MARCO TÚLIO NOGUEIRA SILVA

ALINHAMENTO MÚLTIPLO GLOBAL DE SEQÜÊNCIAS PELA REPRESENTAÇÃO DE PROFILE E CLUSTERIZAÇÃO: COMPARAÇÃO COM OS RESULTADOS DO CLUSTALW (EMBL-EBI)

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

LAVRAS
MINAS GERAIS – BRASIL
2006

MARCO TÚLIO NOGUEIRA SILVA

ALINHAMENTO MÚLTIPLO GLOBAL DE SEQÜÊNCIAS PELA REPRESENTAÇÃO DE PROFILE E CLUSTERIZAÇÃO: COMPARAÇÃO COM OS RESULTADOS DO CLUSTALW (EMBL-EBI)

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Área de Concentração:

Biologia, Otimização, Engenharia de Software

Orientador:

Prof. Ricardo Martins Silva de Abreu

Co-Orientador:

Prof. Fortunato Silva de Menezes

LAVRAS
MINAS GERAIS – BRASIL
2006

Ficha Catalográfica preparada pela Divisão de Processos Técnico da Biblioteca Central da UFLA

Silva, Marco Túlio Nogueira

Alinhamento Múltiplo Global de Seqüências pela Representação de Profile e Clusterização: Comparação com os Resultados do ClustalW (EMBL-EBI). Lavras – Minas Gerais, 2006

Monografia de Graduação –Universidade Federal de Lavras. Departamento de Ciência da Computação.

1. BioInformática. 2. Otimização. 3. Biologia. I. SILVA, M. T. N.. II. Universidade Federal de Lavras. III. Título.

MARCO TÚLIO NOGUEIRA SILVA

ALINHAMENTO MÚLTIPLO GLOBAL DE SEQUENCIAS PELA REPRESENTAÇÃO DE PROFILE E CLUSTERIZAÇÃO: COMPARAÇÃO COM OS RESULTADOS DO CLUSTALW (EMBL-EBI)

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em (20 de Setembro de 2006)

_____

Prof. MSc. Deive Ciro de Oliveira

_____

Profª.Dsc. Marluce Rodrigues Pereira

_____

Prof. Dsc.Fortunato Silva de Menezes
(Co-Orientador)

_____

Prof. Dsc.Ricardo Martins Silva de Abreu
(Orientador)

LAVRAS
MINAS GERAIS – BRASIL
2006

Aos meus pais Antônio Claret da Silva(Thuya)

e Maria Lizete Nogueira Silva

E a todos que me apoiaram

Dedico.

Agradecimentos

Agradeço a Deus,
Aos meus pais e minha irmã Natália,
À minha grande amiga Érika,
À PRP e ao CNPq pelo apoio financeiro,
Ao Prof. Ricardo com quem sempre pude contar,
Ao Prof. e amigo Fortunato epor sempre me apoiar e incentivar e
A todos meus amigos em especial ao Adriano e Bruno.

Resumo

O avanço da biologia, sobretudo no contexto de bioinformática, aliado à necessidade de sistemas específicos, tanto para fins de estudo quanto de utilização prática, gerou uma necessidade de elaboração e desenvolvimento de softwares e equipamentos para trabalhar com grande massa de dados. Para isto, há uma necessidade de se obter um conhecimento dos métodos utilizados no alinhamento de seqüências, o que possibilita o desenvolvimento de novos métodos e teorias mesclando com os já existentes na literatura. Neste trabalho estudamos os vários métodos utilizados no alinhamento de seqüências em geral e, verificamos o mais apropriado para o tratamento de alinhamento múltiplo global de seqüências. Implementamos este alinhamento e comparamos o resultado obtido com o fornecido pela ferramenta ClustalW (`http://www.ebi.ac.uk/clustalW`). A implementação em paralela do método usado é necessária para tratar seqüências de escalas práticas.

Abstract

The progress of the biology, mainly in the context of bioinformatics, in addition to requirements of specific systems, with the purpose of studying as well as practical use, raises the development of softwares and hardware to deal with huge amount of data. To this end, it is essential the knowledge of methods used in sequence aligment, what allows the development of new methods and theories mixing with the existent ones. In this work, we study the several methods used in sequence alignment in general and, verify the more appropriate to deal with global multiple sequences alignment. We implement this method and compared the result obtained with the one provided by the ClustalW tool (`http://www.ebi.ac.uk/clustalW`). The parallel implementation of the method used is required to deal with practical sequences.

Sumário

Lista de figuras

Lista de tabelas

1.INTRODUÇÃO

1.1.Contextualização e Motivação

A descoberta do código genético e do fluxo da informação biológica, dos ácidos nucléicos para as proteínas, fez surgir uma nova ciências, a Biologia Molecular. Desde então, inúmeros métodos de seqüênciamento e análise de DNA e proteínas vêm sendo propostos. O objetivo de tais métodos é conhecer a "receita" que a natureza criou e aperfeiçoou durante milhões de anos e que ela segue para criar um ser vivo. Na década de 90, com o surgimento de seqüenciadores automáticos ocorreu uma explosão no número de seqüências a serem armazenadas e analisadas, o que torna indispensável a utilização de ferramentas computacionais cada vez mais eficientes em termos de interpretação e armazenamento dos resultados obtidos. Nascia, assim, a Bioinformática, uma nova área do conhecimento que compreende a intersecção da Biologia, Informática, Matemática e Estatística.

Assim, estudos envolvendo DNA e proteínas, que tem por objetivo predizer características e funções de determinados genes e proteínas em um ser vivo, passam essencialmente por medidas de Homologia entre Seqüências. Em se tratando de alinhamento de múltiplas seqüências, suas aplicações na Biologia envolvem, segundo Napoli (2003) [1], dentre outras aplicações: encontrar relações evolucionárias entre diferentes seres vivos, domínios conservados em DNA ou proteínas, semelhanças de estruturas bidimensionais ou tridimensionais em proteínas, inferências sobre semelhança em termos de funcionalidade entre diferentes seqüências. No entanto, de acordo com Mclure *et.al.* (1994) [2], alinhamentos envolvendo múltiplas seqüências pertencem à classe de problemas NP - completos, problemas não – determinísticos polinomialmente. Para tanto, torna-se necessário a utilização de algoritmos aproximativos ou heurísticas capazes de encontrar um alinhamento próximo de um alinhamento ótimo.

1.2.Fundamentos de Biologia Molecular

Foi intuitivamente que o homem começou a fazer uso da genética a seu favor. Já em 9000 A.C., mesmo sem compreender alguns conceitos, que seriam descobertos mais tarde, o homem selecionou as melhores sementes para o plantio e escolheu os animais mais

vigorosos para a reprodução. Os primeiros filósofos da humanidade já falavam de alguns fenômenos genéticos (sem saber a suas causas) e com o desenvolver da sociedade, pessoas de todas as áreas como médicos, matemáticos, físicos, padres e filósofos, também contribuíram com idéias para o entendimento da hereditariedade. No século XVII, um monge Agustiniano chamado Gregor Mendel deu o primeiro grande passo para desvendar a hereditariedade. Através da análise dos cruzamentos entre ervilhas, Mendel deduziu a presença de fatores hereditários que eram propagados de forma estável de geração a geração, sendo responsáveis pela formação de características individuais. No século XX, ocorreu um considerável progresso nos estudos de biologia celular. Trabalhos como de Frederick Griffth (1928), Avery (1942) e Alfred Hershey e Martha Chase (1952) citados em Napoli (2003) [1] deram uma grande contribuição na associação da hereditariedade ao DNA (ácido desoxirribonucléico). Em 1953, James Watson e Francis Crick, baseados em vários trabalhos da época sobre o DNA, descreveram esta molécula como uma dupla fita, enrolada em hélice ao redor de um eixo, sendo as fitas antiparalelas. O DNA possui uma estrutura periódica que se repete a cada 10 nucleotídeos. As bases nitrogenadas das duas fitas, que estão voltadas para o interior desta, pareiam de forma complementar entre si, na qual a Adenina se liga à Timina e a Citosina à Guanina.

O dogma central define o paradigma da biologia molecular, em que a informação é perpetuada através da replicação do DNA e é traduzida através de dois processos – a transcrição que converte a informação do DNA em uma forma mais acessível (uma fita de RNA complementar) e através da tradução que converte a informação contida no RNA em proteínas, como ilustra a figura 1.1.

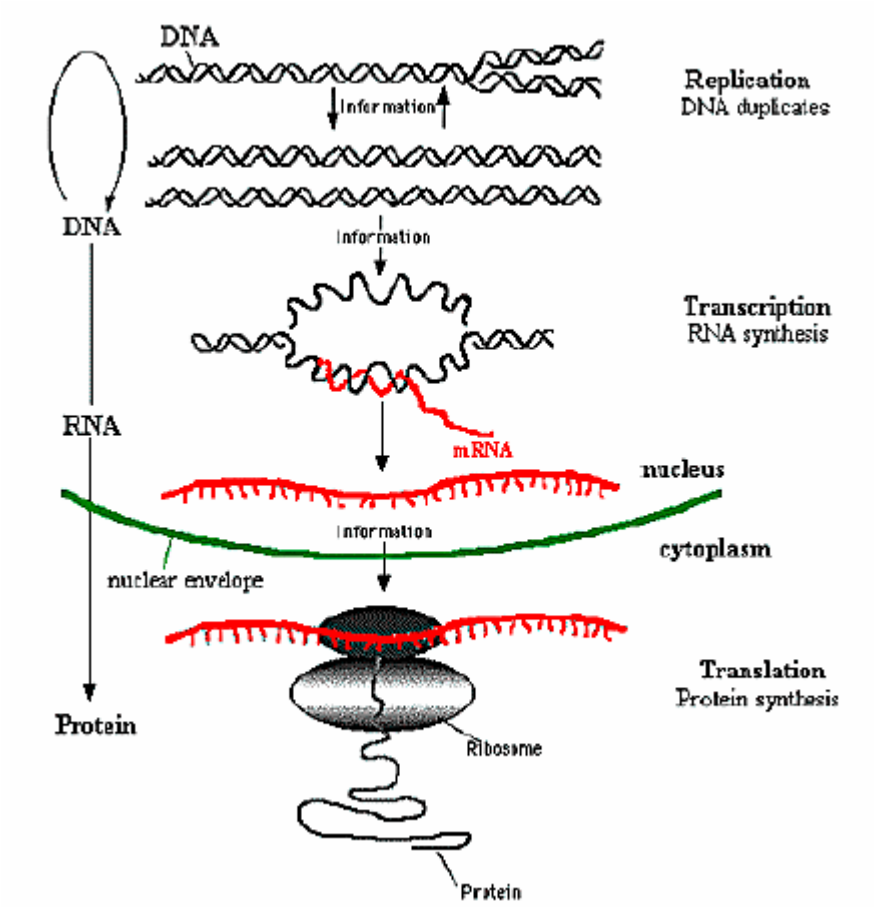


Figura 1.1: O dogma central da biologia Molecular.(Fonte: Bioinformática: do seqüenciamento a função biológica [11])

1.3.Genoma

A seqüência completa de DNA que codifica um ser vivo é chamada genoma. Assim como uma receita é composta de várias instruções , o genoma também é composto de milhares de comandos chamados genes. Cada gene é uma instrução específica, uma seqüência específica de bases nitrogenadas no DNA para a síntese de uma proteína, ou seja, são modelos para gerar proteínas. No caso de um ser humano, portanto, os genes regulam todas as características como altura, quantidade de cabelo, cor dos olhos, distribuição de gordura, etc.

1.4.Mutação

As mutações, modificações na receita de um ser vivo de uma espécie, podem fazer com que ele tenha uma doença ou uma má formação. Mas também pode dar-lhe uma nova característica que, se conferir alguma vantagem de sobrevivência e reprodução sobre seus companheiros, será passado aos seus descendentes. O acúmulo dessas "mutações vantajosas" eventualmente dará origem a indivíduos tão diferentes dos originais que estes constituirão uma nova espécie. Ou seja, a evolução das espécies se dá por meio da seleção natural e da mutação.

1.5.Proteínas

As proteínas são os personagens principais na formação de um ser vivo. Elas dirigem todas as estruturas que compõem as células e algumas proteínas constituem elas mesmas outras partes das células e, logo, do organismo (os cabelos e unhas, por exemplo, consistem basicamente em proteína, a queratina), outras são responsáveis por catalizar as milhões de reações químicas que acontecem em um organismo. As proteínas são polímeros lineares criados a partir de um conjunto de pequenas moléculas denominadas aminoácidos. Segundo Napoli (2003) [1], cada um dos vinte aminoácidos encontrados com mais freqüência nas proteínas tem uma natureza química diferente, determinada por sua cadeia lateral – um grupo químico que varia de aminoácido para aminoácido (Tabela 1.1). A seqüência química da proteína chama–se estrutura primária, mas a maneira pela qual a seqüência se dobra para formar uma molécula compacta é tão importante para a função da proteína quanto sua estrutura primária. Os elementos das estruturas secundária e terciária que compõem a dobra final da proteína podem juntar partes distantes da seqüência química da proteína para formar sítios funcionais.

Tabela 1.1: Listagem dos aminoácidos existentes (Fonte: D.C. Oliveira [3])

Aminoácido	Abreviação com três letras	Abreviação com uma letra
Alalina	Ala	A
Arginina	Arg	R
Asparagina	Asn	N
Aspartato (Ácido Aspártico)	Asp	D
Aspartato ou asparagina	Asx	B
Cisteína	Cis	C
Fenilalanina	Fen	F
Glicina	Gli	G
Glutamato (Ácido Glutâmico)	Glu	E
Glutamato ou glutamina	Gix	Z
Glutamato (glutamida)	Gin	Q
Histidina	His	H
Isoleucina	Ile	I
Leucina	Leu	L
Lisina	Lis	K
Metionina	Met	M
Prolina	Pro	P
Serina	Ser	S
Tirosina	Tir	Y
Treonina	Tre	T
Triptofano	Trp	W
Valina	val	V

2.REFERENCIAL TEÓRICO

2.1.Bioinformática

Do início até meados do século passado os geneticistas e químicos se questionavam sobre a natureza química do material genético. Das pesquisas desenvolvidas, surgiu a conclusão de que o DNA era a molécula que armazenava a informação genética e, em 1953, sua estrutura química foi descoberta no clássico trabalho de Watson e Crick. Com a posterior descoberta do código genético e do fluxo da informação biológica dos ácidos nucléicos para as proteínas, tais polímeros passaram a constituir os principais objetos de estudo da Biologia Molecular. Logo surgiram métodos de seqüenciamento desses polímeros, principalmente do DNA, que permitia a investigação de suas seqüências monoméricas constituintes. Desde então, um grande número dessas seqüências já foram produzidas e então disponíveis em bancos de dados públicos. Com o surgimento dos seqüenciadores automáticos, na década de 90, ocorreu uma explosão nos números de seqüências a serem armazenadas, exigindo recursos computacionais cada vez mais eficientes. Além do armazenamento ocorria, paralelamente, a necessidade de análise desses dados, o que tornava indispensável a utilização de plataformas computacionais eficientes para a interpretação dos dados seqüenciados. Assim nascia a Bioinformática. Essa nova ciência envolveria as seguintes linhas do conhecimento, a Engenharia de Software e a Biologia Molecular. Portanto, a Bioinformática é a aplicação do quadro ferramental das áreas do conhecimento mencionadas acima a problemas associados à Biologia, em especial da Biologia Molecular, pois esta área demanda maior esforço em formalizações teóricas e práticas. Existem vários segmentos dentro da Bioinformática que tratam de problemas específicos. Um exemplo é o tratamento de bases de dados biológicas, que necessitam de conhecimentos em Biologia Molecular (seqüências de DNA e proteínas) além da área de criação e manipulação de Banco de Dados. Outro segmento é o tratamento e realização do seqüênciamento, que demanda grande esforço computacional, principalmente em se tratando de genomas de grandes dimensões. Além disso, tem-se os sistemas de análise de resultados obtidos a partir do processo de seqüenciamento. Dentro da Bioinformática, existe um importante segmento chamado Biologia Computacional. Segundo Oliveira D.C. (2002) [3], este segmento tem

como objetivo propor uma abordagem matemática computacional aos problemas encontrados nos processos biológicos, em especial aos problemas ligados ao seqüenciamento. Nesta abordagem, são definidas questões de otimização, o que implica em economia de esforço computacional.

2.2.Biologia Computacional

Segundo Oliveira D.C. (2002) [3], a Biologia Computacional, em seu significado mais amplo, pode ser entendida como a aplicação de técnicas e quadro ferramental da Ciência da Computação à Biologia. Áreas como a Teoria da Computação e Projeto e Análise de Algoritmos têm notada importância no desenvolvimento de algoritmos para solução de problemas formulados. Além disso, tem-se o campo teórico de Banco de Dados, que trata de bases de dados de grande dimensão.

Ainda segundo Oliveira D.C. (2002) [3], a Biologia Computacional se difere da Bioinformática pois a primeira trata de modelagens e formulações teóricas. Enquanto a segunda está associada à parte prática como desenvolvimento de software, gerência de bases de dados e otimização.

2.3.Programação dinâmica

Segundo Cormem et. al. (2002) [4] programação dinâmica, assim como o método de dividir e conquistar resolve problemas combinando as soluções de subproblemas. Os algoritmos de dividir e conquistar particionam o problema em subproblemas independentes, resolvem os sub-problemas recursivamente e então combinam suas soluções para resolver o problema original. Em contraste, a **programação dinâmica** é aplicável quando os subproblemas não são independentes, isto é, quando os subproblemas compartilham subproblemas. Um algoritmo de dividir e conquistar, nesse contexto, trabalha mais que o algoritmo de programação dinâmica, pois ele calcula soluções para o mesmo subproblema. Já o algoritmo de programação dinâmica não comete esta falha, pois os valores calculados de seus subproblemas são armazenados em uma tabela e evitando assim o trabalho de recalcular a resposta toda vez que um subproblema é encontrado.

A programação dinâmica é em geral aplicada a problemas de otimização. Nesse contexto, na quase totalidade de problemas de alinhamento de seqüência, tanto pareados

quanto múltiplos a programação dinâmica é aplicada, pois o que sempre se desejam neles é encontrar um valor que minimize o custo do alinhamento ou que maximize a similaridade deste alinhamento.

Para se reduzir a complexidade da Programação Dinâmica devido a recursão nela encontrada, pode-se utilizar o método tabular, que consiste na utilização de uma matriz para armazenar os passos já computados na PD (Programação Dinâmica).

2.4.Formalizações

Segundo Chan *et.al.* (1992) [5] *string* é uma seqüência ordenada cujos elementos são símbolos ou letras de um alfabeto e é representada pela concatenação simples destes elementos. Na literatura *strings* e seqüências são sinônimos.

2.4.1.Distância de Edição

Frequentemente deseja-se encontrar uma medida da diferença ou distância entre duas seqüências (por exemplo, evolucionária, estrutural, estudar funções de seqüências biológicas, etc.). Segundo Dun Gusfield (1997) [6] há muitos modos de se formalizar a noção de distância entre duas *string* (seqüências). A mais comum e simples, é chamada de distancia de edição. Este modo foca na transformação (edição) de uma seqüência em outra através de uma série de operações de edição em caracteres individuais. As operações de edição permitidas são: inserção (I) de um caracter da segunda seqüência dentro da primeira seqüência, deleção (D) de um caractere da primeira seqüência, ou substituição (R) de um caractere da primeira seqüência por um caractere da segunda seqüência. A seguir são apresentados exemplos referentes às operações citadas:

1.Operação de inserção de um caractere *x* na seqüência:

S.S= agtc → S = agtcx;

2.Operação de deleção de um caractere *c* na seqüência:

S.S= agtc → S = agt;

3.Operação de substituição de um caractere *x* de seqüência *S* por outro *y*:

S.S= agtc,x = t* e *y = a* → tem-se *S = agac;

Quando ocorre um match, ou seja, os dois caracteres comparados são iguais nos denotamos por M.

R I M D M D M M I
v - i n t n e r -
w r i - t - e r s

Figura 2.1: Distância de edição entre duas *strings*: vintner e writers

A *string* RIMDMDMMI é denominada *transcript* de edição de duas *strings*. Esta *string* se encontra sobre o alfabeto R (substituição), M (match), I (inserção), D (deleção) (Figura 2.1).

Este *transcript* de edição é usado para alinhar as duas seqüências comparadas. Quando for encontrada uma Inserção ou uma Deleção é inserido um *gap* (-) na posição do caractere correspondente que está sendo comparado.

O número mínimo de operações necessárias para transformar uma *string* S em outra *string* T constitui a Distância de Edição.

2.4.1.1.Distância de edição ponderada

Em Dan Gusfield (1997) [6] operações com pesos arbitrários, o problema da operação de distancia de edição ponderada é encontrar um *transcript* de edição que transforma a seqüência S1 na seqüência S2 com o mínimo total de tais operações. Para isso precisamos definir valores para cada operação. Por se tratar de um problema de minimização, devemos dar valores aos matchs (M) menores que os valores dados às operações de Inserção (I) e Deleção (D) e também aos valores de substituição (R). As operações de inserção e deleção têm o mesmo peso, pois pode ser visto que a inserção de um caractere em uma seqüência é o mesmo de se dizer que houve uma deleção de um caractere na outra seqüência.

Como em uma operação de substituição ocorre uma deleção e uma inserção ao mesmo tempo ela deve ser no máximo igual ao peso de duas operações de inserção ou deleção. Vamos denotar um match tendo um peso *e,* uma inserção ou uma deleção tendo um peso *d,* e uma substituição tendo um peso *r*. Então:

$$e < d\ ;$$
$$r \leq 2d;$$

2.4.2.Similaridade

Uma alternativa de se relacionar duas seqüências é através da medida de similaridade. A medida de similaridade procura (como o próprio nome diz) a similaridade entre as duas seqüências comparadas, ao invés de procurar a distância entre as mesmas. Segundo Dan Gusfield (1997) [6] esta aproximação é mais utilizada na maioria das aplicações biológicas por causa de sua flexibilidade. Ainda segundo Dan Gusfield (1997) [6] quando se utiliza a similaridade, a linguagem do alinhamento é usualmente mais conveniente que a linguagem do *transcript* de edição. A similaridade pode ser definida como:

Seja ∑ o alfabeto utilizado para as seqüências S_1 e S_2, e seja ∑' o alfabeto ∑ com o caractere "-" adicionado denotando *gap*. Então para dois caracteres x, y em ∑', s(x, y) denota o valor obtido pelo alinhamento do caractere x com o caractere y.

Este valor s, é obtido após procura feita em uma tabela de similaridade. Então a medida de similaridade é a soma de todos os valores de s obtidos do alinhamento entre as duas seqüências. O tamanho da tabela é de |∑'|x |∑'|.

s	a	b	c	d	_
a	1	-1	-2	0	-1
b		3	-2	-1	0
c			0	-4	-2
d				3	-1
_					0

Figura 2.2: Matriz de similaridade

Utilizando esta tabela (Figura 2.2) o alinhamento de:

c a c - d b d
c a b b d b -

tem o valor total de 0 +1 – 2 + 0 + 3 + 3 – 1 = 4.

2.5.Alinhamento pareado

Segundo Oliveira D.C. (2002) [3], um alinhamento simples é caracterizado pelas seguintes definições:

Definição 2.5.1. Segundo Oliveira D.C. (2002) [3] dado um alfabeto ∑, define-se uma seqüência como uma palavra gerada por este alfabeto*:*

• ∑ = {*a,g,c,t* }

• *S* = *aggctta* é uma seqüência de ∑

Definição 2.5.2. Segundo Oliveira D.C. (2002) [3], dada uma seqüência, define-se um *gap* (_) como um caracter desconhecido da seqüência:

• *S* = *ag_tc*

Definição 2.5.3. Segundo Oliveira D.C. (2002) [3], dados dois resíduos (um par de caracteres) a e b, então *σ* (a,b) define a pontuação de a emparelhado com b e *σ* ( ) é denominado função de pontuação. Existem inúmeras variações em relação à função de pontuação de um alinhamento. Para se obter alinhamentos mais significantes do ponto de vista biológico, são utilizadas matrizes de substituição como função de pontuação. Estas matrizes indicam os diferentes valores a serem contabilizados para cada par de unidade e são normalmente utilizadas em alinhamento de seqüências de proteínas. Assim, o valor de cada uma das células da matriz indica a chance de ocorrência de substituição correspondente ao par de caracteres deste casamento. As matrizes de pontuação mais utilizadas são as pertencentes à família PAM e BLOSUM, ambas descritas em Dan Gusfield (1997) [6]. Uma forma mais simples de propor uma função de pontuação é:

σ(a,a) = 1

σ (b,b) = 1

σ (a,b) = -2

σ (a , _) ou σ (_ , a) = -2

Definição 2.5.4. Segundo Oliveira D.C. (2002) [3], se S é uma *string*, |S| denota o tamanho de S e S[i] denota o i-ésimo caracter de S, onde 1 ≤ i ≤ |S|

Definição 2.5.5. Dados S, T, S' e T' *strings*, um alinhamento A mapeia S e T em S' e T' que podem conter *gaps*. Com isso, são satisfeitas as seguintes conclusões:

1.|S'| = |T'|;

2. A remoção dos *gaps* em S' e T' leva a S e T, respectivamente;

3. A pontuação de um alinhamento é dada por:

$$\sum_{i=1}^{l} \sigma(S'[i], T'[i]), \text{ onde } l = |S'| = |T'|. \qquad (2.5.1)$$

Exemplo 2.5.1. Dadas S = agtc e T = aggt, um alinhamento de S com T, utilizando a função de pontuação σ ( ) proposta acima, é ilustrado na figura 2.3.

a	g	_	_	t	c
a	g	g	t	_	_

Valor = -6

Figura 2.3: Alinhamento de duas seqüências e sua pontuação (score)

2.6.Métodos para Comparação de seqüências

A definição de Comparação de Seqüência é estabelecer parâmetros de semelhança entre duas ou mais seqüências respeitando critérios específicos. Sua aplicação em Biologia Computacional é essencial. A dimensão das seqüências e a complexidade dos algoritmos são em determinadas aplicações fatores limitantes.

2.6.1.Alinhamento de duas seqüências

I. Programação dinâmica aplicada ao alinhamento de duas seqüências:

Segundo Chan et.al [5], o método mais utilizado para comparação de duas seqüências é a programação dinâmica. A utilização da **programação dinâmica** para comparação de duas macromoléculas foi proposta inicialmente por Needleman e Wunsch citado em Chan *et.al.* [5]. Neste método, ele procura a similaridade entre as seqüências utilizando uma função de recorrência.

A programação dinâmica se mostra um método eficaz para solucionar o problema de alinhamento de duas seqüências.

Sejam então duas seqüências S_1 e S_2. D (i, j) é definido como a distancia de S_1[1..i] e S_2 [1 ..j]. Isto é, D (i, j) denota o número mínimo de operações de edição necessárias para transformar os primeiros caracteres i de S_1 nos primeiros caracteres j de S_2. Utilizando esta

notação se S_1 tem n letras e S_2 tem m letras a distância de edição entre elas é precisamente o valor de D (n, m).

Segundo Dan Gusfield (1997) [6] a **programação dinâmica** tem 3 componentes essenciais: **(1) a relação de recursão (recorrência), (2) a computação tabular e (3) o *traceback*.**

1. **Relação de recorrência**

A relação de recorrência estabelece a relação de recursividade entre os valores de D(i, j), para i e j positivos. Quando se trata do menor índice de i ou de j, o valor de D(i, j) deve ser declarado explicitamente. Isto é chamado de condição base para D(i, j).

$$D(i, 0) = i$$

e

$$D(0, j) = j$$

A condição base D(i, 0) = i é fácil de se verificar, pois há somente um modo de se transformar os primeiros i caracteres de S_1 para os zero caracteres de S_2 é deletando todos os i caracteres de S_1 . Similarmente, a condição base D(0, j) = j é correta porque j caracteres devem ser inseridos para converter zero caracteres de S_1 para j caracteres de S_2 .

A relação de recorrência para D(i, j), quando i e j são positivos é dada na equação 2.5.2,

$$D(i, j) = min \begin{cases} D\,(i-1, j) + 1, \\ D(\,i, j-1) + 1, \\ D(\,i-1, j-1) + t(i, j), \end{cases} \qquad (2.5.2)$$

onde t(i, j) é 1 se S_1 (i) ≠ S_2 (j), e t(i, j) é 0 se S_1 (i) = S_2 (j).

Isto, considerando que um match de dois caracteres assume valor 0 e uma inserção, deleção e substituição de um caractere assume valor 1.

Na Figura 2.4 ilustra as três possibilidades de onde podem ser preenchidos um campo da matriz de Programação Dinâmica.

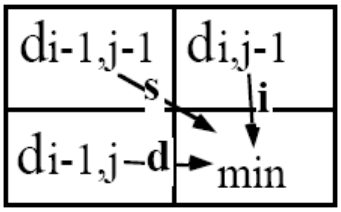


Figura 2.4: Célula (i, j) recebe o mínimo da célula (i -1, j -1) ou da célula (i, j - 1) ou da célula (i – 1, j)

2. Computação tabular

Segundo Dan Gusfield (1997) [6] o segundo componente principal de qualquer programação dinâmica é usar eficientemente a relação de recorrência para computar eficientemente o valor de D(n, m). Isto pode ser feito de maneira mais intuitiva, utilizando-se de uma linguagem de programação, pela criação de uma função recursiva. O algoritmo elementar,de complexidade O(*m***n*), para o cálculo da pontuação de um alinhamento ótimo é mostrado na Figura 2.5.

```
Sim(S1,S2){
%Supondo que m=|S1| e n=|S2|
  for i=0 to m {
    a[i,0]=i * (S[i],-);
  }
  for j=0 to n{
    a[0,j]=j * (-,T[j]);
  }
  for i=1 to m{
    for j=1 to n{
      a[i,j] = max (
        a[i-1,j]   + (S[i],-)
        a[i,j-1]   + (-,T[j])
        a[i-1,j-1] + (S[i], T[j]));
    }
  }
  retorna a[m,n];
}
```

Figura 2.5: Pseudo-codigo do Alinhamento de duas seqüências

Utilizando-se desta idéia, uma tabela de (n +1) x (m + 1) posições será criada. Isto porque, será adicionado aos n caracteres da seqüência S_1 e aos m caracteres da seqüência S_2 um caractere especial, o *gap* (-).

O preenchimento desta tabela será feito pela recursividade utilizando a relação de recorrência. Com esta estratégia, primeiro é computado D(i, j) com o maior (ou menor

valor, caso se esteja interessado na Distância de Edição) valor possível para i e para j, e então é computado o valor de D(i, j) para o incremento de i e j. Estes valores encontrados são armazenados nesta tabela criada (Figura 2.6). Note que a seqüência S_1 corresponde ao eixo vertical da tabela enquanto a seqüência S_2 corresponde ao eixo horizontal da tabela.

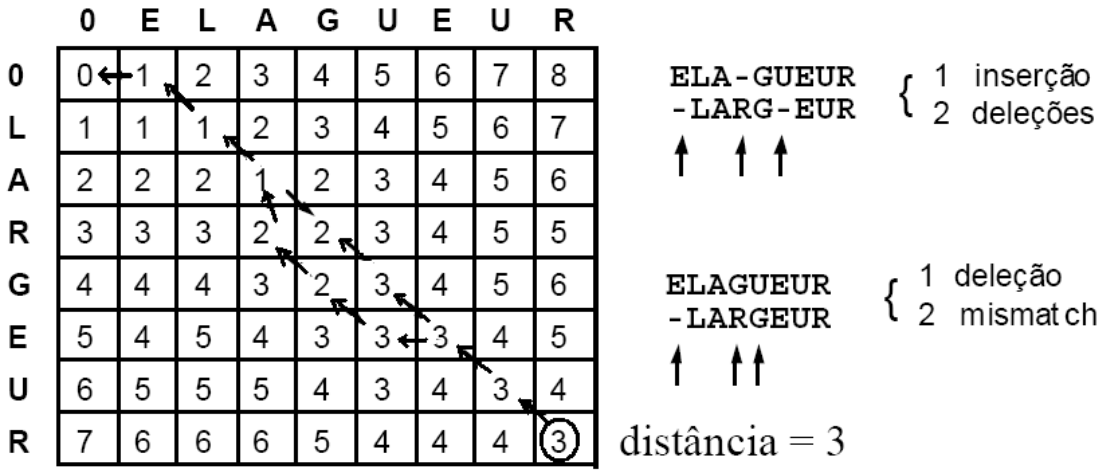


Figura 2.6: Tabela preenchida com dois *transcripts* de edição:
1° DMMIMDMMM e o 2° DMMSSMMM

Utilizando-se das condições bases, pode-se preencher a linha 0 e a coluna 0 desta tabela. Para se entender como preencher o restante das posições, note que pela relação de recorrência geral de D(i, j) todos os valores necessários para a computação de D(1, 1) são conhecidos uma vez que D(0, 0), D(1, 0) e D(0, 1) já foram computados. Então D(1, 1) só pode ser computado depois que a linha 0 e a coluna 0 já tiverem sido computadas. Novamente a relação de recorrência é chamada (recursividade), depois que D(1, 1) foi computado, todos os valores necessários para se computar D(1, 2) estão disponíveis. Seguindo esta idéia, todos os valores até a posição D(n, m) serão computados, e então a distância de edição ótima deste alinhamento será conhecida, na posição D(n, m).

A modificação deste algoritmo de programação dinâmica para distância de edição é facilmente alterado para similaridade, modificando os pesos dos matchs e *mismatchs* (Figura 2.7).

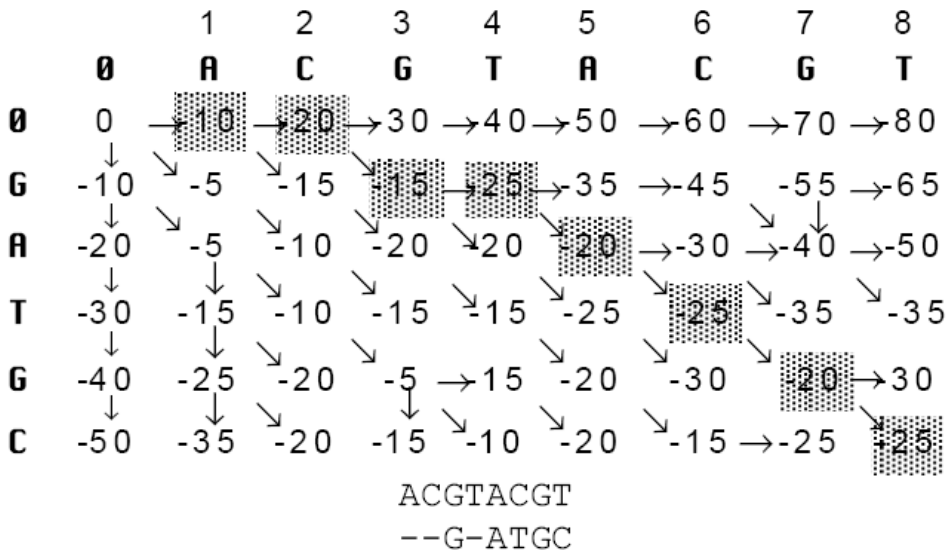


Figura 2.7: Tabela de similaridade matchs valor 5 e *mismatchs* com valor -5

3. *Traceback*

Segundo Dan Gusfield (1997) [6], o modo mais fácil de se obter o *transcript* de edição é estabelecer ponteiros na tabela assim que os valores são computados.

Em particular quando um valor na célula (i, j) é computado, um ponteiro da célula (i, j) é setado para:

- a célula (i, j - 1) se D(i, j) = D (i, j - 1) +1;
- a célula (i - 1, j) se D(i, j) = D (i - 1, j) +1;
- a célula (i -1, j - 1) se D(i, j) = D (i - 1, j - 1) + t(i, j).

Com estes ponteiros setados em cada célula é facilmente encontrado o transcrpit de edição ótimo percorrendo a tabela da posição (n, m) para a célula (0, 0), e qualquer caminho de (n, m) para (0, 0), seguindo os ponteiros estabelecidos durante a computação de D(i, j) especifica um *transcript* de edição com um número mínimo de operações de edição. O trancript de edição é recuperado do caminho interpretando cada seta horizontal da célula (i, j) para a célula (i, j - 1) como uma inserção (I) do caractere $S_2(j)$ em S_1; interpretando cada seta vertical da célula (i, j) para a célula (i - 1, j) como uma deleção (D) do caractere $S_1(i)$ de S_1 e interpretando cada seta diagonal da célula (i, j) para a célula (i - 1, j - 1) como um match (M) se $S_1(i) = S_2(j)$ e como uma substituição (R) se $S_1(i) \neq S_2(j)$. Este caminho especifica o *transcript* de edição ótimo.

Em termos de alinhamento de S_1 e S_2, cada seta horizontal no caminho especifica um espaço inserido em S_1, e cada seta vertical no caminho especifica um espaço inserido em S_2 e por fim, cada seta diagonal no caminho especifica um match ou um *mismatch*, dependendo do caracter específico (Figura 2.8).

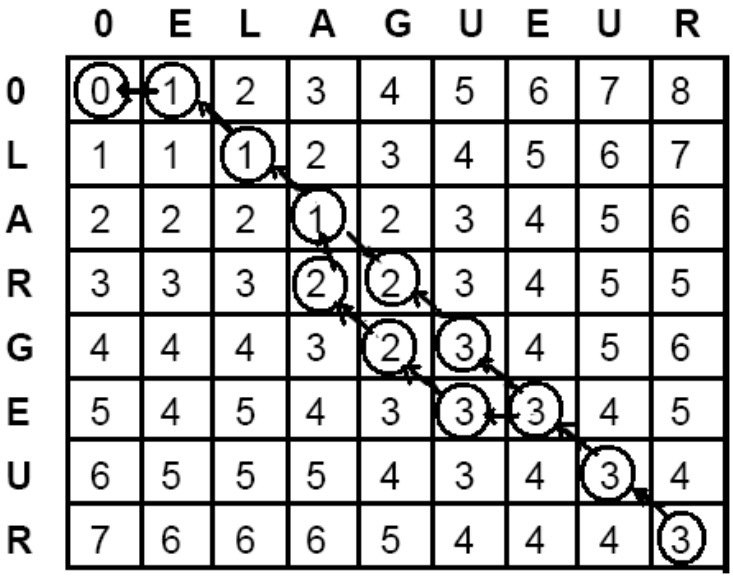


Figura 2.8: *Traceback*

II. Método Dot Plots

Dot Plots é uma representação visual das similaridades entre duas seqüências. Sempre que há uma similaridade entre um caracter de uma seqüência e um caracter de outra seqüência um ponto é inserido na posição referente a esses caracteres.

Segundo [7] existem essencialmente duas formas de realizar dot-plots:

1. **A forma exata**: as seqüências a serem comparadas são arrumadas ao longo de uma matriz. A cada célula (i, j) da matriz onde ocorre um match, ou que i e j se assemelham segundo algum critério (do ponto de vista da matriz de scores escolhida). Uma seqüência diagonal de pontos indica uma região onde as seqüências analisadas são semelhantes. Mas este gráfico é útil quando aplicada a seqüências de proteínas, porque seu alfabeto possui 24 caracteres. No caso das seqüências de DNA e RNA este método não é utilizado porque o gráfico gerado ficaria demasiadamente carregado, gerando um resultado não muito significante a

análise. Alguns autores sugerem a utilização de filtros para reduzir o ruído provocado por matchs aleatórios (Maizel and Lenk 1981 em [7]). Muitos filtros são possíveis, mas o mais comum consiste em colocar um ponto na célula (i, j) se uma janela de 10 bases centrada em (i, j) contêm mais de 6 matches positivos. Independentemente do filtro, este método, como na programação dinâmica requer a construção de uma matriz m x n, e portanto cresce com o produto do comprimento das seqüências ($O(N^2)$). Isto acarreta um peso computacional conseqüente.

2. **Blocos de identidade**. Este método envolve "hashing" e em vez de ter em conta a matriz completa e calcular os pontos para cada célula da matriz, pode-se poupar um tempo computacional considerável se procurar-se apenas por matches exatos de certo comprimento. Este método procura unicamente blocos de identidade (semelhança) perfeita. A complexidade deste algoritmo cresce linearmente com N. O algoritmo simplesmente subdivide as duas seqüências em "palavras" de comprimento pré-determinado. Para cada seqüência a localização de cada palavra é registrada. Estes vetores de "palavras" são então ordenados em paralelo com as palavras. Então, por comparação do vetor ordenado de uma seqüência com a da outra, obtêm-se automaticamente as localizações de todas as "palavras" idênticas. As heurísticas de alinhamento na base de Fast e Blast utilizam este método para selecionar as regiões de alinhamento mais promissoras.

2.6.2.Alinhamento múltiplo de seqüências

Segundo Dan Gusfield (1997) [6], no contexto da biologia molecular, comparação de seqüências múltiplas (DNA, RNA ou proteínas) é muito mais que um exercício técnico. É a ferramenta mais eficiente para extrair e representar importância biológica.

Ainda segundo Dan Gusfield (1997) [6] um alinhamento múltiplo de k > 2 seqüências $S=\{S_1, S_2, ..., S_k\}$ é uma generalização do alinhamento para duas seqüências. Espaços escolhidos são inseridos dentro de cada k seqüências de forma que as seqüências resultantes tenham o mesmo tamanho (l). Então as seqüências são formadas por k linhas e l colunas cada, de forma que cada caractere e espaço de cada seqüência fiquem em uma única coluna.

A principal importância do alinhamento de seqüências múltiplo segundo Dan Gusfield (1997) [6] é para solucionar **dois problemas** que podem ocorrer quando se analisa as seqüências duas a duas. O primeiro problema que pode ocorrer é quando as duas seqüências de aminoácidos, por exemplo, estão tão pouco conservadas ou eles podem estar altamente dispersos que o melhor alinhamento entre estas duas seqüências relacionadas é estatisticamente indistinguível do melhor alinhamento de duas seqüências de aminoácidos randômicas. O segundo problema é o oposto do primeiro. A comparação de duas seqüências de espécies altamente relacionadas poderia não revelar importância biologicamente de padrões conservados, porque as similaridades críticas são perdidas no grande numero de similaridades. Então quando se faz a comparação de duas seqüências para encontrar padrões críticos em comum, o desafio é escolher espécies que o nível de divergência seja mais informativo.

O alinhamento de seqüências múltiplas é a resposta para estes problemas. Utilizando este método, não é crucial escolher espécies que o nível de divergência é mais informativo. Freqüentemente, padrões biologicamente importantes que não podem ser revelados pela comparação de duas seqüências sozinhas se torna claro quando muitas seqüências relacionadas são simultaneamente comparadas. Além disso, com o alinhamento múltiplo, é possível algumas vezes organizar um conjunto de seqüências relacionadas (freqüentemente em uma árvore) para demonstrar mudanças contínuas ao longo do caminho conectando duas seqüências extremas, onde estas mesmas seqüências mostram pouca similaridade pareada. Em geral, alinhamento de seqüências deste tipo é útil para deduzir a história evolucionária.

Alinhamento Pareado Induzido: dado um alinhamento *M*, o alinhamento pareado induzido de duas seqüências S_i e S_j é obtido de *M* removendo todas as linhas, exceto as linhas S_i e S_j**.** Isto é, o alinhamento pareado induzido é o alinhamento múltiplo de *M* restrito para S_i e S_j .Quaisquer dois espaços neste alinhamento induzido pode ser removido se desejado. Então, o score de um alinhamento pareado induzido é determinado usando algum esquema de pontuação escolhido para alinhar duas seqüências de maneira padrão (Figura 2.9).

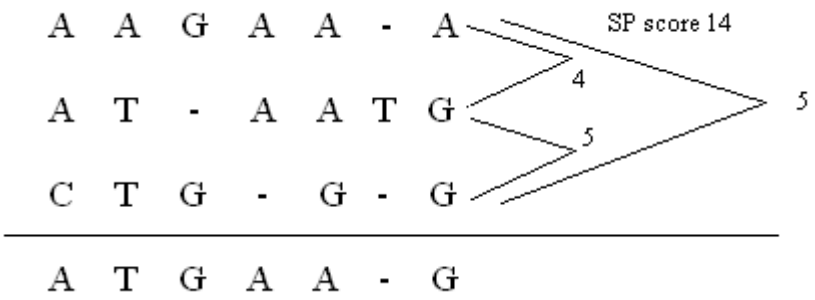


Figura 2.9: Alinhamento múltiplo M de três seqüências mostradas acima. Utilizando o esquema de pontuação pareada, os três alinhamentos pareados induzidos têm scores de 4, 5, 5 para um total de SP score de 14. Note que ocorre match entre *gaps*. A seqüência consenso é mostrada abaixo da linha horizontal (Fonte Dun Gusfield [6]).

Seqüência de consenso: dado um alinhamento múltiplo *M* de um conjunto de seqüências S, o caractere de consenso da coluna i de *M* é o caractere que minimiza a distância somada dele com os outros caracteres da coluna i. Seja então d(i) a soma mínima na coluna i. A seqüência de consenso S_M então é a concatenação dos caracteres de consenso para cada coluna de *M*.

Pode-se generalizar a seqüência de consenso S_M, como sendo a seqüência que melhor representa o conjunto S de seqüências.

Alinhamento consistente: seja S um conjunto de seqüências, e T uma árvore onde cada nó é rotulado com uma seqüência distinta de S. Então, um alinhamento múltiplo consistente *M* de S é chamado consistente com T se o alinhamento pareado induzido de S_i e S_j têm pontuação $D(S_i, S_j)$, para cada par de seqüências (S_i, S_j), que são nos adjacentes em T. Por exemplo, veja Figura 2.10:

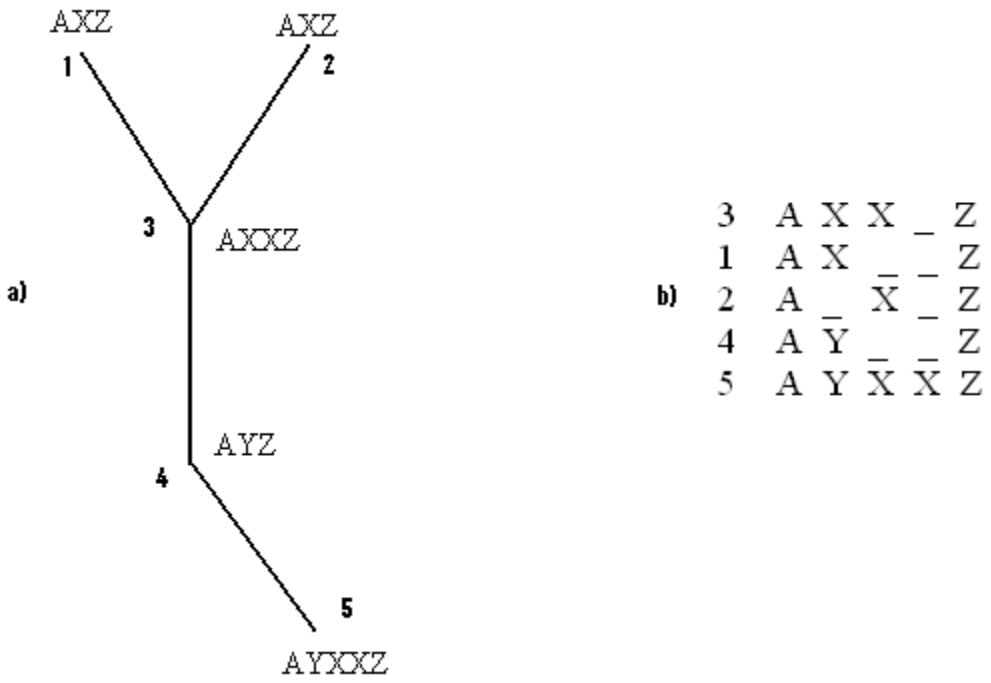


Figura 2.10: a)A árvore com nós rotulados, b)Um alinhamento múltiplo que é consistente com a árvore (Fonte: Dun Gusfield [6])

Alinhamento Múltiplo Via *Profile*

Definição: segundo Dan Gusfield (1997) [6], dado um alinhamento múltiplo de um conjunto de seqüências, um *profile* para este alinhamento especifica para cada coluna a freqüência que cada caractere aparece nesta coluna. Um *profile* algumas vezes é chamado também de matriz de peso na literatura biológica.

Alinhando Seqüência e *Profile*

Dado um *profile* P e uma nova seqüência S, a tarefa é saber quão bem S ou alguma subseqüência de S ajusta o *profile* P. Como o *gap* também é um caractere legal em P, um ajuste de S em P poderia também permitir a inserção de *gaps* dentro de S, então o alinhamento de S em P é facilmente generalizado como um simples alinhamento de seqüências.

2.6.2.1.Métodos exaustivos

Os métodos exaustivos de comparação de seqüências múltipla garantem um alinhamento ótimo. Alguns deles podem também utilizar bordas heurísticas para limitar a procura por alinhamentos ótimos. Dentre estes métodos destaca-se a Programação Dinâmica.

1) Programação dinâmica

Esta aproximação utiliza uma matriz de N dimensões para alinhar N seqüências. Como visto anteriormente este método desprende muito tempo de processamento. Mas para minimizar este problema, Frickett citado em [5] propôs um algoritmo para procurar por um alinhamento ótimo de duas *strings* limitando a busca em uma faixa da matriz apenas. Esta faixa de procura se concentra próxima à diagonal, pois é onde se localiza a maior significância do alinhamento.

Utilizando esta idéia de Frickett, e a medida de soma de pares Carrillo e Lipman citado em Chan *et.al.* (1992) [5] propuseram uma estratégia para se alinhar N seqüências. É então obtido um limite superior para o alinhamento em cada par de seqüências. O limite superior para o custo do alinhamento das duas seqüências, X e Y, é definida em uma região do plano bi-dimensional definido (X, Y). Dentro desta região está a projeção do caminho ótimo das N seqüências sobre (X, Y). A procura pelo caminho ótimo das N seqüências é então limitada em certa região da matriz de N dimensões, reduzindo com isso seu tempo de busca.

I. Programação dinâmica com Soma de Pares (SP)

Segundo Dan Gusfield (1997) [6], o problema de soma de pares, pode ser resolvido de maneira ótima via programação dinâmica. Mas, se há k seqüências e cada uma delas têm tamanho n, a programação dinâmica necessita de tempo θ (n^k), e então este método é praticável para somente um número pequeno de seqüências. Além disso foi provado que este problema, está na classe de problemas NP-Completo.

Definição: sejam S_1, S_2 e S_3 três seqüências de tamanho n_1, n_2 e n_3 respectivamente e seja também D(i, j, k) o alinhamento ótimo de SP para alinhar S_1 [1..i], S_2 [1..j] e S_3 [1..k]. O score para match, *mismatch* ou espaço é especificado pelas variáveis *smatch, smis,* e *sspace*, respectivamente.

A tabela D utilizada pela programação dinâmica forma um cubo (três dimensões). Cada célula (i, j, k), que não fazem parte das bordas da tabela tem sete vizinhos que devem ser consultados para determinar o valor da célula (i, j, k). A função de recorrência está codificada no seguinte pseudocódigo (Figura 2.11):

```
1   for i := 1 to n1 do
2       for j := 1 to n2 do
3           for k := 1 to n3 do
4               begin
5               if (S1(i) = S2(j)) then cij := smatch
6               else cij := smis;
7               if (S1(i) = S3(k)) then cik := smatch
8               else cik := smis;
9               if (S2(j) = S3(k)) then cjk := smatch
10              else cjk := smis;
11
12              d1 := D(i – 1, j –1, k – 1) + cij + cik + cjk ;
13              d2 := D(i – 1, j –1, k) + cij + 2*sspace;
14              d3 := D(i – 1, j, k – 1) + cik + 2*sspace;
15              d4 := D(i, j –1, k – 1) +  cjk + 2*sspace;
16              d5 := D(i – 1, j, k) + 2*sspace;
17              d6 := D(i, j –1, k) + 2*sspace;
18              d7 := D(i , j, k – 1) + 2*sspace;
```

Figura 2.11: Recorrência para células que não fazem parte da borda da tabela

Resta ainda computar os valores de D quando as células estão na borda em uma das faces da tabela, isto é quando i = 0, ou j = 0 ou k = 0. Para isto considere $D_{12}(i, j)$ denotar a família das distâncias pareadas entre as subseqüências S_1 [1 .. i] e S_2 [1 .. j], e seja $D_{13}(i, k)$ e $D_{23}(j, k)$ as distâncias pareadas análogas envolvendo os pares de seqüências S_1, S_3 e S_2, S_3. Então:

$$D(i, j, 0) = D_{12}(i, j) + (i, j)*sspace,$$
$$D(i, 0, k) = D_{13}(i, k) + (i, k)*sspace,$$
$$D(0, j, k) = D_{23}(j, k) + (j, k)*sspace,$$

e

$$D(0, 0, 0) = 0.$$

II. Aproximação por Árvore

A árvore descreve as relações existentes entre um conjunto de seqüências de entrada. Uma aproximação do problema do alinhamento por árvore é construir uma seqüência de ancestral (nós internos da árvore) e então alinhar as seqüências de entrada da árvore com estes nós internos respeitando as relações de incidência da árvore (topologia).

2.6.2.2 - Métodos heurísticos

Segundo Chan *et.al.* (1992) [5], métodos heurísticos tentam encontrar em tempo razoável, alinhamentos bons que não são necessariamente ótimos. Estes métodos heurísticos podem se classificar dentro de cinco aproximações:

1. Subseqüências
2. Árvores
3. Seqüência de consenso
4. Clusterização
5. Template

1. **Subseqüências:** uma subseqüência refere-se a um segmento, uma região. Existem quatro métodos diferentes de aproximação por subseqüências.
    a. Jonhson e Doolitle citado em Chan *et.al.* (1992) [5], propuseram um método para alinhar três ou mais seqüências pela comparação de segmentos selecionados das seqüências. Começando do termino aminoácido da seqüência, uma janela é estabelecida para limitar o número de comparações de segmento. Isto é usado para alinhar um resíduo de cada segmento. A janela é movida à frente da posição corrente de forma que novos segmentos são comparados e os seguintes resíduos são alinhados. O processo continua ao longo da seqüência até o carbono termini for alcançado.
    b. Região: R é representada por uma tripla (w, i, j) indicando que há um match da palavra w cujo qual começa na posição i de uma seqüência e na posição j em outra seqüência. Para obter a lista de regiões de match para a comparação de duas seqüências, pode-se concatenar as duas seqüências

dentro de uma seqüência simples S, e então ordenar S para repetidas palavras. Isto pode ser feito pelo conceito de hashing.

c. Ao invés de procurar por repetições exatamente iguais em todas as seqüências como no método de regiões, Waterman citado em Chan *et.al.* (1992) [5] propõe encontrar o consenso padrão, que ocorre imperfeitamente em uma freqüência prefixa. Um consenso padrão é uma palavra k (k>1) que é comum em pelo menos uma porcentagem prefixa (β) de seqüências. Com as seqüências organizadas em linhas, ocorre à procura da palavra mais freqüente no bloco de j a j + W – 1, onde j = 1, 2,..., e W é o tamanho da janela.

d. Bacon e Anderson citado em Chan *et.al.* (1992) [5] propôs um algoritmo baseado no alinhamento de segmentos onde a significância da pontuação do alinhamento é julgada usando diferentes modelos estatísticos. A similaridade entre um par de segmentos é definida como a soma da similaridade dos resíduos individuais correspondendo às posições relativas no início de cada segmento.

2. **Árvores:** Segundo Chan *et.al.* (1992) [5], árvore é um gráfico acíclico em que as folhas representam um conjunto de amostras e relações de incidências do grafo representam as taxonômicas ou filogenéticas entre as amostras. A busca por um alinhamento ótimo utilizando uma árvore ainda requer um tempo exponencial, por isso várias heurísticas têm sido criadas para obter bons alinhamentos em tempo razoável. Podemos descrever três diferentes métodos.

   a. Sankoff e Cedergren citado em Chan *et.al.* (1992) [5], diz que dado um método heurístico para alinhar N seqüências que estão relacionados por uma representação de uma árvore filogenética da história evolucionária destas seqüências. Na árvore, os N nós terminais correspondem ao conjunto de seqüências de entrada enquanto os M nós interiores correspondem às seqüências mais antigas que são inferidas pelos nós descendestes delas. Então cada nó interior representa um ponto evolucionário onde as seqüências se derivaram em pelo menos duas seqüências descendentes diferentes.

b. Segundo Chan *et.al.* (1992) [5], se as seqüências são relacionadas por uma árvore binária, sua complexidade de tempo é reduzida. O método começa com a construção de uma seqüência "média" de duas seqüências originais (seqüências de entrada) que estão relacionadas por nós da árvore. É nomeado um peso baseado no número de sucessões originais das quais uma sucessão construída é derivada. O processo de construção da seqüência continua seguindo as relações de incidência da árvore subindo para a raiz onde a seqüência média final é derivada. O alinhamento global é obtido pelo alinhamento de cada seqüência original com esta seqüência média final.

c. Hein citado em [5], introduziu o conceito de grafo de seqüências para alinhamento múltiplo de seqüências e reconstrução de seqüências ancestrais quando é dada a filogenia na seqüência de entrada.

3. **Seqüência de consenso:** segundo Chan *et.al.* (1992) [5], há basicamente três métodos distintos baseados nesta aproximação:

   a. Alinhamento de um grupo de seqüências relacionadas é freqüentemente feito à mão. Esta aproximação é subjetiva. Em Patthy citado em Chan *et.al.* (1992) [5] simula esta aproximação por um processo de alinhamento iterativo controlado que pesa as características básicas da família de proteína e então força o alinhamento dos resíduos conservados. O processo basicamente determina a seqüência de consenso que incorpora as principais características das seqüências relacionadas. O processo de alinhamento de seqüências utilizando este método segue os seguintes passos:

      i. Pontuação de similaridade de todos os possíveis pares de seqüência são obtidos e as seqüências relacionadas mais próximas são agrupadas juntas baseadas nesta pontuação.

      ii. Para cada grupo, os resíduos conservados na maioria das comparações pareadas são identificados e as localizações dos *gaps* são determinadas. Destes dados, uma tentativa de uma seqüência de consenso é deduzida. As seqüências do grupo são então alinhadas com estas seqüências de consenso para maximizar a similaridade das

regiões conservadas. Do alinhamento resultante, um grupo de seqüências de consenso corrigido é determinado.

iii. Pela comparação da seqüência de consenso dos vários grupos, uma seqüência de consenso unificada da maioria destes caracteres das seqüências é deduzida.

iv. As seqüências são alinhadas com a seqüência de consenso unificada e o resultado global do alinhamento é utilizado para produzir outra seqüência de consenso unificada

.

b. MULTAN – é um programa que pode alinhar um grande número de seqüências de ácidos nucléicos. Primeiro uma das seqüências é escolhida como seqüência de consenso inicial. As outras seqüências são então alinhadas uma vez com esta seqüência de consenso para gerar um alinhamento no qual uma nova seqüência de consenso é derivada. O processo é repetido se a nova seqüência de consenso não for igual à seqüência anterior. Em cada interação, cada seqüência é alinhada com a seqüência de consenso assim introduzindo *gaps* dentro de cada seqüência ou na seqüência de consenso. Se um *gap* é introduzido dentro de uma seqüência de consenso, então o *gap* correspondente também será introduzido em cada uma das seqüências.

c. SEQCMP – é um programa que encontra a seqüência de consenso para um conjunto de seqüências de ácidos nucléicos. O principal emprego é para gerar uma matriz de pontos para cada par de seqüências e todas as matrizes geradas são então sobrepostas uma com as outras para identificar os pontos em comum em cada matriz e portanto a seqüência de consenso.

Ainda dentro desta classificação, Dan Gusfield (1997) [6] propõem a resolução através de três métodos aproximativos: *String* de consenso de Steiner, Alinhamento múltiplo consensual ótimo e Alinhamento múltiplo consensual ótimo no nível das colunas. Neste caso, apesar de impraticável, retornam o valor ótimo do alinhamento. Então, métodos aproximativos são

aplicados neles, fazendo com isso convergirem para uma mesma solução que é no máximo duas vezes pior que o Alinhamento Múltiplo por Soma de Pares, que retorna o valor ótimo deste alinhamento.

4. **Clusterização** – esta aproximação tenta construir uma árvore filogenética para o alinhamento de seqüências ou arrumar a seqüências dentro de uma ordem particular nas quais as seqüências são alinhadas uma a uma.

Aproximação por template – template é uma seqüência de consenso de um seguimento de um alinhamento correspondendo uma parte da estrutura secundária. Muitos templates ao longo do alinhamento podem ser obtidos e são usados para guiar o alinhamento de outras seqüências relacionadas com uma estrutura secundária desconhecida. As seqüências relacionadas são alinhadas com os templates e incluídas no alinhamento inicial uma a uma. Os templates são modificados para incluir os resíduos variáveis destas seqüências durante o processo de alinhamento.

2.7.Ferramentas atuais para se obter Alinhamento Múltiplo

Há basicamente duas aproximações via software na determinação de similaridades entre proteínas. Os seguintes métodos globais constroem um alinhamento com o tamanho da seqüência de entrada: Amult, DFalign, Multal, MAS, Tulla, Clustal V, MST. Uma sub-classe do método global tenta primeiro identificar uma série ordenada de motifs e então processa o alinhamento intervindo em regiões. Ex.: Genalign, Assemble (Tabela 2.1).

Métodos locais por outro lado, somente tentam identificar séries ordenadas de motifs enquanto ignoram regiões entre motifs. Ex.: (Pima, Pralign, Macaw). Neste trabalho não entraremos em detalhes com respeito aos métodos de alinhamento local.

O alinhamento global alinha toda a seqüência que está sendo analisada, enquanto o alinhamento local alinha somente regiões que são mais similares, descartando o restante da seqüência.

Motifs são regiões contínuas de três a nove resíduos (caracteres) conservados frequentemente envolvidos na função ou na integridade estrutural das proteínas. Por este

motivo é tão importante identificar séries ordenadas de motifs em alinhamentos de proteínas.

Tabela 2.1: Tabela contendo os métodos com seus respectivos algoritmos e matrizes de pontuação mais utilizados na atualidade (Fonte: M. McLure, T. Vasi, and W. Fitch. [2]).

Method (Developer)	Algorithm	Matrix*
Global:		
AMULT (G. Barton)	NW	Any
ASSEMBLE (M. Vingron)	Dot matrix NW	Log odds
CLUSTAL V (D. Higgens)	WL	Any
DFALIGN (D.-F. Feng)	NW	Log odds
GENALIGN[g] (H. Martinez)	CW, NW	UM
MSA (S. Altschul)	CL	PAM250
MULTAL (W. Taylor)	NW	UM, PAM250
MWT (J. Kececioglu)	maximum weight trace	Any
TULLA (S. Subbiah)	NW	Any
Local:		
MACAW (G. Schuler)	SW	PAM250
PIMA (P. Smith)	SW	AACH
PRALIGN (M. Waterman)	CW	PAM250

Métodos Globais

O diagrama da Figura 2.12 resume a implementação básica dos vários algoritmos empregados nos nove diferentes métodos de alinhamento múltiplo empregados.

Segundo McCure etal [2], todos os métodos começam pela comparação de forma pareada de todas as seqüências. Muitos ferramentas agrupam as seqüências dentro de sub-alinhamentos usando medidas de similaridade (por exemplo Genalign e Multal) ou árvore filogenética (Clustal V, Amult e DFalign). As ferramentas Genalign, Multal e Clustal V subsequentemente alinham os sub-alinhamentos agrupados um com os outros empregando vários métodos de consenso que reduz cada sub-alinhamento em uma seqüência simples de consenso, permitindo os sub-alinhamentos serem combinados alinhando suas seqüências de consenso produzindo um alinhamento múltiplo progressivo. Além disso, a ferramenta Genalign permite o usuário escolher o algoritmo Needleman- Wunsch citado em McCure etal [2] (NW - Programação Dinâmica) ou palavra de consenso (CW) para fazer o alinhamento enquanto Clustal V permite ao usuário especificar parâmetros tanto no estágio de alinhamento pareado quanto no estágio de alinhamento múltiplo.

As ferramentas Amult e DFalign produzem um alinhamento múltiplo progressivo diretamente do estágio de clusterização. Amult então produz um alinhamento múltiplo final através da otimização do alinhamento múltiplo progressivo. Uma nova característica do

Amult provê a opção de produzir o Alinhamento Múltiplo Progressivo diretamente do estágio de ordenação pareada, evitando a clusterização filogenética. Os métodos MAS e Tulla produzem um Alinhamento Múltiplo Progressivo e então o Alinhamento Múltiplo Final. O MSA também pode produzir o Alinhamento Múltiplo Final evitando passar pelo alinhamento múltiplo progressivo, caso o usuário supra os limites superiores para todos os pares de seqüências que são necessários na programação dinâmica multidimensional em um espaço restrito.

Assemble e MWT produzem um Alinhamento Múltiplo Final diretamente da análise pareada. MAS e MWT diferem dos outros métodos porque eles computam um alinhamento múltiplo ótimo utilizando uma função de alinhamento múltiplo bem definida.

2.7.1.Algoritmos Básicos

As formulações biológicas interessantes (que contém maior significância biológica) estão na classe dos problemas NP-Completos (ou seja, não há solução serial para qualquer entrada em tempo polinomial).

Nesta seção serão apresentados os principais algoritmos utilizados pelos métodos citados acima.

A aproximação pelo método da matriz de pontos (dot plots ou dor matrix) tem sido muito utilizada na analise de seqüências. Em resumo, um vetor de duas dimensões de duas seqüências é criado e um ponto é colocado em cada match. No método Assembly a matriz de pontos é inicialmente empregada como um filtro para identificar e retirar somente os motifs que estão conservados entre um dado conjunto de seqüências, antes de se usar a programação dinâmica.

Segundo McCure *et.al.* [2], muitos métodos utilizam a Programação Dinâmica, que por sua vez encontra o alinhamento ótimo entre duas seqüências usando vários sistemas de pontuação.

Segundo McCure *et.al.* [2], uma redução significante em tempo de CPU para o caso de duas seqüências, com pouca perda de sensibilidade, foi alcançado pelo uso do método da matriz de pontos (dot matrix) em conjunto com o algoritmo NW (Needleman and Wunsch) citado em McCure *et.al.* [2], resultando no algoritmo WL (Wilbur-Lipman). Outro melhoramento do algoritmo NW, quando estendido para seqüências múltiplas, foi

alcançado pelo alinhamento pareado para re*string*ir a procura por caminhos ótimos entre seqüências múltiplas, então criando o algoritmo CL (Carrillo and Lipman) citado em McCure *et.al.* [2].

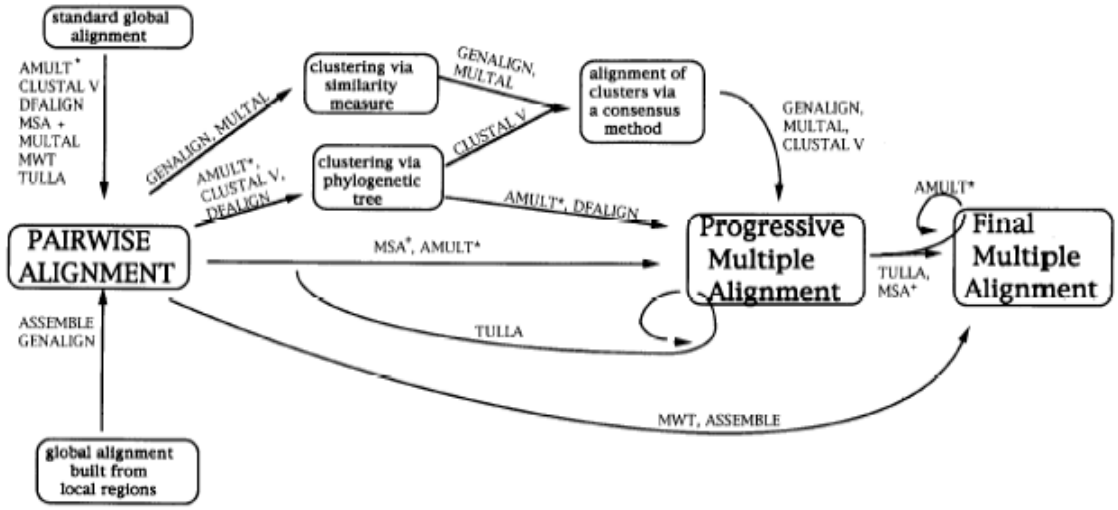


Figura 2.12: Implementação básica dos vários algoritmos empregados no alinhamento múltiplo de seqüências (Fonte: M. McLure, T. Vasi, and W. Fitch.[2])

2.8.Agrupamento

Segundo Furtado (1996) [8], as análises rudimentares e exploratórias de dados como os procedimentos gráficos, auxiliam em geral, o entendimento da complexa natureza da análise multivariada. Encontrar nos dados uma estrutura natural de agrupamento é uma importante técnica exploratória. A análise de agrupamento não considera o número de grupos e é realizada com base na similaridade ou dissimilaridade (distância).

O objetivo desta análise é agrupar objetos semelhantes segundo suas características (variáveis). Mas isto leva a dois problemas:

1. não existem impedimentos para realizar o agrupamento de variáveis semelhantes segundo as realizações obtidas pelos objetos amostrados.
2. verificar se um individuo A é mais parecido com B do que com C.

No segundo problema, ainda segundo Furtado (1996) [8], quando o número de variáveis envolvidas é pequeno, uma inspeção visual pode resolver o problema, caso contrário deve-se utilizar instrumentos estatísticos para estabelecer esta parecença.

2.8.1.Medidas de parecença (similaridade e dissimilaridade)

Segundo Furtado (1996) [8], é necessário especificar um coeficiente de parecença que indique a proximidade entre os indivíduos. É importante considerar, em todos os casos semelhantes a este, a natureza da variável (discreta, contínua, binária) e a escala de medida (nominal, ordinal, real ou razão).

Dentre as principais medidas de distância podemos destacar:

- Distância Euclidiana

$$D(\underset{\sim 1}{X}, \underset{\sim 2}{X}) = \sqrt{(X_{11} - X_{21})^2 + (X_{12} - X_{22})^2 + \ldots + (X_{1p} - X_{2p})^2}$$

- Distância de Mahalanobis

$$D(\underset{\sim 1}{X}, \underset{\sim 2}{X}) = \sqrt{(\underset{\sim 11}{X} - \underset{\sim 21}{X})^2 \, S^{-1} \, (\underset{\sim 12}{X} - \underset{\sim 22}{X})^2}$$

- Métrica de Minkowski, a qual depende de funções modulares.

$$D(\underset{\sim 1}{X}, \underset{\sim 2}{X}) = \left[\sum_{i-1}^{p} | \underset{\sim li}{X} - \underset{\sim z}{X} |^m \right]^{l/m}$$

Sempre que possível é conveniente usar distâncias verdadeiras, ou seja, aquelas que obedecem a desigualdade triangular, para agrupamento de objetos, embora alguns algoritmos de agrupamento não exijam o atendimento desta preocupação.

- Distância Euclidiana media,

$$d_{h,i} = \sqrt{\sum_{j-1}^{p} \frac{\left(X_{hj} - X_{ij}\right)^2}{p}}$$

- Distância Euclidiana Padronizada,

$$d_{h,i} = \sqrt{\sum_{j=1}^{p} \left(\frac{X_{hj} - X_{ij}}{\sqrt{S_{ij}}} \right)^2} = \sqrt{(\underset{\sim h}{X} - \underset{\sim i}{X})^t \, D^{-1} (\underset{\sim h}{X} - \underset{\sim i}{X})}$$

Muitas vezes os objetos não podem ser mensurados em variáveis quantitativas. Estas variáveis podem ser transformadas em dicotônicas (binárias), determinando um ponto de corte de interesse prático. Da mesma forma, variáveis qualitativas podem ser transformadas em variáveis binárias tomando-se como valor 1 a presença de uma determinada realização e

o valor 0 para as demais. Estas ocorrências de dados binários são bastante comuns em genética molecular.

Muitas vezes também as medidas de dissimilaridade podem ser transformadas em medidas de similaridade pela utilização desta relação:

$$S_{h,i} = \frac{1}{1 + d_{h,i}}$$

Outra forma de se obter coeficientes de similaridade a partir da distância euclidiana, calculada com variáveis padronizadas, pode ser obtida pelo coeficiente de Cattel:

$$S_{h,i} = \frac{2\left(p - \frac{2}{3}\right) - d^2{}_{h,j}}{2\left(p - \frac{2}{3}\right) + d^2{}_{h,j}}$$

No entanto nem sempre é possível construir distâncias a partir de similaridades. Isso só pode ser feito se a matriz de similaridades for não negativa definida.

$$d_{h,j} = \sqrt{2(1 - S_{h,j})}$$

2.8.2.Agrupamentos Hierárquicos

Segundo Furtado (1996) [8], os agrupamentos hierárquicos são realizados por sucessivas fusões ou por sucessivas divisões. Os métodos hierárquicos aglomerativos iniciam com tantos grupos quanto aos objetos. Inicialmente, os objetos mais similares são agrupados e fundidos formando um único grupo. Eventualmente o processo é repetido, e com o decréscimo da similaridade, todos os subgrupos são fundidos, formando um único grupo com todos os objetos.

A seguir está apresentado um algoritmo geral para agrupamentos hierárquicos aglomerativos com n objetos (itens ou variáveis).

1. Iniciar com n grupos, cada um com único elemento e com uma matriz simétrica nxn de dissimilaridades (distâncias) D = $\{d_{hi}\}$.
2. Buscar na matriz D o par de grupos mais similar (menor distância) e fazer a distância entre os grupos mais similares U e V igual a d_{UV}.

3. Fundir os grupos U e V e nomeá-los por (UV). Recalcular e rearranjar as distâncias na matriz D:

(a) eliminando as linhas e colunas correspondentes a U e V e

(b) acrescentando uma linha e coluna com as distâncias entre os grupo (UV) e os demais grupos.

4. Repetir os passos 2 e 3 num total de (n-1) vezes (todos os objetos estarão em um único grupo). Anotar a identidade dos grupos que vão sendo fundidos e os respectivos níveis (distâncias) nas quais isto ocorre.

Ligação simples (vizinho mais próximo).

Segundo Johnson [12], no método de ligação simples, os objetos menos distantes (min (d_{hi})) devem, inicialmente, ser fundidos. Em seguida é calculada a distância entre os objetos fundidos e o restante dos objetos, utilizando para formar a nova matriz de distância dos vizinhos mais próximos. Este processo é executado até que todos os objetos estejam em apenas um grupo.

Para exemplificar é considerado um exemplo no qual se destacam quatro objetos (A,B,C,D), e para o qual a matriz das distâncias D_{DIST} entre os objetos é apresentada a seguir. (Figura 2.13)

	A	B	C	D
A	0			
B	3	0		
C	7	9	0	
D	8	6	5	0

Figura 2.13: Matriz das distâncias D_{DIST}

Para ilustrar o método da ligação simples, os objetos menos distantes devem, inicialmente, ser fundidos. Então, min ($D_{DIST}^{A,B}$) = $d_{A,B}$=3. O próximo passo é fundir A com B formando o grupo (AB) e em seguida calcular as distâncias deste grupo e os objetos remanescentes. As distâncias dos vizinhos mais próximos são,

$d_{(AB),C} = \min\{ d_{AC}, d_{BC} \} = \min \{7,9\} = 7$

$d_{(AB),D} = \min\{ d_{AD,} d_{BD} \} = \min \{8,6\} = 6$

A nova matriz D_{DIST} para o próximo passo é (Figura 2.14)

	AB	C	D
AB	0		
C	7	0	
D	6	5	0

Figura 2.14: Nova Matriz das distâncias D_{DIST}

A menor distância entre D e C, com $d_{DC} = 5$, os quais foram fundidos formando o subgrupo DC, no nível 5. Recalculando as distâncias têm-se,

$d_{(DC),(AB)} = \min\{ d_{D(AB),} d_{C(AB)} \} = \min \{6,7\} = 6$

A nova matriz D_{DIST} fica, (Figura 2.15)

	DC	AB
DC	0	
AB	6	0

Figura 2.15: Matriz das distâncias D_{DIST}

Conseqüentemente o grupo DC é fundido com AB na distância 6.

Os resultados finais destes agrupamentos podem ser apresentados por gráficos denominados dendogramas. Os dendogramas apresentam os elementos e os respectivos pontos de fusão ou divisão dos grupos formados em cada estágio. Um exemplo de um dendograma é apresentado na Figura 2.16.

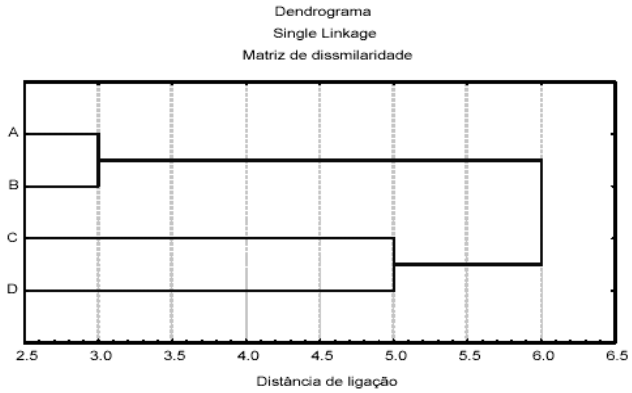


Figura 2.16: Dendograma para agrupar 4 objetos (A,B,C,D) pelo método da ligação simples (vizinho mais próximo). (Fonte: Furtado [8])

(b) **Ligação completa** (vizinho mais distante)

Segundo Johnson [12], o método da ligação completa é realizado da mesma forma que o do vizinho mais próximo, com a exceção de que a distância entre grupos é tomada como a máxima distância entre dois elementos de cada grupo.

(c) **Ligação média** (método do centróide)

Segundo Johnson [12], método da ligação média é realizado da mesma forma que o do vizinho mais próximo e mais distante, com exceção de que a distância entre grupos é tomada como a média entre dois elementos de cada grupo.

3.METODOLOGIA

3.1.Desenvolvimento do Programa

O trabalho foi realizado no Ambiente do Laboratório de Departamento do Curso de Ciência da Computação da Universidade Federal de Lavras – DCC-UFLA, utilizando uma máquina com 512 mb de memória Ram, processador AMD Semprom 2800+, com o sistema operacional Windows 2000 Professional, linguagem de programação C++ com o compilador Mingw, e para avalição dos resultados retornados pelo programa, foi feita uma comparação com os resultados do alinhamento retornado pela ferramenta ClustalW [9].

O trabalho foi desenvolvido utilizando a técnica de programação dinâmica explicada anteriormente (seção 2.6.1). Esta foi empregada com o propósito de se encontrar o melhor caminho entre dois *profile*s (seção 2.6.2).

Existem diferenças entre a relação de recorrência apresentada para a programação dinâmica entre:

I. apenas duas seqüências,
II. uma seqüência e um *profile*,
III. e *profile* com *profile*, sendo esta última empregada no presente projeto.

Nesta última, se alinha de forma pareada as colunas de um *profile* com as colunas de outro *profile* obedecendo a relação de recorrência abaixo:

$$D(i, j) = \max (D(i-1, j-1)+ \sum_i \sum_j s(x, y)*p_1(x,i)*p_2(y,j), D(i-1, j) + \sum_i s(x, -)*p_1(x,i), D(i, j-1)+ \sum_j s(-, y)*p_2(y,j)) \quad (3.1)$$

Onde:

$s(x, y)$ é dado pela distância na matriz de pontuação entre o caractere x e y,

$p_1(x, i)$ é a freqüência do caracter x na coluna i do *profile* 1,

$p_2(y, j)$ é a freqüência do caracter y na coluna j do *profile* 2.

A idéia da geração de um *profile* é mostrada na Figura 3.1. São dadas as freqüências de cada caracter do alfabeto ∑ em cada coluna do *profile*.

Profile da seqüência tgactg:

N = 6

N= número de caracteres do alinhamento.

∑		C_1	C_2	C_3	C_4	C_5	C_6
	A	0	0	1	0	0	0
	C	0	0	0	1	0	0
	T	1	0	0	0	1	0
	G	0	1	0	0	0	1
	-	0	0	0	0	0	0

a)

Profile do alinhamento de tgactg e tg_cag:

N = 6

N= número de caracteres do alinhamento.

∑		C_1	C_2	C_3	C_4	C_5	C_6
	A	0	0	0,5	0	0,5	0
	C	0	0	0	1	0	0
	T	1	0	0	0	0,5	0
	G	0	1	0	0	0	1
	-	0	0	0,5	0	0	0

b)

Figura 3.1: Exemplo de um *Profile*. Em a) *profile* representa apenas uma seqüência e em b) o *profile* representa um alinhamento

A idéia de alinhamento entre colunas de *profile*s é mostrada abaixo e Figura 3.2. Para se alinhar dois *profile*s, deve-se alinhar todos os caracteres da colunas do *profile* 1 com todos os caracteres das colunas do *profile* 2.

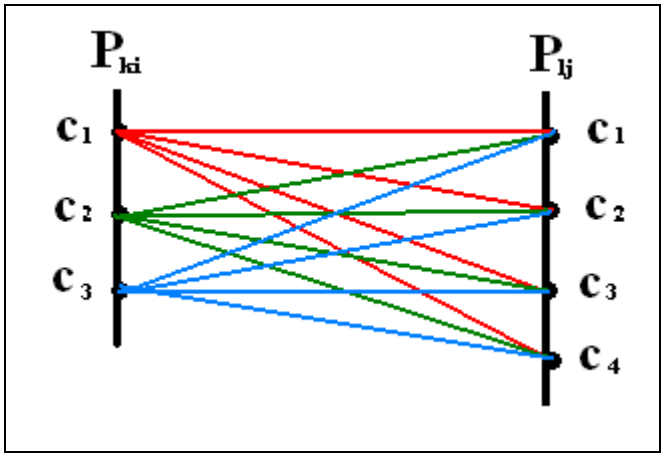


Figura 3.2: Alinhamento a coluna P_{kl} do *profile* 1 e P_{lj} do *profile* 2

O esquema de pontuação utilizado para saber o valor do alinhamento de um caracter com outro foi a matriz Blosum62 (Figura 3.3).

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0	-2	-1	0	-4
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3	-1	0	-1	-4
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3	3	0	-1	-4
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3	4	1	-1	-4
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1	-3	-3	-2	-4
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2	0	3	-1	-4
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3	-1	-2	-1	-4
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3	0	0	-1	-4
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3	-3	-3	-1	-4
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1	-4	-3	-1	-4
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2	0	1	-1	-4
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1	-3	-1	-1	-4
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1	-3	-3	-1	-4
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2	-2	-1	-2	-4
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2	0	0	0	-4
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0	-1	-1	0	-4
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3	-4	-3	-2	-4
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1	-3	-2	-1	-4
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4	-3	-2	-1	-4
B	-2	-1	3	4	-3	0	1	-1	0	-3	-4	0	-3	-3	-2	0	-1	-4	-3	-3	4	1	-1	-4
Z	-1	0	0	1	-3	3	4	-2	0	-3	-3	1	-1	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
X	0	-1	-1	-1	-2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-2	0	0	-2	-1	-1	-1	-1	-1	-4
*	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	1

Figura 3.3: Matriz de pontuação Blosum 62

A idéia de se utilizar a programação dinâmica para implementar o alinhamento múltiplo de seqüências é devido ao fato dela retornar o valor ótimo para as entradas fornecidas. Como o custo computacional dispensado para sua execução é elevado, utilizamos a caracterização de seqüências via *profile* para efetuar mais de um alinhamento múltiplo de seqüências. Com isso, através de um alinhamento pareado global de *profile* conseguimos chegar em um alinhamento múltiplo global de seqüências, com um gasto computacional menor do que seria necessário para se fazer o mesmo alinhamento de forma múltipla diretamente com as seqüências. Para conseguirmos isto, utilizamos também a clusterização. Representamos um grupo de seqüências, quando conseguimos alinhar estas e assim geramos um novo *profile* destas seqüências alinhadas. Então o conceito de grupo é a geração de um *profile* que só pode ser obtido através de um conjunto de seqüências alinhadas.

No primeiro passo, o *profile* é referente a cada seqüência de entrada. Isto significa que se tivermos N seqüências de entrada teremos N grupos, ou seja N *profile*s. Então é feito o alinhamento (programação dinâmica) para toda combinação 2x2 destes N *profile*s. Este resultado é com o intuito de estabeler uma métrica entre as seqüências analisadas, que serão armazenadas em uma matriz A_{NxN}. É utilizada apenas a diagonal inferior da matriz, pois o cálculo do *profile* i com o *profile* j é o mesmo valor do *profile* j com o *profile* i. Junto com esta matriz que armazena os valores referentes a programação dinâmica é criada uma nova matriz T_{NxN} de mesma ordem, onde se armazena os *transcripts* de edição obtidos no final da programação dinâmica que por sua vez serão necessários para se construir os *tracebacks* dos *profile*s alinhados (Figura 3.4).

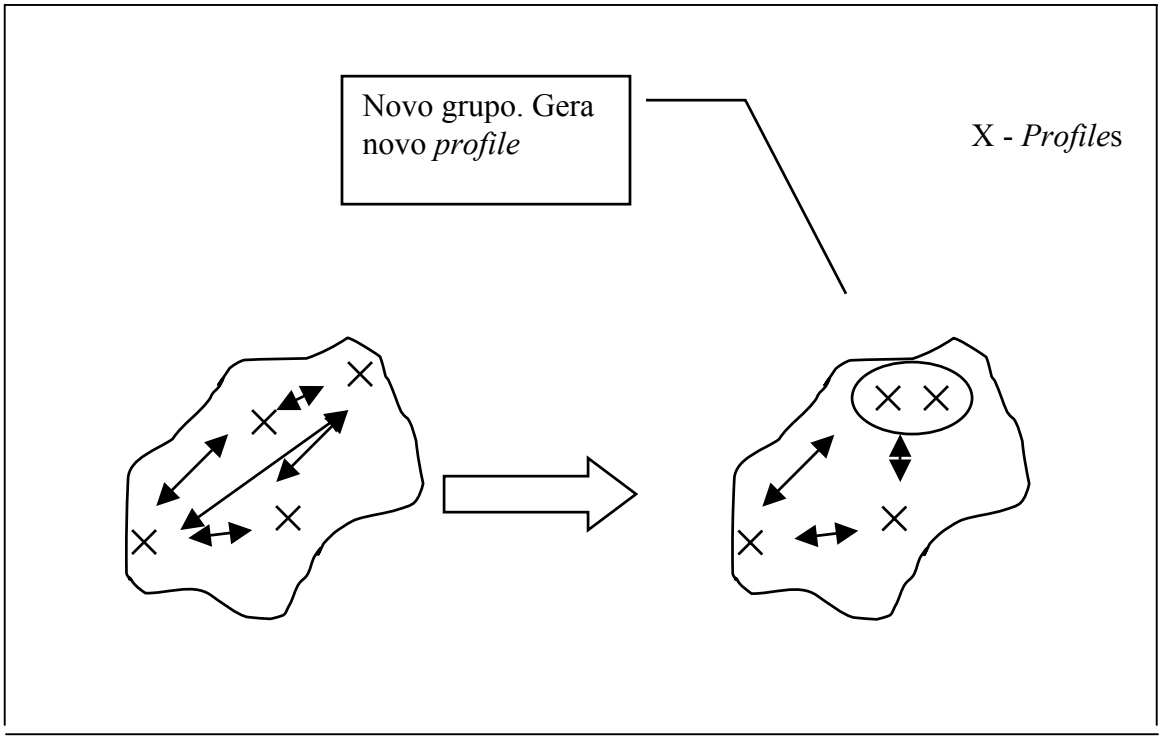


Figura 3.4: Processo de Clusterização. X representa cada *profile*.

Após o preenchimento da matriz A_{NxN}, busca-se o menor valor do elemento de matriz (i, j) e clusteriza-se seqüências referentes à respectiva posição (i, j) de menor valor desta matriz. Como dito anteriormente este processo de clusterização refere-se ao alinhamento destes dois *profile*s que resultaram na menor pontuação da programação dinâmica. Após este alinhamento é gerado um novo *profile* finalizando o processo de clusterização dos *profile*s antigos. É recalculado o valor de similaridade entre este novo grupo e o restante dos *profile*s via Programação Dinâmica. Então uma nova matriz $A_{N-1xN-1}$ contendo o valor entre as similaridades entre cada grupo é gerada. Esta matriz possui uma ordem a menos que a matriz anterior como é mostrado na Figura 3.5. Este passo é repetido até que se tenha uma matriz de uma dimensão apenas.

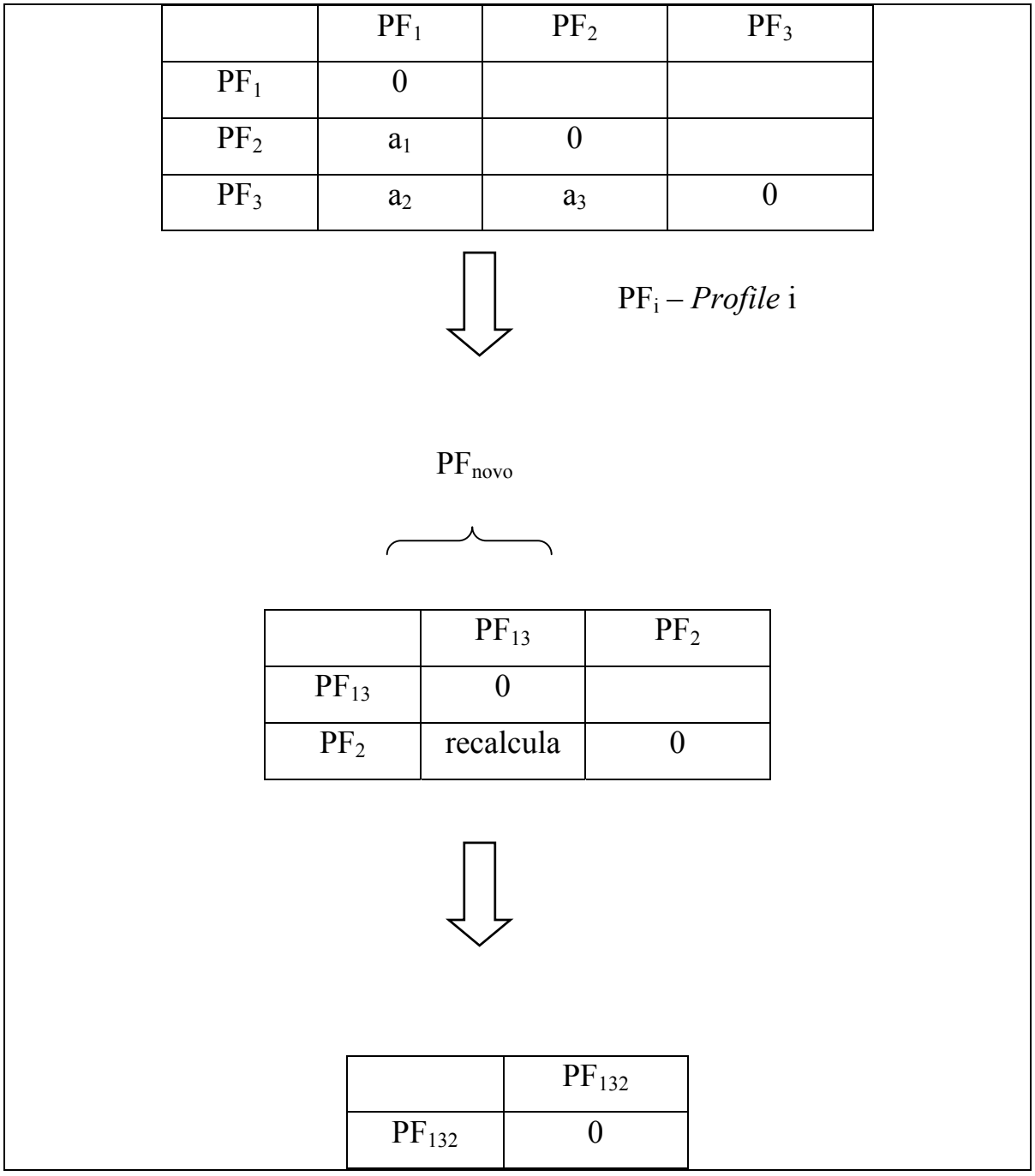


Figura 3.5: Neste ponto obtém-se o *profile* PF_{132} com o mesmo número de seqüências iniciais, só que agora elas estão alinhadas.

Comparação

A metodologia de comparação foi desenvolvida da seguinte forma (Figura 3.6):

1. escolha das seqüências de proteínas a serem comparadas;
2. obtenção do conjunto de seqüências de proteínas alinhadas de forma pareada no Blast, com a seqüência obtida na etapa anterior;
3. escolha da melhor seqüência (maior pontuação), pior seqüência (pior pontuação) e uma seqüência intermediaria (próximo da média entre a melhor e a pior seqüência selecionadas). O motivo de se utilizar três seqüências foi de assim produzir um alinhamento múltiplo em tempo razoável, devido a complexidade do algoritmo utilizado (de Programação Dinâmica) e o tamanho das seqüências analisadas (cerca de 150 caracteres);
4. alinhamento múltiplo das três seqüências na ferramenta ClustalW e no programa desenvolvido;
5. comparação entre o alinhamento obtido no ClustalW e o alinhamento obtido no programa desenvolvido.

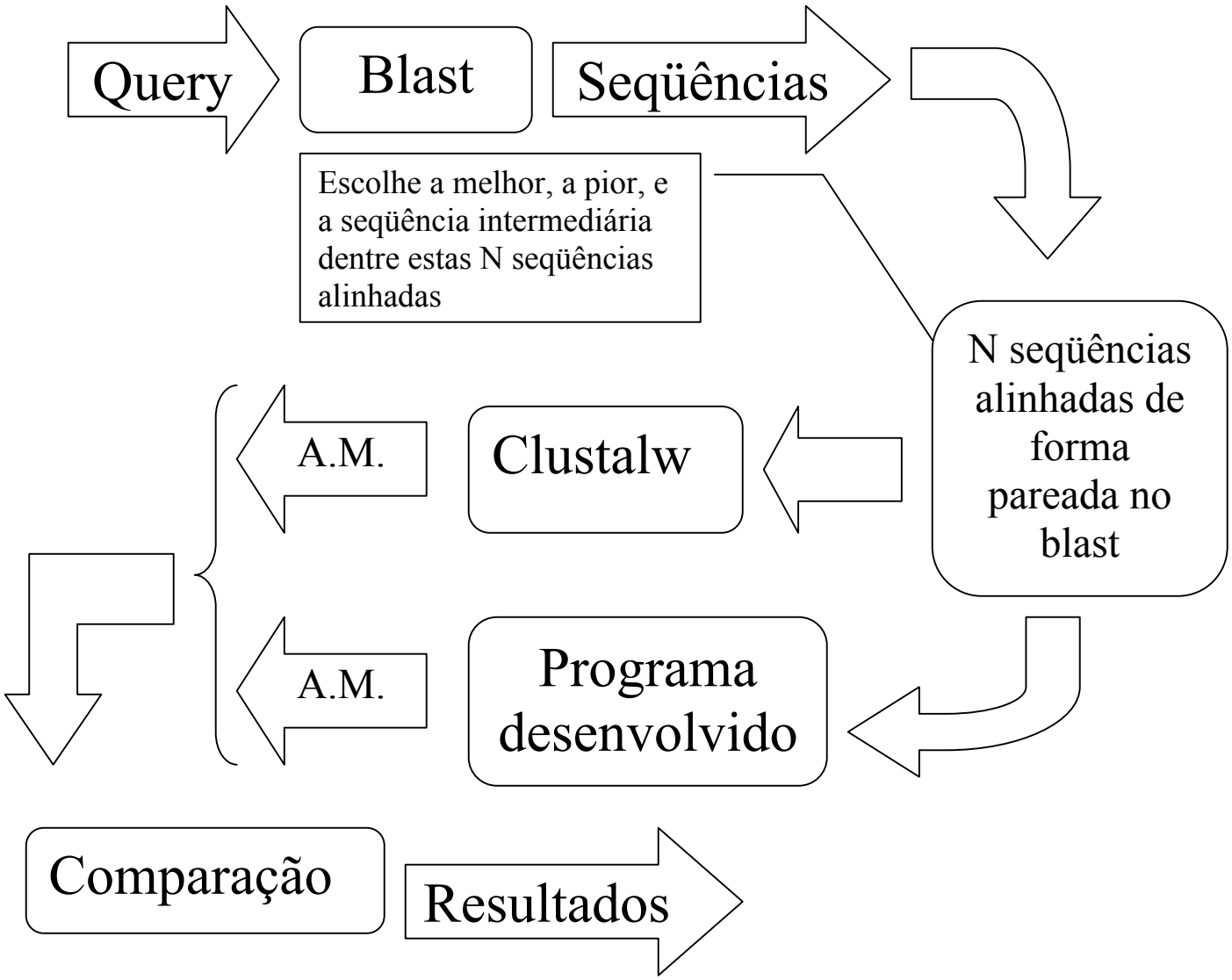


Figura 3.6: Processo de comparação

4.RESULTADOS E DISCUSSÕES

Para se comparar as seqüências, como dito anteriormente, foi necessário fazer uma busca com a query gi|85860063 na ferramenta Blast (NCBI – [10]) utilizando o algoritmo blastp, que alinha seqüências de proteínas (Figura4.1).

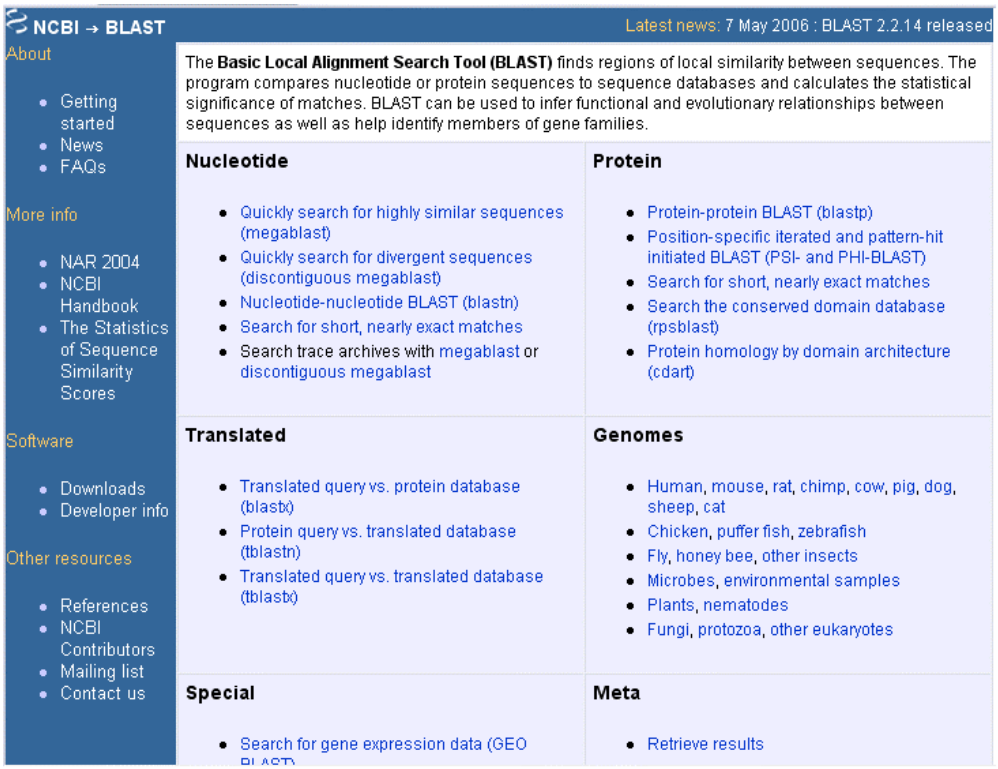


Figura 4.1: Tela inicial do Blast

A tela para executar o alinhamento com a query e o banco de dados é mostrada na Figura 4.2.

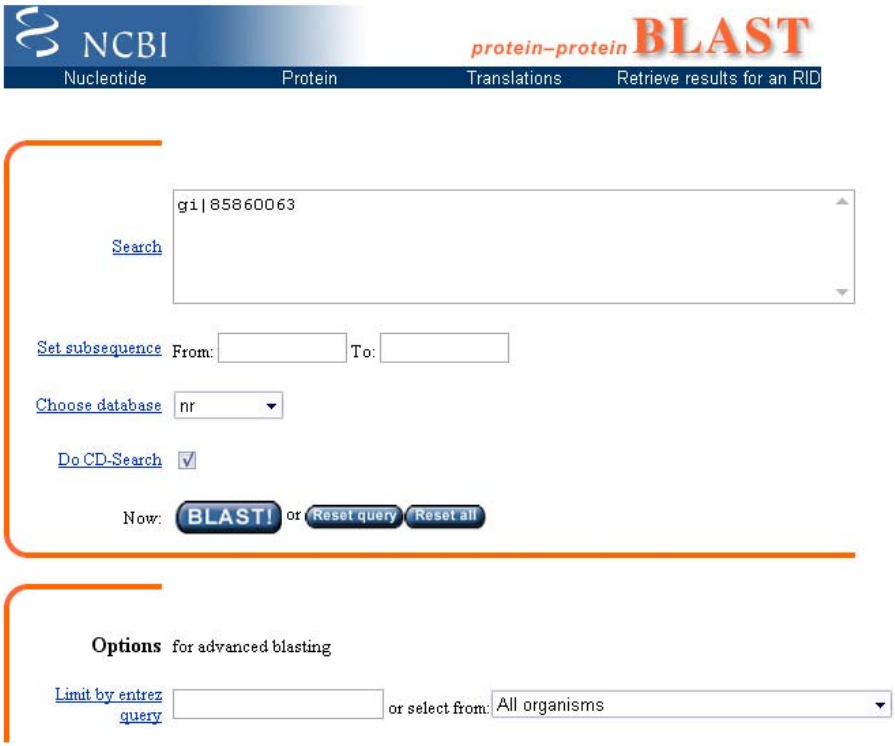


Figura 4.2: Tela de pesquisa do Blast

Após o Blast alinhar a query com as seqüências contidas em seu banco de dados, uma nova tela apresentando os melhores alinhamento será exibida. Estes alinhamentos estão ordenados em ordem decrescente de similaridade (do melhor para o pior).

O critério para seleção das seqüências foi a escolha da melhor seqüência (maior bits), a pior e uma seqüência com pontuação (bits) próximo da média da melhor e pior seqüências. A tela retornada pelo Blast que exemplifica a ordenação destas seqüências pode ser vista na Figura 4.3 e 4.4.

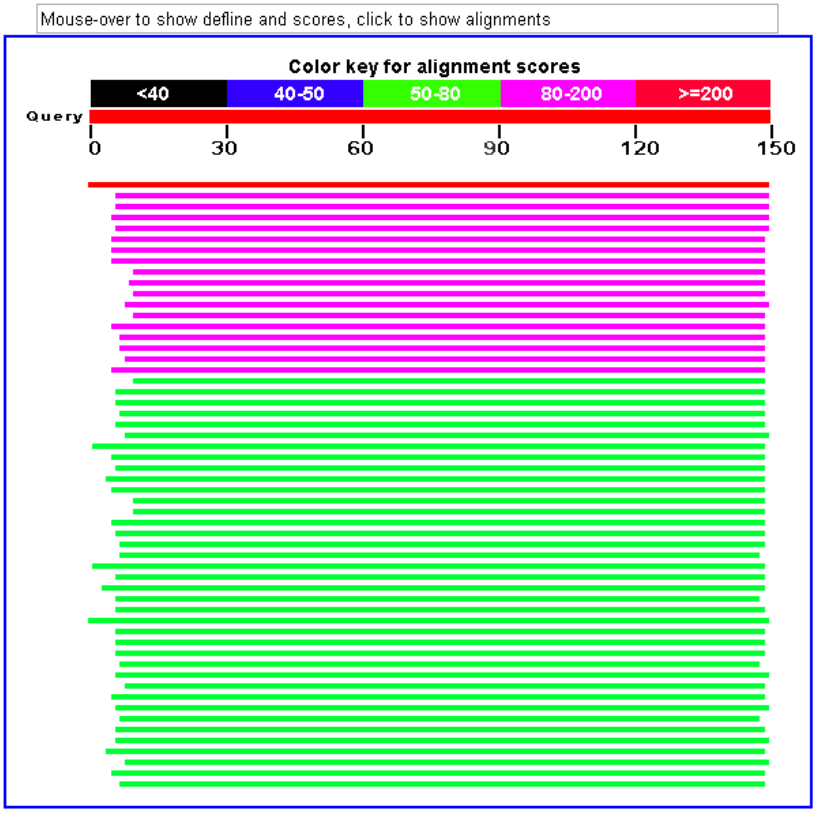
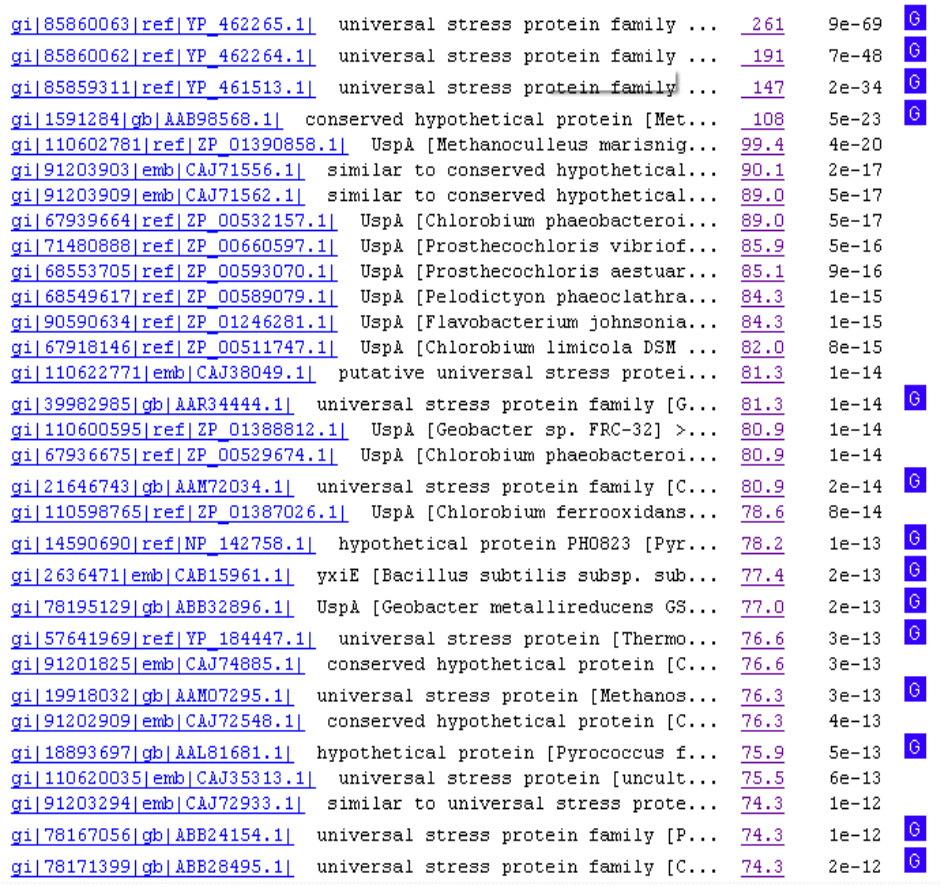


Figura 4.3: Representação gráfica da similaridade entre as seqüências alinhadas pelo Blast. Fonte: [11]



gi|85860063|ref|YP_462265.1|	universal stress protein family ...	261	9e-69
gi|85860062|ref|YP_462264.1|	universal stress protein family ...	191	7e-48
gi|85859311|ref|YP_461513.1|	universal stress protein family ...	147	2e-34
gi|1591284|gb|AAB98568.1|	conserved hypothetical protein [Met...	108	5e-23
gi|110602781|ref|ZP_01390858.1|	UspA [Methanoculleus marisnig...	99.4	4e-20
gi|91203903|emb|CAJ71556.1|	similar to conserved hypothetical...	90.1	2e-17
gi|91203909|emb|CAJ71562.1|	similar to conserved hypothetical...	89.0	5e-17
gi|67939664|ref|ZP_00532157.1|	UspA [Chlorobium phaeobacteroi...	89.0	5e-17
gi|71480888|ref|ZP_00660597.1|	UspA [Prosthecochloris vibriof...	85.9	5e-16
gi|68553705|ref|ZP_00593070.1|	UspA [Prosthecochloris aestuar...	85.1	9e-16
gi|68549617|ref|ZP_00589079.1|	UspA [Pelodictyon phaeoclathra...	84.3	1e-15
gi|90590634|ref|ZP_01246281.1|	UspA [Flavobacterium johnsonia...	84.3	1e-15
gi|67918146|ref|ZP_00511747.1|	UspA [Chlorobium limicola DSM ...	82.0	8e-15
gi|110622771|emb|CAJ38049.1|	putative universal stress protei...	81.3	1e-14
gi|39982985|gb|AAR34444.1|	universal stress protein family [G...	81.3	1e-14
gi|110600595|ref|ZP_01388812.1|	UspA [Geobacter sp. FRC-32] >...	80.9	1e-14
gi|67936675|ref|ZP_00529674.1|	UspA [Chlorobium phaeobacteroi...	80.9	1e-14
gi|21646743|gb|AAM72034.1|	universal stress protein family [C...	80.9	2e-14
gi|110598765|ref|ZP_01387026.1|	UspA [Chlorobium ferrooxidans...	78.6	8e-14
gi|14590690|ref|NP_142758.1|	hypothetical protein PHO823 [Pyr...	78.2	1e-13
gi|2636471|emb|CAB15961.1|	yxiE [Bacillus subtilis subsp. sub...	77.4	2e-13
gi|78195129|gb|ABB32896.1|	UspA [Geobacter metallireducens GS...	77.0	2e-13
gi|57641969|ref|YP_184447.1|	universal stress protein [Thermo...	76.6	3e-13
gi|91201825|emb|CAJ74885.1|	conserved hypothetical protein [C...	76.6	3e-13
gi|19918032|gb|AAM07295.1|	universal stress protein [Methanos...	76.3	3e-13
gi|91202909|emb|CAJ72548.1|	conserved hypothetical protein [C...	76.3	4e-13
gi|18893697|gb|AAL81681.1|	hypothetical protein [Pyrococcus f...	75.9	5e-13
gi|110620035|emb|CAJ35313.1|	universal stress protein [uncult...	75.5	6e-13
gi|91203294|emb|CAJ72933.1|	similar to universal stress prote...	74.3	1e-12
gi|78167056|gb|ABB24154.1|	universal stress protein family [P...	74.3	1e-12
gi|78171399|gb|ABB28495.1|	universal stress protein family [C...	74.3	2e-12

Figura 4.4: Seqüências ordenadas de forma decrescente de similaridade

Logo após a escolha destas três seqüências, elas são alinhadas de forma múltipla na ferramenta ClustalW (ebi). A tela do ClustalW é mostrada na figura 4.4.

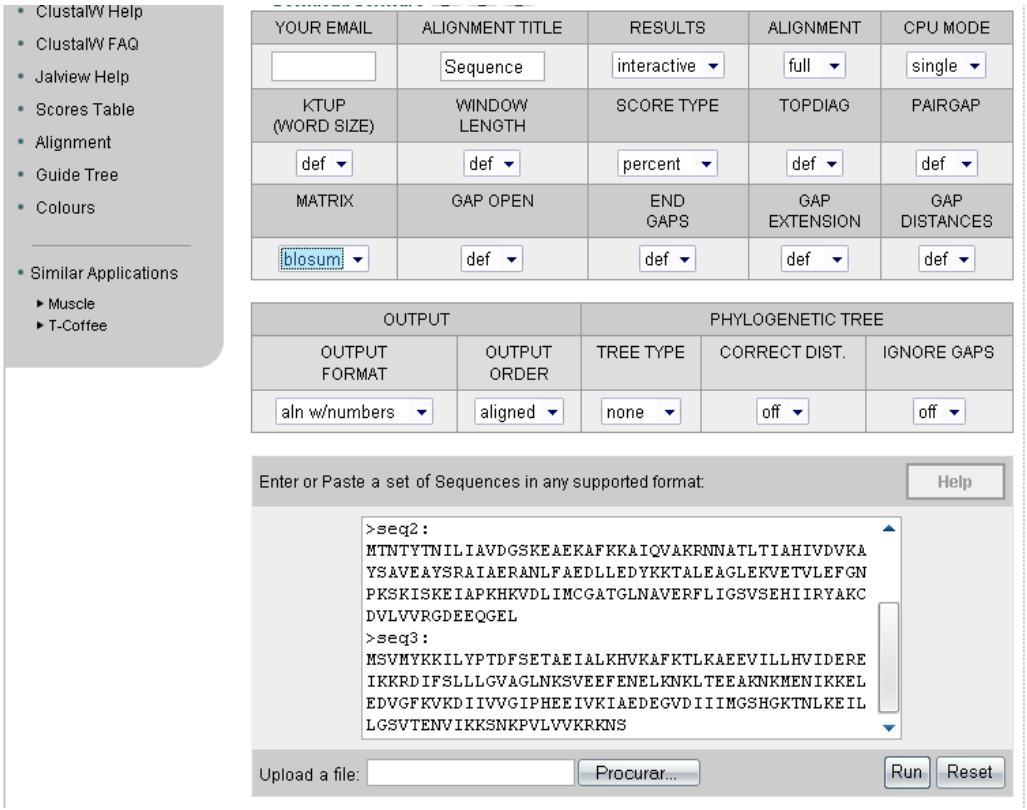


Figura 4.4: Tela inicial da ferramenta ClustalW

O ClustalW alinha as seqüências de forma múltipla e retorna seu alinhamento bem como seu score. Isto pode ser observado nas figuras 4.5 e 4.6.

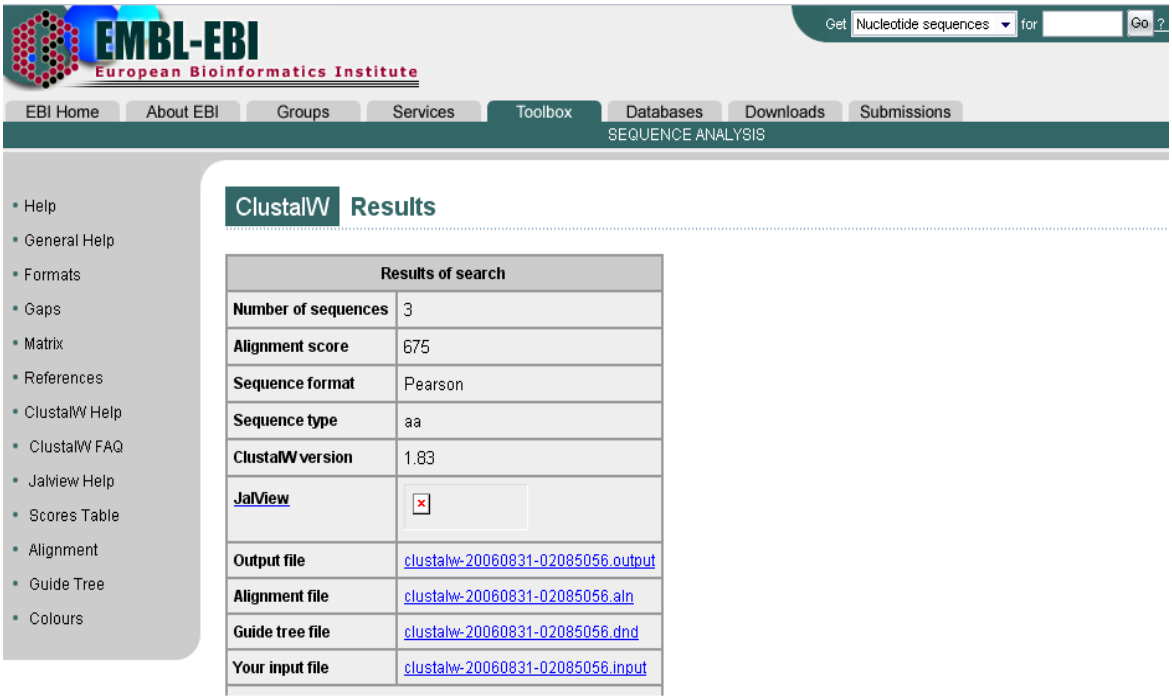


Figura 4.5: Resultados obtidos pelo alinhamento múltiplo

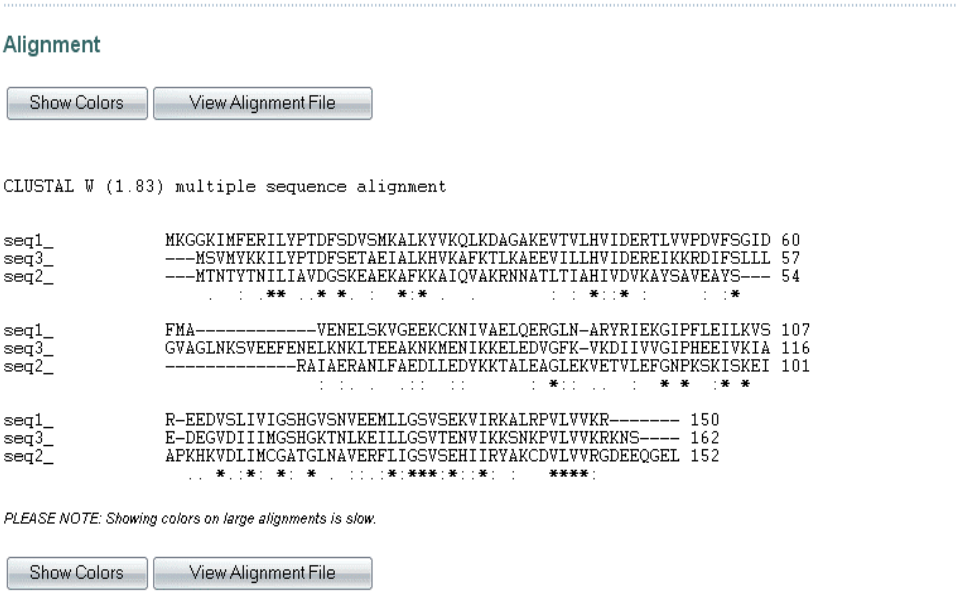


Figura 4.6: Seqüências alinhadas

4.1.Comparação dos Resultados

Nesta seção as seqüências serão apresentadas em ordem de similaridade. A primeira seqüência (linha 1) possui o maior score entre todas as seqüências retornadas pelo Blast (*score* = 261) (*universal stress protein family* [*Syntrophus aciditrophicus SB*), a seqüência 2 (segunda linha) é uma seqüência de *score* intermediário (score = 108) (*Methanocaldococcus jannaschii DSM 2661, complete genome*), e a terceira e última seqüência é uma seqüência de *score* baixo (*score* = 45.4) (terceira linha) (*Bacillus cereus subsp. cytotoxis NVH 391-98*).

Resultados obtidos pelo alinhamento múltiplo no programa ClustalW (Figura 4.7):

```
Linha 1: MKGGKIMFERILYPTDFSDVSMKALKYVKQLKDAGAKEVTVLHVIDERTLVVPDVFSGID
Linha 2: ---MSVMYKKILYPTDFSETAEIALKHVKAFKTLKAEEVILLHVIDEREIKKRDIFSLLL
Linha 3: ---MTNTYTNILIAVDGSKEAEKAFKKAIQVAKRNNATLTIAHIVDVKAYSAVEAYS---

Cont. 1: FMA------------VENELSKVGEEKCKNIVAELQERGLN-ARYRIEKGIPFLEILKVS
Cont. 2: GVAGLNKSVEEFENELKNKLTEEAKNKMENIKKELEDVGFK-VKDIIVVGIPHEEIVKIA
Cont. 3: -------------RAIAERANLFAEDLLEDYKKTALEAGLEKVETVLEFGNPKSKISKEI

Cont. 1: R-EEDVSLIVIGSHGVSNVEEMLLGSVSEKVIRKALRPVLVVKR-------
Cont. 2: E-DEGVDIIIMGSHGKTNLKEILLGSVTENVIKKSNKPVLVVKRKNS----
Cont. 3: APKHKVDLIMCGATGLNAVERFLIGSVSEHIIRYAKCDVLVVRGDEEQGEL
```

Figura 4.7: Alinhamento obtido pela ferramenta ClustalW

Resultados obtidos pelo alinhamento pareado por *profile* com clusterização via programação dinâmica (Figura 4.8):

```
Linha 1: MKGGKIMFERILYPTDFSDVSMKALKYVKQL-KDAGAKEVTVLHVIDERTLVVPDVFS--
Linha 2: MS--V-MYKKILYPTDFSETAEIALKHVKAF-KTLKAEEVILLHVIDEREIKKRDIFSLL
Linha 3: MT--N-TYTNILIAVDGSKEAEKAFKKAIQVAKRNNA-TLTIAHIVDVKAYSAVEAYS--

Cont. 1: -GID--FMAV---ENELSK-VGEE-KCK--NIVAELQERGL-NARYRIEKGIPFLEILK-
Cont. 2: LGVAGLNKSVEEFENELKNKLTEEAKNKMENIKKELEDVGF-KVKDIIVVGIPHEEIVK-
Cont. 3: RAIA--ERA-----NLFAEDLLED--YK--K-TA-L-EAGLEKVETVLEFGNPKSKISKE

Cont. 1: VSREEDVSLIVIGSHGVSNVEEMLLGSVSEKVIRKALRPVLVVK----R---
Cont. 2: IAEDEGVDIIIMGSHGKTNLKEILLGSVTENVIKKSNKPVLVVK---RKNS-
Cont. 3: IAPKHKVDLIMCGATGLNAVERFLIGSVSEHIIRYAKCDVLVVRGDEEQGEL
```

Figura 4.8: Alinhamento obtido pelo programa desenvolvido neste trabalho

Pode ser observado diferenças entre os alinhamentos obtidos pela ferramenta ClustalW e os alinhamento obtidos pelo programa em questão. Essas diferenças se devem a vários fatores tais como, diferença na matriz de pontuação, diferentes *tracebacks* possíveis, dentre vários outros métodos estatísticos aplicados na ferramenta ClustalW para assim ter seu alinhamento refinado.

Mas como visto anteriormente, em um alinhamento múltiplo a região importante para análise biológica são os motifs. Analisando os resultados de forma mais qualitativa, o método utilizado para se obter o alinhamento e os alinhamentos obtidos foram satisfatórios pela grande semelhança em porções das *strings* alinhadas tanto no ClustalW quanto no alinhamento obtido via Programação Dinâmica utilizando Clusterização e *Profile*.

Podemos observar isto na Figura 4.9.

Trecho 1 (ClustalW):

```
MFERILYPTDFSDVSMKALKYVKQL
MYKKILYPTDFSETAEIALKHVKAF
TYTNILIAVDGSKEAEKAFKKAIQV
```

Trecho 1 (Programa Desenvolvido):

```
MFERILYPTDFSDVSMKALKYVKQL
MYKKILYPTDFSETAEIALKHVKAF
TYTNILIAVDGSKEAEKAFKKAIQV
```

Trecho 2 (ClustalW):

```
KDAGAKEVTVLHVIDERTLVVPDVFS
KTLKAEEVILLHVIDEREIKKRDIFS
AKRNNATLTIAHIVDVKAYSAVEAYS
```

Trecho 2 (Programa Desenvolvido):

```
KDAGAKEVTVLHVIDERTLVVPDVFS
KTLKAEEVILLHVIDEREIKKRDIFS
KRNNA-TLTIAHIVDVKAYSAVEAYS
```

Trecho 3 (ClustalW):

```
---
EEF
---
```

Trecho 3 (Programa Desenvolvido):

```
---
EEF
---
```

Trecho 4 (ClustalW):

```
ARYRIEKGIPFLEILK
VKDIIVVGIPHEEIVK
VETVLEFGNPKSKISK
```

Trecho 4 (Programa Desenvolvido):

```
ARYRIEKGIPFLEILK
VKDIIVVGIPHEEIVK
VETVLEFGNPKSKISK
```

Trecho 5 (ClustalW):

```
EEDVSLIVIGSHGVSNVEEMLLGSVSEKVIRKALRPVLVVKR DEGVDIIIMGSHGKTNLKEILLGSVTENVIKKSNKPVLVVKR
KHKVDLIMCGATGLNAVERFLIGSVSEHIIRYAKCDVLVVRG
```

Trecho 5 (Programa Desenvolvido):

```
EEDVSLIVIGSHGVSNVEEMLLGSVSEKVIRKALRPVLVVKR DEGVDIIIMGSHGKTNLKEILLGSVTENVIKKSNKPVLVVKR
KHKVDLIMCGATGLNAVERFLIGSVSEHIIRYAKCDVLVVRG
```

Figura 4.9: Comparação entre alinhamento obtido no ClustalW e no programa desenvolvido para diferentes seqüências. Trechos equivalentes ao alinhamento obtido na ferramenta ClustalW e o Programa Desenvolvido para as demonstradas nas Figuras 4.8 e 4.9

Analisando por este ponto de vista, há uma grande possibilidade entre as similaridades encontradas pelos dois métodos nas seqüências analisadas serem regiões onde se encontram motifs. Isto significa que tanto o alinhamento obtido pelo ClustalW quanto o alinhamento obtido no programa desenvolvido neste projeto foram iguais ou na pior das hipóteses com mínimas diferenças, onde com a intervenção de um técnico humano (pela procura das similaridades existentes) poderiam ser solucionadas.

6.CONCLUSÃO

A utilização de técnicas de alinhamento é importante para se conhecer novos membros de uma determinada família, sua história evolutiva bem como no processo de seqüenciamento de DNA.

Um problema grave encontrado é que com a utilização da Programação Dinâmica para se obter um alinhamento ótimo leva a um grande uso de CPU, prejudicando assim seu desempenho. Isto pode ser solucionado utilizando métodos aproximativos ou heurísticas, mas isto leva uma perda de qualidade na solução, ou então uma solução para estes problemas é o uso de programação paralela.

Com a utilização de representação de seqüências via *profile* ao invés de consenso também garante uma melhor representação destas seqüências alinhadas, garantindo assim um melhor alinhamento final.

6.1.Trabalhos Futuros

Uma proposta para trabalhos futuros consiste na implementação paralela do problema de alinhamento múltiplo global via programação dinâmica utilizando *profile* e clusterização utilizando MPI. Este projeto já se encontra em andamento.

7.BIBLIOGRAFIA

[1]M. Napoli, Introdução à Bioinformática, Universidade Federal de Goiás, 2003

[2].M. McLure, T. Vasi, and W. Fitch. Comparative analysis of multiple protein sequence alignment methods. *Mol. Biol. Evolution,* 11:571-92, 1994

[3]D. C. Oliveira, Alinhamento de Seqüências, Universidade Federal de Lavras, 2002. Monografia final de Curso.

[4]Cormen, Thomas H. ... [et al.]. Algoritmos: teoria e pratica: Rio de Janeiro: Campus, 2002.

[5] S. Chan, A. Wong, D. Chiu. A Survey of Multiple Sequence Comparison Methods. Bolletin of Mathetical Biology, 1992.

[6] Gusfield, Dan. Algorithms on *Strings*, Trees, and Sequences – Computer Science and Computational Biology. Cambrigde University Prees, 1997.

[7] Rocha, Eduardo. Módulo de BioInformática Análise de seqüências Cadeira de Algorítmica e Programação, Atelier de BioInformatique, U. Paris 6 & Institut Pasteur, Paris.

[8] Furtado Ferreira, Daniel – Apostila de Análise Multivariada – Lavras 1996.

[9] http://www.ebi.ac.uk/clustalw/

[10] http://130.14.29.110/BLAST/

[11] Campos, Magnólia de Araújo, et al - Bioinformática: do Seqüenciamento a Função Biológica – Lavras 2006.

[12] Johnson, Richard A. e Wichern, Dean W. Applied Multivariate Statistical Analysis, 2a edition, Prentice Hall International, Inc. Cap.12 Clustering, seção 12.4 - Nonhierarchical Clustering Methods K-means Method eh o das medias.