é feita entre alinhar uma coluna de letras com outra coluna de letras, uma coluna de letras com uma de buracos, ou uma coluna de buracos com uma coluna de letras, pela combinação que alcançar maior pontuação do par de colunas.

Algoritmo 9 mostra como é calculada a pontuação de um alinhamento entre duas colunas específicas dos dois alinhamentos de entrada. Onde **scoreAlg** é a pontuação (*match, mismatch* ou *gap*) do alinhamento da letra de uma coluna, com a letra de outra coluna.

Algoritmo 9 Algoritmo para calcular pontuação linear de *gaps* para o Alinhamento de Perfil

```
Entrada: gap, Alignment1, Alignment2,col1, col2
Saída: score
score ← 0
for i ← 1 to Alignment1.length do
  for j ← 1 to Alignment2.length do
    score ← score + scoreAlg(Alignment1[i,col1],Alignment2[j,col2])
return score
```

O Algoritmo 10 ilustra como é construída a matriz de pesos. Cada célula (i, j) da matriz indica a pontuação da coluna i de um alinhamento com a coluna j do outro alinhamento. Onde **scoreColumn** é o Algoritmo 9.

Algoritmo 10 Algoritmo para agrupamento por alinhamento de perfil

```
Entrada: align1,align2, nCol1,nCol2
Saída: matrixAlign
scoreMax ← -∞
for i ← 1 to align1.length do
  for j ← 1 to align2.length do
    matrixAlign[i,j] ← -∞
    score ← matrixAlign[i-1,j] + scoreColumn(align1[i],gaps)
    if score > matrixAlign[i,j] then
      matrixAlign[i,j] ← score
    score ← matrixAlign[i-1,j-1] + scoreColumn(align1[i],align2[j])
    if score > matrixAlign[i,j] then
      matrixAlign[i,j] ← score
    score ← matrixAlign[i,j-1] + scoreColumn(gaps,align2[j])
    if score > matrixAlign[i,j] then
      matrixAlign[i,j] ← score
  if matrixAlign[i,j] > scoreMax then
    scoreMax ← matrixAlign[i,j]
return (scoreMax,matrixAlign)
```

Utilizando as mesmas sequências e os dois grupos produzidos no alinhador com agrupamento de consenso:

```
SAPANAVAADNATAIALKYNQDATKSERVAAARPGLPPEEQHCADCQFMQADAAGATDEWKGCQLFPGKLINVNGWCASWTLKAG
EDLPHVDAATNPIAQSLHYIEDANASERNPVTKTELPGSEQFCHNCSFIQADSGA---WRPCTLYPGYTVSEDGWCLSWAHKTA
AAPLVAETDAN--AKSLGYVADTTKADK---TKYPKHTKDQSCSTCALYQ----GKTAPQGACPLFAGKEVVAKGWCSAWA-KKA
MERLSED---DPAAQALEYRHDAS------SVQHPAYEEGQTCLNC-LLYTDASAQ--DWGPCSVFPGKLVSANGWCTAWVAR--
```

e,

```
QDLPPLDPSAEQAQALNYVKDTAEAADHPAHQEGEQCDNCMFF-QADSQGCQL-----FPQNSVEPAGWCQSWTAQN
------EPRAEDGHAHDYVNEAADASGHPRYQEGQLCENCAFWGEAVQDGWGRCTHPDFDEVLVKAEGWCSVYAPAS
```

O alinhamento de perfil, utilizando pontuação *gap* = -8, foi o seguinte:

```
SAPANAVAADNATAIALKYNQDATKSERVAAARPGLPPEEQHCADCQFM-QADAAGATDEWKGCQ-L-FPGKLINVNGWCASWTLKAG
EDLPHVDAATNPIAQSLHYIEDANASERNPVTKTELPGSEQFCHNCSFI-QADSGA----WRPCT-L-YPGYTVSEDGWCLSWAHKTA
AAPLVAETDAN--AKSLGYVADTTKADK---TKYPKHTKDQSCSTCALY-Q----GKTAPQGACP-L-FAGKEVVAKGWCSAWA-KKA
MERLSED---DPAAQALEYRHDAS------SVQHPAYEEGQTCLNC-LL-YTDASAQ--DWGPCS-V-FPGKLVSANGWCTAWVAR--
QDLPPLDPSAEQ-AQALNYVKDTA--E---AADHPAHQEGEQCDNCMFF-QADSQG----CQL-----FPQNSVEPAGWCQSWTAQN-
------EPRAED-GHAHDYVNEAA--D---ASGHPRYQEGQLCENCAFWGEAVQDG----WGRCTHPDFDEVLVKAEGWCSVYAPAS-
```

Por exemplo, tomando a posição (12,12) da matriz que corresponde ao alinhamento da décima segunda coluna do primeiro alinhamento com a décima segunda do segundo. São obtidas as seguintes pontuações (utilizando valores da matriz de substituição BLOSUM62):

- Alinhamento da décima segunda coluna do primeiro alinhamento com a décima segunda do segundo alinhamento, respectivamente $A, P, -, P$ e Q, D, cuja pontuação é: $score(A, Q) + score(A, D) + 2 \times score(P, Q) + 2 \times score(P, D) + score(-, Q) + score(-, D) = -1 - 2 + (2 \times -1) + (2 \times -1) - 8 - 8 = -23$
- Alinhamento da décima segunda coluna do primeiro com buracos: $6 \times gap + 2 \times score(gap, gap) = -48 + 0 = -48$.
- Alinhamento da décima segunda coluna do segundo com uma coluna de buracos: $8 \times gap = -64$.

Como podemos notar a maior pontuação foi a do alinhamento da décima segunda coluna do primeiro alinhamento com a décima segunda do segundo alinhamento. Portanto, no alinhamento final utilizou-se esta combinação.

Assim como no alinhamento por consenso também foram utilizadas variações para o alinhamento de perfil, tanto no tipo de alinhador como na função de penalidade de *gaps* para os pares de sequências. Estas variações são:

- Alinhamento de perfil semi-global com pontuação linear (como descrito na Seção 3.2.1).
- Alinhamento de perfil global com pontuação afim (como descrito na Seção 3.2.2).
- Alinhamento de perfil semi-global com pontuação afim (como descrito na Seção 3.2.2).
- Alinhamento de perfil global com pontuação logarítmica (como descrito na Seção 3.2.3).
- Alinhamento de perfil semi-global com pontuação logarítmica (como descrito na Seção 3.2.3).

4.6 Melhoria de Parâmetros

Tendo em vista a alta sensibilidade do alinhador de perfil em relação aos parâmetros de entrada, da matriz de substituição e das penalidades de abertura e extensão de buracos, iniciou-se um novo estudo para melhorar estes parâmetros do alinhador de perfil com pontuação afim de acordo com as sequências a serem alinhadas. Para isto utilizamos a idéia do ClustalW [66] determinando a matriz de substituição de acordo com a similaridade das sequências, e adaptando os valores de abertura (*gop*) e extensão (*gep*) de blocos de buracos a partir de um valor inicial.

No intuito de balancear o valor do *gop* em relação ao tamanho da sequências, utiliza-se o logaritmo do tamanho da menor entre os tamanhos máximos de cada alinhamento. O valor do *gep* também é balanceado buscando evitar buracos longos nas sequências menores.

O Algoritmo 11, mostra como são calculados os valores para *gop* e *gep* a partir da matriz de substituição escolhida. A escolha da matriz é feita com base na similaridade das sequências, quanto mais similares as sequências, maior é a BLOSUM utilizada. Os valores de *gop* e *gep* são determinados levando em consideração a média dos elementos não diagonais da matriz (determinada pelo algoritmo **avgNonDiagonal**) e também os tamanhos das sequências de entrada. A função **avgNonDiagonal** retorna o tamanho da maior sequência do alinhamento e **getMax** o tamanho da maior sequência do alinhamento.

Utilizando os valores -17 e -1, como pontuação inicial para *gop* e *gep*, e as seguintes sequências como entrada:

```
>seq1
MHIKKPLNAFMLYMKEMRANVVAESTLKESAAINQILGRRWHALSREEQAKYYELARKERQLHMQLYPGWSARDNYGKKKKRKREK
>seq2
MQDRVKRPMNAFIVWSRDQRRKMALENPRMRNSEISKQLGYQWKMLTEAEKWPFFQEAQKLQAMHREKYPNYKYRPRRKAKMLPK
>seq3
GKGDPKKPRGKMSSYAFFVQTSREEHKKKHPDASVNFSEFSKKCSERWKTMSAKEKGKFEDMAKADKARYEREMKTYIPPKGE
>seq4
MKKLKKHPDFPKKPLTPYFRFFMEKRAKYAKLHPEMSNLDLTKILSKKYKELPEKKKMKYIQDFQREKQEFERNLARFREDHPDLIQNAKK
```

Na primeira vez em que o algoritmo de melhoria de parâmetros é executado, ele recebe como entrada $seq3$ como o primeiro alinhamento, e $seq4$ como segundo alinhamento. Com essas entradas sugere a utilização da matriz BLOSUM45, de $gop = -7.63$ e $gep = -1.0$, e tem o alinhamento como resultado (chamaremos de $align1$):

```
MH--IKKPLNAFMLYMKEMRANVVAESTLKESAAINQILGRRWHALSREEQAKYYELARKERQLHMQLYPGWSARDNYGKKKKRKREK
MQDRVKRPMNAFIVWSRDQRRKMALENPRMRNSEISKQLGYQWKMLTEAEKWPFFQEAQKLQAMHREKYPNYKYRP---RRKAKMLPK
```

Na segunda vez o algoritmo recebe como entrada $seq1$ e $seq2$. Como saída sugere a utilização de BLOSUM45, $gop = -7.65$ e $gep = -1.0$. Gerando o seguinte alinhamento (chamaremos de $align2$):

```
GK-----GD-PKKPRGKMSSYAFFVQTSREEHKKKHPDASVNFSEFSKKCSERWKTMSAKEKGKF-EDMAKADKARYEREMKTY-------IPPKGE
MKKLKKHPDFPKKP---LTPYFRFFMEKRAKYAKLHPEMS-NL-DLTKILSKKYKELPEKKKMKYIQDFQR-EKQEFERNLARFREDHPDLIQNAKK
```

Finalmente, o algoritmo recebe como entrada os alinhamentos $align1$ e $align2$. Como saída atribui BLOSUM62 para a matriz e -11.56 e -1.0 para os valores de *gop* e *gep*, respectivamente. Além disso, o alinhamento resultante é o seguinte:

```
GK-----GD-PKKPRGKMSSYAFFVQTSREEHKKKHPDASVNFSEFSKKCSERWKTMSAKEKGKF-EDMAKADKARYEREMKTY----------IPPKGE
------MQDRVKRP---MNAFIVWSRDQRRKMALENPRMR--NSEISKQLGYQWKMLTEAEKWPFFQEAQK-LQAMHREKYPNYKYRP---RRKAKMLPK
MKKLKKHPDFPKKP---LTPYFRFFMEKRAKYAKLHPEMS-NL-DLTKILSKKYKELPEKKKMKYIQDFQR-EKQEFERNLARFREDH---PDLIQNAKK
------MH--IKKP---LNAFMLYMKEMRANVVAESTLKE--SAAINQILGRRWHALSREEQAKYYELARK-ERQLHMQLYPGWSARDNYGKKKKRKREK
```

Algoritmo 11

```
Entrada: alignment1, alignment2, sequences, similarityMatrix, initialGop, initialGep
Saída:groupGop, groupGep, substitutionMatrix
count ← 0
identity ← 0
numberOfSequences ← sequences.length
for i ← 1 numberOfSequences do
  for j ← 1 numberOfSequences do
    identity ← identity + similarityMatrix[i,j];
    count ← count +1
identity ← identity/count;
if identity > 80.0 then
  substitutionMatrix ← BLOSUM80
else
  if identity >60.0 then
    substitutionMatrix ← BLOSUM62
  else
    if identity >30.0 then
      substitutionMatrix ← BLOSUM45
    else
      substitutionMatrix ← BLOSUM30
scale ← identity/100;
m ← getMax(alignment1)
n ← getMax(alignment2)
matAvscore ← avgNonDiagonal(substitutionMatrix)
logmin ← log2(min(m, n))
if matAvscore ≤ 0 then
  groupGop ← (initialGop + logmin)
else
  groupGop ← scale × matAvscore × (initialGop + logmin)
if m > n then
  logNM ← log2(m / n)
else
  logNM ← log2(n / m)
groupGep ← initialGep × (1 + logNM)
return (groupGop, groupGep, substitutionMatrix)
```

Capítulo 5

Alinhadores Progressivos

Apesar de existir um grande número de alinhadores múltiplos de sequências que utilizam a abordagem progressiva, eles não realizam uma comparação das possíveis combinações de métodos que podem ser utilizados em cada etapa do alinhamento. Neste sentido implementamos vários alinhadores progressivos no intuito de combinar diferentes métodos que podem ser utilizados em cada etapa do alinhamento.

Para obter a matriz de distâncias foram utilizados quatro métodos do PHYLIP [19]. O primeiro deles é baseado nas matrizes PAM de Dayhoff, seguindo o modelo DCMut, descrito na Seção 3.3.1. O segundo é baseado no modelo de *Jones-Taylor-Thornton*, o JTT, descrito na Seção 3.3.2. O terceiro baseia-se no modelo PMB (*Probability Matrix from Blocks*), apresentado na Seção 3.3.3. O quarto utiliza o modelo de categorias descrito na Seção 3.3.4, para o qual utilizaremos a sigla PCM. Além dos quatro modelos do PHYLIP utilizados para determinar a matriz de distâncias, utilizamos o algoritmo de pontuação recursiva apresentado na Seção 4.1.1 (abreviada por LD) e o algoritmo de pontuação logarítmica apresentado na Seção 4.1.2 (abreviada por LOGD).

Para a construção da árvore filogenética utilizamos dois métodos do pacote estatístico R [22], o UPGMA (abreviado por UP) e o *Neighbor Joining* (NJ), descritos nas Seções 4.2.1 e 4.2.2, respectivamente.

A construção do alinhamento múltiplo é realizada por métodos de agrupamento. Utilizamos o agrupamento por consenso, abreviado como AC descrito na Seção 4.5.1 e suas variações: alinhamento global com pontuação logarítmica (ACLog), alinhamento semi-global com pontuação afim (ACb), alinhamento semi-global com pontuação logarítmica (ACLogb) e alinhamento recursivo (LC). Utilizamos ainda, o agrupamento por perfis apresentado na Seção 4.5.2, abreviado por AP, e suas variações: alinhamento de perfil semi-global (APb), alinhamento de perfil global com pontuação afim (APA), alinhamento de perfil semi-global com pontuação afim (APAb), alinhamento de perfil global com pontuação afim e melhoria de parâmetros (APAp), alinhamento de perfil global com

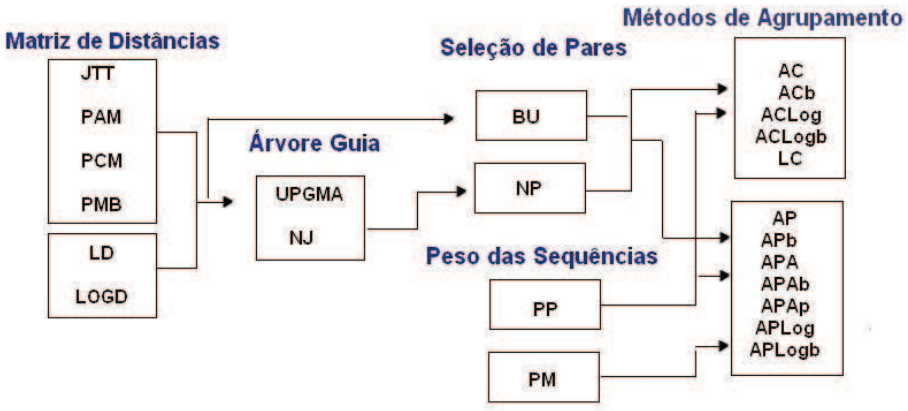


Figura 5.1: Métodos utilizados em cada etapa dos 342 alinhadores múltiplos progressivos construídos

pontuação logarítmica (APLog), e o alinhamento de perfil semi-global com pontuação logarítmica (APLogb).

Os métodos de agrupamento alinham as sequências de acordo com os pares selecionados pelos métodos de seleção de pares e levam em consideração os pesos das sequências. As sequências não são utilizadas na ordem em que foram colocadas na árvore, pois desta forma podemos a partir de qualquer árvore determinar a ordem dos alinhamentos entre pares de sequências ou alinhamentos.

Para selecionar a ordem em que os pares são alinhados utilizamos os métodos: par mais próximo (NP, de *Nearest Pair*) descrito na Seção 4.3.1 e bloco único (BU) descrito na Seção 4.3.2. É importante ressaltar que a seleção por bloco único não faz uso da árvore para escolher qual par será alinhado em um certo momento. O esquema de pesos das sequências é ativado quando utilizamos a média das distâncias (abreviada por PM) como apresentado 4.4.1, e quando o esquema está desativado utilizamos a abreviação PP.

A Figura 5.1 mostra cada um dos métodos utilizados na construção dos 342 alinhadores múltiplos progressivos. Nesta figura podemos observar a relação, sequência, em que cada método, de cada etapa é executado.

Os alinhadores foram implementados na linguagem JAVA. Utilizamos a biblioteca biojava [31], pois permite a manipulação de sequências. Esta disponibiliza, por exemplo, parsers para formatos comuns de arquivos e métodos que realizam o alinhamento de pares de sequências e calculam sua pontuação.

A Tabela 5.1 mostra os métodos utilizados por cada um dos 342 alinhadores implementados, na qual o nome dos alinhadores segue o seguinte padrão: XXX utilizam alinhamento global ou local, XXXb utilizam alinhamento semi-global, e XXXp utilizam o algoritmo para melhoria de parâmetros. Por exemplo, na primeira linha da Tabela 5.1, o alinhador global 049 utliza o modelo JTT para determinar a matriz de distância, UPGMA para construir a árvore filogenética, par mais próximo para seleção de pares, agrupamento por consenso e esquema de pesos desativado. A numeração tem início no alinhador 049, os alinhadores anteriores utilizavam Distância de Edição [40] e Distância de Kimura [37], estes foram descartados pois em testes preliminares não apresentaram bons resultados.

Alinhador	**Distância**	**Árvore**	**Seleção Pares**	**Agrupamento**	**Peso**
049	JTT	UP	NP	AC	PP
051	JTT	UP	NP	AP	PP
052	JTT	UP	NP	AP	PM
053	JTT	UP	NP	APA	PP
054	JTT	UP	NP	APA	PM
055	JTT	–	BU	AC	PP
057	JTT	–	BU	AP	PP
058	JTT	–	BU	AP	PM
059	JTT	–	BU	APA	PP
060	JTT	–	BU	APA	PM
061	JTT	NJ	NP	AC	PP
063	JTT	NJ	NP	AP	PP
064	JTT	NJ	NP	AP	PM
065	JTT	NJ	NP	APA	PP
066	JTT	NJ	NP	APA	PM
073	PMB	UP	NP	AC	PP
075	PMB	UP	NP	AP	PP
076	PMB	UP	NP	AP	PM
077	PMB	UP	NP	APA	PP
078	PMB	UP	NP	APA	PM
079	PMB	–	BU	AC	PP
081	PMB	–	BU	AP	PP
082	PMB	–	BU	AP	PM
083	PMB	–	BU	APA	PP
084	PMB	–	BU	APA	PM
085	PMB	NJ	NP	AC	PP

Continua na próxima página

Tabela 5.1 – continuação da página anterior

Alinhador	Distância	Árvore	Seleção Pares	Agrupamento	Peso
087	PMB	NJ	NP	AP	PP
088	PMB	NJ	NP	AP	PM
089	PMB	NJ	NP	APA	PP
090	PMB	NJ	NP	APA	PM
097	PAM	UP	NP	AC	PP
099	PAM	UP	NP	AP	PP
100	PAM	UP	NP	AP	PM
101	PAM	UP	NP	APA	PP
102	PAM	UP	NP	APA	PM
103	PAM	–	BU	AC	PP
105	PAM	–	BU	AP	PP
106	PAM	–	BU	AP	PM
107	PAM	–	BU	APA	PP
108	PAM	–	BU	APA	PM
109	PAM	NJ	NP	AC	PP
111	PAM	NJ	NP	AP	PP
112	PAM	NJ	NP	AP	PM
113	PAM	NJ	NP	APA	PP
114	PAM	NJ	NP	APA	PM
121	PCM	UP	NP	AC	PP
123	PCM	UP	NP	AP	PP
124	PCM	UP	NP	AP	PM
125	PCM	UP	NP	APA	PP
126	PCM	UP	NP	APA	PM
127	PCM	–	BU	AC	PP
129	PCM	–	BU	AP	PP
130	PCM	–	BU	AP	PM
131	PCM	–	BU	APA	PP
132	PCM	–	BU	APA	PM
133	PCM	NJ	NP	AC	PP
135	PCM	NJ	NP	AP	PP
136	PCM	NJ	NP	AP	PM
137	PCM	NJ	NP	APA	PP
138	PCM	NJ	NP	APA	PM

Continua na próxima página

Tabela 5.1 – continuação da página anterior

Alinhador	Distância	Árvore	Seleção Pares	Agrupamento	Peso
145	LD	UP	NP	AC	PP
147	LD	UP	NP	AP	PP
148	LD	UP	NP	AP	PM
149	LD	UP	NP	APA	PP
150	LD	UP	NP	APA	PM
151	LD	–	BU	AC	PP
153	LD	–	BU	AP	PP
154	LD	–	BU	AP	PM
155	LD	–	BU	APA	PP
156	LD	–	BU	APA	PM
157	LD	NJ	NP	AC	PP
159	LD	NJ	NP	AP	PP
160	LD	NJ	NP	AP	PM
161	LD	NJ	NP	APA	PP
162	LD	NJ	NP	APA	PM
185	JTT	UP	NP	LC	PP
187	JTT	–	BU	LC	PP
189	JTT	NJ	NP	LC	PP
193	PMB	UP	NP	LC	PP
195	PMB	–	BU	LC	PP
197	PMB	NJ	NP	LC	PP
201	PAM	UP	NP	LC	PP
203	PAM	–	BU	LC	PP
205	PAM	NJ	NP	LC	PP
209	PCM	UP	NP	LC	PP
211	PCM	–	BU	LC	PP
213	PCM	NJ	NP	LC	PP
217	LD	UP	NP	LC	PP
219	LD	–	BU	LC	PP
221	LD	NJ	NP	LC	PP
281	JTT	UP	NP	APLog	PP
282	JTT	UP	NP	APLog	PM
283	JTT	–	BU	APLog	PP
284	JTT	–	BU	APLog	PM

Continua na próxima página

Tabela 5.1 – continuação da página anterior

Alinhador	Distância	Árvore	Seleção Pares	Agrupamento	Peso
285	JTT	NJ	NP	APLog	PP
286	JTT	NJ	NP	APLog	PM
289	PMB	UP	NP	APLog	PP
290	PMB	UP	NP	APLog	PM
291	PMB	–	BU	APLog	PP
292	PMB	–	BU	APLog	PM
293	PMB	NJ	NP	APLog	PP
294	PMB	NJ	NP	APLog	PM
297	PAM	UP	NP	APLog	PP
298	PAM	UP	NP	APLog	PM
299	PAM	–	BU	APLog	PP
300	PAM	–	BU	APLog	PM
301	PAM	NJ	NP	APLog	PP
302	PAM	NJ	NP	APLog	PM
305	PCM	UP	NP	APLog	PP
306	PCM	UP	NP	APLog	PM
307	PCM	–	BU	APLog	PP
308	PCM	–	BU	APLog	PM
309	PCM	NJ	NP	APLog	PP
310	PCM	NJ	NP	APLog	PM
313	LD	UP	NP	APLog	PP
314	LD	UP	NP	APLog	PM
315	LD	–	BU	APLog	PP
316	LD	–	BU	APLog	PM
317	LD	NJ	NP	APLog	PP
318	LD	NJ	NP	APLog	PM
321	LOGD	UP	NP	AC	PP
322	LOGD	UP	NP	LC	PP
323	LOGD	UP	NP	AP	PP
324	LOGD	UP	NP	AP	PM
325	LOGD	UP	NP	APA	PP
326	LOGD	UP	NP	APA	PM
327	LOGD	UP	NP	APLog	PP
328	LOGD	UP	NP	APLog	PM

Continua na próxima página

Tabela 5.1 – continuação da página anterior

Alinhador	Distância	Árvore	Seleção Pares	Agrupamento	Peso
329	LOGD	–	BU	AC	PP
330	LOGD	–	BU	LC	PP
331	LOGD	–	BU	AP	PP
332	LOGD	–	BU	AP	PM
333	LOGD	–	BU	APA	PP
334	LOGD	–	BU	APA	PM
335	LOGD	–	BU	APLog	PP
336	LOGD	–	BU	APLog	PM
337	LOGD	NJ	NP	AC	PP
338	LOGD	NJ	NP	LC	PP
339	LOGD	NJ	NP	AP	PP
340	LOGD	NJ	NP	AP	PM
341	LOGD	NJ	NP	APA	PP
342	LOGD	NJ	NP	APA	PM
343	LOGD	NJ	NP	APLog	PP
344	LOGD	NJ	NP	APLog	PM
353	JTT	UP	NP	ACLog	PP
354	JTT	–	BU	ACLog	PP
355	JTT	NJ	NP	ACLog	PP
357	PMB	UP	NP	ACLog	PP
358	PMB	–	BU	ACLog	PP
359	PMB	NJ	NP	ACLog	PP
361	PAM	UP	NP	ACLog	PP
362	PAM	–	BU	ACLog	PP
363	PAM	NJ	NP	ACLog	PP
365	PCM	UP	NP	ACLog	PP
366	PCM	–	BU	ACLog	PP
367	PCM	NJ	NP	ACLog	PP
369	LD	UP	NP	ACLog	PP
370	LD	–	BU	ACLog	PP
371	LD	NJ	NP	ACLog	PP
373	LOGD	UP	NP	ACLog	PP
374	LOGD	–	BU	ACLog	PP
375	LOGD	NJ	NP	ACLog	PP

Continua na próxima página

Tabela 5.1 – continuação da página anterior

Alinhador	Distância	Árvore	Seleção Pares	Agrupamento	Peso
049b	JTT	UP	NP	ACb	PP
051b	JTT	UP	NP	APb	PP
052b	JTT	UP	NP	APb	PM
053b	JTT	UP	NP	APAb	PP
054b	JTT	UP	NP	APAb	PM
055b	JTT	–	BU	ACb	PP
057b	JTT	–	BU	APb	PP
058b	JTT	–	BU	APb	PM
059b	JTT	–	BU	APAb	PP
060b	JTT	–	BU	APAb	PM
061b	JTT	NJ	NP	ACb	PP
063b	JTT	NJ	NP	APb	PP
064b	JTT	NJ	NP	APb	PM
065b	JTT	NJ	NP	APAb	PP
066b	JTT	NJ	NP	APAb	PM
073b	PMB	UP	NP	ACb	PP
075b	PMB	UP	NP	APb	PP
076b	PMB	UP	NP	APb	PM
077b	PMB	UP	NP	APAb	PP
078b	PMB	UP	NP	APAb	PM
079b	PMB	–	BU	ACb	PP
081b	PMB	–	BU	APb	PP
082b	PMB	–	BU	APb	PM
083b	PMB	–	BU	APAb	PP
084b	PMB	–	BU	APAb	PM
085b	PMB	NJ	NP	ACb	PP
087b	PMB	NJ	NP	APb	PP
088b	PMB	NJ	NP	APb	PM
089b	PMB	NJ	NP	APAb	PP
090b	PMB	NJ	NP	APAb	PM
097b	PAM	UP	NP	ACb	PP
099b	PAM	UP	NP	APb	PP
100b	PAM	UP	NP	APb	PM
101b	PAM	UP	NP	APAb	PP

Continua na próxima página

Tabela 5.1 – continuação da página anterior

Alinhador	Distância	Árvore	Seleção Pares	Agrupamento	Peso
102b	PAM	UP	NP	APAb	PM
103b	PAM	–	BU	ACb	PP
105b	PAM	–	BU	APb	PP
106b	PAM	–	BU	APb	PM
107b	PAM	–	BU	APAb	PP
108b	PAM	–	BU	APAb	PM
109b	PAM	NJ	NP	ACb	PP
111b	PAM	NJ	NP	APb	PP
112b	PAM	NJ	NP	APb	PM
113b	PAM	NJ	NP	APAb	PP
114b	PAM	NJ	NP	APAb	PM
121b	PCM	UP	NP	ACb	PP
123b	PCM	UP	NP	APb	PP
124b	PCM	UP	NP	APb	PM
125b	PCM	UP	NP	APAb	PP
126b	PCM	UP	NP	APAb	PM
127b	PCM	–	BU	ACb	PP
129b	PCM	–	BU	APb	PP
130b	PCM	–	BU	APb	PM
131b	PCM	–	BU	APAb	PP
132b	PCM	–	BU	APAb	PM
133b	PCM	NJ	NP	ACb	PP
135b	PCM	NJ	NP	APb	PP
136b	PCM	NJ	NP	APb	PM
137b	PCM	NJ	NP	APAb	PP
138b	PCM	NJ	NP	APAb	PM
145b	LD	UP	NP	ACb	PP
147b	LD	UP	NP	APb	PP
148b	LD	UP	NP	APb	PM
149b	LD	UP	NP	APAb	PP
150b	LD	UP	NP	APAb	PM
151b	LD	–	BU	ACb	PP
153b	LD	–	BU	APb	PP
154b	LD	–	BU	APb	PM

Continua na próxima página

Tabela 5.1 – continuação da página anterior

Alinhador	Distância	Árvore	Seleção Pares	Agrupamento	Peso
155b	LD	–	BU	APAb	PP
156b	LD	–	BU	APAb	PM
157b	LD	NJ	NP	ACb	PP
159b	LD	NJ	NP	APb	PP
160b	LD	NJ	NP	APb	PM
161b	LD	NJ	NP	APAb	PP
162b	LD	NJ	NP	APAb	PM
281b	JTT	UP	NP	APLogb	PP
282b	JTT	UP	NP	APLogb	PM
283b	JTT	–	BU	APLogb	PP
284b	JTT	–	BU	APLogb	PM
285b	JTT	NJ	NP	APLogb	PP
286b	JTT	NJ	NP	APLogb	PM
289b	PMB	UP	NP	APLogb	PP
290b	PMB	UP	NP	APLogb	PM
291b	PMB	–	BU	APLogb	PP
292b	PMB	–	BU	APLogb	PM
293b	PMB	NJ	NP	APLogb	PP
294b	PMB	NJ	NP	APLogb	PM
297b	PAM	UP	NP	APLogb	PP
298b	PAM	UP	NP	APLogb	PM
299b	PAM	–	BU	APLogb	PP
300b	PAM	–	BU	APLogb	PM
301b	PAM	NJ	NP	APLogb	PP
302b	PAM	NJ	NP	APLogb	PM
305b	PCM	UP	NP	APLogb	PP
306b	PCM	UP	NP	APLogb	PM
307b	PCM	–	BU	APLogb	PP
308b	PCM	–	BU	APLogb	PM
309b	PCM	NJ	NP	APLogb	PP
310b	PCM	NJ	NP	APLogb	PM
313b	LD	UP	NP	APLogb	PP
314b	LD	UP	NP	APLogb	PM
315b	LD	–	BU	APLogb	PP

Continua na próxima página

Tabela 5.1 – continuação da página anterior

Alinhador	Distância	Árvore	Seleção Pares	Agrupamento	Peso
316b	LD	–	BU	APLogb	PM
317b	LD	NJ	NP	APLogb	PP
318b	LD	NJ	NP	APLogb	PM
321b	LOGD	UP	NP	ACb	PP
323b	LOGD	UP	NP	APb	PP
324b	LOGD	UP	NP	APb	PM
325b	LOGD	UP	NP	APAb	PP
326b	LOGD	UP	NP	APAb	PM
327b	LOGD	UP	NP	APLogb	PP
328b	LOGD	UP	NP	APLogb	PM
329b	LOGD	–	BU	ACb	PP
331b	LOGD	–	BU	APb	PP
332b	LOGD	–	BU	APb	PM
333b	LOGD	–	BU	APAb	PP
334b	LOGD	–	BU	APAb	PM
335b	LOGD	–	BU	APLogb	PP
336b	LOGD	–	BU	APLogb	PM
337b	LOGD	NJ	NP	ACb	PP
339b	LOGD	NJ	NP	APb	PP
340b	LOGD	NJ	NP	APb	PM
341b	LOGD	NJ	NP	APAb	PP
342b	LOGD	NJ	NP	APAb	PM
343b	LOGD	NJ	NP	APLogb	PP
344b	LOGD	NJ	NP	APLogb	PM
353b	JTT	UP	NP	ACLogb	PP
354b	JTT	–	BU	ACLogb	PP
355b	JTT	NJ	NP	ACLogb	PP
357b	PMB	UP	NP	ACLogb	PP
358b	PMB	–	BU	ACLogb	PP
359b	PMB	NJ	NP	ACLogb	PP
361b	PAM	UP	NP	ACLogb	PP
362b	PAM	–	BU	ACLogb	PP
363b	PAM	NJ	NP	ACLogb	PP
365b	PCM	UP	NP	ACLogb	PP

Continua na próxima página

Tabela 5.1 – continuação da página anterior

Alinhador	Distância	Árvore	Seleção Pares	Agrupamento	Peso
366b	PCM	–	BU	ACLogb	PP
367b	PCM	NJ	NP	ACLogb	PP
369b	LD	UP	NP	ACLogb	PP
370b	LD	–	BU	ACLogb	PP
371b	LD	NJ	NP	ACLogb	PP
373b	LOGD	UP	NP	ACLogb	PP
374b	LOGD	–	BU	ACLogb	PP
375b	LOGD	NJ	NP	ACLogb	PP
053p	JTT	UP	NP	APAp	PP
054p	JTT	UP	NP	APAp	PM
059p	JTT	–	BU	APAp	PP
060p	JTT	–	BU	APAp	PM
065p	JTT	NJ	NP	APAp	PP
066p	JTT	NJ	NP	APAp	PM
077p	PMB	UP	NP	APAp	PP
078p	PMB	UP	NP	APAp	PM
083p	PMB	–	BU	APAp	PP
084p	PMB	–	BU	APAp	PM
089p	PMB	NJ	NP	APAp	PP
090p	PMB	NJ	NP	APAp	PM
101p	PAM	UP	NP	APAp	PP
102p	PAM	UP	NP	APAp	PM
107p	PAM	–	BU	APAp	PP
108p	PAM	–	BU	APAp	PM
113p	PAM	NJ	NP	APAp	PP
114p	PAM	NJ	NP	APAp	PM
125p	PCM	UP	NP	APAp	PP
126p	PCM	UP	NP	APAp	PM
131p	PCM	–	BU	APAp	PP
132p	PCM	–	BU	APAp	PM
137p	PCM	NJ	NP	APAp	PP
138p	PCM	NJ	NP	APAp	PM
149p	LD	UP	NP	APAp	PP
150p	LD	UP	NP	APAp	PM

Continua na próxima página

Tabela 5.1 – continuação da página anterior

Alinhador	Distância	Árvore	Seleção Pares	Agrupamento	Peso
155p	LD	–	BU	APAp	PP
156p	LD	–	BU	APAp	PM
161p	LD	NJ	NP	APAp	PP
162p	LD	NJ	NP	APAp	PM
325p	LOGD	UP	NP	APAp	PP
326p	LOGD	UP	NP	APAp	PM
333p	LOGD	–	BU	APAp	PP
334p	LOGD	–	BU	APAp	PM
341p	LOGD	NJ	NP	APAp	PP
342p	LOGD	NJ	NP	APAp	PM

Tabela 5.1: Lista dos alinhadores implementados, explicitando os métodos utilizados. Onde JTT indica modelo de *Jones-Taylor-Thornton*, PMB o modelo de *Probability Rate Matriz*, PAM o de PAM de *Dayhoff* e PCM o modelo de categorias, LD distância local, $LogD$ distância logarítmica, para cálculo da distância. Para construção da árvore temos UP indicando $UPGMA$ e NJ indicando *Neighbor Joining*. A seleção de pares é representada por NP (*Nearest Pair*) e BU (*Unique Block*). Os métodos de agrupamento são abreviados por $AC, ACLog, LC, AP, APA, APLog$, que indicam respectivamente: agrupamento por consenso, agrupamento por consenso com pontuação logarítmica, agrupamento de consenso local, agrupamento de perfil, agrupamento de perfil com função afim e agrupamento de perfil com pontuação logarítmica. Finalmente PP indica ausência de peso das sequências e PM o uso de média das distâncias

5.1 BAliBASE

BAliBASE (*Benchmark Alignment dataBASE*) [67, 68] é um banco de dados de alinhamentos múltiplos de sequências manualmente refinado, dividido em categorias de acordo com o tamanho dos blocos conservados, similaridade e presença de inserções e de extensões N/C-terminais nas sequências.

A determinação da qualidade dos alinhamentos produzidos por algoritmos que realizam o MSA usavam um pequeno número de casos de testes selecionados pelo desenvolvedor do algoritmo. Tendo em vista que o desempenho de alinhadores depende do número de sequências, do grau de similaridade entre elas e do número de inserções do alinhamento, é interessante ter um conjunto de testes que apresente todos esses requisitos.

Neste sentido o BAliBASE foi construído. Ele contém alinhamentos de sequências de proteínas de alta qualidade para identificar os pontos fortes e fracos dos diversos alinhadores disponíveis atualmente.

A versão 3.0 do BAliBASE, possui sequências divididas em 5 conjuntos. No primeiro conjunto existem poucas sequências, com tamanho similar e pequena porcentagem de aminoácidos diferentes entre elas, subdivide-se em 2 grupos o RV11 com menos de 20% de identidade entre as sequências e o RV12 com identidade entre 20% e 40%. O segundo conjunto (RV20) contém alinhamentos de uma família (sequências com mais de 25% de identidade) e três sequências órfãs (menos de 20% de identidade). O terceiro conjunto (RV30) é composto por alinhamentos de quatro diferentes famílias de proteínas com menos de 25% de identidade. O conjunto quatro (RV40) contém sequências com grandes extensões terminais e o conjunto cinco (RV50) com grandes inserções internas.

O BAliBASE 3.0 oferece 218 entradas com sequências completas para testes e 168 entradas com sequências que possuem apenas regiões homólogas. Todos os conjuntos, com exceção do RV40, contêm arquivos com sequências de proteínas completas e arquivos que possuem as regiões homólogas correspondentes às completas. Cada arquivo de teste do BAliBASE possui as sequências em formato FASTA, o alinhamento de referência em formato MSF e o alinhamento de referência com mais informações, como de blocos conservados, em formato XML.

Além das sequências para testes, o BAliBASE disponibiliza um programa, denominado `bali_score`, que calcula pontuações comparativas entre seu alinhamento de referência e um alinhamento fornecido (em formato MSF). As duas pontuações calculadas são a de soma de pares (SP) e pontuação total de colunas (TC). SP é a razão entre o número de pares corretamente alinhados e o no número de pares alinhados no alinhamento de referência. TC é a razão entre o número de colunas alinhadas corretamente no alinhamento de teste e o número de colunas alinhadas no alinhamento de referência [17]. Ambos tem escala entre 0.0 e 1.0, e quanto maior a pontuação em relação ao de referência, melhor é o alinhamento.

A Tabela 5.2 apresenta as características dos 5 conjuntos de referência. Características relacionadas a todas as sequências são nomeadas como A_{xxx} e S_{xxx}, as relacionadas com as sequências que são completas são nomeadas como AC_{xxx} e SC_{xxx}, e as relacionadas com as sequências que possuem apenas as regiões homólogas são nomeadas como AH_{xxx} e SH_{xxx}.

Parâmetro	RV11	RV12	RV20	RV30	RV40	RV50
A_{num}	76	88	82	60	49	31
S_{num}/A_{num}	6,87	9,00	45,59	63,17	27,63	28,55
S_{min}	47	49	36	50	55	151
S_{max}	1192	2766	1520	1293	7923	1902
S_{med}	275,20	345,38	314,96	320,86	465,51	422,12
S_{desv}	40,10	53,01	50,75	56,87	281,27	99,75
AC_{num}	38	44	41	30	49	16
SC_{min}	51	49	52	63	55	172
SC_{max}	1192	2766	1520	1293	7923	1902
SC_{med}	309,47	387,44	384,65	380,66	465,51	515,06
SC_{desv}	62,86	93,25	93,30	98,25	281,87	139,02
AH_{num}	38	44	41	30	0	15
AH_{min}	47	49	36	50	-	151
AH_{max}	544	1006	602	621	-	957
AH_{med}	240,93	303,33	245,26	261,07	-	327,28
AH_{desv}	17,33	12,77	8,21	15,49	-	57,86

Tabela 5.2: Características dos conjuntos do BAliBASE. A_{num} corresponde ao número total de alinhamentos em cada conjunto de referência. S_{num}/A_{num} corresponde ao número médio de sequências em cada alinhamento, S_{min}, ao comprimento mínimo de sequência no conjunto, S_{max}, ao comprimento máximo, S_{med}, ao comprimento médio e S_{desv}, ao desvio padrão do tamanho das sequências. Os respectivos valores para AC, SC_{xxx} e AH, SC_{xxx} correspondem aos valores anteriores para alinhamentos com sequências completas e alinhamentos com sequências que possuem apenas as regiões homólogas

Assim como o BAliBASE existem outros *benchmarks* como SABmark [74], HOMSTRAD [44] e PREFAB [16], que poderiam ser utilizados para comparar os alinhadores implementados. Escolhemos o BAliBASE (versão 3.0) para ser usado na avaliação dos alinhadores desenvolvidos neste trabalho devido ao refinamento manual e a divisão de categorias que possui. Além de utilizar as sequências de proteínas como entrada, também utilizamos a comparação da pontuação, calculada pelo `bali_score`, dos alinhamentos de referência do BAliBASE contra os produzidos pelos nossos alinhadores.

5.2 Escolha de Parâmetros

Devido ao grande número de alinhadores originados pela combinação dos métodos estudados realizamos testes preliminares, em uma máquina Intel Core 2 Duo de 2.33GHz e 3GB de memória, e notamos a alta sensibilidade dos alinhadores em relação aos valores dos parâmetros utilizados. Decidimos realizar um estudo para adequar os parâmetros com as sequências de entrada, de acordo com os alinhadores a serem utilizados.

5.2.1 Alinhamento Recursivo

Testes preliminares com o alinhador local resultaram em alinhamentos com muitos buracos e *mismatches*. Isso poderia estar ocorrendo devido à possibilidade do alinhamento recursivo ser de qualquer tamanho, o que resultaria na quebra de regiões mais conservadas das sequências.

A Tabela 5.3, mostra os piores alinhadores encontrados quando realizamos testes preliminares com um conjunto que continha 6 arquivos de referência do BAliBASE (o menor de cada conjunto de referência RV11 a RV50). Como podemos observar em todos eles as distâncias entre os pares de sequências foram calculadas utilizando o algoritmo de distância local, mas que não exigia um limite mínimo para o tamanho do alinhamento local. Por exemplo, o pior alinhador foi o 165b que obteve pontuação de soma de pares igual a $32,43$ e pontuação total de colunas igual a $21,67$. Este alinhador utilizou a distância local para calcular a matriz de distâncias, bloco único para selecionar os pares, alinhamento de perfil semi-global e esquema de pesos desativado.

Uma alternativa para evitar a quebra dos blocos conservados foi utilizar o Algoritmo 3.1.1, para alinhamento local de pares, com a alteração de exigir um tamanho mínimo para o alinhamento local a ser retornado em cada recursão. No entanto, nos deparamos com o problema de escolher qual o tamanho mínimo a ser exigido para o alinhamento, uma vez que este depende das sequências que deseja-se alinhar. Caminhamos para um estudo para determinar o melhor limite a ser utilizado quando a entrada dos alinhadores são os conjuntos de referência do BAliBASE 3.0.

Alinhador	SP	TC	CD	AG	SL	MA	PS
147b	37,93	24,25	LD	UP	NP	APb	PP
148b	37,92	24,33	LD	UP	NP	APb	PM
154	37,83	22,67	LD	-	BU	AP	PM
159	37,20	22,75	LD	NJ	NP	AP	PP
160	35,56	18,92	LD	NJ	NP	AP	PM
160b	34,62	22,75	LD	NJ	NP	APb	PM
159b	33,70	22,58	LD	NJ	NP	APb	PP
165b	32,43	21,67	LD	-	BU	APb	PP

Tabela 5.3: Pontuação SP e TC para os piores alinhadores, nos testes iniciais. Na primeira coluna está o identificador do alinhador, em seguida estão as pontuações SP e TC e depois os métodos utilizados por ele para cálculo de distância, geração de árvore guia, seleção do par, agrupamento de alinhamentos e esquema de peso.

Escolha do Tamanho Mínimo do Alinhamento

Para observar qual o melhor limite mínimo de usado no alinhamento local foi utilizado um algoritmo que calculava a porcentagem *core blocks* corretamente alinhados (tomando por base os *core blocks* indicados pelo BAliBASE nos arquivos com extensão XML). Este algoritmo marca as posições de início e fim de cada um dos *core blocks* e então pega as subsequências correspondentes a cada um deles. As subsequências identificadas como alinhamentos de *core blocks* são comparadas com subsequências correspondentes alinhadas pelo alinhamento recursivo. Note que, são observados os aminoácidos alinhados das sequências para comparar as subsequências, e não as posições dos mesmos em relação as sequências completas.

A análise da porcentagem de *core blocks* alinhados foi feita da seguinte forma. Combinamos todos os pares de sequências e cada um dos dez arquivos com maior número de sequências de cada conjunto de referência do BAliBASE, totalizando 103.759 pares. Por exemplo, do conjunto RV11 foram selecionados os 10 arquivos com maior número de sequência, estes foram: 5, 7, 18, 19, 20, 28, 30, 31, 33 e 38. Em seguida, para cada um deles foi feita uma combinação de pares de todas sequências com todas sequências, por exemplo, o arquivo 5 que possui 14 sequências originou 91 pares.

O número de pares de cada arquivo de sequências de cada um dos cinco conjuntos de referência pode ser visto na Tabela 5.4.

Cada um dos pares de sequências foram alinhados com o alinhador recursivo (Algoritmo 3), utilizando diferentes valores mínimos permitidos para o alinhamento local (utilizamos 0, 50, 100, 150 e 200 como limite). Com isso, para conjunto do BAliBASE

RV11			RV12			RV20			RV30			RV40			RV50		
A	S	P	A	S	P	A	S	P	A	S	P	A	S	P	A	S	P
5	14	91	4	15	105	3	74	2701	1	116	6670	2	55	1485	1	34	561
7	9	36	8	13	78	14	65	2080	3	142	10011	4	67	2211	3	43	903
18	14	91	11	12	66	25	81	3240	5	113	6328	11	39	741	6	60	1770
19	10	45	15	12	66	31	60	1770	13	86	3655	16	42	861	7	23	253
20	9	36	26	18	153	32	61	1830	18	78	3003	17	42	861	8	26	325
28	10	45	27	13	78	34	59	1711	21	140	9730	37	46	1035	9	28	378
30	14	91	29	12	66	36	91	4095	23	77	2926	41	55	1485	11	36	630
31	11	55	35	27	351	37	65	2080	26	81	3240	44	47	1081	12	49	1176
33	11	55	37	13	78	39	91	4095	28	89	3916	47	54	1431	14	30	435
38	8	28	43	34	561	40	87	3741	29	98	4753	49	62	1891	15	32	496

Tabela 5.4: Número de sequências de cada conjunto usado nos alinhamentos para escolha do limite do alinhador local. A coluna A indica o arquivo de sequências, S é o número de sequências por arquivo e a coluna P indica o número de pares de sequência para aquele arquivo

construímos 5 alinhamentos locais diferentes. Estes pares também foram alinhados por um alinhador global.

Os alinhamentos produzidos foram submetidos ao Algoritmo 12, que calcula a porcentagem de *core blocks* corretamente alinhados, e calculou-se a média dessas porcentagens para todos os pares de cada conjunto e para cada alinhador. As médias foram calculadas não só levando em consideração os pares formados por qualquer sequência do conjunto. Calculou-se também as médias de pares de sequências formados apenas por aquelas cujo tamanho era menor que o valor da mediana do conjunto. As médias de pares de sequências formados apenas por sequências cujo tamanho era maior que o valor da mediana do conjunto, e de pares de sequências em que uma era de tamanho maior que a mediana e a outra era menor.

Posteriormente, para cada grupo do BAliBASE (RV11 a RV50) foi calculada a média das médias de porcentagens dos *core blocks* alinhados nos 10 conjuntos de cada grupo utilizado. Na Tabela 5.5 podemos observar as porcentagens de *core blocks* alinhados para os pares dos 10 conjuntos do RV11, na Tabela 5.6 para os do RV12, na Tabela 5.7 para os do RV20, na Tabela 5.8 para os pares do conjunto RV30, na Tabela 5.9 para os do RV40 e na Tabela 5.10 para os pares do conjunto RV50.

A Tabela 5.11 mostra a média das pontuações dos 10 maiores conjuntos dos grupos RV11 a RV50, para cada um dos limites de tamanho de alinhamento utilizados. As melhores pontuações dos alinhamentos aconteceram quando o limite para o tamanho

Tamanho mínimo alinhamento	Média mm	Média mM	Média MM	Média $Total$
0	35,31	30,35	30,76	31,22
50	38,14	33,21	33,60	34,07
100	46,75	41,01	41,22	41,90
150	46,84	41,74	41,98	42,55
200	46,78	41,43	41,80	42,34
Global	45,42	37,86	37,73	38,89

Tabela 5.5: Média da porcentagem de *core blocks* bem alinhados nos alinhamentos de pares de sequências dos 10 maiores conjuntos do RV11. Média mm, corresponde a média do alinhamento de pares de sequência que são menores que a mediana, mM,corresponde a média do alinhamento de pares de sequência que são menores que a mediana com as maiores que a mediana, MM corresponde a média do alinhamento de pares de sequência que são maiores que a mediana e $Total$ são de todas as sequências contra todas as sequências.

Tamanho mínimo alinhamento	Média mm	Média mM	Média MM	Média $Total$
0	74,56	68,95	69,44	69,94
50	78,22	74,13	74,00	74,62
100	78,41	75,23	75,42	75,76
150	77,81	75,20	75,87	75,88
200	75,61	74,88	75,60	75,31
Global	73,26	67,17	67,55	68,16

Tabela 5.6: Média da porcentagem de *core blocks* bem alinhados nos alinhamentos de pares de sequências dos 10 maiores conjuntos do RV12. Média mm, corresponde a média do alinhamento de pares de sequência que são menores que a mediana, mM,corresponde a média do alinhamento de pares de sequência que são menores que a mediana com as maiores que a mediana, MM corresponde a média do alinhamento de pares de sequência que são maiores que a mediana e $Total$ são de todas as sequências contra todas as sequências.

Tamanho mínimo alinhamento	Média mm	Média mM	Média MM	Média $Total$
0	84,48	84,50	85,03	84,77
50	86,09	86,11	86,56	86,34
100	86,67	86,94	87,37	87,13
150	85,98	86,08	86,44	86,26
200	86,00	86,00	86,30	86,16
Global	84,80	83,28	83,18	83,47

Tabela 5.7: Média da porcentagem de *core blocks* bem alinhados nos alinhamentos de pares de sequências dos 10 maiores conjuntos do RV20. Média mm, corresponde a média do alinhamento de pares de sequência que são menores que a mediana, mM,corresponde a média do alinhamento de pares de sequência que são menores que a mediana com as maiores que a mediana, MM corresponde a média do alinhamento de pares de sequência que são maiores que a mediana e $Total$ são de todas as sequências contra todas as sequências.

Tamanho mínimo alinhamento	Média mm	Média mM	Média MM	Média $Total$
0	68,60	60,16	61,31	61,86
50	69,13	62,02	63,07	63,49
100	72,14	64,83	65,88	66,32
150	73,86	66,72	67,55	68,09
200	73,86	66,48	67,23	67,84
Global	71,17	61,82	62,45	63,39

Tabela 5.8: Média da porcentagem de *core blocks* bem alinhados nos alinhamentos de pares de sequências dos 10 maiores conjuntos do RV30. Média mm, corresponde a média do alinhamento de pares de sequência que são menores que a mediana, mM,corresponde a média do alinhamento de pares de sequência que são menores que a mediana com as maiores que a mediana, MM corresponde a média do alinhamento de pares de sequência que são maiores que a mediana e $Total$ são de todas as sequências contra todas as sequências.

Tamanho mínimo alinhamento	Média mm	Média mM	Média MM	Média $Total$
0	68,51	55,52	52,91	55,93
50	78,27	67,64	64,09	67,27
100	75,83	67,50	65,72	67,70
150	71,56	57,90	56,97	59,21
200	71,86	57,48	56,33	58,78
Global	65,68	48,93	47,67	50,48

Tabela 5.9: Média da porcentagem de *core blocks* bem alinhados nos alinhamentos de pares de sequências dos 10 maiores conjuntos do RV40. Média mm, corresponde a média do alinhamento de pares de sequência que são menores que a mediana, mM,corresponde a média do alinhamento de pares de sequência que são menores que a mediana com as maiores que a mediana, MM corresponde a média do alinhamento de pares de sequência que são maiores que a mediana e $Total$ são de todas as sequências contra todas as sequências.

Tamanho mínimo alinhamento	Média mm	Média mM	Média MM	Média $Total$
0	59,82	41,78	46,46	47,86
50	63,4	48,17	51,89	52,76
100	67,3	55,03	58,73	58,93
150	67,02	53,6	59,89	58,31
200	63,09	50,92	59,27	54,94
Global	60,37	47,93	55,54	50,55

Tabela 5.10: Média da porcentagem de *core blocks* bem alinhados nos alinhamentos de pares de sequências dos 10 maiores conjuntos do RV50. Média mm, corresponde a média do alinhamento de pares de sequência que são menores que a mediana, mM,corresponde a média do alinhamento de pares de sequência que são menores que a mediana com as maiores que a mediana, MM corresponde a média do alinhamento de pares de sequência que são maiores que a mediana e $Total$ são de todas as sequências contra todas as sequências.

Tamanho mínimo alinhamento	Média mm	Média mM	Média MM	Média $Total$
0	65,22	56,88	57,65	58,60
50	68,88	61,88	62,20	63,09
100	71,19	65,09	65,72	66,29
150	70,52	63,54	64,78	65,05
200	69,54	62,87	64,42	64,23
Global	66,79	57,83	59,02	59,16

Tabela 5.11: Média da porcentagem de *core blocks* bem alinhados nos alinhamentos de pares de sequências dos 10 maiores conjuntos de cada um dos grupos RV11 a RV50. Média mm, corresponde a média do alinhamento de pares de sequência que são menores que a mediana, mM,corresponde a média do alinhamento de pares de sequência que são menores que a mediana com as maiores que a mediana, MM corresponde a média do alinhamento de pares de sequência que são maiores que a mediana e $Total$ são de todas as sequências contra todas as sequências.

mínimo do alinhamento de pares foi 100. Portanto, este valor foi usado nos testes finais dos alinhadores que envolveram agrupamento utilizando alinhamento local e o cálculo de distância utilizando distância local.

5.2.2 Alinhamento de Perfil

Os resultados dos testes preliminares com os alinhadores não foram condizentes com o esperado pela literatura, pois alinhadores utilizando agrupamento por consenso foram superiores (produziram alinhamentos com maior pontuação SP e TC em relação às sequências do BAliBASE) aos que utilizam agrupamento por perfil. Uma explicação para isto seriam os parâmetros utilizados para os testes. No entanto, determinar os parâmetros para penalizar buracos e qual a melhor matriz de substituição de aminoácidos deve ser utilizada não são tarefas triviais, principalmente por não existir um padrão na literatura.

Diante da ausência de um consenso sobre o assunto, partiu-se para o estudo empírico a fim de determinar quais os melhores parâmetros a serem utilizados por cada alinhador. Os nossos testes começaram pelos alinhadores de perfil com pontuação global e perfil com pontuação semi-global, por terem tido pior desempenho. Arbitrariamente escolhemos o alinhador 051 (perfil), que recebeu como entrada 5 grupos de sequências. Cada um desses grupos era composto por um arquivo (o menor deles) de cada um dos conjuntos de entrada do BAliBASE, por exemplo, o primeiro grupo possuía seis conjuntos de sequências, os seis com menor número de sequências de cada conjunto RV11 a RV50 do BAliBASE. O segundo conjunto era composto pelos seis com menor número de sequências de cada

conjunto, excluindo os que já estavam no primeiro grupo, e assim sucessivamente. A escolha de tamanho a partir dos menores deve-se ao tempo de processamento necessário para os testes, o tempo despendido seria excessivo se escolhêssemos os maiores conjuntos.

Avançamos os testes variando a pontuação de *gap* de -1 a -20 e matrizes BLOSUM 45, 50, 55, 60, 62, 65, 70, 75, 80. Na Figura 5.2 observa-se o gráfico com a média dos valores para cada conjunto de referência. E na Figura 5.3 os resultados para cada um dos seis conjuntos de referência, em todos os casos a média utilizada foi o de cinco resultados da pontuação de soma de pares em regiões *core blocks* contra os alinhamentos de referência do BAliBASE.

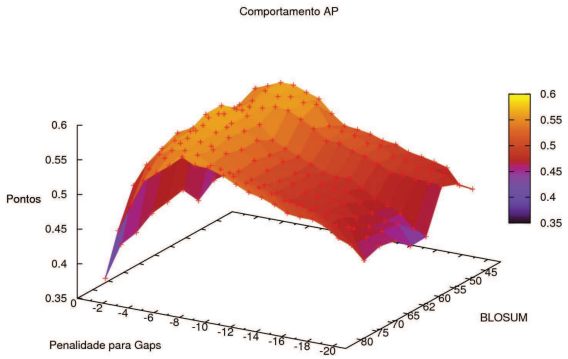


Figura 5.2: Pontuação do alinhador de perfil com alinhamento global (051), com pontuação para *gap* variando entre -1 e -20, e BLOSUM de 45 a 80.

A mesma metodologia foi usada para observar o comportamento de um alinhador de perfil com pontuação semi-global, para isto foi escolhido, aleatoriamente, o 051b. Na Figura 5.4 observa-se o gráfico com a média dos valores para cada conjunto de referência, e na Figura 5.5 os resultados para cada um dos seis conjuntos de referência. Em todos os casos a média utilizada foi a de cinco resultados da pontuação de soma de pares em regiões *core blocks*.

Em paralelo realizamos o estudo dos melhores parâmetros para os alinhadores de perfil que utilizam função afim para penalidade de *gap*, seja com alinhamento global ou semi-global. Os valores de *gep* variaram de -1 a -10, de *gop* de -1 a -20 e as matrizes BLOSUM utilizadas foram 45, 50, 55, 60, 62, 65, 70, 75, 80. Todas as combinações possíveis para os valores de *gep*, *gop* e BLOSUM foram usadas para testar os parâmetros dos alinhadores 053 e 053b, escolhidos aleatoriamente. As Tabelas 5.12 e 5.13 mostram os 20 valores de *gop* e das 9 matrizes para o *gep* que obteve maior pontuação SP para os

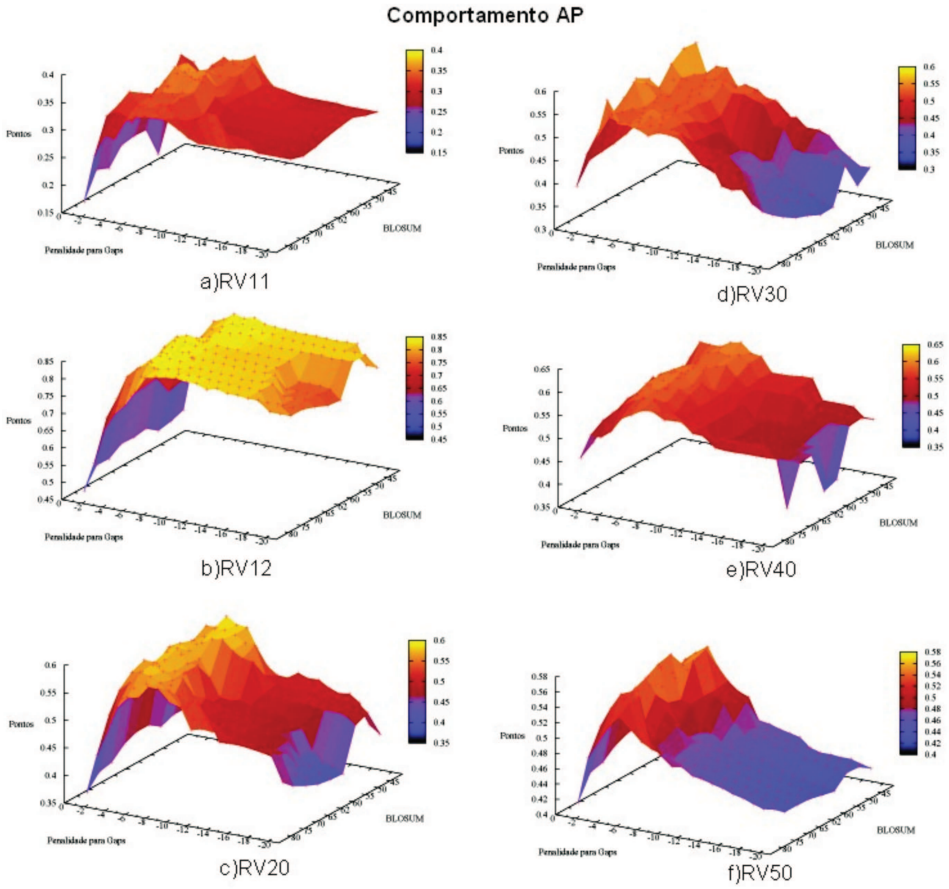


Figura 5.3: Pontuação do alinhador de perfil com alinhamento global (051), com pontuação para *gap* variando entre -1 e -20, e BLOSUM de 45 a 80. Em a) temos o gráfico para o conjunto RV11, em b) para RV12, em c) para RV20, em d) para RV30, em e) para RV40 e em f) para RV50

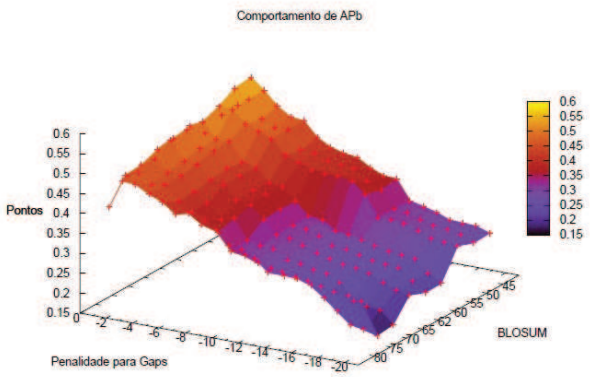


Figura 5.4: Pontuação do alinhador de perfil com alinhamento semi-global (051b), com pontuação para *gap* variando entre -1 e -20, e BLOSUM de 45 a 80.

alinhamentos, para os alinhadores 053 e 053b, respectivamente.

Os alinhadores de perfil com pontuação logarítmica também dependem de um bom ajuste nos valores de *gop*, *c* (pontuação para um grupo de buracos) e da matriz de substituição. Procedemos os testes de forma semelhante ao que foi realizado com o alinhador de perfil com pontuação afim. Escolhemos aleatoriamente os alinhadores 281 e 281b, e variamos os valores de c de -1 a -10, de *gop* de -1 a -20 e as matrizes BLOSUM 45, 50, 55, 60, 62, 65, 70, 75, 80. As Tabelas 5.14 e 5.15 mostram os 20 valores de *gop* e das 9 matrizes para o valor de c que obteve maior pontuação SP para os alinhamentos, para os alinhadores 281 e 281b, respectivamente.

Através dos gráficos das Figuras 5.2 a 5.5 observamos que o alinhador de perfil global deve utilizar BLOSUM 62 e *gap* -5, que o de perfil com pontuação semi-global deve usar BLOSUM 45 e *gap* -2. As Tabelas 5.12 e 5.13 mostraram que o alinhador global de perfil com função afim deve usar BLOSUM 55, *gop* -17 e *gep* -1 e que o alinhador semi-global de perfil com função afim deve utilizar BLOSUM 45, *gop* -10 e *gep* -1. Finalmente, observando as Tabelas 5.14 e 5.15 percebemos que o alinhador de perfil global com pontuação logarítmica deve utilizar qualquer BLOSUM, exceto a 45, desde que o *gop* seja -4 e c seja -9, e que o alinhador de perfil semi-global com pontuação logarítmica deve utilizar qualquer BLOSUM, com *gop* -5 e c -8.

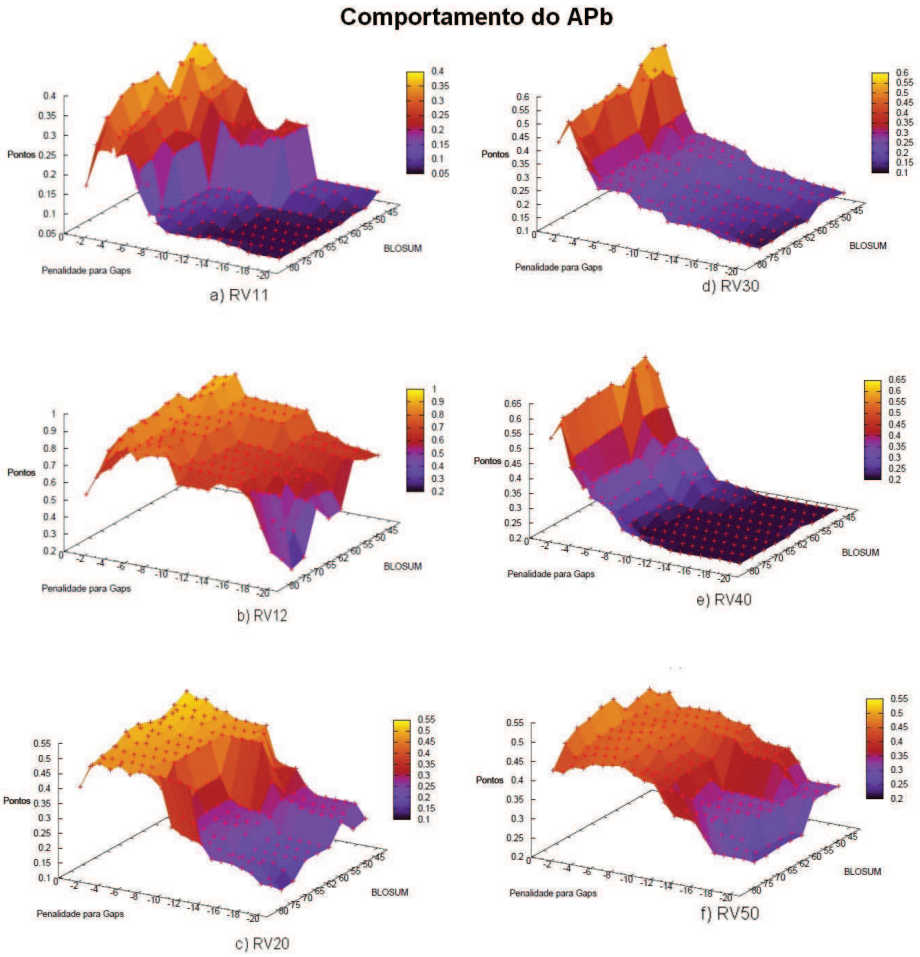


Figura 5.5: Pontuação do alinhador de perfil com alinhamento semi-global (051b), com pontuação para *gap* variando entre -1 e -20, e BLOSUM de 45 a 80. Em a) temos o gráfico para o conjunto RV11, em b) para RV12, em c) para RV20, em d) para RV30, em e) para RV40 e em f) para RV50

		BLOSUM								
		45	50	55	60	62	65	70	75	80
gop	-1	0.385	0.426	0.442	0.452	0.461	0.461	0.415	0.430	0.443
	-2	0.422	0.509	0.512	0.528	0.536	0.537	0.484	0.513	0.528
	-3	0.480	0.569	0.587	0.580	0.588	0.607	0.553	0.570	0.582
	-4	0.524	0.622	0.626	0.637	0.647	0.644	0.594	0.619	0.648
	-5	0.573	0.655	0.644	0.666	0.673	0.682	0.642	0.656	0.656
	-6	0.597	0.657	0.668	0.673	0.683	0.689	0.669	0.645	0.675
	-7	0.623	0.674	0.684	0.690	0.704	0.699	0.673	0.665	0.695
	-8	0.638	0.693	0.704	0.700	0.708	0.703	0.697	0.677	0.707
	-9	0.650	0.699	0.707	0.693	0.713	0.708	0.713	0.699	0.708
	-10	0.653	0.685	0.703	0.697	0.709	0.708	0.701	0.704	0.711
	-11	0.661	0.692	0.709	0.707	0.707	0.710	0.705	0.711	0.710
	-12	0.677	0.701	0.707	0.704	0.706	0.712	0.712	0.711	0.714
	-13	0.683	0.702	0.711	0.704	0.707	0.707	0.714	0.715	0.712
	-14	0.687	0.701	0.707	0.703	0.705	0.702	0.717	0.711	0.712
	-15	0.695	0.695	0.705	0.701	0.701	0.702	0.717	0.717	0.713
	-16	0.702	0.692	0.696	0.698	0.699	0.693	0.720	0.713	0.712
	-17	0.711	0.693	0.694	0.695	0.701	0.693	0.721	0.702	0.706
	-18	0.716	0.694	0.694	0.697	0.699	0.698	0.710	0.701	0.705
	-19	0.721	0.692	0.687	0.691	0.697	0.690	0.705	0.700	0.701
	-20	0.714	0.680	0.687	0.687	0.692	0.682	0.702	0.700	0.704

Tabela 5.12: Pontuação de SP para *gep*=-1 para o alinhador de perfil global com pontuação afim para cada BLOSUM e para cada uns dos 20 *gop* do alinhador 053

		BLOSUM								
		45	50	55	60	62	65	70	75	80
gop	-1	0.396	0.463	0.482	0.492	0.495	0.514	0.451	0.476	0.491
	-2	0.459	0.553	0.564	0.590	0.602	0.600	0.529	0.550	0.576
	-3	0.525	0.631	0.639	0.644	0.648	0.660	0.586	0.617	0.627
	-4	0.555	0.657	0.685	0.670	0.680	0.678	0.644	0.650	0.677
	-5	0.608	0.658	0.665	0.671	0.676	0.687	0.665	0.677	0.707
	-6	0.634	0.645	0.668	0.680	0.684	0.691	0.685	0.690	0.718
	-7	0.670	0.629	0.666	0.711	0.713	0.715	0.705	0.712	0.733
	-8	0.670	0.624	0.654	0.708	0.705	0.709	0.726	0.733	0.736
	-9	0.678	0.590	0.609	0.634	0.695	0.649	0.715	0.728	0.743
	-10	0.658	0.562	0.600	0.637	0.660	0.644	0.718	0.720	0.758
	-11	0.663	0.566	0.581	0.656	0.656	0.662	0.710	0.714	0.731
	-12	0.662	0.538	0.576	0.628	0.641	0.646	0.708	0.716	0.736
	-13	0.618	0.539	0.549	0.615	0.642	0.643	0.712	0.718	0.729
	-14	0.619	0.535	0.550	0.590	0.617	0.623	0.718	0.682	0.731
	-15	0.593	0.525	0.547	0.584	0.602	0.602	0.682	0.688	0.694
	-16	0.587	0.524	0.546	0.584	0.602	0.584	0.675	0.683	0.689
	-17	0.583	0.524	0.539	0.491	0.573	0.583	0.678	0.678	0.689
	-18	0.578	0.523	0.539	0.462	0.548	0.548	0.678	0.660	0.684
	-19	0.586	0.517	0.528	0.462	0.548	0.548	0.676	0.660	0.684
	-20	0.556	0.516	0.522	0.462	0.549	0.550	0.676	0.664	0.684

Tabela 5.13: Pontuação de SP para *gep*=-1 para o alinhador de perfil semi-global com pontuação afim para cada BLOSUM e para cada uns dos 20 *gop* do alinhador 053b

		BLOSUM								
		45	50	55	60	62	65	70	75	80
gop	-1	0.465	0.533	0.533	0.533	0.533	0.533	0.533	0.533	0.533
	-2	0.543	0.664	0.664	0.664	0.664	0.664	0.664	0.664	0.664
	-3	0.627	0.746	0.746	0.746	0.746	0.746	0.746	0.746	0.746
	-4	0.764	0.795	0.795	0.795	0.795	0.795	0.795	0.795	0.795
	-5	0.795	0.820	0.820	0.820	0.820	0.820	0.820	0.820	0.820
	-6	0.838	0.859	0.859	0.859	0.859	0.859	0.859	0.859	0.859
	-7	0.867	0.884	0.884	0.884	0.884	0.884	0.884	0.884	0.884
	-8	0.874	0.890	0.890	0.890	0.890	0.890	0.890	0.890	0.890
	-9	0.901	0.903	0.903	0.903	0.903	0.903	0.903	0.903	0.903
	-10	0.890	0.890	0.890	0.890	0.890	0.890	0.890	0.890	0.890
	-11	0.874	0.874	0.874	0.874	0.874	0.874	0.874	0.874	0.874
	-12	0.876	0.876	0.876	0.876	0.876	0.876	0.876	0.876	0.876
	-13	0.875	0.875	0.875	0.875	0.875	0.875	0.875	0.875	0.875
	-14	0.875	0.875	0.875	0.875	0.875	0.875	0.875	0.875	0.875
	-15	0.889	0.889	0.889	0.889	0.889	0.889	0.889	0.889	0.889
	-16	0.887	0.887	0.887	0.887	0.887	0.887	0.887	0.887	0.887
	-17	0.850	0.850	0.850	0.850	0.850	0.850	0.850	0.850	0.850
	-18	0.819	0.819	0.819	0.819	0.819	0.819	0.819	0.819	0.819
	-19	0.820	0.820	0.820	0.820	0.820	0.820	0.820	0.820	0.820
	-20	0.820	0.820	0.820	0.820	0.820	0.820	0.820	0.820	0.820

Tabela 5.14: Pontuação de SP para c=-4 para o alinhador de perfil global com pontuação logarítmica para cada BLOSUM e para cada uns dos 20 *gop* do alinhador 281

		BLOSUM								
		45	50	55	60	62	65	70	75	80
gop	-1	0.454	0.454	0.454	0.454	0.454	0.454	0.454	0.454	0.454
	-2	0.562	0.562	0.562	0.562	0.562	0.562	0.562	0.562	0.562
	-3	0.625	0.625	0.625	0.625	0.625	0.625	0.625	0.625	0.625
	-4	0.679	0.679	0.679	0.679	0.679	0.679	0.679	0.679	0.679
	-5	0.693	0.693	0.693	0.693	0.693	0.693	0.693	0.693	0.693
	-6	0.705	0.705	0.705	0.705	0.705	0.705	0.705	0.705	0.705
	-7	0.737	0.737	0.737	0.737	0.737	0.737	0.737	0.737	0.737
	-8	0.758	0.758	0.758	0.758	0.758	0.758	0.758	0.758	0.758
	-9	0.582	0.582	0.582	0.582	0.582	0.582	0.582	0.582	0.582
	-10	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581
	-11	0.582	0.582	0.582	0.582	0.582	0.582	0.582	0.582	0.582
	-12	0.584	0.584	0.584	0.584	0.584	0.584	0.584	0.584	0.584
	-13	0.583	0.583	0.583	0.583	0.583	0.583	0.583	0.583	0.583
	-14	0.583	0.583	0.583	0.583	0.583	0.583	0.583	0.583	0.583
	-15	0.519	0.519	0.519	0.519	0.519	0.519	0.519	0.519	0.519
	-16	0.519	0.519	0.519	0.519	0.519	0.519	0.519	0.519	0.519
	-17	0.518	0.518	0.518	0.518	0.518	0.518	0.518	0.518	0.518
	-18	0.518	0.518	0.518	0.518	0.518	0.518	0.518	0.518	0.518
	-19	0.518	0.518	0.518	0.518	0.518	0.518	0.518	0.518	0.518
	-20	0.518	0.130	0.518	0.518	0.518	0.518	0.518	0.518	0.518

Tabela 5.15: Pontuação de SP para c=-5 para o alinhador de perfil global com pontuação logarítmica para cada BLOSUM e para cada uns dos 20 *gop* do alinhador 281b

5.2.3 Pontuação Logarítmica

Inicialmente implementamos o Algoritmo 4 de complexidade de tempo $O(n^3)$ e $O(n^2)$ de espaço para alinhar pares de sequências com penalidade logarítmica de buracos.

Após testes preliminares percebemos que o custo computacional (de tempo) para realização dos alinhamentos múltiplos seria muito grande e portanto, inviável. Diante disso, foi necessário estudar um novo algoritmo com menor complexidade de tempo. O algoritmo escolhido foi o desenvolvido por Miller e Myers [43], que possui complexidade $O(n^2 \log n)$ descrito na Seção 3.2.3.

A Figura 5.6 mostra a comparação do tempo, em segundos, de dois alinhadores que utilizam função logarítimca para penalizar buracos. Um deles usa o algorítmico cúbico para o alinhamento de pares de sequências, o outro usa o algoritmo de complexidade $O(n^2 \log n)$. O gráfico que aparece na Figura 5.6 ilustra o alinhamento de 237 pares de sequências, combinações de pares de sequência dos conjuntos BB11001.tfa, BB12020.tfa, BB20020.tfa e BB30017.tfa do BAliBASE. O eixo X representa a produto do tamanho das sequências de cada par alinhado, e o eixo Y representa o tempo, em segundos, que o alinhador levou para alinhar aquele par. Como podemos notar, o alinhador que utiliza o algoritmo cúbico representa o tempo em uma função curva e bem mais alta que a do algoritmo de Miller e Myers.

Com base no gráfico da Figura 5.6, optamos por utilizar nos testes finais o algoritmo de complexidade $O(n^2 \log n)$.

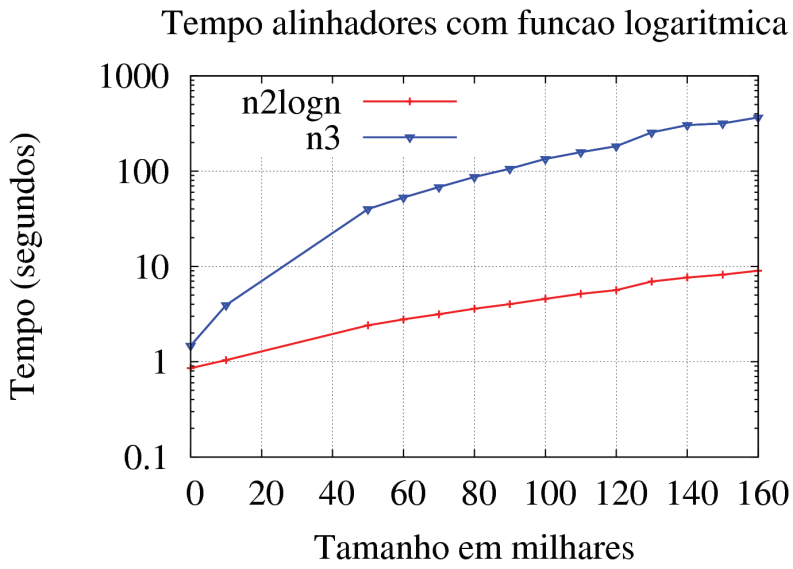


Figura 5.6: Comparação de tempo dos algoritmos de pontuação logarítmica para alinhamento de pares de sequências. No eixo X temos o produto do tamanho das sequências do par (em milhares) e no eixo Y o tempo em segundos que o alinhador utilizou para alinhar os pares de sequências

Algoritmo 12 Pseudocódigo para calcular porcentagem de *core blocks* alinhados

```
Entrada:AlignSequencesCBlocks[], algnSeq[], numberOfcore blocks
Saída: percentage of aligned core blocks
iniBlocks ← []; fimBlocks ← []; alignedTotal ← 0; sizeTotal ← 0
for i ← 1 to numberOfcore blocks do
  sequence ← AlignSequencesCBlocks[1].subsequence(iniBlocks[i],fimBlocks[i])
  sequencePair ← AlignSequencesCBlocks[2].subsequence(iniBlocks[i],fimBlocks[i])
  tam ← 0; k ← 0; found ← False
  while not found do
    fail ← False
    if (tam<alignSeq[1].length) and (sequence[k] ≠ alignSeq[1,tam]) then
      tam ← tam + 1
    else
      ini ← tam
      while (not fail) and (tam<alignSeq[1].length) and(k<sequence.length) do
        if sequence[k] = "-" then
          k ← k + 1
        else
          if alignSeq[1,j] = "-" then
            j ← j + 1
          else
            if sequence[k] = alignSeq[1,j] then
              k ← k + 1; j ← j + 1
            else
              fail ← True, k ← 0
      if not fail then
        found ← True
  k ← 0; ini ← 0
  while  (k<sequence.length) and (ini<j) do
    if sequence[k] = alignSeq[1,ini] then
      if sequencePair[k] = alignSeq[2,ini] then
        alignedTotal ← alignedTotal + 1
      k ← k + 1
    ini ← ini + 1
  sizeTotal ← sizeTotal + fimBlocks[i] - iniBlocks[i]
return (100 × alignedTotal/sizeTotal)
```

Capítulo 6

Avaliação dos Alinhadores

O BAliBASE, apresentado na Seção 5.1, é utilizado como ferramenta de avaliação deste trabalho, tanto nos testes preliminares e ajustes de parâmetros realizados no Capítulo 5, como também nos testes finais dos alinhadores, que apresentaremos neste capítulo.

Apesar de ser possível obter pontuações dos conjuntos do BAliBASE levando em consideração as regiões *core block* em alinhamentos de sequências completas e truncadas ou o alinhamento das sequências em todo o seu comprimento para entradas de sequências completas e truncadas, optamos por analisar apenas os alinhamentos de sequências completas e em regiões *core block*. Esta escolha foi feita, pois as sequências completas representam situações mais realistas e complexas para cada alinhador e as regiões *core block* têm alinhamentos de referência mais confiáveis.

Os alinhadores progressivos, listados na Tabela 5.1, foram testados recebendo como entrada dois subconjuntos de referência daqueles presentes no BAliBASE, que chamamos de RVS1 e RVS2. O conjunto RVS1 possui entradas que contêm os arquivos com menor número de sequências de cada um dos seis conjuntos do BAliBASE. O conjunto RVS2 possui entradas que contêm os arquivos com segundo menor número de sequências de cada um dos seis conjuntos do BAliBASE. A Tabela 6.1 mostra as características das sequências do conjuntos RVS1 e a Tabela 6.2 mostra as características das sequências do conjunto RVS2.

Escolhemos as menores sequências de cada conjunto do BAliBASE devido ao tempo de processamento necessário para executar todos os 342 alinhadores implementados. A Tabela 6.3 mostra o tempo médio, em segundos, de execução dos alinhadores para os conjuntos RVS1 e RVS2 separados para cada um dos métodos de agrupamento utilizados. Esses testes, assim como os realizados para determinar os parâmetros, foram realizados em uma máquina Intel Core 2 Duo de 2.33GHz e 3GB de memória. Podemos observar que os alinhadores que utilizam alinhamento de perfil com função de penalidade logarítmica para buraco levam maior tempo para execução.

Arquivo	**Sequências**	**Min**	**Max**	**Med**	**Mediana**
BB11001	4	83,00	91,00	86,25	85,50
BB12020	4	118,00	129,00	125,50	127,50
BB20020	16	247,00	527,00	288,19	271,50
BB30017	15	231,00	370,00	313,00	331,00
BB40032	6	259,00	1179,00	846,00	960,50
BB50004	9	386,00	505,00	429,56	415,00

Tabela 6.1: Características das sequências do conjunto RVS1. A coluna Arquivo indica o nome do arquivo. A coluna Sequências indica o número de sequências do arquivo. As colunas Min, Max, Med e Mediana indicam os tamanhos mínimo, máximo, médio, assim como a mediana do tamanho das sequências do arquivo.

Arquivo	**Sequências**	**Min**	**Max**	**Med**	**Mediana**
BB11025	4	64,00	103,00	82,75	82,00
BB12021	6	71,00	85,00	75,67	73,00
BB20001	16	74,00	697,00	295,31	212,50
BB30006	18	97,00	923,00	303,39	108,50
BB40010	9	67,00	214,00	109,00	70,00
BB50002	13	172,00	819,00	416,69	400,00

Tabela 6.2: Características das sequências do conjunto RVS2. A coluna Arquivo indica o nome do arquivo. A coluna Sequências indica o número de sequências do arquivo. As colunas Min, Max, Med e Mediana indicam os tamanhos mínimo, máximo, médio, assim como a mediana do tamanho das sequências do arquivo.

Método de Agrupamento	**tempo RVS1**	**tempo RVS2**
AC	187,74	401,58
ACb	220,08	39,11
ACLog	192,33	-
ACLogb	191,45	-
LC	92,18	356,35
AP	232,77	591,70
APb	143,80	152,20
APA	246,65	327,21
APAb	131,74	58,68
APLog	4013,65	-
APLogb	3681,00	-
APAp	129,70	994,80

Tabela 6.3: Tempo médio, em segundos, de execução dos alinhadores para cada sequência dos conjuntos RVS1 e RVS2, separados para cada um dos métodos de agrupamento utilizados

O conjunto total de alinhadores implementados possui 342 alinhadores, destes 147 utilizam função logarítmica para penalidade de buraco nos métodos de agrupamento e foram testados apenas com o conjunto RVS1, devido ao grande consumo de tempo necessário para execução destes alinhadores. Os 195 alinhadores restantes foram avaliados utilizando ambos os conjuntos, RVS1 e RVS2, para os testes. Denominamos o conjunto completo dos alinhadores de Grupo 1, todos estes alinhadores tiveram como entrada o RVS1. Os 195 alinhadores testados com RVS1 e RVS2 serão denominados como Grupo 2, o Grupo 2 está contido no Grupo 1 de alinhadores.

Para cada conjunto de alinhadores, determinado pelo método de agrupamento, utilizamos os parâmetros definidos através dos testes preliminares apresentados no Capítulo 5. Para os alinhadores em que houve o estudo de parâmetros, apresentados na Seção 5.2, utilizamos os parâmetros com os quais os alinhadores alcançaram a melhor pontuação em relação aos casos testados. A Tabela 6.4 apresenta os valores dos parâmetros utilizados em cada tipo de agrupamento.

Do conjunto de alinhadores progressivos que envolvem algoritmos locais, seja para determinar distância ou para agrupar alinhamentos, 34 haviam passado por testes preliminares com os alinhadores locais desenvolvidos inicialmente (aqueles em que era permitido qualquer tamanho de alinhamento como resultado de cada recursão). Comparamos os resultados desses alinhadores, utilizando as pontuações SP e TC, com os resultados dos alinhadores locais equivalentes, que exigem o limite mínimo de 100 bases para o tamanho

Agrupamento	**gap**	**gop**	**gep**	**c**	**matriz**
AC	-	12	3	-	BLOSUM62
ACb	-	-12	-3	-	BLOSUM62
ACLog	-	-8	-	-5	BLOSUM62
ACLogb	-	-8	-	-5	BLOSUM62
LC	-	12	-3	-	BLOSUM62
AP	-5	-	-	-	BLOSUM62
APb	-2	-	-	-	BLOSUM45
APA	-	-17	-1	-	BLOSUM55
APAb	-	-10	-1	-	BLOSUM45
APLog	-	-9	-	-4	BLOSUM75
APLogb	-	-8	-	-5	BLOSUM62
APAp	-	-17	-1	-	BLOSUM55

Tabela 6.4: Parâmetros utlizados nos testes dos alinhadores. A primeira coluna indica o método de agrupamento, a última coluna indica a matriz de substituição e as demais indicam os valores para penalidade para buraco. Quando há "-"indica que o método não usa o parâmetro daquela coluna. A coluna "c"representa a penalidade logarítmica para extensão de buraco

para o alinhamento. Com esta comparação notamos que apesar de em alguns casos as pontuações dos alinhadores que exigem limite terem sido inferiores às dos que não exigem, obtivemos uma melhora média de 7,35% para a pontuação SP e 2,18% para a pontuação TC quando utilizamos os alinhadores que exigem o limite mínimo de 100 bases no alinhamento retornado em cada chamada do alinhamento local recursivo. A Tabela 6.5 mostra as pontuações de SP e TC para os dez piores alinhadores locais antes e depois da inclusão do limite mínimo para o tamanho do alinhamento.

Os alinhadores de perfil com função afim para penalidade de buraco que usaram a melhoria dos parâmetros iniciais foram comparados com os que não utilizaram melhoria. Observando as pontuações SP e TC para os conjuntos de entrada RVS1 e RVS2, a melhora média foi de 1,97% para SP, e 10,34% para TC. Ao observar a pontuação de cada entrada do conjunto RVS1, observamos que a melhora média foi de 0,02% para SP e 2,53% para TC. E ao observar a pontuação para cada entrada do conjunto RVS2 foi possível notar que a melhora média foi de 15,14% para SP e 3,73% para TC. A baixa melhoria pode ser devido a utilização dos determinados nos testes empíricos como valores iniciais de abertura e extensão de buraco.

Na Tabela 6.6 mostramos as pontuações de SP mínima, máxima, média e mediana para cada método utilizado em cada etapa do alinhadores do Grupo 1. Já na Tabela 6.7

Alinhador	SP Antes	TC Antes	SP Depois	TC Depois
160	35,56	18,92	47,24	25,42
159	37,20	22,75	47,66	25,42
154	37,83	22,67	52,85	21,50
147	38,12	22,67	45,39	22,75
148	39,57	21,75	53,06	21,50
153	43,37	23,75	45,50	22,75
162	46,10	23,42	62,71	39,50
150	47,09	24,08	69,83	44,08
156	50,06	25,00	69,83	44,08
221	51,46	28,58	50,15	24,42

Tabela 6.5: Pontuações dos dez piores alinhadores que usam agrupamento local, antes e depois da inclusão do limite mínimo de 100 bases para o tamanho do alinhamento.

apresentamos as mesmas pontuações para o Grupo 2. Escolhemos a pontuação SP pelo fato dela ser mais sensível que a pontuação TC. Por exemplo, na Tabela 6.6, o método JTT que determina a matriz de distâncias tem a pontuações SP $52, 40, 82, 87, 70, 44, 71, 53$, respectivamente, mínima, máxima, média e mediana. As pontuações TC para este método são $17, 86, 65, 17, 47, 96, 48, 10$, respectivamente, mínima, máxima, média e mediana.

Na Tabela 6.6 podemos observar que dos alinhadores de agrupamento por consenso, os que usam ACLog foram os melhores alinhadores. O método *Neighbor Joining* mostrou-se com uma pequena superioriadade em relação ao UPGMA. Os métodos de distância local e de logarítimica apresentaram desempenho inferiores aos demais.

Em ambas as Tabelas, 6.6 e 6.7, podemos notar, que os agrupamentos de perfil e perfil semi-global com pontuação afim são métodos com maior pontuação SP, tanto a média quanto a mediana, e que a utilização da média de distâncias é melhor do que quando a desativamos. Observa-se ainda que os métodos do PHYLIP (JTT, PAM, PCM e PMB) apresentam valores muito próximos tanto para média, quanto para mediana, o mesmo ocorre com os métodos para seleção de pares.

Os alinhadores que alcançaram as vinte melhores pontuações médias de SP e TC quando utilizamos o Grupo 1 de alinhadores para os testes, e também os métodos usados por eles em cada etapa do alinhamento progressivo, são listados na Tabela 6.8. O melhor alinhador foi o 125, ele obteve pontuação média 83,40 para SP e 64,67 para TC, e utilizou PCM para calcular a matriz de distâncias, UP para construção da árvore guia, NP para seleção de pares, APA para o agrupamento e PP para média das distâncias.

Observamos na Tabela 6.8 que os vinte melhores alinhadores utilizam agrupamento de perfil com pontuação afim ou agrupamento de perfil semi-global com pontuação afim.

	Método	Mínimo	Máximo	Média	Mediana
Distância	JTT	52,40	82,87	70,44	71,53
	PAM	49,87	82,13	69,79	70,87
	PCM	52,50	83,40	70,47	70,73
	PMB	52,33	82,63	70,42	71,67
	LD	47,27	76,88	61,26	61,58
	LOGD	38,67	77,35	51,71	52,60
Árvore	NJ	46,63	82,72	65,74	69,55
	UP	38,67	83,40	64,22	66,86
Seleção de Pares	BU	47,02	82,08	63,91	66,19
	NP	38,67	83,40	64,97	67,34
Agrupamento	AC	48,37	75,70	67,76	70,89
	ACb	43,88	58,30	53,27	53,72
	ACLog	58,88	74,50	69,04	72,19
	ACLogb	57,17	72,37	66,71	66,59
	LC	57,18	68,18	51,83	59,30
	AP	44,73	83,40	71,20	70,73
	APb	39,75	57,83	53,17	54,50
	APA	47,43	83,40	72,84	79,90
	APAb	53,65	82,87	73,90	76,56
	APAp	47,43	82,08	72,13	79,19
	APLog	38,67	72,67	62,82	66,20
	APLogb	39,07	75,77	59,25	70,69
Média das Distâncias	PM	38,67	82,63	66,88	69,24
	PP	38,92	83,40	63,77	66,80

Tabela 6.6: Pontuação SP mínima, máxima, média e mediana do conjunto RVS1 para cada um dos métodos utilizados nos alinhadores progressivos, do Grupo 1, implementados

	Método	Mínimo	Máximo	Média	Mediana
Distância	JTT	42,22	67,39	55,72	55,00
	PAM	42,26	66,84	55,82	59,01
	PCM	41,82	67,11	55,72	56,38
	PMB	42,08	66,16	55,50	55,10
	LD	31,26	62,21	44,50	44,41
Árvore	NJ	32,77	65,29	52,66	52,92
	UP	31,26	67,39	53,26	52,85
Seleção de Pares	BU	36,14	67,11	52,00	51,66
	NP	31,26	67,39	52,96	52,92
Agrupamento	AC	48,18	56,73	53,76	54,29
	ACb	41,59	48,81	45,33	45,56
	LC	44,61	52,65	49,37	49,01
	AP	40,62	53,97	50,40	51,76
	APb	31,26	44,78	41,33	42,53
	APA	54,23	65,29	62,95	63,67
	APAb	42,88	67,39	60,70	63,34
	APAp	32,87	64,83	52,04	60,88
Média das Distâncias	PM	31,49	67,07	53,93	53,71
	PP	31,26	67,39	51,93	51,67

Tabela 6.7: Pontuação SP mínima, máxima, média e mediana dos conjuntos RVS1 e RVS2 para cada um dos métodos utilizados nos alinhadores progressivos, do Grupo 2, implementados

Alinhador	**SP**	**TC**	**MD**	**AG**	**SP**	**MA**	**PS**
125	83,40	64,67	PCM	UP	NP	APA	PP
053b	82,87	64,00	JTT	UP	NP	APAb	PP
137	82,72	64,50	PCM	NJ	NP	APA	PP
53	82,67	64,83	JTT	UP	NP	APA	PP
077b	82,63	63,00	PMB	UP	NP	APAb	PP
77	82,43	64,17	PMB	UP	NP	APA	PP
114	82,13	63,50	PAM	NJ	NP	APA	PM
113	82,08	64,17	PAM	NJ	NP	APA	PP
59p	82,08	64,50	JTT	–	BU	APA	PP
83p	82,05	65,50	PMB	–	BU	APA	PP
65	82,03	64,17	JTT	NJ	NP	APA	PP
138	82,03	62,83	PCM	NJ	NP	APA	PM
101b	82,03	62,67	PAM	UP	NP	APAb	PP
89	81,75	63,50	PMB	NJ	NP	APA	PP
66	81,70	63,50	JTT	NJ	NP	APA	PM
60p	81,65	61,67	JTT	–	BU	APA	PM
138p	81,63	65,17	PCM	NJ	NP	APA	PM
137p	81,60	63,00	PCM	NJ	NP	APA	PP
59	81,47	64,33	JTT	–	BU	APA	PP
90	81,42	64,17	PMB	NJ	NP	APA	PM

Tabela 6.8: Pontuação de média de soma de pares, na segunda coluna, e de total de colunas, na terceira coluna, dos vinte melhores alinhadores, do Grupo 1, testados com o conjunto RVS1. MD indica os métodos para o cálculo da matriz de distância, AG indica os métodos para construção da árvore guia, SP indica os métodos para seleção de pares, MA indica os métodos de agrupamento e PS indica o esquema de peso das sequências

Podemos notar que a seleção de pares quando feita pelos pares mais próximos é superior ao Bloco Único, que aparece apenas quatro vezes na tabela. Quando a média das distâncias estava desativada os alinhadores tiveram melhor desempenho. Apesar dos métodos de agrupamento que utilizam alinhamento local terem melhorado seu desempenho com a inclusão do limite a ser utilizado no tamanho do alinhamento, como mostramos na Tabela 6.5, ele não aparece entre em nenhum dos vinte melhores alinhadores.

A Tabela 6.9 mostra os alinhadores que alcançaram as vinte melhores pontuações médias de SP e TC quando utilizamos os conjuntos RVS1 e RVS2 como entrada para o Grupo 2 de alinhadores. A tabela mostra também os métodos utilizados por cada alinhador na geração do alinhamento progressivo. O alinhador 53b foi o melhor entre os

alinhadores. Ele obteve pontuação média 67,39 para SP e 43,42 para TC, e utilizou JTT para calcular a matriz de distâncias, UPGMA para construção da árvore guia, par mais próximo para seleção de pares, alinhamento de perfil semi-global com pontuação afim para penalidade de buraco para o agrupamento e sem uso do esquema de pesos.

As pontuações da Tabela 6.8 mostram que os alinhadores de perfil com pontuação afim, e os de perfil semi-global com pontuação afim foram os melhores em relação ao conjunto de 195 alinhadores. Observe que dos vinte melhores alinhadores listados, oito utilizam o método de Bloco Único para seleção de pares. Observe ainda que dos doze métodos, dentre os vinte, que utilizam NP para selecionar os pares para o agrupamento, oito utilizam UPGMA para construção da árvore guia. Os métodos de distância, com exceção da distância local, apresentam-se bem distribuídos dentre os vinte melhores.

Os vinte alinhadores com pior desempenho quando testamos o Grupo 1 de alinhadores são listados na Tabela 6.10. Observamos que entre os alinhadores listados, mais da metade utiliza o método LOGD, definido na Seção 4.1.2, para calcular a matriz de distâncias. Observamos ainda que dos dezessete métodos que utilizam a árvore guia (pois a seleção de pares por bloco único não utiliza), doze fazem uso do método UPGMA.

Os vinte piores alinhadores, do Grupo 2 de testes, são apresentados na Tabela 6.11. Podemos notar, que quatorze dos vinte alinhadores obtêm a matriz de distâncias pelo método de distância local, confirmando os valores da Tabela 6.7, que mostravam um baixo desempenho deste método.

Os alinhadores listados nas Tabelas 6.8 e 6.9 reforçam o que observamos nas Tabelas 6.6 e 6.7, pois eles utilizam agrupamento de perfil com pontuação afim para penalidade de buraco, seja global ou semi-global. Além disso, os métodos JTT, PAM, PCM e PMB para calcular as distâncias, que possuem pontuações média e mediana próximos, apresentam-se bem distribuídos entre os vinte melhores alinhadores para ambos os conjuntos de tetes.

Os métodos para determinar árvore guia podem ser vistos tanto nas tabelas que apresentam os melhores alinhadores, como nas que apresentam os piores. Além disso eles alcançaram valores de média e mediana muitos próximos nas Tabelas 6.6 e 6.7, e apesar do *Neighbor Joining* parecer ser levemente superior, não é possível afirmar qual dos métodos é realmente melhor.

A semelhança entre as pontuações dos métodos que realizam a seleção de pares, par mais próximo e bloco único, pode estar ocorrendo devido a árvore construída pelos métodos UPGMA e *Neighbor Joining*, pois esta pode não estar agregando informações relevantes para a seleção de pares de acordo com o par mais próximo.

Alinhador	**SP**	**TC**	**MD**	**AG**	**SP**	**MA**	**PS**
53b	67,39	43,42	JTT	UP	NP	APAb	PP
131b	67,11	42,33	PCM	-	BU	APAb	PP
132b	67,07	41,42	PCM	-	BU	APAb	PM
101b	66,84	42,58	PAM	UP	NP	APAb	PP
77b	66,16	42,00	PMB	UP	NP	APAb	PP
83b	65,70	40,08	PMB	-	BU	APAb	PP
60b	65,47	41,25	JTT	-	BU	APAb	PM
66	65,29	42,92	JTT	NJ	NP	APA	PM
138	65,24	42,25	PCM	NJ	NP	APA	PM
90	65,22	42,58	PMB	NJ	NP	APA	PM
84b	65,21	39,5	PMB	-	BU	APAb	PM
125b	65,15	41,08	PCM	UP	NP	APAb	PP
59b	65,14	40,33	JTT	-	BU	APAb	PP
114	65,11	42,67	PAM	NJ	NP	APA	PM
126b	65,10	40,83	PCM	UP	NP	APAb	PM
60p	64,83	43,33	JTT	-	BU	APA	PM
126	64,68	42,25	PCM	UP	NP	APA	PM
125	64,60	40,75	PCM	UP	NP	APA	PP
108b	64,53	38,83	PAM	-	BU	APAb	PM
53	64,33	40,83	JTT	UP	NP	APA	PP

Tabela 6.9: Pontuação média de soma de pares, na segunda coluna, e de total de colunas, na terceira coluna, dos vinte melhores alinhadores, do Grupo 2, testados com o conjunto RVS1 e RVS2. MD indica os métodos para o cálculo da matriz de distância, AG indica os métodos para construção da árvore guia, SP indica os métodos para seleção de pares, MA indica os métodos de agrupamento e PS indica o esquema de peso das sequências

Alinhador	**SP**	**TC**	**MD**	**AG**	**SP**	**MA**	**PS**
160b	49,48	24,00	LD	NJ	NP	APb	PM
159b	49,22	24,17	LD	NJ	NP	APb	PP
331b	48,68	21,33	LOGD	–	BU	APb	PP
321	48,37	7	LOGD	UP	NP	AC	PP
339b	48,37	23,67	LOGD	NJ	NP	APb	PP
148b	47,80	22,33	LD	UP	NP	APb	PM
339	47,63	20,83	LOGD	NJ	NP	AP	PP
325p	47,47	19,67	LOGD	UP	NP	APA	PP
326p	47,43	19,17	LOGD	UP	NP	APA	PM
147b	47,27	21,83	LD	UP	NP	APb	PP
329	47,02	6,83	LOGD	–	BU	AC	PP
329b	47,02	6,83	LOGD	–	BU	ACb	PP
340	46,63	20,17	LOGD	NJ	NP	AP	PM
323	45,87	20,17	LOGD	UP	NP	AP	PP
324	45,43	19,00	LOGD	UP	NP	AP	PM
321b	43,88	7,17	LOGD	UP	NP	ACb	PP
323b	39,75	13,83	LOGD	UP	NP	APb	PP
327b	39,07	26,83	LOGD	UP	NP	APLogb	PP
327	38,92	26	LOGD	UP	NP	APLog	PP
328	38,67	25,83	LOGD	UP	NP	APLog	PM

Tabela 6.10: Pontuação média de soma de pares, na segunda coluna, e de total de colunas, na terceira coluna, dos vinte piores alinhadores, do Grupo 1, testados com o conjunto RVS1. MD indica os métodos para o cálculo da matriz de distância, AG indica os métodos para construção da árvore guia, SP indica os métodos para seleção de pares, MA indica os métodos de agrupamento e PS indica o esquema de peso das sequências

Alinhador	**SP**	**TC**	**MD**	**AG**	**SP**	**MA**	**PS**
82b	42,25	17,58	PMB	–	BU	APb	PM
51b	42,22	16,5	JTT	UP	NP	APb	PP
81b	42,22	17,75	PMB	–	BU	APb	PP
76b	42,08	16,17	PMB	UP	NP	APb	PM
157b	41,99	14,92	LD	NJ	NP	ACb	PP
99b	41,89	16,67	PAM	UP	NP	APb	PP
123b	41,82	16,42	PCM	UP	NP	APb	PP
159	41,73	20,17	LD	NJ	NP	AP	PP
151b	41,59	12,42	LD	–	BU	ACb	PP
160	41,48	20,17	LD	NJ	NP	AP	PM
153	40,68	18,08	LD	–	BU	AP	PP
147	40,62	18,08	LD	UP	NP	AP	PP
153b	39,14	15,64	LD	–	BU	APb	PP
162p	38,13	18,67	LD	NJ	NP	APA	PM
161p	37,98	19,5	LD	NJ	NP	APA	PP
154b	37,32	14,83	LD	–	BU	APb	PM
160b	35,65	14,83	LD	NJ	NP	APb	PM
159b	32,77	14,58	LD	NJ	NP	APb	PP
148b	31,49	13,08	LD	UP	NP	APb	PM
147b	31,26	12,83	LD	UP	NP	APb	PP

Tabela 6.11: Pontuação média de soma de pares, na segunda coluna, e de total de colunas, na terceira coluna, dos vinte piores alinhadores, do Grupo 2, testados com o conjunto RVS1 e RVS2. MD indica os métodos para o cálculo da matriz de distância, AG indica os métodos para construção da árvore guia, SP indica os métodos para seleção de pares, MA indica os métodos de agrupamento e PS indica o esquema de peso das sequências

Capítulo 7

Conclusão e Trabalhos Futuros

Tendo como motivação a relevância do alinhamento múltiplo de sequências na identificação de características conservadas em famílias de proteínas, trabalhamos nesta pesquisa com o objetivo de estudar diferentes métodos que podem ser utilizados na construção de um alinhador progressivo e de analisar as possíveis combinações dentre os métodos estudados.

Os métodos utilizados para o desenvolvimento dos alinhadores progressivos foram apresentados nos Capítulos 3 e 4. No Capítulo 3 foram descritos os métodos mais genéricos para o alinhamento de sequências e as possíveis funções para penalidade de buraco, já no Capítulo 4 foram apresentados os métodos mais direcionados ao alinhamento progressivo.

Foram utilizados seis métodos para determinação da matriz de distâncias, dois para construção da árvore guia e doze para o agrupamento das sequências. Estes doze métodos de agrupamento foram utilizados na última etapa do alinhamento progressivo, que é a construção do alinhamento múltiplo, e foram auxiliados por dois métodos para a seleção dos pares que são alinhados a cada iteração e pelo esquema de peso das sequências através da média das distâncias.

A combinação de todos esses métodos levou a implementação de 342 alinhadores progressivos, que foram avaliados utilizando sequências dos seis conjuntos de referência do BAliBASE.

No Capítulo 5 foram apresentados os 342 alinhadores progressivos implementados e também o estudo realizado para determinar os melhores valores para penalidade de buraco e as melhores matrizes de substituição de aminoácidos a serem utilizados por cada grupo de alinhadores de acordo com os métodos de agrupamento que possuem e com as sequências de entrada. Ainda naquele capítulo foi determinado o melhor valor a ser utilizado como limite mínimo para o tamanho de um alinhamento local recursivo.

Através destes ajustes de parâmetros conseguimos melhorar a pontuação dos nossos alinhadores em relação aos conjuntos de referência do BAliBASE. Pois, em testes realiza-

dos inicialmente os alinhadores com agrupamento por consenso estavam com pontuação superior aos de perfil, que estavam sendo prejudicados pelos parâmetros utilizados.

Estudamos ainda dois algoritmos para determinar a penalidade de buracos através da função logarítmica. Um deles é cúbico em relação ao tamanho das sequências e o outro possui complexidade $O(n^2 \log n)$, onde n é o tamanho das sequências. Observamos que o ganho de tempo de processamento com o algoritmo de complexidade $O(n^2 \log n)$ é bastante satisfatório e, por isso, passamos a utilizá-lo nos alinhadores que fazem uso da função logarítmica para penalizar buracos.

Após a conclusão do estudo para determinar os parâmetros para os alinhadores, iniciamos os testes com dois subconjuntos de sequências do BAliBASE a fim de determinar os melhores alinhadores dentre os 342 implementados. Estes alinhadores foram separados em dois grupos, o primeiro deles envolveu todos os alinhadores e teve como entrada apenas um dos subconjuntos de teste, o RVS1. Já o segundo grupo possuía 195 alinhadores, pois exclui os que utilizam pontuação logarítmica. Este foi testado com ambos os subconjuntos do BAliBASE, RVS1 e RVS2.

Os testes mostraram que os alinhadores que realizaram agrupamento por perfil com pontuação afim e por perfil semi-global com pontuação afim obtiveram melhores desempenhos. Mostraram ainda, para o grupo que contém todos os alinhadores, que o alinhamento de consenso com pontuação logarítmica foi o melhor de todos os alinhadores de consenso. Com os testes, também foi possível notar que, apesar dos métodos que envolvem alinhamento local recursivo terem seu desempenho melhorado com a inclusão do limite mínimo de tamanho para o alinhamento local, eles não foram capazes de superar os alinhadores de perfil. Notamos também que os métodos para do PHYLIP para determinar a matriz de distâncias, os métodos para construção da árvore e os métodos para seleção de pares, apresentaram-se bem equilibrados, e com isso não foi possível afirmar qual deles é realmente o melhor.

As principais contribuições deste trabalho foram:

- Estudo e desenvolvimento de diferentes algoritmos para realização do agrupamento das sequências em um alinhamento progressivo.
- Estudo sobre os parâmetros a serem utilizados pelos alinhadores considerando os métodos de agrupamento.
- Avaliação dos alinhadores progressivos, observando os métodos de cada etapa da construção do alinhamento.

Diferentes trabalhos futuros podem ser desenvolvidos a partir do trabalho realizado nesta dissertação. O principal deles é o uso de estratégias iterativas para refinar os alinhamentos produzidos. Algoritmos iterativos podem ser determinísticos ou estocásticos

e dependem de um outro algoritmo que produza alinhamentos iniciais. Os alinhamentos gerados pelos 342 alinhadores podem ser utilizados como iniciais para o refino do iterativo, principalmente no caso de algoritmos genéticos (que fazem parte dos estocásticos) que precisam de uma grande população para realizar processo de refinamento. Desta forma, os alinhamentos produzidos para cada arquivo de entrada podem ser usados como população para os algoritmos genéticos.

Propomos ainda a adaptação do tamanho do alinhamento recursivo, de acordo com as sequências que serão alinhadas, a cada nível do algoritmo. O limite fixo para a tamanho do alinhamento recursivo pode ter influenciado no seu baixo desempenho.

Seria interessante realizar um estudo em relação à penalidade de alinhamentos de pares de buracos. Além disso, a utilização de técnicas como o *bootstrapping* para melhorar a árvore guia construída.

Determinar quais dos alinhadores são melhores para um determinado conjunto de sequências, de acordo com as características das mesmas, também pode ser citado como um trabalho futuro. Além disso, é importante realizar uma análise estatística dos alinhamentos produzidos como por exemplo a utilizada por Thompson e colaboradores [69].

Apêndice A

Revisão Bibliográfica

Neste apêndice apresentamos uma revisão bibliográfica dos artigos relacionados ao nosso trabalho. Os textos escolhidos envolvem assuntos semelhantes ao do nosso estudo e tem como objetivo proporcionar conhecimento mais abrangente sobre alinhamento de sequências.

A.1 Progressive Sequence Alignment as a Prerequisite to Correct Phylogenetic Trees

Uma das abordagens para o alinhamneto múltiplo é a progressiva, que utiliza alinhamento de pares [48] para construir o MSA. Este trabalho [20] apresenta o método progressivo, que constrói árvores filogenéticas evolucionárias. Em tal abordagem, assume-se que as sequências possuem um ancestral comum, e as árvores são construídas a partir de matrizes de distâncias.

O método utiliza o princípio: "uma vez *gap*, sempre *gap*". E leva em consideração não só a importância da topologia das árvores, mas também o tamanho dos ramos, que deve ser proporcional às distâncias evolucionárias. As sequências são alinhadas progressivamente, iniciando pelos pares mais similares e segue buscando as sequências mais similares.

O alinhamento progressivo apresentado segue os seguintes passos:

- Alinhamento de pares: São possíveis $(n-1)n/2$ alinhamentos. Os autores utilizaram o programa SCORE.

- Identificação dos pares de sequências mais próximas: Usa o programa BORD para ordenar preliminarmente as sequências, o BLEN na comparação das sequências e o PREalign no caso de árvores compostas.

- Inserção progressiva de elementos neutros: Usa o programa DFalign, que insere elementos neutros (Xs) em qualquer *gap* que ocorre em pares alinhados com alta similaridade.

- Pontuação do alinhamento final: Não é necessário usar programa de alinhamento. Utiliza o programa SHUFFLE para calcular o alinhamento randômico de cada sequência (valor usado no cálculo da pontuação).

- Construção da árvore: Usa o programa BORD para obter a ordem dos ramos e o BLEN para determinar seus tamanhos.

Sequências de *Superoxide Dismutase*, enzima catalisadora importante na ação antioxidante das células expostas ao oxigênio, de sete espécies diferentes foram submetidas aos processos de alinhamento progressivo para construção da árvore filogenética e também ao tradicional método de Fitch. O método progressivo apresenta resultados mais próximos da filogenia real dos organismos. Onze diferentes sequências de hemoglobina foram alinhadas usando o método proposto, gerando um alinhamento próximo à realidade histórica. Finalmente aplicou-se o alinhamento progressivo e o simples alinhamento de pares em sequências de nove Tirosinas, as árvores geradas foram bem parecidas diferindo apenas no tamanho dos ramos.

Os experimentos mostraram que o método enquanto heurística proporcionou um alinhamento múltiplo baseado em critérios objetivos e que as árvores geradas aproximam-se bastante a filogenia real dos organismos.

A.2 CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice

O alinhamento múltiplo de sequências de é essencial na biologia molecular. Assim, o desenvolvimento de ferramentas para realizá-lo torna-se importante. Estes auxiliam em atividades como: encontrar padrões que caracterizam famílias de proteínas, demonstrar homologia entre sequências, e predizer estruturas secundárias e terciárias de proteínas.

O alinhamento múltiplo progressivo consiste basicamente de três etapas: cálculo da matriz de distâncias, construção da árvore guia a partir da matriz e alinhamento progressivo das sequências. Possui dois grandes problemas. O do mínimo local, pois faz o alinhamento de maneira gulosa, isto é, nem sempre o melhor para as duas sequências que

está comparando no momento, o que não garante um ótimo global. O outro está na escolha dos parâmetros do alinhamento (matriz de pesos, PAM ou BLOSUM por exemplo, e penalidades para *gaps*).

Neste trabalho [66], os autores descrevem melhorias propostas para a abordagem progressiva de alinhamento múltiplo [20] através do programa CLUSTAL W, que melhora a sensibilidade do alinhador sem perder rapidez e eficiência. É importante mencionar que se as sequências, seja de nucleotídeos ou de aminoácidos, forem muito divergentes (mais 70% - 75% de divergência) a abordagem progressiva perde a confiabilidade.

As melhorias apresentadas no CLUSTAL W são principalmente em relação à escolha dos parâmetros a serem usados no alinhamento. Cada etapa pode ser resumida da seguinte forma:

- Matriz de distâncias: originalmente utilizava-se uma pontuação fixa para cada *gap*. Neste trabalho passaram a usar duas penalidades para *gaps* (uma para abertura e outra para extensão) e uma matriz de pesos de aminoácidos completa. A pontuação do alinhamento é calculada como o número de identidades no melhor alinhamento dividido pelo número de resíduos comparados (pontuação em porcentagem é dividida por 100 e subtraída de 1 para ser colocada na matriz).

- Árvore guia: construída à partir da matriz de distâncias da etapa anterior. Inicialmente é construída uma árvore sem raiz, cujos ramos terão tamanho proporcional a distância estimada. Então, coloca-se a raiz na árvore usando o método do ponto médio. Em seguida, calcula-se o peso das sequências na árvore, que dependem da distância até a raiz, do número de sequências com que compartilham um ramo e do pesos dos ramos (cada sequência tem seu peso mais a proporção do peso de cada ramo, por exemplo, se um ramo tem peso $0,8$ e é compartilhado por duas sequências, cada uma delas têm $0,4$ acrescido no seu peso).

- Alinhamento progressivo: realizado das folhas para raiz da árvore. A cada estágio, um algoritmo de programação dinâmica completo é usado com o resíduo da matriz de pesos e penalidades para abertura e extensão de *gaps*. Para calcular a pontuação de uma posição do alinhamento de uma sequência com outra (ou com alinhamento) utiliza-se a média de todas as comparações necessárias. Para pontuar as sequências, cada valor da matriz de pesos é multiplicado para as sequências. Nesta etapa aplicam-se as melhorias:

  1. Pesos das sequências são normalizados. Sequências mais parecidas têm pesos menores e mais divergentes pesos maiores. Os valores variam entre 0 e 1.
  2. Possui dois tipos de penalidades para *gaps*: GOP (*gap open penalty*), que é o custo para abrir um *gap* de qualquer tamanho e GEP (*gap extension penalty*)

que corresponde ao custo de incluir um item no *gap*. O GOP é calculado da seguinte forma:
$GOP = GOP \times log[min(N, M)] \times (Med) \times (P)$,
onde N,M são os tamanhos das duas sequências, Med é a pontuação média de resíduos de mismatches, P porcentagem de identidade entre as sequências. Já o GEP é calculado da seguinte forma:
$GEP = GEP \times [1.0 + |log(N/M)|]$,
onde N,M são o tamanho das duas sequências.

3. Penalidades para *gaps* podem ser alteradas de acordo com a posição em que aparecem, de forma a penalizar menos o *gap* se já existir algum outro naquela posição do alinhamento. Assim a penalidade para abertura de um *gap* quando existem *gaps* na posição é a seguinte:
$GOP = GOP \times 0.3(ns/n)$
onde ns é o número de sequências sem *gap* e n o número de sequências. A penalidade para abertura de *gap* próximo a *gaps* existentes:
$GOP = GOP \times 2 + [(8 - distanciadogap) \times 2]/8$

   Além disso, a penalidade é reduzida em um terço quando não há *gap* na posição, quando se trata de trechos hidrofilicos.

4. Podem ser usadas as matrizes de distância: PAM , Gonnet e BLOSUM (default).

5. Alinhamento de sequências muito divergentes pode ser adiado até que todas as outras estejam alinhadas.

CLUSTAL W encontrou um alinhamento difícil de ser contestado a olho nu quando testado com sequências que não eram mais que 35% divergentes. Quando testado com sequências mais divergentes produziu pequenas regiões com alinhamentos falsos. Neste ponto pode-se aplicar o refinamento manual, removendo sequências e alinhando as demais automaticamente, ou ainda realizar a mudança de parâmetros.

A.3 A comprehensive comparasion of multiple sequence alignment programs

O alinhamento de proteínas tem sido essencial na biologia molecular, pois é usado em situações como: encontrar motifs e regiões conservadas em famílias de proteínas, predição das estruturas secundária e terciária e ainda análise da evolução filogenética.

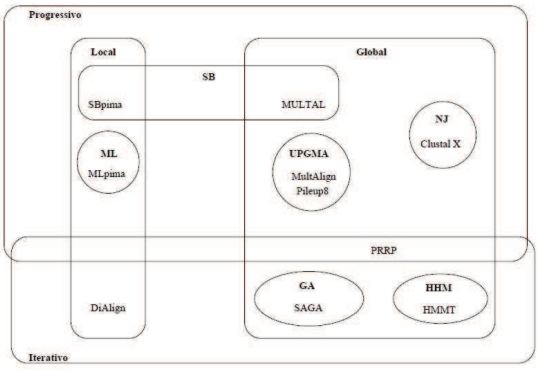


Figura A.1: Alinhadores usados na comparação

Encontrar uma solução computacional exata para resolver o problema é difícil. Desta forma têm sido desenvolvidas heurísticas na tentativa de encontrar boas soluções a custo computacional aceitável. Os alinhadores tentam se aproximar ao máximo de um alinhamento ótimo matemáticamente, que pode não corresponder ao ótimo biológico. O alinhamento progressivo e o alinhamento iterativo são as abordagens mais conhecidas. A figura A.1 mostra um esquema com diferentes programas que implementam tais abordagens de forma local ou global.

Neste trabalho [69] os autores realizaram uma comparação entre os programas da Figura A.1 utilizando como referência os casos de teste do BAliBASE [67, 68]. Neste, os casos de teste são distribuídos em 5 conjuntos e cada um destes é utilizado para avaliar um alinhador em uma situação específica. No primeiro conjunto há poucas sequências com tamanho similar. O segundo conjunto contém alinhamentos de uma família (sequências com mais de 25% de identidade) e três sequências órfãs (menos de 20% de identidade). O terceiro conjunto é composto por alinhamentos de quatro diferentes famílias de proteínas com menos de 25% de identidade entre sequências de diferentes famílias. Os conjuntos 4 e 5 contêm sequências com grandes extensões terminais e grandes inserções, respectivamente.

Duas formas de pontuação foram usadas para comparar os alinhamentos produzidos pelos programas em teste com os alinhamentos do BAliBASE. Para cada teste utilizou-se a pontuação mais coerente. A primeira, soma de pares, é usada para determinar a extensão dos alinhamentos entre pares de sequências em que houve sucesso, seu valor cresce conforme o número de sequências alinhadas corretamente cresce. Esta pontuação é obtida da seguinte forma:

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

onde $S_i = \sum_{j=1}^{N} \sum_{k=1,k\neq j}^{N} p_{ijk}$ e $p_{ijk} = 1$ se o par de resíduos A_{ij} e A_{ik} estiverem alinhados com o alinhamento de referência e 0 caso contrário, M_r é o número de colunas do alinhamento de referência, S_{ri} é a pontuação da coluna S_i do alinhamento de referência.

A segunda forma de pontuação é a de colunas, que pontua de forma binária a habilidade dos programas em alinhar toda uma coluna corretamente. E pode ser calculada como segue:

$$CS = \sum_{i=1}^{i=M} \frac{C_i}{M} \tag{A.2}$$

onde $C_i = 1$ se os resíduos da i-ésima coluna do alinhamento equivalem aos de referência e $C_i = 0$ caso contrário.

No primeiro conjunto de referência observaram-se que regiões ambíguas representam 42% dos resíduos no BAliBASE e somam respectivamente, em média, 32%, 22% e 11% das pontuações *full-length* nas categorias V1 (menos de 25% identidade), V2 (20 a 40% de identidade) e V3 (mais de 35% de identidade). O programa PRRP efetuou o melhor alinhamento da categoria V1 e observou-se que Clustal X tem melhor desempenho quando alinha sequências de comprimrnto médio ou curto.

No segundo conjunto de referência realizaram-se dois tipos de testes. O primeiro realizando o alinhamento das famílias de proteínas nas sequências órfãs e o segundo adicionando-se uma, duas e, por fim, três sequências órfãs. Através da soma de pares observaram que os alinhadores globais foram melhor que os locais, sendo que Clustal X e SAGA foram melhores que o PRRP.

O terceiro conjunto de referência foi avaliado pela pontuação de colunas, esta estima melhor o alinhamento entre famílias. Dos métodos progressivos Clustal X se mostrou o melhor. No entanto foi pior que os iterativos PRRP e SAGA.

No quarto teste, que continha sequências de diferentes tamanhos, os programas SBpima, DiAlign e MLpima (alinhamento local) foram melhores que os demais. Finalmente, com a avaliação do quinto conjunto de referência o DiAlign foi o melhor dos alinhadores. Em seguida vieram os métodos com alinhamento global e por fim SBpima e MLpima.

Com este estudo os autores concluíram a importância em se ter um mecanismo para avaliar e comparar o desempenho dos alinhadores múltiplos, além da necessidade de se utilizar pontuações baseadas em sequências completas e restritas a *core blocks*, aquelas regiões de alta confiabilidade nos alinhamentos de referência.

A.4 Logarithmic *gap* costs decrease alignment accuracy

O alinhamento de sequências é essencial para o estudo da biologia molecular. Neste trabalho [8] o foco de estudo é a qualidade de algoritmos de alinhamento global para pares de sequências.

Algoritmos de alinhamento global se dividem em duas categorias: FSA (*Finite State Automata*) e HMM (*Hidden Markov Models*). Por ser possível realizar a conversão entre as abordagens FSA e HMM, foi utilizada a FSA focando uma implementação de custo mínimo.

A corretude dos alinhamentos depende dos parâmetros utilizados (para *match*, *mistach*, *gap*). Neste trabalho os autores comparam três diferentes funções usadas para determinar o custo de *gaps*: função afim ($G(k) = a + bk$, onde k é o número de *gaps* consecutivos), função logarítmica ($G(k) = a + c \times ln(k)$, onde k é o número de *gaps* consecutivos e c a penalidade logarítmica) e log-afim ($G(k) = a + bk + c \times ln(k)$, onde k é o número de *gaps* consecutivos e c a penalidade logarítmica).

A comparação entre alinhamentos é feita dividindo-os em três categorias: 1) colunas aparecem apenas no primeiro, 2) colunas aparecem apenas no segundo e 3) colunas aparecem em ambos. Através da contagem do número de colunas de cada categoria podemos medir quão idênticos, I, são os alinhamentos.

$$I = \frac{2 * K_3}{2 * K_3 + K_1 + K_2} \tag{A.3}$$

onde K_c é o número de colunas na categoria c.

Os autores destacam que quanto maior a divergência entre as sequências, menor é a corretude do alinhamento. Assim, a corretude esperada para um alinhamento com uma certa função para penalidade de *gaps* depende das sequências utilizadas e também do tamanho dos ramos (medidos em tempo de substituição. Uma unidade é igual a uma substituição, em média, por nucleotídeo).

Cinco mil pares de sequências foram gerados em árvores não enraizadas usando um programa de simulação de sequências. O tamanho médio dos ramos foi $0,2$ e, utilizou-se $match = 1$, $mismatch = 0$ e $a = 16$. Os alinhamentos produzidos foram observados de duas maneiras. Na primeira compara-se diferentes alinhamentos utilizando a mesma função para penalidade de *gap* e na segunda investiga-se a melhoria de cada alinhamento quando otimiza-se o custo de *gap* para cada um deles.

Melhores custos de *gaps* usados nos alinhamentos foram identificados pela alta identidade dos alinhamentos. As funções afim e log-afim tiveram picos de identidade próximos

a 100%, mostrando-se melhores que a logarítmica. Interessantemente o maior ramo melhorou a identidade do alinhamento produzido. Isto pode ter ocorrido devido ao fato de que as sequências que o compõem, embora relacionadas, estão saturadas por *indels* e por isso tenham poucos nucleotídeos homólogos com as outras.

Com este trabalho, os autores mostraram que para os parâmetros usados, e em algoritmos de alinhamento FSA, função log-afim para penalidade de *gap* são superiores a logarítmica e afim, no entanto esta última apresentou resultados semelhantes a log-afim.

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

O BAliBASE é uma ferramenta que possui alinhamentos de referência alta qualidade com sequências protéicas e também conjuntos de referência que podem ser usados para avaliar a performance de um alinhador afim de identificar seus problemas ao alinhar diferentes tipos de entradas.

O uso de *benchmarks* como o BAliBASE é importante pois os alinhadores estão sendo aperfeiçoados para enfrentarem diversos problemas que dificultam o processo, como sequências com erros de sequenciamento e sequências heterogêneas.

No BAliBASE 3.0 [67], os alinhamentos são baseados na superposição de estruturas 3D de proteínas submetidas a refinamento e validação manual. Estes alinhamentos são separados em conjuntos de referência de forma a agrupar aqueles que representem os mesmos problemas reais dos alinhamentos, detalhados na Talela A.1.

Referência	**Característica**
1	Sequências do alinhamento são equidistantes Subdividido em 2 subconjuntos: um com menos de 20% de identidade entre as sequências e outro com identidade entre 20% e 40%
2	Alinhamentos de uma família (sequências com mais de 25% de identidade) e três sequências órfãs (menos de 20% de identidade)
3	Alinhamentos de quatro diferentes famílias de proteínas com menos de 25% de identidade entre sequências de diferentes famílias
4	Sequências com grandes extensões terminais
5	Sequências com grandes inserções internas

Tabela A.1: Conjuntos de referência do BAliBASE

Ainda são fornecidas as regiões em que ocorrem alinhamentos mais confiáveis, *core blocks*, pois são regiões mais conservadas. Os conjuntos de referência desta versão 3.0 são provenientes da versão 2.0 (82 alinhamentos, totalizando 532 sequências) e do SCOP (*Structural Classification of Proteins* [47]). Sendo o grande diferencial do BAliBASE em relação a outros *benchmarks* [16, 44, 74] a alta qualidade de seus alinhamentos devido à avaliação e refinamento manual por especialistas, além da divisão dos conjuntos de testes que facilitam a avaliação de um alinhador em uma certa condição.

A.6 MUSCLE: multiple sequence alignment with high accuracy and high throughput

O alinhamento múltiplo de proteínas é importante em muitas aplicações como estimação da árvore filogenética e predição de estrutura . Este trabalho [16] descreve um alinhador de sequências de proteínas, o MUSCLE, que conta com um algortimo rápido para determinar distância, utilizando *kmer*. Além de uma função logarítmica para pontuação do alinhamento de perfil e refinamento do alinhamento usando uma árvore dependente de partição.

MUSCLE usa duas medidas para determinar a distância entre pares de sequências: *kmer distance* e Kimura. As matrizes de distância dos pares de sequências são clusterizadas usando UPGMA [46] para formar a árvore filogenética. O alinhamento de perfil se dá pela definição de cada par de colunas do alinhamento múltiplo.

O algoritmo se divide em três estágios:

- Esquema progressivo: estágio em que se busca construir um alinhamento múltiplo exato e com rapidez. Constrói-se uma matriz de distância, D1, usando o método de *kmer*. A partir de D1 constrói-se uma árvore binária TREE1 utilizando UPGMA. E a partir de TREE1, percorrida em ordem de prefixo, o alinhamento progressivo (MSA1) é construído pelo sistema de perfil.

- Melhoria do progressivo: A utilização do método de *kmer* para determinar a distância entre pares de sequências pode levar a construção de uma árvore sub ótima. Por isso, uma nova estimativa de distância usando Kimura para determinar a distâncias entre os pares de sequência do MSA1 e formar a matriz D2 a partir da qual o método UPGMA construirá a árvore TREE2. Então, o alinhamento MSA2 é formado considerando TREE2.

- Refinamento: Inicia pela escolha de uma aresta de TREE2 (visitadas em ordem decrescente à distância da raiz), e divide-se a TREE2 em duas subárvores excluindo

a aresta escolhida, e realiza-se o alinhamento de perfil em cada subárvore. Em seguida é produzido um novo alinhamento, este alinha novamente os dois perfis e faz a pontuação de soma de pares para o novo alinhamento, se ela melhora considera-se o novo, caso contrário permanece o antigo. O refinamento é repetido até que haja convergência ou atinja um limite definido pelo usuário.

O desempenho do MUSCLE foi avaliado utilizando quatro conjuntos de alinhamentos de referência: BAliBASE [67, 68], SABmark [74], SMART [55] e PREFAB [16]. Comparando-o com outros quatro métodos de alinhamento CLUSTAL W [66], T-Coffee [51] e dois scripts do MAFFT [34]. Foram utilizadas três métricas de avaliação: Q (qualidade, número de resíduos alinhados corretamente dividido pelo número de resíduos do alinhamento de referência), TC (pontuação total de coluna, número de colunas alinhadas corretamente dividida pelo número total de colunas) e APDB (derivada diretamente das estruturas, sem necessidade do alinhamento de referência).

Em todos os casos de teste e medidas de qualidade o MUSCLE atingiu melhores resultados. Com isso os autores observaram é possível construir um novo algoritmo para alinhamento múltiplo que produz resultados comparáveis ou superiores aos melhores métodos descritos até então.

A.7 Multiple sequence alignment with hierarchical clustering

Desde a década de 70 foram criados diversos programas para realizar o alinhamento de sequências. No entanto, os métodos desenvolvidos até então podem não fornecer resultados adequados quando é necessário alinhar mais que duas sequências. Para suprir essa deficiência foram propostas abordagens que utilizaram as seguintes estratégias: limitar problema a três sequências curtas ou a sequências muito relacionadas, utilizar uma árvore pré-determinada, encontrar subsequências comuns, e selecionar o melhor alinhamento de par diante da pontuação da comparação de todos pares.

Este trabalho [10] foi descrito um algoritmo que usa a formação de *clusterings* de uma forma simples em que não há pré-alinhamento desses *clusterings*.

São realizadas seis etapas:

1. Realização dos alinhamentos de pares e cálculo da pontuação correspondente.
2. Construção de uma hierarquia de clusterização das sequências usando a pontuação calculada.
3. Percorre a árvore alinhando pares de *clusters* até que todos sejam alinhados.

4. Exibição do alinhamento e cálculo da pontuação do mesmo.

5. Construção de uma nova hierarquia de clusterização com as novas pontuações.

6. Retorno à etapa 3 para construção de um novo alinhamento caso haja um novo *clustering* diferente do anterior. Este processo se repete até que o *clustering* das sequências não seja alterado.

O algoritmo para alinhar pares de sequências utiliza a matriz de Dayhoff [12]. Neste algoritmo, dadas duas sequências A e B, i e j elementos das respectivas sequências e $w(i, j)$ o peso que indica $D(A(i), B(j))$. E dado $S(i, j)$ o máximo, para todos os caminhos da célula (i, j), e $M(i, j)$ ser o valor máximo para todas as células da matriz de pesos M. Temos que:

$$S(i,j) = w(i,j) + max(S(i+1, j+1), M(i+1, j+1) - g)$$

$$M(i,j) = max(S(i,j), M(i+1, j), M(i, j+1))$$

Com esta forma de preencher a matriz de pesos, temos a pontuação do alinhamento como a primeira célula da matriz.

Para alinhar os *clusters* segue-se o mesmo raciocínio do alinhamento de pares para obter S e M. No entanto, utiliza-se a pontuação $w(i, j)$, onde i é a posição de um resíduo nas sequências $B_1...B_P$ de um *cluster*, e j é a posição de um resíduo nas sequências $C_1...C_Q$ de um outro cluster. Calculada da seguinte forma:

$w(i,j) = \frac{1}{P.Q} \sum_{R=1}^{R=P} \sum_{S=1}^{S=Q} D(B_R(i), C_S(j))$.
onde D é matriz de substituição.

Os *clusters* são agrupados de maneira hierárquica utilizando a pontuação dos alinhamentos de pares como índice de similaridade entre as sequências. Unem-se a cada iteração dois *clusters* que forem mais próximos.

Apesar de ser difícil provar matematicamente a convergência do processo iterativo, devido à dependência do alinhamento global para o cálculo da distância entre duas sequências, nos testes realizados observou-se que a convergência ocorreu em uma ou duas iterações. O programa descrito pode ser usado tanto no alinhamento de proteínas como no de nucleotídeos, encontrando as sequências mais relacionadas em um dado conjunto.

Referências Bibliográficas

[1] S. Altschul. Amino acid substitution matrices from an information theoretic perspective. *Journal of Molecular Biology*, 219(3):555–565, 1991.

[2] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[3] A. Bairoch. *PC/Gene: a protein and nucleic aminoacid sequence analysis microcomputer package, PROSITE: a dictionary of sites and patterns in proteins and SWISS-PROT: a protein sequence data bank.* PhD thesis, University of Geneva, USA, 1990.

[4] C. Baudet. Uma abordagem para detecção e remoção de artefatos em sequências ESTs. Master's thesis, University of Campinas, Brazil, 2006. In Portuguese.

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

[6] L. Brocchieri and S. Karlin. A symmetric-iterated multiple alignment of protein sequences. *Journal of Molecular Biology*, 276(1):249–264, 1998.

[7] T. A. Brown. *Genomes*. John Wiley and Sons, Inc, 1999.

[8] R. A. Cartwright. Logarithmic gap costs decrease alignment accuracy. *BMC Bioinformatics*, 7:527–538, 2006.

[9] N. Castells-Brooke. Beginner's guide to molecular biology. http://www.rothamsted.bbsrc.ac.uk/notebook/index.php.

[10] F. Corpet. Multiple sequence alignment with hierarchical clustering. *Nucleic Acids Research*, 16(22):10881–10890, 1988.

[11] A. M. da Silva, G. C. Coelho, and E. M . Reis. Proteomica: Uma aborgagem funcional do estudo do genoma. *Saúde e Ambiente*, 2:01–10, 2007.

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

[13] J. Devereux, P. Haeberli, and O. Smithies. GCG package: A comprehensive set of sequence analysis programs for the vax. *Nucleic Acids Research*, 12:387–395, 1984.

[14] L. Duret and S. Abdeddaim. *Bioinformatics: sequence, structure and databanks.* Oxford University Press, 2000.

[15] R. Edgar. MUSCLE: A multiple sequence alignment method with reduced time and space complexity. *Bioinformatics*, 5(1):113, 2004.

[16] R. Edgar. MUSCLE: Multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797, 2004.

[17] R. C. Edgar and K. Sjolander. Simultaneous sequence alignment and tree construction using hidden markov models. *Pacific Symposium on Biocomputing*, 8:180–191, 2003.

[18] M. Farach, S. Kannan, and T. Warnow. A robust model for finding optimal evolutionary trees. *Algorithmica*, 13:155–179, 1995. Special Issue on Computational Biology.

[19] J. Felsenstein. Phylip home page. http://evolution.genetics.washington.edu/phylip.html.

[20] D. Feng and R. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Biology*, 25(4):351–360, 1987.

[21] National Center for Biotecnology Information. http://www.ncbi.nlm.nih.gov, July 2008.

[22] R. Gentleman and R. Ihaka. The R project for statistical computing, July 2009. http://www.r-project.org/.

[23] D. G. George, L. T. Hunt, and W. C. Barker. *Current methods in sequence comparison and analysis in Macromolecular Sequencing and Synthesis.* D. H. Schlesinger, 1980.

[24] G. H. Gonnet, M. A. Cohen, and S. A. Benner. Exhaustive matching of entire protein sequence database. *SCIENCE*, 256(5062):1443–1445, 1992.

[25] O. Gotoh. Optimal alignment between groups of sequences and its application to multiple sequence alignment. *Computer Applications in the Biosciences*, 9(3):361–370, 1993.

[26] Xun Gu and Wen-Hsiung Li. The size distribution of insertions and deletions in human and rodent pseudogenes suggests the logarithmic gap penalty for sequence alignment. *Journal of Molecular Evolution*, 40:464–473, 1995. 10.1007/BF00164032.

[27] S. Henikoff and J. Henikoff. Amino acid substituition matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[28] S. Henikoff, J. Henikoff, and S. Pietrovski. Blocks: A non-redudant database of protein alignment blocks derived from multiple compilations. *Bioinformatics*, 15(6):471–479, 1999.

[29] J. Heringa. Two strategies for sequence comparison: profile-preprocessed and secundary structure-induced multiple alignment. *Computers and Chemistry*, 23(3):341–364, 1999.

[30] P. Hogeweg and B. Hesper. The alignment of sets of sequences and the construction of phyletic trees: An integrated method. *Journal of Molecular Evolution*, 20(2):175–186, 1984.

[31] R.C.G. Holland, T. Down, M. Pocock, A. Prlic, D. Huen, K. James, S. Foisy, A. Dräger, A. Yates, M. Heuer, and M.J. Schreiber. Biojava: an open-source framework for bioinformatics. *Bioinformatics*, 24(18):btn397–2097, August 2008.

[32] H. Jakubowski. Biochemistry online: An approach based on chemical logic, 2010. http://www.biochemweb.org/genes.shtml.

[33] D. Jones, W. Taylor, and J. Thornton. The rapid generation of mutation data matrices from protein sequences. *Computer Applications in the Biosciences*, 8(3):275–282, 1992.

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

[35] J. Kececioglu. The maximum weight trace problem in multiple sequence alignment. *Lecture Notes In Computer Science. Proceedings of the 4th Annual Symposium on Combinatorial Pattern Matching*, 684:106–119, 1993.

[36] M. Kimura. A simple model for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution*, 16(2):111–120, 1980.

[37] M. Kimura. *The Natural Theory of Molecular Evolution*. Cambridge University Press, 1983.

[38] C. Kosiol and N. Goldman. Different versions of the dayhoff rate matrix. *Molecular Biology and Evolution*, 22(2):193–199, 2005.

[39] A. L. Lehninger, Cox, and N. K. Yarborough. *Principles of Biochemistry*. Sarvier, 2004.

[40] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[41] D. Lipman, S. Altschul, and J. Kececioglu. A tool for multiple sequence alignment. *Proceedings of the National Academy of Sciences*, 86(12):4412–4415, 1989.

[42] G. Mendel. Experiments in Plant Hybridization. Mendel's Paper in English.

[43] W. Miller and E. Myers. Sequence comparison with concave weighting functions. *Bulletin of Mathematical Biology*, 50(2):97–120, 1988.

[44] K. Mizuguchi, C. Deane, T. Blundell, and J. Overington. HOMSTRAD: A database of protein structure alignments for homologous families. *Protein Science*, 7(11):2469–2471, 1998.

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

[46] D. Mount. *Bioinformatics Sequence and Genome Analysis*. Cold Spring Harbor Laboratory Press, 2001.

[47] A. Murzim, S. Brenner, and .Hubbard. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995.

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

[49] M. Nei and S. Kumar. *Molecular Evolution and Phylogenetics*. Oxford University Press, 2000.

[50] C. Notredame. Recent evolutions of multiple sequence alignment algorithms. *PLoS Computational Biology*, 3(8):1405–1408, 2002.

[51] C. Notredame, D. Higgins, and J. Heringa. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 302(1):205–217, 2000.

[52] C. Notredame, L. Holm, and D. Higgins. COFFEE: an objective function for multiple sequence alignments. *Bioinformatics*, 14(5):407–422, 1998.

[53] D. Notredame and D. Higgins. SAGA: sequence alignment by genetic algorithm. *Nucleic Acids Research*, 24(8):1515–1524, 1996.

[54] O. O'Sullivan, K. Suhre, C. Abergel, DG. Higgins, and C. Notredame. 3DCoffee: combining protein sequences and structures within multiple sequence alignments. *Journal of Molecular Biology*, 340:385–395, 2004.

[55] C. Ponting. SMART identification and annotation of domains from signalling and extracellular protein sequences. *Nucleic Acids Research*, 27(1):229–232, 1999.

[56] J. A. Quitzau. Um consenso completamente resolvido entre árvores filogenéticas completamente resolvidas. Master's thesis, University of Campinas, Brazil, 2005. In Portuguese.

[57] M. Ronaghi. Pyrosequencing sheds light on DNA sequencing. *Genome Research*, 11:3–11, 2001.

[58] M. Ronaghi, S. Kramohamed, B. Pettersson, M. Uhlen, and P. Nyren. Real-time DNA sequencing using detection of pyrophosphate release. *Analitical Biochemistry*, 242(1):84–89, 1996.

[59] M. Ronaghi, M. Uhlen, and P. Nyren. DNA sequencing: A sequencing method based on real-time pyrophosphate. *Science*, 281:363–365, 1998.

[60] M. Rossignol, J. B. Peltier, A. Matros, A. M. Maldonado, and J. V. Jorrin. Plant proteome analysis: A 2004-2006 update. *Proteomics*, 6:5529–5548, 2006.

[61] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.

[62] F. Sanger, S. Nicklen, and A. R. Coulson. DNA sequencing with chain termination inhibitors. *Proceedings of the National Academy Science, USA*, 74(12):5463–5467, 1977.

[63] J. Setubal and J. Meidanis. *Introduction to computational molecular biology.* PWS Publishing Company, 1997.

[64] T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

[65] A. Sundquist, M. Ronaghi, H. Tang, P. Pevzner, and S. Batzoglou. Whole-genome sequencing and assembly with high-throughput, short-read technologies. *PloS one*, 2(5):e484, 2007.

[66] J. Thompson, D. Higgins, and T. Gibson. CLUSTAL W: improving the sesitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weigh matrix choice. *Nucleic Acids Research*, 22(22):4673–4680, 1994.

[67] J. Thompson, P. Koehl, R. Ripp, and O. Poch. BAliBASE 3.0: latest developments of the multiple alignment benchmark. *Proteins*, 61(1):127–136, 2005.

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

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

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

[71] S. Veerassamy, A. Smith, and E. Tillier. A transition probability model for aminoacid substitutions from blocks. *Journal Computational Biology*, 10(6):997–1010, 2003.

[72] Carlos Vogt. Bioinformática, genes e inovação. http://www.comciencia.br/reportagens/bioinformatica/bio01.shtml.

[73] I. Wallace, O. O'Sullivan, D. Higgins, and C. Notredame. M-Coffee: combining multiple sequence alignment methods with T-Coffee. *Nucleic Acids Research*, 34(6):1692–1699, 2006.

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

[75] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1:337–348, 1994.

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