Padrões em Biossequências

Melissa Lemos
e-mail: melissa@inf.puc-rio.br

Marcus V. S. Poggi de Aragão
e-mail: poggi@inf.puc-rio.br

Marco Antônio Casanova
e-mail: casanova@inf.puc-rio.br

PUC-RioInf.MCC17/03 Junho, 2003

Abstract: The goal of this work is to investigate algorithms for the automatic discovery of patterns in a set of biosequences, one of the fundamental problems in Bioinformatics. Such algorithms find applications in areas such as multiple biosequence alignment, protein structure and function prediction, characterization of protein families, promoter signal detection.
Useful patterns typically correspond to functionally or structurally important elements regions in proteins or DNA sequences. The underlying assumption is that these important regions are better conserved in evolution since they are important to the structure or function of the molecule. The discovery of such patterns may help understand the relationships between biosequence, structure and function of proteins and in, a larger context, it may help interpret the working of living organisms.

Keywords: Pattern Discovery Algorithm, Bioinformatics, DNA, Protein, Genome.

Resumo: O objetivo deste trabalho é apresentar algoritmos conhecidos para descoberta automática de padrões em um conjunto de biossequências, um dos problemas fundamentais em bioinformática. Estes algoritmos podem ser usados em alinhamento de múltiplas biossequências, predição de estrutura e função de proteínas, caracterização de famílias de proteínas, detecção de sinais de promotores.
Os padrões geralmente correspondem a elementos importantes para a estrutura e função das sequências de DNA e de proteínas. Existem suposições de que estas regiões são melhores conservadas durante a evolução porque são importantes para a estrutura ou função da molécula. A descoberta de tais padrões pode ajudar no entendimento dos relacionamentos entre as biossequências, na estrutura e função de proteínas e além disso, na interpretação da atividade dos organismos vivos.

Palavras-chave: Algoritmo de descoberta de padrões, Bioinformática, DNA, Proteína, Genoma.

Sumário

1 Preliminares

Este trabalho tem como objetivo a apresentação de um levantamento sobre descoberta de padrões biológicos.

Primeiramente será apresentado o contexto biológico onde estão as definições, os conceitos e o interesse biológico na análise de padrões.

Depois será descrito o contexto computacional onde estão definições importantes para o entendimento dos próximos capítulos. A seguir estão os principais algoritmos de acordo com os métodos que utilizam, que são: busca exaustiva, métodos heurísticos iterativos, métodos de *machine learning* e métodos que utilizam informações adicionais às sequências biológicas.

Entre os algoritmos descritos, estão os mais utilizados atualmente pelos biólogos, como Pratt, Teiresias e o que baseia-se no modelo Hidden Markov.

1.1 Contexto Biológico

As macromoléculas biológicas, DNA, RNA e proteínas, são cadeias de moléculas orgânicas relativamente pequenas. Existem poucos tipos diferentes destas moléculas – 4 bases para o DNA e o RNA e 20 aminoácidos para proteínas. Uma macromolécula pode ser codificada por uma cadeia em um alfabeto de comprimento 4 (para DNA e RNA) ou 20 (para proteínas). As cadeias para as moléculas de DNA/RNA são chamadas de sequências de nucleotídeos e cada elemento nesta sequência é chamado de base. Analogamente, as cadeias que representam as proteínas são chamadas de sequências de proteínas, e cada elemento desta sequência é chamado de aminoácido ou resíduo. As sequências de nucleotídeos e proteínas são chamadas de biossequências ou, simplesmente, de sequências [EJT00].

O DNA é o principal armazenador da informação genética. Esta informação é copiada ou *transcrita* para moléculas de RNA, cujas sequências de nucleotídeos contêm o "código" para a ordenação específica de aminoácidos. As proteínas são então sintetizadas num processo que envolve a *tradução* do RNA (veja a Figura 1). A série de eventos acima relacionada é conhecida como o *dogma central da biologia molecular*; ela pode ser resumida da forma apresentada na Figura 1 [Rob85].

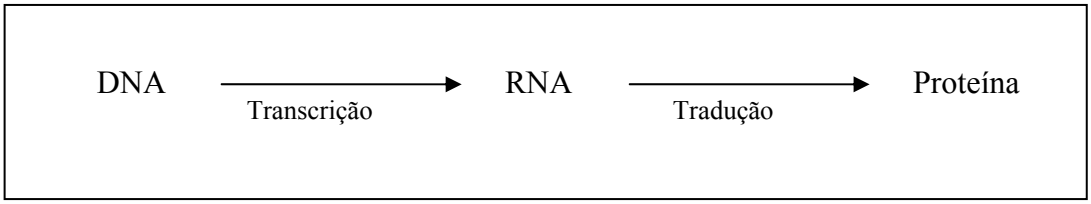


Figura 1. Transcrição e Tradução.

Quando um gene se *expressa*, sua informação é primeiramente copiada no RNA, que por sua vez dirige a síntese dos produtos elementares dos genes, as proteínas específicas. O termo *transcrição* é empregado como sinônimo de síntese de RNA e *translação* como sinônimo de síntese protéica [Rob85].

Os *códons*, ou unidades hereditárias que contém o código de informação para um aminoácido, são compostos por três nucleotídeos (um trio). Esta informação encontra-se no DNA, de onde é transcrita para o *RNA mensageiro* ou mRNA. Assim, o mRNA possui a sequência de bases complementar à do DNA do qual foi copiado. Dessa maneira, o código é lido em grupos de três bases, sendo três o número mínimo necessário para a codificação de 20 aminoácidos [Rob85].

Observa-se que a informação genética contida no DNA não pode agir diretamente como molde para a síntese de proteínas, precisando ser primeiramente transcrita no RNA mensageiro. O RNA é sintetizado pela enzima *RNA polimerase*. Esta enzima copia a sequência de bases de uma das cadeias do DNA, de acordo com as regras do pareamento de bases de Watson-Crick. Além de copiar precisamente a sequência de nucleotídeos do molde de DNA, a RNA polimerase é capaz de reconhecer diversos sinais genéticos no cromossomo, como os responsáveis pelo início e terminação da síntese do RNA em locais precisos. Os sinais de iniciação do DNA são chamados *promotores* e representam o sítio inicial de ligação da RNA polimerase. Esta ligação é um estágio crucial na regulação da expressão do gene. Por exemplo, todos os promotores de organismos procarióticos possuem a sequência comum TATAATG (ou pequenas variações dela) localizada em torno de 10 nucleotídeos antes da extremidade final do mRNA. Esta região rica em AT provavelmente favorece a separação local das cadeias do DNA, um estágio necessário para a RNA polimerase obter acesso às bases do DNA. Além da sequência TATAATG, a enzima também reconhece uma região do DNA localizada aproximadamente 35 bases antes do início do mRNA. Pequenas alterações nestas bases podem desativar os promotores; porém diferentes promotores não possuem sequências comuns evidentes nesta região. Não surpreendem as

variações individuais entre os promotores, pois genes diferentes são expressos com eficiência variável; isto é, alguns promotores são "mais fortes" que outros. Por exemplo, a *E.coli* possui proteínas chamadas *repressoras* que podem se ligar à sequências específicas de DNA (denominadas *operadoras*), inibindo a expressão de um gene ou conjunto de genes [Rob85].

O comprimento da porção codificadora de um gene depende da extensão da mensagem a ser traduzida, isto é, o número de aminoácidos da proteína. Por exemplo, uma sequência de 1.500 nucleotídeos pode conter 500 códons que codificam para uma proteína que contém 500 aminoácidos. A mensagem é lida a partir de um ponto inicial fixo sinalizado por *códons de iniciação* especiais e finalizada em um ponto final sinalizado por *códons de terminação* [Rob85].

A sequência de trios determina a sequência dos aminoácidos de uma proteína. Os aminoácidos, no entanto, não são capazes de reconhecer por si sós um dado trio do mRNA; para que isso aconteça, cada aminoácido precisa se ligar a uma molécula adaptadora denominada *RNA de transferência* (tRNA). Cada molécula de tRNA possui um sítio de ligação do aminoácido e um outro local para o reconhecimento dos trios do mRNA. Este último sítio é denominado de *anti-códon* e consiste em três nucleotídeos que podem estabelecer um pareamento de bases com o códon complementar do mRNA [Rob85].

Por volta de 1964, todos os 64 códons possíveis haviam sido decifrados, sendo que 61 códons correspondem a aminoácidos e 3 representam sinais para a terminação das cadeias polipeptídicas. Sabendo que existem somente 20 aminoácidos, fica evidente que vários trios podem codificar o mesmo aminoácido; isto é, alguns dos trios são sinônimos. A prolina, por exemplo, é codificada por CCU, CCA, CCG e CCC [Rob85].

Inesperadamente, observou-se que, nos eucariontes, a informação para mRNAs covalentemente contíguos está frequentemente localizada em segmentos de DNA não contíguos. Em outras palavras, os genes são interrompidos por inserções de DNA não-codificador. Estas sequências de DNA inseridas, que não são encontradas no mRNA maduro, são denominadas sequências intercaladas ou *íntrons*. Foram encontrados íntrons em vários genes, mas nem todos os genes são interrompidos.

Simplificadamente, é possível resumir este assunto da seguinte maneira. Todas as células possuem DNA e os genes podem estar ativos, inativos ou em um estágio entre ativo e inativo. Existem três etapas em que é possível regular um gene: transcrição,

tradução e pós-tradução (veja a Figura 2). Todos os fatores envolvidos nestas etapas são fontes de pesquisa biológica. Tanto as regiões codificantes do DNA (que produzirão proteínas), quanto as regiões não codificantes, são importantes e consequentemente, são estudadas. Como exemplos de regiões não codificantes, tem-se os íntrons e as proteínas ou enzimas que podem ativar ou desativar a RNA polimerase tornando possível ou não a etapa de transcrição. Estas proteínas ou enzimas são chamadas de *fatores de transcrição* e o locais onde elas estão no DNA são chamados de *sítios de ligação dos fatores de transcrição*.

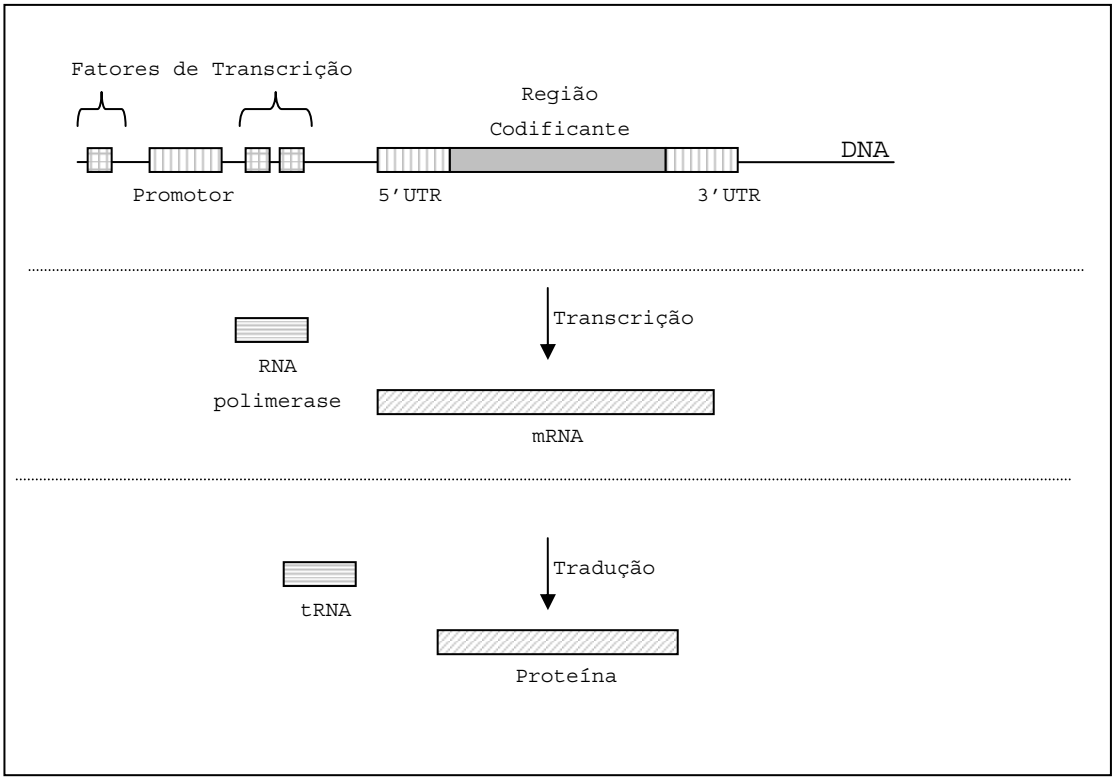


Figura 2. DNA e Proteína.

Atualmente, um número considerável de biossequências já foram identificadas e já existem diversos bancos de dados armazenando tais informações. O Genbank e o EMBL são exemplos de bancos de dados de sequências de nucleotídeos e o SWISS-PROT, de sequências de proteínas. Além disso, existem também os bancos de dados da estrutura tridimensional das proteínas, o PDB é um exemplo [EJT00].

O maior problema hoje em dia não é mais obter as sequências, mas sim analisá-las. O principal objetivo desta análise é extrair todo tipo de significado biológico destas biossêquencias [EJT00].

Como foi ressaltado é importante estudar tanto a região codificante quanto a não codificante. Todos estes dois pontos podem ser analisados através de descoberta de padrões.

Por exemplo, um modo de analisar as sequências de proteínas é agrupá-las em *famílias*, sendo cada família um conjunto de sequências que se acredita estarem relacionadas biologicamente (isto é, por sua evolução, estrutura ou função). Para cada família, tenta-se descobrir as características comuns que podem ser expressas puramente em termos de suas sequências. Estas descrições de características "sintáticas" comuns são chamadas de *padrões* [EJT00].

A terminologia utilizada na literatura para descrever similaridades comuns entre sequências não é bem estabelecida. As palavras *pattern, motif, fingerprint, template, fragment, core* e *site* são utilizadas. Por um padrão entende-se aqui qualquer descrição de características da sequência tal que (1) permite decidir se uma sequência se casa ou não com ele; ou (2) permite atribuir um valor ao casamento da proteína com o padrão. No primeiro caso, diz-se que o padrão é *determinístico* e, no segundo caso, diz-se que o padrão é *probabilístico* [EJT00].

Existem diversos métodos de organização das proteínas em famílias baseados na presença de sequências comuns. Os membros de uma mesma família de proteína são frequentemente caracterizados por mais de um padrão. As proteínas são atribuídas às famílias de acordo com homologias determinadas por alinhamento de sequências. Se um alinhamento descobrir homologia entre uma sequência de proteína e uma família de proteínas, um relacionamento filogenético entre elas é assumido automaticamente [BPG96]. Mas isto não é totalmente seguro, já que similaridades significativas entre sequências nem sempre indicam relacionamentos evolutivos e, além da homologia entre as sequências, as proteínas também podem se relacionar por similaridade estrutural, o que pode não ser descoberto através de alinhamento. Uma idéia seria levar em consideração outras informações na classificação de proteínas que são altamente divergentes no alinhamento, mas que são equivalente funcionalmente [BDV+00]. É por isso que as proteínas são organizadas em árvores hierárquicas que revelam seus relacionamentos funcionais e evolutivos [BDV+00].

A identificação de padrões conservados entre sequências durante a evolução pode mostrar que elas estão relacionadas funcionalmente e estruturalmente. O primeiro passo para a predição da função da proteína é procurar por características na sequência que são comuns a grupos de proteínas com alguma atividade específica e

que estão ausentes de grupos de proteínas que não possuem tal atividade. Estes métodos para descobrir a função em proteínas, através do reconhecimento de padrões, baseiam-se na premissa de que as sequências altamente conservadas (onde conservação indica a estabilidade da sequência ao longo da evolução) são preservadas através da evolução porque elas são importantes para a função ou estrutura da proteína. Intuitivamente isto parece uma hipótese válida, mas é possível que algumas sequências conservadas correspondam simplesmente a regiões com baixa taxa de mutação[BDV+00].

Apesar de parecer que alguns padrões funcionais não estão presentes na sequência da proteína, por consistirem de resíduos separados por regiões longas e variáveis, tais resíduos conservados podem estar formando um grupo funcional quando a proteína é colocada em sua estrutura tridimensional [Cal00]. Além disso, padrões de sequências conservados podem destacar elementos que são responsáveis pela similaridade estrutural entre proteínas e podem ser utilizados para predizer a estrutura tridimensional da proteína [BDV+00].

Como alguns aminoácidos possuem características similares, tais como tamanho e carga, as substituições são frequentemente permitidas em padrões de proteínas. Em [NWB98] há a descrição de um método para caracterizar famílias de proteínas através de padrões conservados, que é um pouco flexível na posição específica de cada aminoácido. Os grupos de aminoácidos que ocorrem em determinada posição com uma frequência significativa são identificados e utilizados para caracterizar subconjuntos de padrões que são biologicamente relevantes. Enquanto um padrão pode ser sensível ao ponto de identificar um novo membro da família com um número mínimo de negativos falsos (sequências que pertencem a família e que não são identificadas pelo padrão), o padrão pode levar também a sequências positivas falsas (sequências que não pertencem a família e que são identificadas pelo padrão) [BDV+00].

Alguns dicionários biológicos de proteínas foram construídos, como o PROSITE [Pro02] e o PRINTS [Pri02], e podem ser utilizados para a predição de função de proteínas novas ou desconhecidas [BDV+00]. O PROSITE é uma fonte de dados que armazena uma coleção de famílias de proteínas com suas anotações [PRO02a][BJE+97].

Além dos padrões das proteínas, também são estudados os padrões em sequências de nucleotídeos, destacando-se os promotores, os códons que especificam as sequências

de aminoácidos de uma proteína, os sítios de íntrons e os sítios de ligação das proteínas regulatórias que ativam ou reprimem a transcrição [Mou99].

Geralmente as regiões não codificantes não são conservadas. Consequentemente a presença de regiões conservadas antes de uma região codificante pode indicar alguma funcionalidade importante [BDV+00].

Encontrar todos os promotores de um determinado genoma também é uma tarefa importante, mas muito difícil. Para descobrí-los em sequências genômicas grandes é necessário identificar as características que são comuns às sequências de promotores e que não são comuns às sequências que não são promotores. Esta é uma tarefa complicada, especialmente em eucariontes que estão associados a vários promotores e que possuem inúmeros fatores regulatórios. Frequentemente, o objetivo não é somente identificar os promotores, mas entender como os genes sob o controle destes promotores são regulados. Isto envolve a identificação de sequências regulatórias específicas em regiões onde os promotores estão localizados, por exemplo, aquelas sequências que se unem a fatores de transcrição específicos [BDV+00].

Um outro caso interessante é a descoberta de padrões para a detecção de repetições *tandem*. Estas repetições são duas ou mais cópias contíguas aproximadas de sequências de nucleotídeos. Estas duplicações ocorrem devido a eventos de mutação no qual um segmento original de DNA, o padrão, é convertido dentro da sequência em cópias individuais. Com o passar do tempo estas cópias podem sofrer outras mutações sem coordenação, o que leva as cópias idênticas do padrão original a se tornarem variações aproximadas dele [BDV+00].

As repetições *tandem* são muito comuns em sequências genômicas. [Ben99] notou que elas estão presentes em provavelmente 10% ou mais do genoma humano, que estão relacionadas com diversas doenças e que parecem possuir um papel importante na regulação do gene. Mas a descoberta de suas propriedades é limitada devido à dificuldade em detectá-las [BDV+00].

1.2 Contexto Computacional

Neste trabalho serão discutidos os algoritmos que descobrem padrões desconhecidos em sequências, problema que será chamado aqui de *descoberta de padrões*. No entanto, em Biologia, muitas sequências consenso são conhecidas e é importante para os biólogos terem ferramentas para encontrá-las em novas sequências. Este problema será chamado de *casamento de padrões*. Os programas para casamento de padrões

podem ser gerais, isto é, recebem como entrada algum padrão para ser encontrado em um conjunto de sequências ou podem ser específicos, isto é, encontram um tipo particular de padrão em algum conjunto de sequências. Geralmente estes programas são refinados para obter a melhor sensibilidade e especificidade.

Por considerar o problema de descoberta de padrões computacionalmente mais interessante do que o de casamento de padrões, este trabalho apresentará os métodos existentes para o primeiro tipo de problema, embora os métodos para o segundo problema sejam muito importantes para os biólogos.

1.2.1 A Ciência da Computação e os Padrões das Biossequências

Um dos objetivos de descoberta de padrões em Biologia é o de classificação. Isto ocorre, por exemplo, nas famílias de proteínas. Os padrões são considerados *padrões classificadores*, se e somente se, dada uma proteína desconhecida, ela irá ser classificada como membro de uma família, se tiver os padrões daquela família. É possível formular esta questão como um problema de *machine learning*: dado um conjunto de sequências que pertencem à família (exemplos positivos) e um conjunto de sequências que não pertencem à família (exemplos negativos), constrói-se uma função f que, para cada proteína decida se ela pertence ou não à família. Neste contexto de descoberta de padrões, as classes de funções f que se casam com as sequências desconhecidas são mais interessantes. Note que os exemplos negativos são simplesmente outras proteínas conhecidas obtidas de bancos de dados de proteínas como o SWISSPROT. Geralmente os métodos começam suas análises com os exemplos positivos e usam os exemplos negativos somente para avaliar seus classificadores. Uma descrição detalhada de possíveis formalizações de descoberta de padrões para classificação pode ser encontrada em [BJE+98].

Além da classificação, há o objetivo de descoberta de *padrões significativos*. Como, por exemplo, encontrar um elemento regulatório em um conjunto de regiões onde ele provavelmente está presente. Isto não significa que este elemento não esteja presente em outros lugares do genoma ou que todas as sequências dadas devam conter tal elemento. Outro exemplo se encontra no contexto de famílas de proteínas. É interessante encontrar elementos estruturais ou funcionais importantes, independentemente deles terem especificidade suficiente para distinguir uma família de outra [BDV+00].

Geralmente para resolver esta questão, define-se uma classe de padrões que se deseja procurar e descobre-se o padrão com maior pontuação que tem *aprovação* suficiente. Os métodos diferem na forma de calcular a pontuação e na definição de aprovação. Frequentemente aprovação indica o número de vezes em que um padrão ocorre nas sequências. O usuário pode exigir que um padrão ocorra em todas as sequências ou determinar um número mínimo de ocorrências. Em alguns casos, o número mínimo de ocorrências não é especificado pelo usuário, mas faz parte de uma função de pontuação. Por exemplo, algumas vezes os padrões longos com poucas ocorrências são mais interessantes que os padrões pequenos com muitas ocorrências [BDV+00].

A situação torna-se mais complicada no caso dos *padrões probabilísticos*, tais como os modelos Hidden Markov. Os *padrões determinísticos* casam-se ou não com uma sequência (0 ou 1), enquanto os probabilísticos dão uma probabilidade de casamento entre 0 e 1. Consequentemente, existem diferentes níveis de casamento. É necessário configurar algum limiar para um casamento ser considerado relevante ou incluir na pontuação do padrão as probabilidades de casamento [BDV+00].

Os métodos para pontuação de padrões diferem muito. A pontuação pode descrever o próprio padrão (como seu comprimento) ou pode ser baseada nas ocorrências do padrão [BDV+00].

O objetivo dos algoritmos pode ser encontrar o melhor padrão (aquele com maior pontuação), encontrar os melhores padrões (aqueles que tiverem pontuações mais altas) ou encontrar todos os padrões com algum nível pré-definido de aprovação e pontuação [BDV+00].

1.2.2 Sequências de Entrada

Geralmente a entrada dos métodos de descoberta de padrões consiste em um conjunto de sequências para as quais se espera encontrar algum padrão. O símbolo $\sum$ será usado como o alfabeto de todos os possíveis caracteres que ocorrem em sequências, isto é, o conjunto {A,T,C,G} para DNA e os 20 caracteres que representam os aminoácidos para as proteínas.

Alguns algoritmos usam outras informações além das sequências como, por exemplo, informações da estrutura secundária ou terciária das sequências, relacionamentos evolutivos entre as sequências, etc [BDV+00].

1.2.3 Tipos de Padrões

Em um nível mais alto de abstração, os padrões podem ser classificados em dois tipos: determinístico e probabilístico. Como foi dito anteriormente, um padrão determinístico pode se casar ou não com determinada sequência, enquanto, geralmente, os padrões probabilísticos são modelados por modelos que dão a probabilidade de uma sequência ser gerada pelo modelo. Quanto maior a probabilidade, melhor é o casamento entre a sequência e o padrão [BDV+00].

1.2.3.1 Padrões Determinísticos

O tipo mais simples de padrão é uma sequência de caracteres de ∑, como TATAAA, a sequência consenso do box TATA. Mas também é possível a ocorrência de padrões mais complexos, com as seguintes características [BDV+00]:

- *Caracter Ambíguo*: é um caracter que se casa com algum caracter pertencente a um subconjunto de ∑. Tais subconjuntos são indicados por uma lista com seus elementos entre colchetes. Por exemplo, [LF] é um conjunto que contém os elementos L e F. A-[LF]-G é uma padrão na notação usada pelo banco de dados PROSITE. Este padrão casa-se com subsequências que começam com A, terminam com G e tem L ou F no meio.
- *Caracter Coringa*: é um tipo especial de caracter que se casa com qualquer caracter de ∑. Coringas são representados por N em sequências de nucleotídeos e por X em proteínas. Frequentemente usa-se “.”. A sequência de um ou mais coringas é chamada de *buraco*. Um buraco pode ter comprimento variável. No banco de dados PROSITE este tipo de buraco é representado por x(i,j), onde i é o comprimento mínimo do buraco e j o comprimento máximo. Por exemplo, x(4,6) casa-se com qualquer buraco de comprimento 4, 5 ou 6. No PROSITE também é usado x(i) para representar um buraco com comprimento fixo e * para buraco com qualquer comprimento, inclusive 0.

A cadeia F-x(5)-G-x(2,4)-G-*-H é um exemplo de um padrão complexo no PROSITE. Geralmente os programas não aceitam todas estas características.

É possível aumentar o poder de expressão dos padrões determinísticos permitindo alguns descasamentos. O tipo mais comum de diferença é a substituição, mas em alguns casos são permitidas inserções ou remoções. Nestes casos o número de

descasamentos seria a distância entre a subcadeia *S* e uma cadeia que se case melhor com o padrão *P* [BDV+00].

1.2.3.2 Matrizes de Pontuações

Mesmo os padrões determinísticos mais complexos não conseguem expressar determinadas informações. Suponha que exista um padrão que contenha na primeira posição o caracter C em 40% dos casos e G nos outros 60%. O símbolo ambíguo [CG] dá a mesma importância a ambos caracteres [BDV+00].

O tipo mais simples de padrão probabilístico é a matriz de pontuações ou PWM (de *Position Weight Matrices* em inglês). PWM é uma tabela que especifica um padrão sem buracos. Esta tabela possui para cada par (posição, caracter), a frequência relativa do caracter naquela posição do padrão [BDV+00].

Suponha que o padrão tenha comprimento k (número de colunas da tabela). A pontuação de uma sequência $x_1...x_k$ é

$$\prod_{i=1}^{k} \frac{A[x_i, i]}{f(x_i)}$$

onde $A[c,i]$ é uma entrada da matriz que corresponde à frequência relativa do caracter c na posição i do padrão, e $f(c)$ é a frequência do caracter c em todas as sequências consideradas. Esta pontuação é chamada de *odd score*. Para simplificar é possível armazenar na tabela as pontuações *log-odd score*

$$A'[c,i] = \log \frac{A[c,i]}{f(c)},$$

em vez das frequências $A[c,i]$. Então a seguinte fórmula para a pontuação será utilizada

$$\prod_{i=1}^{k} A'[x_i, i],$$

em vez da anterior [BDV+00].

1.2.3.3 Modelos Estocásticos

Todos os tipos de padrões discutidos anteriormente são construídos para que o usuário possa identificar facilmente características importantes de ocorrências de padrões. Algumas vezes é vantajoso representar um padrão de forma mais implícita, geralmente através de uma regra de discriminação, que decide se uma dada sequência é ou não uma ocorrência de um padrão modelado. Tal regra pode ser baseada em

algum modelo estocástico, como o HMM (de *Hidden Markov Model* em inglês), ou pode empregar métodos de *machine learning* como, por exemplo, redes neurais [BDV+00].

As regras podem ser treinadas com o objetivo de formarem um padrão (o que corresponde a descoberta de padrões) e então elas podem ser usadas para discriminação (o que corresponde ao casamento de padrões) [BDV+00].

2 Busca Exaustiva

Muitos problemas de descoberta de padrões são NP-difíceis e por isso não se espera obter um algoritmo rápido que encontre a melhor solução possível. Muitos métodos são baseados na busca exaustiva. Apesar de terem complexidade de tempo exponencial, no pior caso, geralmente os programas usam técnicas sofisticadas de podas ou cortes que tornam possível a busca em dados de entrada típicos [BDV+00].

2.1 Enumeração de Todos os Padrões

O método mais simples de descobrimento de padrões é enumerar todos os padrões que satisfazem às restrições do usuário, descobrir as ocorrências destes padrões nas sequências de entrada e atribuir pontuação ou significado estatístico para cada padrão, baseando-se nas ocorrências encontradas. É possível, por exemplo, gerar como saída todos os padrões com pontuações acima de um determinado valor [BDV+00].

Por exemplo, suponha o caso em que se queira descobrir o padrão de nucleotídeos mais significativo de comprimento 10, permitindo 2 descasamentos no máximo. Enumeram-se todas as cadeias de comprimento 10 do alfabeto {A,C,T,G} (existem 4^{10}=1.048.576 cadeias). Cada cadeia é um padrão em potencial. Descobrem-se todas as ocorrências com no máximo 2 descasamentos em sequências de entrada, calculam-se as pontuações dos padrões e relata-se o padrão com maior pontuação ou os padrões com pontuações acima de um determinado valor [BDV+00].

Este método é cabível somente para padrões pequenos e simples porque seu tempo de execução aumenta exponencialmente com o comprimento do padrão. Além disso o número de possibilidades cresce se forem permitidos caracteres coringas, caracteres ambíguos, buracos, etc. E, ainda, o tempo de execução cresce linearmente com o comprimento das sequências de entrada [BDV+00].

A vantagem deste método é a garantia da descoberta do melhor padrão. É possível também permitir descasamentos, inserções e remoções [BDV+00].

2.1.1 Aplicações do Método de Enumeração

Muitos sítios de ligação de proteínas em DNA são padrões pequenos sem buracos, com uma certa variação. Estes sítios são bem modelados por padrões simples que permitem um número pequeno de descasamentos e é possível utilizar busca exaustiva para encontrá-los. Recentemente estes métodos foram utilizados em [Tom99,VAC98]. Nestes trabalhos foram enumerados todos os padrões possíveis e estimados os

significados estatísticos das ocorrências. [VAC98] tenta encontrar em um conjunto de sequências o padrão que aparece em várias cópias (box GATA). São considerados padrões de 4 a 9 nucleotídeos, não são permitidos descasamentos e são descobertos os padrões que ocorrem mais vezes que os outros, considerando uma distribuição conhecida. [Tom99] permite descasamentos e tenta descobrir o significado estatístico através do número de ocorrências em que o padrão foi encontrado [BDV+00].

2.1.2 Enumeração de Padrões com Buracos

Em alguns casos, é mais razoável buscar padrões com buracos. Um exemplo de um sistema que considera estes tipos de padrões é o MOTIF [SAC90]. O MOTIF descobre padrões com 3 aminoácidos conservados separados por 2 buracos fixos (por exemplo A...Q...I). Os buracos podem ter comprimentos de 0 a *d*, sendo *d* um parâmetro especificado pelo usuário. O MOTIF não permite descasamento, mas não exige que o padrão ocorra em todas as sequências. Se as sequências possuírem mais que 3 posições com regiões conservadas, existirão muitos padrões, cada um contendo um subconjunto diferente de regiões conservadas. Neste caso o algoritmo remove os padrões que ocorrem próximos uns dos outros. Depois, todos os casamentos de um determinado padrão são alinhados e baseado no consenso das colunas deste alinhamento, o padrão é estendido [BDV+00].

2.2 Enumeração de Padrões com Podas

Para descobrir padrões maiores ou mais ambíguos, não é indicado usar a busca exaustiva diretamente. Suponha que se queira encontrar um padrão longo e sem buracos que ocorra, possivelmente com alguns descasamentos, em pelo menos *k* sequências. É possível começar com padrões pequenos (por exemplo padrões de comprimento igual a 1) que aparecem em *k* sequências e estendê-lo enquanto ele ainda esteja de acordo com as *k* sequências. Em cada passo é preciso estender o padrão de todas as maneiras possíveis e checar se o novo padrão ainda ocorre em no mínimo *k* sequências. Uma vez que o padrão não possa mais ser estendido sem deixar de existir nas *k* sequências, ele é considerado o máximo e é dado como saída. Esta estratégia de busca é uma pesquisa em profundidade da árvore de todas as sequências possíveis (veja a Figura 3). Os galhos que não suportam padrões são podados [BDV+00], ou seja, quando se descobre um nó com um padrão que não ocorre em pelo menos k sequências, a busca não continua sendo feita em seus filhos. Os nós com

linhas pontilhadas aparecem em pelo menos k sequências. Os nós da árvore em negrito representam aqueles que não podem ser mais estendidos.

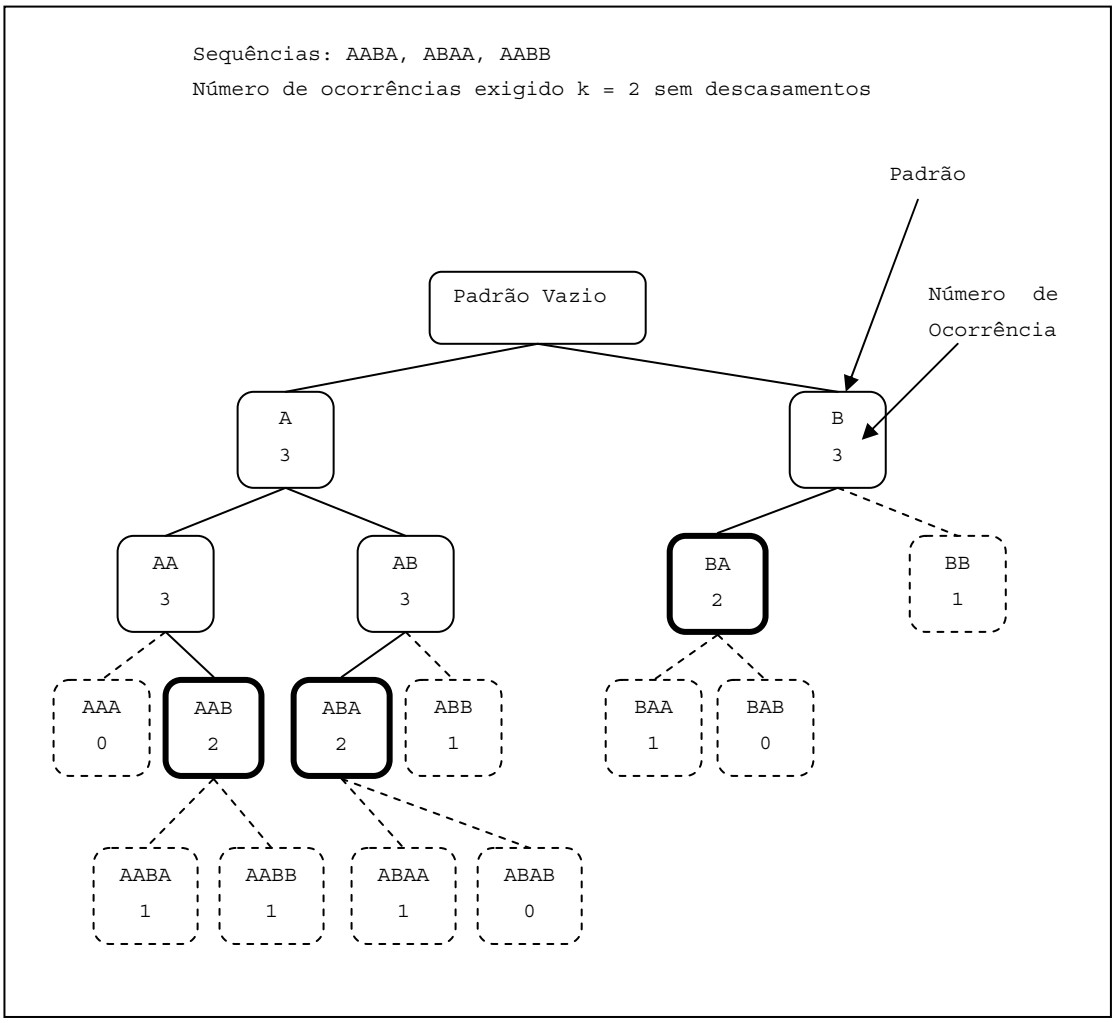


Figura 3. Busca em uma árvore com todos os padrões possíveis.

Este tipo de melhora funciona bem em situações reais, embora o tempo teórico de execução, no pior caso, continue sendo exponencial. A principal vantagem é que eles permitem a busca de padrões maiores e mais complexos do que as buscas exautivas simples. Como exemplos desta estratégia, podem ser citados o algoritmo Pratt [Jon96] e a fase de varredura do algoritmo TEIRESIAS [RF98b]. Ambos encontram padrões com buracos [BDV+00].

2.2.1 Pratt

O Pratt possui atualmente duas versões. A primeira é descrita em [JCH95] e a segunda, em [Jon96]. A descrição da segunda versão será aqui apresentada.

O objetivo deste algoritmo é descobrir os padrões mais interessantes que se casam com um número mínimo de sequências dadas.

O usuário especifica uma classe de padrões definindo restrições quanto à ambiguidade permitida, o comprimento e o número de buracos. O algoritmo explora exaustivamente uma árvore de busca que representa a classe de padrões procurada, e relata os padrões conservados, que são identificados de acordo com uma pontuação baseada em uma função pré-definida. Além disso, ele poda a busca, não explorando extensões dos padrões que não são conservados (se um padrão não é conservado então nenhuma extensão do padrão será conservada). No pior caso, o algoritmo possui tempo exponencial [Jon96].

2.2.1.1 Definições

Os padrões que podem ser encontrados pelo Pratt compõem um subconjunto dos padrões usados pelo PROSITE [BBH95]. Um padrão P pode ser escrito da seguinte maneira:

$$(1)\ P = A_1 - x(i_1,j_1) - A_2 - x(i_2,j_2) - \ldots - x(i_{p-1},j_{p-1}) - A_p,$$

$A_1,\ldots,A_p$ são chamados de *componentes do padrão P* e são subconjuntos de $\sum$. Um componente de padrão é chamado de *identificador* se ele contiver apenas um símbolo, senão ele é chamado de *ambíguo*. Um padrão que contém p componentes é chamado de um *p-padrão*. As expressões $x(i_k,j_k)$ são as *regiões coringas* (buracos) de P, onde $i_1 \leq j_1$, $i_2 \leq j_2,\ldots, i_{p-1} \leq j_{p-1}$ são inteiros positivos.

Uma região coringa é chamada de *fixa* se $i=j$, e *flexível*, se $j > i$. A *flexibilidade* da região coringa é definida como j-i. A representação de uma região coringa é omitida quando $i=j=0$, é 'x' quando $i=j=1$ e é $x(i)$ quando $i=j$.

Como exemplo, 'A-[DE]-x(3)-G-x(3,4)-L' é um padrão com $p=4$, sendo $A_1=\{A\}$, $A_2=\{D,E\}$, $A_3=\{G\}$ e $A_4=\{L\}$, $i_1=j_1=0$, $i_2=j_2=i_3=3$ e $j_3=4$.

O usuário pode definir um conjunto de limites para seus padrões que é definido por uma tupla $B = (A,P,L,W,F,N,FP)$, sendo A o conjunto dos valores permitidos para os componentes de padrões, P o número máximo de componentes, L o comprimento máximo de um padrão, W o comprimento máximo de uma região coringa, F a flexibilidade máxima de uma região coringa, N o número máximo de regiões coringas flexíveis, e FP o produto máximo de flexibilidade.

O conjunto de padrões que satisfazem a expressão (1) e que estão dentro dos limites B definidos pelo usuário é representado por C. Este conjunto é o que deve ser descoberto durante a busca.

Um padrão P_1 é uma *generalização* de um padrão P_2 se qualquer sequência que se casa com P_2 também se casa com P_1; neste caso P_2 é uma *especialização* de P_1.

Dada uma classe C de padrões, uma família de *operadores de transformação* $\xrightarrow{i}_c$, para $i \in \{1,2,3\}$, é definida. Cada operador pode ser aplicado ao padrão P em C, e produzir outro padrão mais geral P' em C. Os operadores são [Jon96]:

1. $P \xrightarrow{1}_c P'$ se P' puder ser obtido através da remoção de um componente c do padrão P;
2. $P \xrightarrow{2}_c P'$ se P' puder ser obtido através da substituição de um componente c em P por um componente menos restritivo $c' \in A$ de tal forma que $c \subset c'$.
3. $P \xrightarrow{3}_c P'$ se P' puder ser obtido permitindo mais flexibilidade em uma das regiões coringas de P.

É definido que $\longrightarrow_c$ como

$$P \longrightarrow_c P' \text{ se e somente se } P \xrightarrow{i}_c P' \text{ para algum } i \in \{1,2,3\},$$

e

$$P \xrightarrow{*}_c P' \text{ se e somente se existirem padrões } P = P_1, ..., P_k = P',$$

para algum $k \geq 1$, tal que $P_1 \longrightarrow_c P_2 \longrightarrow_c ... \longrightarrow_c P_k$.

Se $P \xrightarrow{*}_c P'$, diz-se que P' é uma *C-generalização* de P.

Por exemplo o padrão A-B-C-D pode ser generalizado para [AB]-B-x(1,3)-D usando as seguinte sequência de operadores: A-B-C-D $\xrightarrow{2}_c$ [AB]- B-C-D $\xrightarrow{1}_c$ [AB]- B-x-D $\xrightarrow{3}_c$ [AB]- B-x(1,3)-D.

Para reduzir o tamanho da saída e do espaço de busca, o algoritmo não relata padrões que são menos específicos que outros.

Uma *função de pontuação* de um padrão é representada por $I(P)$ [JCH95]. Ela é usada para classificar os padrões descobertos pelo Pratt de tal forma que os padrões com pontuações mais altas são aqueles que se acredita serem os mais interessantes.

Um *grafo de padrão* é um grafo dirigido $G=(V,E)$ onde os nós V representam os componentes de padrão e os arcos E representam as regiões coringa. Um caminho no grafo define um padrão. Um nó $v \in V$ é rotulado por um componente de padrão $\alpha(v)$ e um arco $e = (u \rightarrow v) \in E$ é rotulado por, no mínimo, $\delta(e)$ e, no máximo, $\delta'(e)$, número de resíduos que se casam com uma região coringa definida pelo arco. Um caminho $p = u_1, u_2 ...., u_n$ em G define o padrão

$$\Pi(p) = \alpha(u_1) - x(\delta(u_1 \rightarrow u_2), \delta'(u_1 \rightarrow u_2)) - \alpha(u_2) - .... - \alpha(u_n).$$

No exemplo da Figura 4, o caminho $p = u,w,x$ define o padrão $\Pi(p)$=A-x(0,1)-C-x(3)-D.

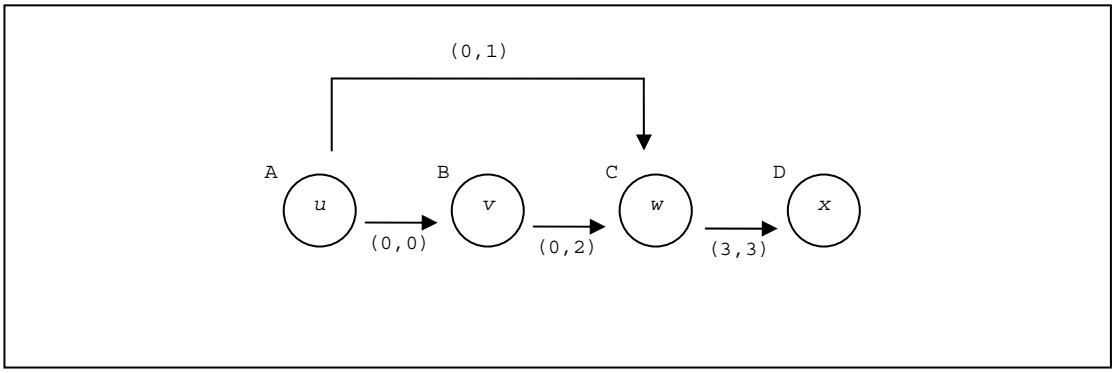


Figura 4. Grafo que define um padrão.

$\Pi(G,C)$ denota o conjunto de todos os padrões definidos pelos caminhos em G que podem ser C-generalizados para um conjunto de padrões em C, i.e.,

$$\Pi(G,C) = \bigcup_{p \text{ é caminho em } G \wedge \Pi(p) \in C} \{P \mid \Pi(p) \xrightarrow{*}_{c} P\}.$$

Em [Jon96] foi escolhido representar padrões obtidos por operações de generalização do tipo 1 diretamente pelo grafo, adicionando arcos que podem começar e terminar em qualquer nó de G. Isto significa que a aplicação de generalizações do tipo 1 a qualquer padrão definido por um caminho em G produzirá padrões que podem ser derivados de outros caminhos em G sem precisar usar operações de generalização do tipo 1.

Consequentemente foi restringido os operadores de generalização como se segue:

$P \xrightarrow[c]{} P'$ se e somente se $P \xrightarrow[c]{i} P'$ para algum $i \in \{2,3\}$,

e

$P \xrightarrow{*}_{c} P'$ se e somente se existirem padrões $P = P_1, ..., P_k = P'$,

para algum $k \geq 1$, tal que $P_1 \xrightarrow{}_{c} P_2 \xrightarrow{}_{c} ... \xrightarrow{}_{c} P_k$.

É definido que $\Pi'(G,C)$ é o conjunto de padrões em C que podem ser derivados de padrões definidos pelos caminhos em G usando a sequência de operadores $\xrightarrow{}_{c}$:

$$\Pi'(G,C) = \bigcup_{p \text{ caminho em } G \wedge \Pi(p) \in C} \{P \mid \Pi(p) \xrightarrow{*}_{c} P\}.$$

Se G contiver arcos que representam as operações de generalização do tipo 1, então:

$$\Pi'(G,C) = \Pi(G,C).$$

2.2.1.2 O Algoritmo

A Figura 5 apresenta uma visão geral do algoritmo de descoberta de padrões implementado pela segunda versão do Pratt. O usuário fornece como entrada um conjunto de sequências $S = S^1, ..., S^n$ desalinhadas, limites B que especificam a classe de padrões C, e um número mínimo k de sequências com as quais o padrão deve se casar. Um grafo de padrões é construído dependendo do tipo de busca que o usuário quiser fazer:

1. Se o usuário não impuser restrições aos padrões a serem procurados, as (n-k+1) menores sequências em S são concatenadas em uma única cadeia s, e o grafo de padrões G é construído a partir de s (representado pela seta rotulada por 1 na Figura 5).
2. Se o usuário tiver uma sequência especial q e quiser procurar por padrões que se casam com q e que ocorram em pelo menos k sequências em S, então um grafo de padrões é construído a partir de q. Por exemplo, analisando a saída de uma pesquisa por homologia em um banco de dados, q poderia ser a sequência de consulta e S o conjunto das sequências que mais se assemelharam com q (representado pela seta rotulada por 2 na Figura 5).
3. Se o usuário tiver um alinhamento múltiplo de um subconjunto de sequências em S, um grafo de padrões pode ser construído a partir dele, e a busca restringida a padrões que estão consistente com o alinhamento (representado pela seta rotulada por 3 na Figura 5).

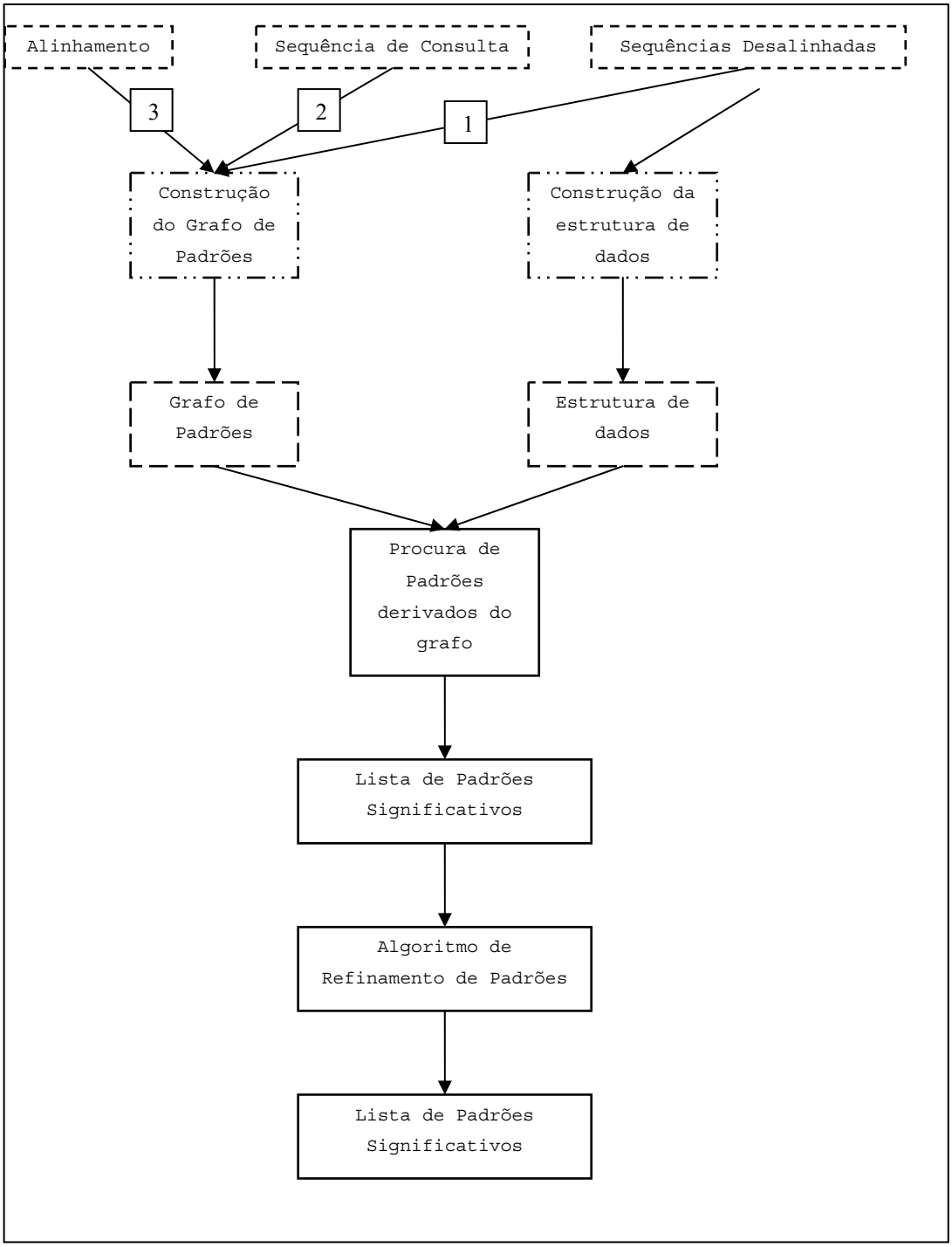


Figura 5. Esquema do Pratt.

Como exemplo de construção de um grafo de padrões, veja o caso em que ele é construído a partir de uma sequência $s = s_1...s_l$ e de um conjunto de limites B = (A,P,L,W,F,N,FP) que define o conjunto de padrões C. Recorde que A é o conjunto dos valores permitidos para os componentes de padrões, P o número máximo de componentes, L o comprimento máximo de um padrão, W o comprimento máximo de

uma região coringa, F a flexibilidade máxima de uma região coringa, N o número máximo de regiões coringas flexíveis, e FP o produto máximo de flexibilidade.

O procedimento que constrói o grafo é o seguinte:

- Construa em G um nó u_i para cada caracter s_i em s. Para cada nó u_i construa arcos para todos os nós u_j tais que $i < j \leq \min(i+W+1, l)$ e rotule o arco (u_i, u_j) com $(j-i-1, j-i-1)$.

Como exemplo, veja o grafo de padrões construído a partir da sequência s = 'ABCDEFG' com $W = 2$ na Figura 6. Neste caso, a sequência s contém $l = 7$ caracteres. O grafo tem então 7 nós rotulados por seus caracteres. O primeiro nó u_1 ($i = 1$) possui arcos para os nós cujos números são $1 < 2 \leq \min(4,7)$, ou seja, u_2=B, u_3=C e u_4=D.

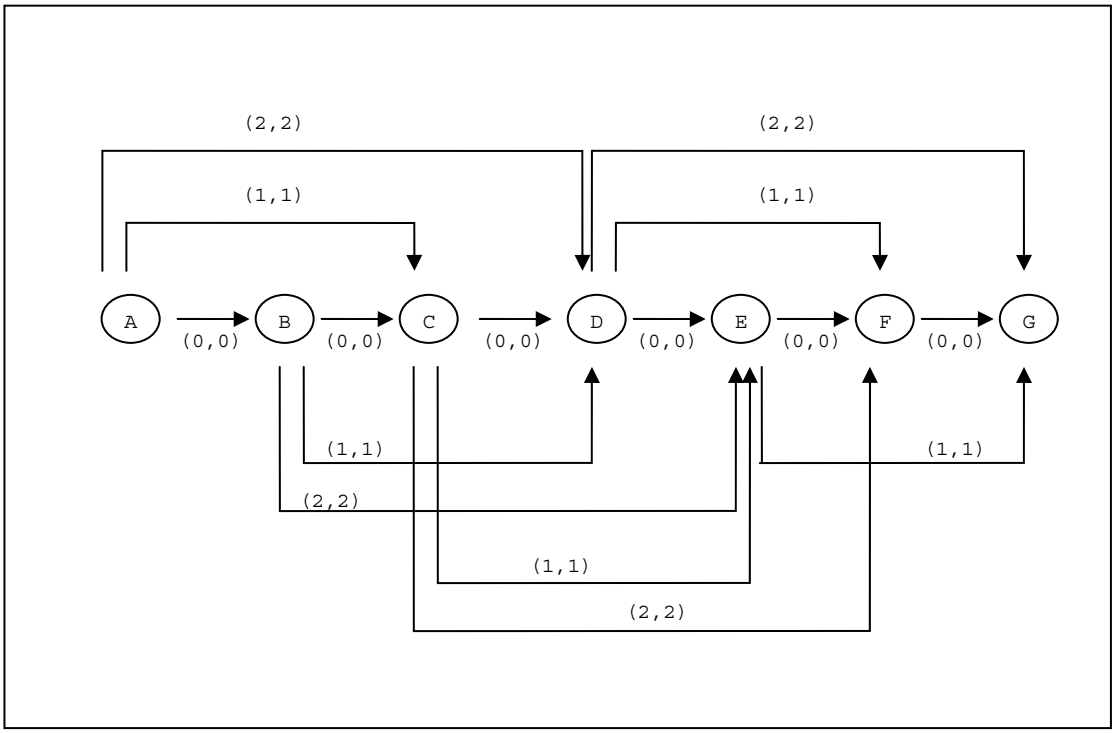


Figura 6. Exemplo de grafo gerado pelo Pratt.

Em [Jon96] há a descrição do algoritmo para construção do grafo a partir do alinhamento múltiplo de sequências.

Construído o grafo, o conjunto de padrões $\Pi'(G,C)$ é pesquisado procurando-se padrões com mais alta pontuação que se casam com no mínimo k sequências.Esta busca é feita usando pesquisa em profundidade sobre uma árvore de busca que contém

todos os padrões de $\Pi'(G,C)$. A árvore de busca possui como raiz um padrão vazio, e contém todos os *k*-padrões possíveis de $\Pi'(G,C)$ na profundidade *k*. Um *k*-padrão é definido por um *k*-caminho em *G* e por operações *c* de generalização em *C*, se existirem. A pesquisa em profundidade foi implementada como um procedimento recursivo, tendo como entrada um *k*-padrão *P* conservado e um *k*-caminho em *G* de onde *P* foi obtido. Todas as possíveis extensões de *P* que estão em *C*, e que podem ser geradas a partir de uma extensão de um nó do caminho *p*, são geradas.

Para cada um destes padrões, verifica-se se ele é conservado (ou seja, se se casa com o número mínimo de *k* sequências). Se for, o padrão é analisado novamente usando o mesmo procedimento. Se nenhuma extensão simples de *P* for conservada, e se *P* tiver uma pontuação *I* alta, então *P* é adicionado à lista de padrões a serem refinados.

Em outras palavras, em cada passo da busca em profundidade obtém-se um padrão existente que tenha se casado com o número *k* mínimo de sequências pedido pelo usuário e adiciona-se um buraco (possivelmente de tamanho 0) e um caracter ou um caracter ambíguo. Tenta-se todas as possibilidades. Isto significa que se for permitido regiões coringas, tenta-se todas as possibilidades de buracos, se for permitido símbolos ambíguos, tenta-se todos os símbolos possíveis. Depois testa-se cada novo padrão obtido para verificar se se casa com pelo menos *k* sequências. Isto é feito com a ajuda de uma estrutura de dados especial. Os padrões que não se casam com pelo menos *k* sequências são descartados. Repete-se este procedimento para todos os padrões que se casaram com as *k* sequências.

Como exemplo, veja o grafo da sequência s = 'ABCDEFG' com $W = 2$, apresentado na Figura 6. Assuma que $F = 1$ e que A={{A},{B},{C},{D},{E},{F},{G}}. Além disso, suponha que o padrão P = A-x-C-D foi derivado do caminho p=A,C,D. Neste momento, o caminho *p* pode ser estendido através dos arcos (D,E), (D,F) ou (D,G). Cada um destes arcos gerará extensões diferentes de *P*. Considerando que F=1, ou seja, que a flexibilidade máxima de uma região coringa é 1, tem-se as seguintes extensões.

Pelo arco (D,E):

- A-x-C-D-x(0,0)-E = A-x-C-D-x(0,0)-E
- A-x-C-D-x(0,1)-E

Pelo arco (D,F):

- A-x-C-D-x(1,1)-F

- A-x-C-D-x(0,1)-F
- A-x-C-D-x(1,2)-F

Pelo arco (D,G):

- A-x-C-D-x(2,2)-G
- A-x-C-D-x(1,2)-G
- A-x-C-D-x(2,3)-G.

Note que, para cada arco, foram consideradas todas as possibilidades de acordo com os limites passados pelo usuário. Por exemplo, como a flexibilidade máxima de uma região coringa era 1, a região coringa x do caminho que passava pelo arco (D,G), poderia ser igual à encontrada no grafo, x(2,2), ou poderia ter diferença de 1, isto é, x(2,3) e x(1,2).

O algoritmo recursivo explora todos os padrões em $\Pi'(G,C)$ e identifica os padrões que se casam (ou que são conservados) em pelo menos k sequências. Ele poda as subárvores na árvore de busca pelos padrões que não são conservados. Dependendo do tamanho do grafo de padrões, este algoritmo pode ser mais eficiente que o método de simples pesquisa em profundidade utilizado na primeira versão do Pratt.

O algoritmo de busca e poda dos possíveis padrões é uma extensão do método de Neuwald e Green [NG94], que é explicado na seção posterior (programa ASSET). Resumidamente em [NG94] é apresentado um método que busca padrões e relata os mais significativos que se casam com qualquer número de sequências. A função que calcula o quanto um padrão é significativo depende do número de sequências que o padrão se casa e dele mesmo. O método é capaz de encontrar padrões com componentes identificadores e ambíguos. Já o Pratt restringe a busca aos padrões que se casam a um número mínimo de sequências. Isto permite podar a busca eficientemente.

Finalmente, depois de identificar os padrões conservados, estes serão entradas de um algoritmo de refinamento de padrões, onde componentes de padrões ambíguos são introduzidos. Para obter informações do algoritmo de refinamento, consulte [Jon96] e [JCH95].

2.2.2 ASSET

Aligned Segment Statistical Evaluation Tool (ASSET) [NG94] é um programa que implementa o algoritmo de busca em profundidade de padrões ou DFPS (de *Depth-*

First Pattern Search em inglês) em um conjunto de sequências de entrada. A seguir serão descritos os detalhes do algoritmo.

2.2.2.1 Definições

Um padrão de comprimento k é uma sequência $Q = Q_1 Q_{2....} Q_k$ onde para cada i, Q_i é um conjunto de caracteres (que representam os aminoácidos) permitidos na posição i. O conjunto que consiste de um único aminoácido é representado por seu caracter, um conjunto com vários aminoácidos é representado por seus caracteres entre colchetes e o conjunto que consiste por todos os resíduos é representado por $\sum$ ou, também, por um ponto ".". Um d-padrão ($2 \le d \le k$) é um padrão que tem um conjunto com um único resíduo em d posições, incluindo a primeira e a k-ésima posição,e as demais *(k-d)* posições são ".".

Uma subsequência $s = s_1 s_{2....} s_l$ (onde cada s_i é um aminoácido) de comprimento l é chamada de l-segmento. Ela se casa com um padrão Q, de comprimento $k \le l$, se s_i for igual a um elemento de Q_i para cada $i \le k$.

O bloco de padrões $B_Q = B_Q^l$ é o conjunto de todos os l-segmentos (de uma dada coleção de sequências de entrada) que se casam com Q. O bloco universal $B_U = B_u^l$ é o conjunto com todos os l-segmentos derivados das sequências de entrada. Por exemplo, se as sequências de entrada forem "MAGGSPRL" e "MWAGST", então B_u^5 consiste de:

<1,1>	MAGGS	<2,1>	MWAGS
<1,2>	AGGSP	<2,2>	WAGST
<1,3>	GGSPR	<2,3>	AGSTx
<1,4>	GSPRL	<2,4>	GSTxx
<1,5>	SPRLx	<2,5>	STxxx
<1,6>	PRLxx	<2,6>	Txxxx
<1,7>	RLxxx		
<1,8>	Lxxxx		

e $B_{AG.[ST]}^5$ consiste de <1,2> e <2,3> (onde <i,j> significa o segmento na sequência i que se inicia na posição j).

O número de l-segmentos em M sequências que podem se casar com um d-padrão de comprimento $k \le l$ é

$$N_k = \sum_{i=1}^{M} \max(0, L_i - k + 1) \quad (1)$$

onde L_i é o comprimento da sequência i.

Neste artigo, um motif significa uma região conservada associada a um ou a mais padrões significativos em um conjunto de sequências de proteínas relacionadas.

2.2.2.2 Álgebra dos Blocos

Para detectar os motifs de um conjunto de sequências de proteínas é necessário, primeiro, determinar os *l*-segmentos (i.e. $B_Q = B_Q^l$) que se casam com todos os padrões de um tipo particular. Os motifs são revelados através dos padrões que contém um número grande (estatisticamente significativo) de segmentos que se casam com eles.

B_Q pode ser calculado através dos blocos elementares $b_{i,r}$, para $1 \le i \le l$ e aminoácido *r*, onde $b_{i,r}$ é o conjunto de todos os *l*-segmentos que têm o resíduo *r* na posição *i*.

O bloco para um padrão arbitrário *Q* de comprimento *k* é dado por:

$$B_Q = \bigcap_{i=1}^{k} \left(\bigcup_{r \in Q_i} b_{i,r} \right). \quad (2)$$

Por exemplo, o bloco $B_{AG.[ST]}$ pode ser derivado por:

$B_{AG.[ST]} = b_{1,A} \cap b_{2,G} \cap (b_{4,S} \cup b_{4,T})$

$= \{<1,2>,<2,3>\} \cap \{<1,2>,<1,3>,<2,3>\} \cap (\{<1,2>,<2,2>\} \cup \{<2,3>\})$

$= \{<1,2>\ \mathrm{AGGSP},<2,3>\ \mathrm{AGST}\}.$

Note que a parte da equação de B_Q com o símbolo da união representa os aminoácidos entre colchetes do padrão *Q*. No exemplo a quarta posição do padrão *Q* é um S **ou** um T. Já o símbolo de intersecção representa cada posição do padrão *Q*. No exemplo a primeira posição é um A **e** a segunda posição é um G **e** a quarta posição é um S ou um T.

Há ainda uma versão recursiva da equação (2). Seja *Q* e *Q'* padrões que se diferem em apenas uma posição i, sendo Q_i um subconjunto de Σ e $Q_i' = \Sigma$, tem-se

$$B_Q = B_{Q'} \cap \left(\bigcup_{r \in Q_i} b_{i,r} \right). \quad (3)$$

Note que *Q'* é uma generalização de *Q* ou, logicamente, que *Q* é uma especialização de *Q'*. Os segmentos que se casam com Q (ou seja B_Q) são os segmentos que se casam com *Q'*, mas (representado pela intersecção) que tenham na posição *i* os resíduos determinados por Q_i, que é um subconjunto de Σ (veja que a posição i de Q_i' não é feita nenhuma restrição porque $Q_i' = \Sigma$).

Um B_Q arbitrário pode ser derivado recursivamente pela fórmula (3) começando pelo bloco universal B_U.

2.2.2.3 Significado Estatístico

Uma importante característica deste método é o uso de um critério estatístico rigoroso para avaliar a significância de um padrão. O critério utiliza a probabilidade de que um padrão particular *Q* se casa com *C* ou mais dos N_k segmentos, calculados com base em um modelo nulo que assume que os segmentos são independentes uns dos outros.e que as posições dentro de um segmento são independentes umas das outras. A primeira suposição é violada quando os segmentos se sobrepõe. De acordo com estas suposições o número de segmentos que se casam com *Q* é distribuído binomialmente, i.e. é modelado pela série de N_k tentativas independentes onde o resultado é positivo (o padrão e o segmento se casam) ou negativo (não se casam). A probabilidade de *C* ou mais segmentos casados é de:

$$P(N_k, C, p_Q) = \sum_{i=C}^{N_k} \binom{N_k}{i} p_Q^i (1 - p_Q)^{N_k - i} \quad . (4)$$

Agui p_Q, a frequência predita do padrão Q é dada por:

$$p_Q = \prod_{i=1}^{k} \left(\sum_{r \in Q_i} f_r \right), (5)$$

onde f_r é a frequência do resíduo *r* em um conjunto de dados como um todo. Por exemplo, $p_q = p_{AG./ST/} = f_G f_A (f_S + f_T)$.

Uma versão recursiva da equação (5) é útil. Seja *Q* e *Q'* padrões que se diferem em apenas uma posição i, sendo Q_i um subconjunto de Σ e $Q_i' = \Sigma$, tem-se

$$p_Q = p_{Q'} \times \left(\sum_{r \in Q_i} f_r \right). (6)$$

2.2.2.4 Busca de Padrão em Profundidade

Combinando uma busca em profundidade [CLR90] com as fórmulas recursivas (3) e (6) é possível encontrar os *l*-segmentos que se casam com todos os possíveis padrões de comprimento *l* ou menos e suas correspondentes probabilidades. Este algoritmo de busca utiliza recursão para evitar cálculos redundantes e para não considerar padrões que se iniciam com subpadrões que não se casam com um segmento.

Para obter eficiência pode ser exigido que os padrões para serem considerados significativos devem se casar com no mínimo C_{min} segmentos, sendo C_{min}>1.

Vários motifs correspondem a padrões que permitem vários aminoácidos em determinadas posições. Uma forma de detectar tais padrões é permitir que os padrões tenham dois resíduos em determinadas posições e restringir o conjunto de pares permitidos, já que permitir os 190 pares possíveis afetaria muita a velocidade da execução do algoritmo. Neste caso, os autores consideraram os 26 pares de aminoácidos que têm pontuações positivas na matriz BLOSUM62 [HH92]; estes pares ocorrem mais frequentemente em alinhamentos de proteínas relacionadas.

Ao considerar posições dos padrões com pares de resíduos, há uma perda de velocidade na execução do algoritmo, que pode ser melhorada exigindo que os padrões contemham pelo menos s_{min} posições com um único resíduo, sendo s_{min}>1. Como o número de segmentos que se casam com um único resíduo é tipicamente menor que aqueles que se casam com pares de resíduos, começar a busca pelas posições de um único resíduo permite uma podagem mais rápida da árvore.

2.2.2.5 Procedimento Básico

O objetivo principal é construir uma árvore que represente todos os padrões significativos existentes em um conjunto de sequências de entrada e encontrar os *l*-segmentos que os possuem.

Os nós da árvore correspondem aos blocos B_Q, sendo que o nó raiz é o bloco universal. Desta forma cada nó representa os l-segmentos que se casam com um determinado padrão Q.

Cada nó filho é uma especialização do nó pai, isto é, o padrão que o filho representa é uma especialização do padrão que o pai representa. Tal especialização do nó pai para o nó filho representa uma diferença em apenas uma posição do padrão.

Quanto maior a profundidade de um nó na árvore, mais especializado é o padrão que ele representa.

A árvore é construída recursivamente gerando os filhos de cada nó até a profundidade d_{max} usando as equações (3) e (6). *dmax* indica o número máximo de posições em que são especificados os resíduos, as outras posições são '.'.

A seguir estão os dois procedimentos utilizados: *search* e *dfps*.

O objetivo do *search* é chamar o *dfps* com todos os *l*-segmentos que existem a partir das sequências de entrada. *search* possui os seguintes parâmetros de entrada:

- *block* b: matriz 20x *l* de blocos de segmentos

Aminoácidos	Posições			
	$b_{1,a1}$	$b_{2,a1}$	...	$b_{l,a1}$
	$b_{1,a2}$	$b_{2,a2}$	...	$b_{l,a2}$
	⋮			
	$B_{1,a20}$	$b_{2,a20}$		$b_{l,a20}$

  Cada $b_{i,j}$ corresponde ao conjunto dos *l*-segmentos que possuem o aminoácido *j* na posição *i*. Por exemplo, $b_{1,a20}$ é o conjunto dos *l*-segmentos que possuem o aminoácido a_{20} na primeira posição.

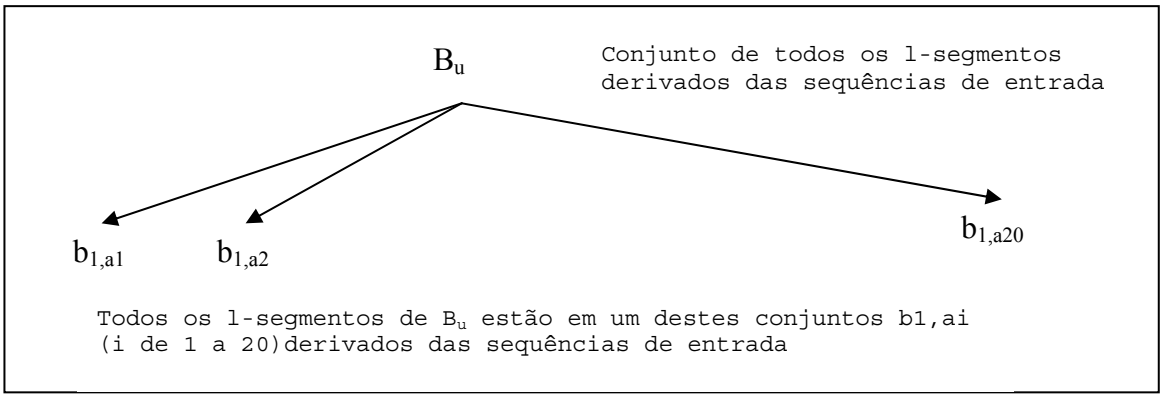
  Note que todos os *l*-segmentos das sequências de entrada pertencem a estes conjuntos e que cada *l*-segmento pertencerá a apenas um dos conjuntos $b_{1,j}$, sendo *j* um aminoácido de a_1 a a_{20}, já que qualquer *l*-segmento de ter em sua primeira posição um dos aminoácidos possíveis.

- *real f*[]: vetor com as frequências dos aminoácidos
- *heap H*: pilha com os padrões mais significativos

  A probabilidade de um padrão associada a cada nó (obtida pela equação (4)) é usado como chave para inserção de padrões na pilha [ASS+86]. Durante o procedimento *dfps* se a pilha estiver cheia, um padrão só é inserido se sua chave for menor que alguma chave existente na pilha; o padrão que tiver a chave maior da pilha é removido antes de tal inserção. Então, no final do processamento do algoritmo, a pilha conterá os padrões mais significativos.

```
procedure search (block b[][],real f[], modifies heap H)
element r;
pattern Q:= U;
for r∈Σ
       Q1 := {r};
       dfps(b,b1,r,Q,fr,f,1,2,H);
end for
end search;
```

O procedimento *search* chama o *dfps* variando *r*, ou seja variando $b_{1,r}$ (percorrendo a primeira coluna da matriz *b*[][]). Desta forma *dfps* é executado para todos os *l*-segmentos que existem a partir das sequências de entrada.



O objetivo de *dfps* é construir uma árvore que representa todos os padrões significativos existentes em um conjunto de sequências de entrada. Ao encontrar estes padrões, encontra-se também os *l*-segmentos que os possuem.

dfps recebe os seguintes parâmetros de entrada:

- *block b[][]:* explicado anteriormente;
- *block B*: um bloco $b_{y,r}$ de l-segmentos (que tem na posição y o aminoácido r) que corresponde ao nó pai, sendo um dos objetivos do procedimento calcular os blocos B's que correspondem aos nós filhos de B.
- *pattern Q*: padrão
- *real p*: frequência do resíduo r
- *real f[]:* explicado anteriormente;
- *integer d*: profundidade da árvore (ou número de posições do padrão que possuem resíduos determinados);
- *integer i*:contador para controle da criação dos nós filhos;
- *heap H*: explicado anteriormente.

O procedimento *search* chama *dfps* para encontrar os nós filhos de B_u (raiz da árvore), ou seja, *search* chama *dfps* para encontrar os filhos de $b_{1,a1}$, $b_{1,a2}$,...., $b_{1,a20}$.

A seguir está o pseudo-código de *dfps*.

```
1. procedure dfps(block b[][],B,pattern Q, realp,f[], integer d,i,
   modifies heap H)
2. element r;
3. real key;
4. block B';
5. integer k,C;
6. for k = i..l
```

```
7.  for r ∈ Σ
              i.  B' := B ∩ b_{k,r};
             ii.  C := |B'|;
            iii.  if C ≥ C_mim
                      1.  Q_k := {r};
                      2.  key := P(N_k,C,p x f_r)
                      3.  insertheap(Q,key,H);
                      4.  if d<d_max
                              a.  dfps(b,B',Q,p x f_r,f,d+1,k+1,H)
                      5.  end if;
             iv.  end if;
8.  end for;
9.  Q_k := Σ;
10. end for;
11. end dfps;
```

Durante cada chamada de dfps para um nó B, os seus filhos B's são calculados através da equação de especialização (6) (linha 7.i). A quantidade de l-segmentos que B' possui é calculada (linha 7.ii). O procedimento só continua (ou seja os filhos de B' serão tratados pelo dfps) se a quantidade de l-segmentos for significativa (linha 7.iii). Caso a quantidade seja significativa, a probabilidade do padrão é calculada (linha 7.iii.2). Se a probabilidade for significativa, tal padrão é inserido na pilha (linha 7.iii.3). A profundidade da árvore é testada para saber se se continua ou não a "descer" a árvore (linha 7.iii.4). Caso seja válido, *dfps* é chamado recursivamente recebendo como parâmetro o filho B' (linha 7.iii.4.a).

A seguir está uma árvore que esquematiza a execução deste algoritmo.

2.2.2.6 Estrutura de Dados dos Blocos

Uma característica importante da execução do algoritmo de *dfps* é o uso de uma estrutura de dados que permite que seu desempenho seja eficiente. Os blocos são representados por vetores de bits, cada bit corresponde a um segmento; o bit tem valor 1 se o segmento tiver no bloco e 0 se não estiver. Isto permite que operações de intersecção sejam feitas rápidas em paralelo, 8 bits por vez usando operadores lógicos AND. Algumas outras características para aumentar a eficiência são descritas em [NG94].

2.2.2.7 Identificação de Regiões com Motifs

O método de agrupamento de padrões usa o fato de que os padrões na mesma região devem ter um número grande e inesperado de segmentos que se casam com ele. Para identificar regiões conservadas que se casam com um grupo de padrões um perfil é construído pelo método de Gribskov *et. al*. [GME87,GLE90].

Uma discussão um pouco mais detalhada da identificação de regiões que contém motifs destacando-se como agrupar padrões e localizar regiões conservadas também são discutidas em [NG94].

2.2.3 WINNOVER

O WINNOVER [PS00] é um outro algoritmo que faz podas em sua estrutura de dados para melhorar o desempenho.

A entrada deste algoritmo é um conjunto *S* de sequências. O objetivo é encontrar um padrão desconhecido que aparece em diferentes posições desconhecidas em cada sequência pertencente ao conjunto dado como entrada. Se um padrão de comprimento *l* aparecesse exatamente em cada sequência, ele poderia ser descoberto através da enumeração de todas as subcadeias de comprimento *l* que existem no conjunto de sequências. No entanto, os padrões biológicos estão sujeitos à mutações e geralmente não aparecem exatamente iguais. Um modelo natural é permitir que o padrão desconhecido apareça com alguns descasamentos, inserções ou remoções no conjunto de sequências [PS00].

Suponha que o conjunto *S* tenha *n* sequências e que se queira encontrar um (*l*,*d*)-padrão, ou seja, um padrão com comprimento *l* que tenha no máximo *d* descasamentos. Se forem encontradas 2 ocorrências de um determinado padrão, elas diferirão em, no máximo, *2d* posições porque ambas diferem do padrão em no máximo *d* posições. A idéia deste algoritmo é procurar por um grupo de *n* subcadeias de comprimento *l*, cada uma obtida de sequências diferentes, sendo que qualquer par de subcadeias seja diferente em no máximo *2d* posições [BDV+00].

Não é garantido que, ao encontrar esta combinação de subcadeias, tenha sido encontrado um padrão. Por exemplo, suponha que se queira encontrar um padrão de comprimento $l = 4$ com no máximo $d = 1$ descasamentos e que tenham sido encontradas 3 ocorrências: AAAA, BBAA, CCAA.

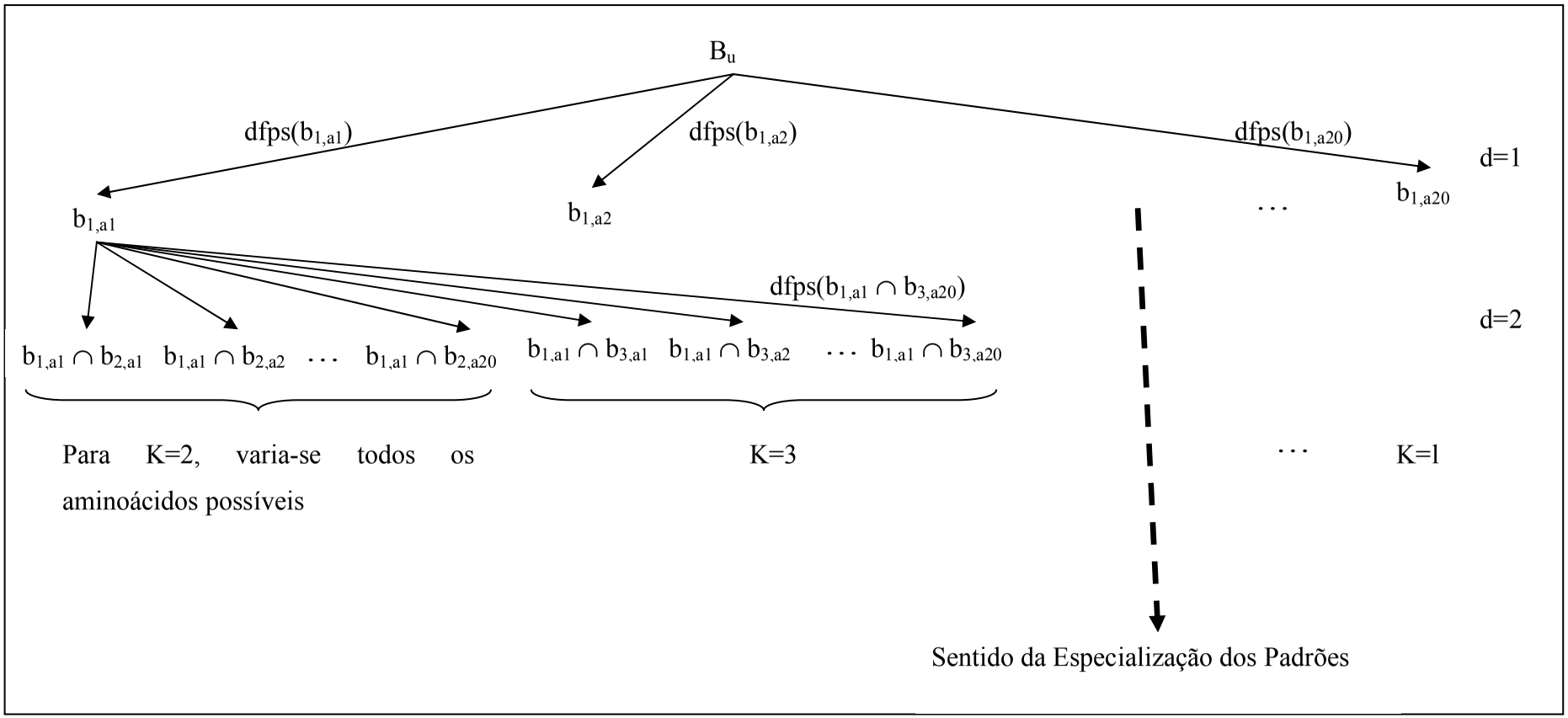


Figura 7.Esquema do algoritmos dfps.

Qualquer par delas difere-se em $2d = 2$ posições, mas nenhum diferencia-se em apenas $d = 1$ posições. No entanto, o algoritmo assume que isso não acontece frequentemente e que a maioria das descobertas é um padrão [BDV+00].

Dado um conjunto de sequências, o WINNOVER constrói um grafo *G* onde os vértices são as subcadeias de comprimento *l* e os arcos correspondem às subcadeias similares [PS00]. Dois vértices, que correspondem a 2 subcadeias, serão conectados por um arco se as duas subcadeias forem obtidas de sequências diferentes e diferenciarem-se em no máximo $2d$ posições. Veja na Figura 8 o grafo G para as sequências ABDE, AFCG, HBCI e JBCK onde $n = 4$, $d = 1$ e $l = 3$.

O grafo obtido é *n*-particionado, isto é, ele pode ser particionado em *n* partições de tal maneira que não exista arco entre vértices da mesma partição. De fato, as partições corresponderão às sequências individuais. O algoritmo encontra o conjunto de *n* vértices tais que quaisquer 2 vértices estejam conectados por um arco. Este conjunto de vértices é chamado de um *clique* [BDV+00]. A idéia é que quaisquer duas ocorrências do padrão em *S* será representada por um arco em *G*. Consequentemente, qualquer (*l*,*d*)-padrão corresponderá a um clique com *n* vértices em *G*, o que reduz o problema de descoberta de padrões a um problema de descoberta de *large clicks* em um grafo [PS00].

O problema de descoberta de *large clicks* em um grafo é NP-Completo [GJ79]. Uma possibilidade é procurar por cliques através de uma busca exaustiva com podas cuidadosas. O algoritmo WINNOVER reduz o número de arcos do grafo *G* removendo os arcos que não podem fazer parte de nenhum clique de tamanho *n*. Alguns arcos do grafo conectam instâncias de padrões reais existentes em S, mas alguns arcos são falsos ou inconsistentes porque correspondem a padrões falsos, isto é, eles representam subcadeias que não são padrões. Os arcos falsos disfarçam os arcos reais e dificultam a descoberta de padrões [PS00]. A poda no grafo tem como objetivo remover os padrões falsos, o que diminui a quantidade de arcos do grafo e, consequentemente, facilita a descoberta de padrões.

O algoritmo é baseado na noção de *clique expansível*. Nota-se que cada vértice de um clique está em uma partição diferente (porque em um clique todos os pares de vértices são conectados por um arco e os vértices de uma partição não são conectados por um arco). O objetivo é encontrar um clique que tenha exatamente um vértice em cada partição. Um vértice é chamado de *vizinho* de um clique se for conectado a todos os vértices deste clique (i.e. acrescentando este vértice, obtém-se um clique maior). Um

clique com $k < n$ vértices é chamado de *expansível* se tiver pelo menos um *vizinho* em cada partição. Um arco é chamado de falso se ele não pertencer a nenhum clique expansível com k vértices.

Suponha que um grafo tenha um clique com n vértices. Um subconjunto contendo k destes n vértices é um clique expansível com k vértices. Consequentemente para cada clique com n vértices, existem $\binom{n}{k}$ cliques expansíveis com k vértices. Qualquer arco pertencente ao clique com n vértices é membro de $\binom{n-2}{k-2}$ cliques expansíveis. Desta forma se for encontrado um arco que não é membro de $\binom{n-2}{k-2}$ cliques expansíveis, este arco não pode fazer parte de um clique com n vértices e pode ser removido. Os arcos podem ser removidos enquanto a condição acima for satisfeita.

Sendo assim, escolhe-se o valor de k, descobre-se todos os cliques expansíveis e remove-se os arcos que não estão em pelo menos $\binom{n-2}{k-2}$ destes cliques expansíveis.

Veja o exemplo na Figura 8 onde o algoritmo WINNOVER foi aplicado com $k = 1$. O grafo contém apenas um clique correspondente ao padrão ABC.

Para $k = 1$, o clique é um subgrafo com um vértice. Neste caso existirá um clique para cada vértice. O vértice é um clique expansível se estiver conectado com pelo menos um vértice em cada uma das outras partições. Os vértices sem esta propriedade podem ser removidos.

Para $k = 2$, o clique é um subgrafo formado pelos vértices u e v com arco (u,v). Neste caso o clique terá apenas um arco (u,v). Ele é expansível se existir um vértice w em cada (n-2) outras partições tais que existam arcos (u,w) e (v,w) (i.e. os vértices u,v,w formam um ciclo de comprimento 3). Cada arco deve estar presente em $\binom{n-2}{k-2} = 1$ clique expansível.

Analogamente, é possível exigir que cada arco esteja em pelo menos $(n\text{-}2)$ cliques expansíveis com 3 vértices e assim por diante. Quanto maior o k, maior o tempo de execução, mas mais arcos serão removidos.

2.2.4 TEIRESIAS

Para que um padrão seja considerado significativo, ele precisa ter um comprimento suficiente. No entanto, padrões longos são difíceis de serem encontrados usando as técnicas de enumeração apresentadas anteriormente. Uma possibilidade para encontrá-

los é através da combinação de padrões pequenos. O algoritmo TEIRESIAS [RF98b] utiliza esta idéia.

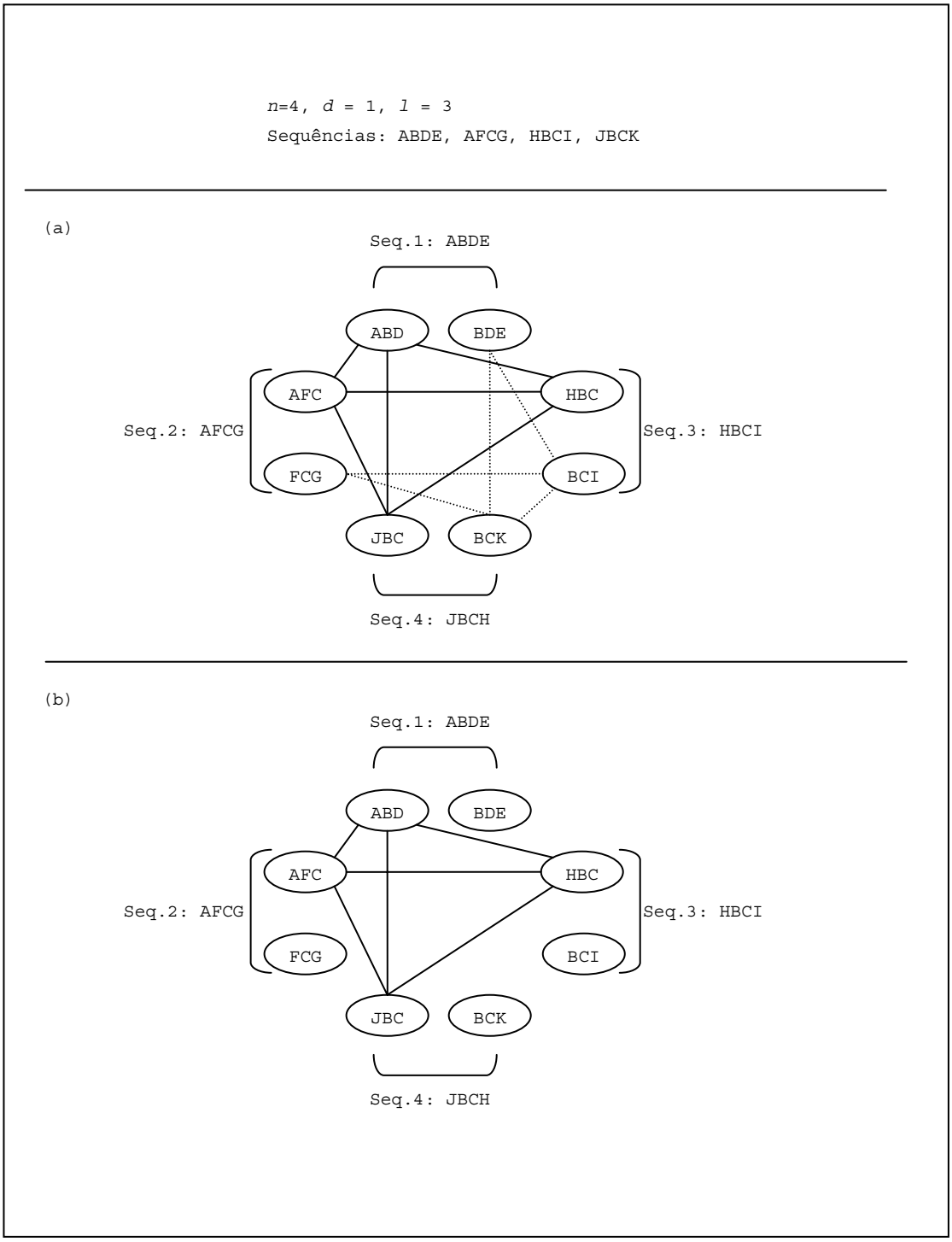


Figura 8. O grafo *G* gerado pelo WINNOVER completo e com podas.

O TEIRESIAS é um algoritmo de enumeração de padrões. O método é combinatorial e relata todos os padrões maximais que possuem um suporte suficiente sem enumerar

todo o espaço de padrões. Isto faz com que ele seja eficiente. Os padrões maximais são gerados primeiro. Como resultado, os padrões redundantes podem ser facilmente detectados através de comparações aos padrões já obtidos. Este algoritmo pode tratar eficientemente padrões com qualquer comprimento [Flo99].

O algoritmo opera em duas fases: varredura e convolução. Durante a fase de varredura, padrões elementares com suporte suficiente são encontrados. Estes padrões elementares constituem blocos bases para a fase de convolução. Eles são combinados progressivamente em padrões maiores até que todos os padrões maximais sejam obtidos. Além disso, a ordem em que a fase de convolução é executada torna fácil a identificação e o descarte de padrões não maximais.

2.2.4.1 Terminologia e Determinação do Problema

Seja $\sum$ o alfabeto definido anteriormente (isto é, o conjunto de aminoácidos e nucleotídeos). A classe de padrões tratada pelo TEIRESIAS é:

$$C_{TEI} = \sum (\sum \cup \{\text{'.'}\}) * \sum.$$

A linguagem $L(P)$ definida para um padrão $P \in C_{TEI}$ é o conjunto de todas as cadeias que podem ser obtidas de P substituindo cada caracter coringa por um caracter arbitrário de $\sum$. Por exemplo, as seguintes cadeias são elementos da linguagem $L(P)$ onde o padrão P é igual a 'A.CH..E':

'ADCHFFE', 'ALCHESE', 'AGCHADE'.

Para qualquer padrão P, qualquer subcadeia de P que também é um padrão é chamada de *subpadrão* de P. Por exemplo 'H..E' é um subpadrão do padrão 'A.CH..E'. Um padrão P é chamado *padrão* $<L,W>$ (com $L \leq W$) se todo subpadrão de P com comprimento W ou mais contiver pelo menos L caracteres.

Dado um padrão P e um conjunto de sequências $S = \{s_1, s_2, ...s_n\}$ define-se como a lista de deslocamentos de P com relação a S o seguinte conjunto:

$$L_S(P) = \{(i,j), \text{a sequência } s_i \text{ casa-se com } P \text{ com deslocamento } j\}.$$

Um padrão P' é dito ser mais específico do que P se P' puder ser obtido de P substituindo um ou mais caracteres coringas por caracteres de $\sum$ ou concatenando uma cadeia arbitrária de caracteres coringas ou caracteres de $\sum$ no lado esquerdo ou direito de P. Todos os padrões seguintes são mais específicos do que 'A.CH..E':

'AFCH..E', 'A.CHL.E.K', 'SA.CH..E'.

Note que se um padrão *P'* é mais específico que um padrão *P* então todos os conjuntos de sequências S terão $|L_S(P')| \leq |L_S(P)|$.

Dado um conjunto S, um padrão P é chamado de maximal em relação a S se não existir um padrão P' que seja mais específico que P tal que $|L_S(P)| = |L_S(P')|$. Se P é não maximal, então existe um padrão maximal P' tal que $|L_S(P)| = |L_S(P')|$.

Usando as definições anteriores, o problema é definido da seguinte maneira:

2.2.4.2 Definição do Problema

Entrada: Um conjunto $S = \{s_1, s_2, ...s_n\}$ de sequências de Σ^* e inteiros *L*, *W* e *K* onde $L \leq W$ e $2 \leq k \leq n$.

Saída: Todos os padrões maximais em C_{TEI} com suporte de no mínimo k no conjunto S.

2.2.4.3 Descrição do Algoritmo

O TEIRESIAS começa varrendo as sequências do conjunto de entrada *S* localizando todos os padrões elementares com suporte de pelo menos *K*. Um *padrão elementar* é uma padrão <*L*,*W*> que contenha exatamente *L* caracteres de Σ. O esquema seguinte mostra um exemplo de processo de varredura para um conjunto particular de sequências *S* e valores específicos de *L* e *W*.

O conjunto é $S = \{s_1, s_2, s_3\}$ = {SDFBASTS, LFCASTS, FDASTSNP}sendo *L*=3, *W*=4 e *K*=3.

O primeiro padrão <3,4> elementar encontrado é o 'F.AS'.

S_1	S	D	**F**	B	**A**	**S**	T	S		
S_2		L	**F**	C	**A**	**S**	T	S		
S_3			**F**	D	**A**	**S**	T	S	N	P

O deslocamento do padrão na primeira sequência é de 3, na segunda é de 2 e na terceira é de 1.

O segundo padrão <3,4> elementar encontrado é o 'AST'.

S_1	S	D	F	B	**A**	**S**	**T**	S
S_2		L	F	C	**A**	**S**	**T**	S

S_3 F D **A** **S** **T** S N P

O deslocamento do padrão na primeira sequência é de 5, na segunda é de 4 e na terceira é de 3.

O terceiro padrão <3,4> elementar encontrado é o ‘AS.S’.

S_1 S D F B **A** **S** T **S**

S_2 L F C **A** **S** T **S**

S_3 F D **A** **S** T **S** N P

O deslocamento do padrão na primeira sequência é de 5, na segunda é de 4 e na terceira é de 3.

Analogamente, o quarto padrão <3,4> elementar encontrado é o ‘STS’.

S_1 S D F B A **S** **T** **S**

S_2 L F C A **S** **T** **S**

S_3 F D A **S** **T** **S** N P

O deslocamento do padrão na primeira sequência é de 6, na segunda é de 5 e na terceira é de 4.

E finalmente o último padrão <3,4> elementar encontrado é o ‘A.TS’.

S_1 S D F B **A** S **T** **S**

S_2 L F C **A** S **T** **S**

S_3 F D **A** S **T** **S** N P

O deslocamento do padrão na primeira sequência é de 5, na segunda é de 4 e na terceira é de 3.

O resultado do processo de varredura é um conjunto EP contendo todos os padrões elementares (e sua listas de deslocamentos associada) que satisfazem o número de ocorrências mínimo. Este conjunto é a entrada da fase de convolução. Para entender como esta fase funciona é importante notar que a fase de varredura corta em pequenos

pedaços todos os padrões que existem. A tarefa da fase de convolução é unir estes pedaços (com tempo e espaço eficientes) para descobrir os padrões originais.

A base principal da fase de convolução é que um padrão P pode ser reconstruído juntando os pares A e B de padrões intermediários tais que o sufixo de A é o mesmo que o prefixo de B. Por exemplo, considere novamente o conjunto S de sequências anterior. Este conjunto contém o padrão 'F.ASTS'. É possível reconstruir este padrão seguindo os seguintes passos:

- Una os padrões elementares 'F.AS' com 'AST' (note que a cadeia 'AS' é sufixo de 'F.AS' e prefixo de 'AST';
- Una o padrão gerado 'F.AST' com o padrão elementar 'STS'.

Dado um padrão P com no mínimo L caracteres de $\sum$, define-se *prefixo*(P) como o subpadrão de P que tem exatamente (L-1) caracteres de $\sum$ e é o prefixo de P. Por exemplo, se $L = 3$, então:

prefixo('F.ASTS') = 'F.A', *prefixo*('AST') = 'AS'.

Analogamente define-se o sufixo(P) como o subpadrão sufixo de P com exatamente (L-1) caracteres de $\sum$. Novamente, para $L = 3$:

sufixo('F.A...S') = 'AS', *prefixo*('ASTS') = 'TS'.

Seja P e Q dois padrões com pelo menos L caracteres de $\sum$ cada, a operação binária *convolução* de P com Q, representada por $\oplus$, é um novo padrão R definido como se segue:

$R = P \oplus Q = PQ'$ se *sufixo*(P) = *prefixo*(Q), senão $R = P \oplus Q = \varnothing$,

onde Q' é a cadeia restante de Q após a retirada do *prefixo*(Q) e $\varnothing$ representa uma cadeia vazia. Os padrões são chamados de *convolúveis* se $P \oplus Q \neq \varnothing$. Veja alguns exemplos quando L=3:

'DF.A.T'$\oplus$ 'A.TSE' = 'DF.A.TSE', 'AS.TF' $\oplus$ 'T.FDE' = $\varnothing$.

Na fase de convolução toma-se cada padrão elementar e tenta-se estendê-lo em ambos os lados unindo um padrão elementar ao outro de todas as formas possíveis. Qualquer padrão que não possa ser estendido sem perda de suporte pode ser um padrão potencialmente maximal. No entanto ainda podem ser gerados padrões não maximais desta forma, além de que alguns padrões podem ser obtidos mais de uma vez. Consequentemente é mantida uma lista de padrões que foram gerados e todos os novos padrões são checados com esta lista. Se a lista contiver um padrão mais específico com o mesmo suporte (ou seja, com o mesmo número de ocorrências), o

padrão novo é descartado. A busca por novos padrões é organizada de tal forma que qualquer padrão maximal P é escrito na saída antes que qualquer padrão não maximal menos específico do que P seja obtido. Desta maneira não é preciso remover nenhum padrão já mostrado na saída [BDV+00]. Desta forma, a fase de convolução tem dois objetivos importantes:

- Obter todos os padrões; e
- Identificar e descartar padrões que não são maximais.

Para atingir estes objetivos é feita uma organização cuidadosa da pesquisa em profundidade. Com ela é possível garantir que: (a) todos os padrões são obtidos, e (b) um padrão maximal P é gerado antes de qualquer padrão não maximal de P. Desta maneira um padrão não maximal é detectado facilmente, apenas comparando-o com todos os padrões obtidos até o momento.

Sejam *P* e *Q* dois padrões arbitrários. Para decidir se *P* tem *forma de prefixo* menor que *Q* (representa-se como $P <_{pf} Q$) executa-se o seguinte procedimento. Primeiro os dois padrões são alinhados de maneira que seus primeiros caracteres estejam todos para a esquerda e nas mesmas colunas. Depois seus caracteres são analisados da esquerda para a direita. Este procedimento é interrompido quando se encontra (se existir) um alinhamento entre uma coluna com um caracter e um coringa. Se o caracter for do padrão *P*, então *P* tem forma de prefixo menor que *Q*. Segue um exemplo.

Seja P = ‘ASD...F’ e Q = ‘SE.ERF.DG’. O primeiro passo alinha os dois padrões começando pelo lado esquerdo.

```
P   A S D . . . F

Q   S E . E R F . D G
```

Depois localiza-se a primeira coluna que contém um coringa e um caracter.

```
P   A S D . . . F
      |   x
Q   S E . E R F . D G
```

Finalmente o padrão na coluna marcada pelo 'x' é o menor. Então $P <_{pf} Q$.

Analogamente é possível determinar se um padrão *P* tem *forma de sufixo* menor que *Q* (representa-se como $P <_{sf} Q$) executa-se o seguinte procedimento. Os dois padrões são alinhados de maneira que seus primeiros caracteres estejam todos para a direita e nas mesmas colunas e seus caracteres são analisados da direita para a esquerda. Este procedimento é interrompido quando encontra-se (se existir) um alinhamento entre uma coluna com um caracter e um coringa. Se o caracter for do padrão *P,* então *P* tem forma de prefixo menor que *Q*.

Com esta definição é possível ordenar os padrões em ordem de prefixo e sufixo.

O algoritmo de convolução pode ser descrito como se segue [BDV+00]:

Fase de convolução

- Para cada padrão elementar *P* (começando com o padrão maior na ordem por prefixo), tentar estendê-lo com outro padrão elementar.

Estendendo o padrão P

- Enquanto existir um padrão elementar Q, que possa ser "colado" ao lado esquerdo de P:
    - Tome o padrão Q que é maior na ordem por sufixos
    - Seja R o padrão resultante da colagem de Q ao lado esquerdo de P
    - Se o padrão R tiver pelo menos K ocorrências e for maximal com relação ao conjunto de padrões já relatados:
        - Tente estender o padrão R com outros padrões elementares (usando o procedimento "Estendendo o padrão P" recursivamente)
        - Se o padrão R tiver o mesmo número de ocorrências do que o padrão P, então o padrão P não é maximal e não é necessário buscar por extensões de P (saída do procedimento). Caso contrário R não é um padrão significativo.
- Repitir este mesmo processo para os padrões elementares que podem ser "colados" ao lado direito de P (começando com o padrão maior na ordenação por prefixos).
- Relatar o padrão P.

2.2.4.4 Desempenho

Pode ser mostrado que o TEIRESIAS produz todos os padrões <*L*,*W*> maximais. Para detalhes veja [RF98b] [BDV+00].

2.2.4.5 Comentários

O algoritmo TEIRESIAS é exato, ou seja, garante-se que todos os padrões <*L*,*W*> maximais que ocorrem em pelo menos K sequências são encontrados. No entanto, o número de tais padrões pode ser alto. Em particular, [PRF+00] mostra que este número pode ser exponencial. No entanto, tal situação não parece acontecer com dados reais. Por exemplo, no banco de dados GenPept que contém 120 mil aminoácidos, existem somente 27mil padrões maximais [PRF+00]. Há estudos experimentais que sugerem que o tempo de execução do TEIRESIAS é linear com o número de padrões gerados na saída [RF98a] [BDV+00].

Outra desvantagem do TEIRESIAS é que a forma dos padrões não é flexível. Os únicos descasamentos permitidos são caracteres coringas. As versões mais novas do TEIRESIAS [RFP+00] permitem padrões com caracteres ambíguos que representam grupos específicos de caracteres de $\sum$. Mas, os padrões do TEIRESIAS não permitem buracos com tamanhos flexíveis. Esta questão pode ser tratada por uma fase pós-processada, onde pode-se combinar padrões, encontrados em padrões maiores, separados por buracos flexíveis [RF98a]. No entanto, este método não garante que todos os padrões da forma especificada serão encontrados [BDV+00].

2.2.4.6 Trabalhos Relacionados ao TEIRESIAS

Uma das desvantagens do TEIRESIAS é que o seu tempo de execução é exponencial. Esta característica foi tratada em [PRF+00] por um novo algoritmo. Este algoritmo impõe mais restrições ao conjunto de padrões escritos na saída. Foi definido um conjunto de padrões não redundantes de tal forma que todos os outros padrões podem ser obtidos a partir deste conjunto. O tamanho deste conjunto é $3n$ onde n é o comprimento da entrada. Foi descrito um algoritmo que encontra todos os padrões não redundantes em um tempo $O(n^3 \log n)$.

Um padrão maximal P é redundante se existir um conjunto de padrões maximais P_1, ..., P_l tais que todas as ocorrências de P são também ocorrências de pelo menos um dos padrões P_1, ...,P_l e todas as ocorrências de P_i (para $1 \leq i \leq l$) é também uma ocorrência de P.

Por exemplo, considere os padrões P_1 = 'AB.D', P_2 = 'A.DDE' e P_3 = 'A..D'. Seja o seguinte conjunto de sequências {FABDDE, ABCDCCD, DDACDDE}. A lista de ocorrências do padrão P_1 neste caso será L_1={(1,2),(2,1)} e a lista de ocorrências do padrão P_2 neste caso será L_2={(1,2),(3,3)}. Cada par representa o número da sequência e a posição na sequência onde o padrão é encontrado. A lista de ocorrências do padrão P é $L_1 \cup L_2$ = {(1,2),(2,1),(3,3)} e, consequentemente, o padrão P é redundante. No entanto, se a sequência EAEDDD for adicionada, o padrão P não será mais redundante porque seu conjunto L será igual a {(1,2),(2,1),(3,3),(4,2)}.

A lista de ocorrências do padrão redundante pode ser construída a partir da lista dos padrões não redundantes. Não é necessário examinar o conjunto original de sequências. Esta característica torna polinomial o tempo de execução do algoritmo que descobre todos estes padrões.

Há ainda um algoritmo diferente, o SPLASH, que descobre padrões do formato do TEIRESIAS. Para detalhes, consulte [Cal00].

3 Métodos Heurísticos Iterativos

Até agora foram descritos os principais algoritmos que descobrem o melhor padrão. No entanto, não é tão simples fazer o mesmo em descoberta de padrões mais complicados. Neste caso são necessárias heurísticas que não necessariamente descobrem o melhor padrão, mas convergem para um máximo local. Um exemplo de algoritmo que utiliza esta técnica é o Gibbs, que será descrito a seguir.

3.1 Gibbs Sampling

[LAB+93] apresenta uma heurística para descoberta de padrões baseada no método Gibbs de amostragem. Em sua versão mais simples, procura-se o melhor padrão conservado, sem buracos, de tamanho fixo W em uma matriz de pontuações ou PWM (descrita anteriormente).

O algoritmo funciona com iterações. A entrada é um conjunto de sequências que se deseja encontrar um padrão. O resultado de cada iteração é um conjunto de subsequências de comprimento W – exatamente uma para cada sequência de entrada. Este conjunto representa as ocorrências do padrão nas sequências. A PWM caracteriza o padrão a partir deste conjunto de ocorrências.

O algoritmo é o seguinte:

1. Passo inicial: Selecione randomicamente uma subsequência de comprimento W de cada sequência de entrada. Estas subsequências formarão o conjunto inicial de ocorrências do padrão. Chame o_i a ocorrência do padrão na sequência i.
2. Passo de Iteração:
    a. Tome randomicamente uma sequência i.
    b. Calcule a PWM baseada em todas as ocorrências exceto o_i. Chame esta matriz de P.
    c. Tome cada subsequência de comprimento W da sequência i e calcule a pontuação dela de acordo com a matriz P.
    d. Escolha uma nova ocorrência o'_i entre todas as subsequências de comprimento W da sequência i usando a distribuição de probabilidade definida pelas pontuações (quanto maior a pontuação, maior a probabilidade).
    e. Substitua o_i por o'_i no conjunto de ocorrências do padrão.
3. Repita o Passo de Iteração até que alguma condição de parada aconteça.

Note que o passo 2.d da Figura 9 tem como objetivo encontrar a subsequência de comprimento *W* da sequência *i* mais próxima de PWM, ou seja, mais próxima do padrão.

O algoritmo de Gibbs não garante que a PWM e o conjunto de ocorrências encontrados serão os melhores, mas converge para um máximo local (ao invés de um máximo global). Por outro lado o método é rápido, o que o torna viável em várias aplicações.

Várias questões relacionadas a este algoritmo e seus padrões têm sido identificadas em trabalhos subsequentes e serão descritas a seguir [BDV+00].

3.1.1 Deslocamentos

Suponha que o conjunto ótimo de ocorrências inicie-se nas posições 8, 14, 22 e 7 de determinadas sequências. Se a posição 21 é obtida ao processar a terceira sequência, o sistema inteiro converge para o conjunto de ocorrências 7, 13, 21 e 6. Este problema foi tratado em [LAB+93] introduzindo um novo passo randômico no algoritmo que calcula pontuações deslocadas por vários caracteres.

Neste passo são calculadas pontuações para todas as possibilidades e são usadas escolhas randômicas com distribuição de probabilidade correspondente. Um método similar foi usado em PROBE [NLL+97] onde, de uma maneira similar, os padrões foram reduzidos ou extendidos em ambos os lados [BDV+00].

3.1.2 Padrões Múltiplos

Algumas vezes é apropriado definir um padrão como uma sequência de várias subsequências consecutivas de comprimento fixo separado por buracos de comprimentos variáveis. Isto significa que uma ocorrência no procedimento de Gibbs seria, neste caso, representada por várias subsequências pequenas em um sequência ao invés de uma. É possível modificar o procedimento de Gibbs [LAB+93,NLL+97] usando programação dinâmica no processo de classificação e escolha da nova ocorrência candidata. Os comprimentos e a quantidade de subsequências são especificadas de antemão [BDV+00].

3.1.3 Largura dos Padrões

Foi suposto anteriormente que a largura do padrão é fixa e que os usuários davam as sequências de entrada.

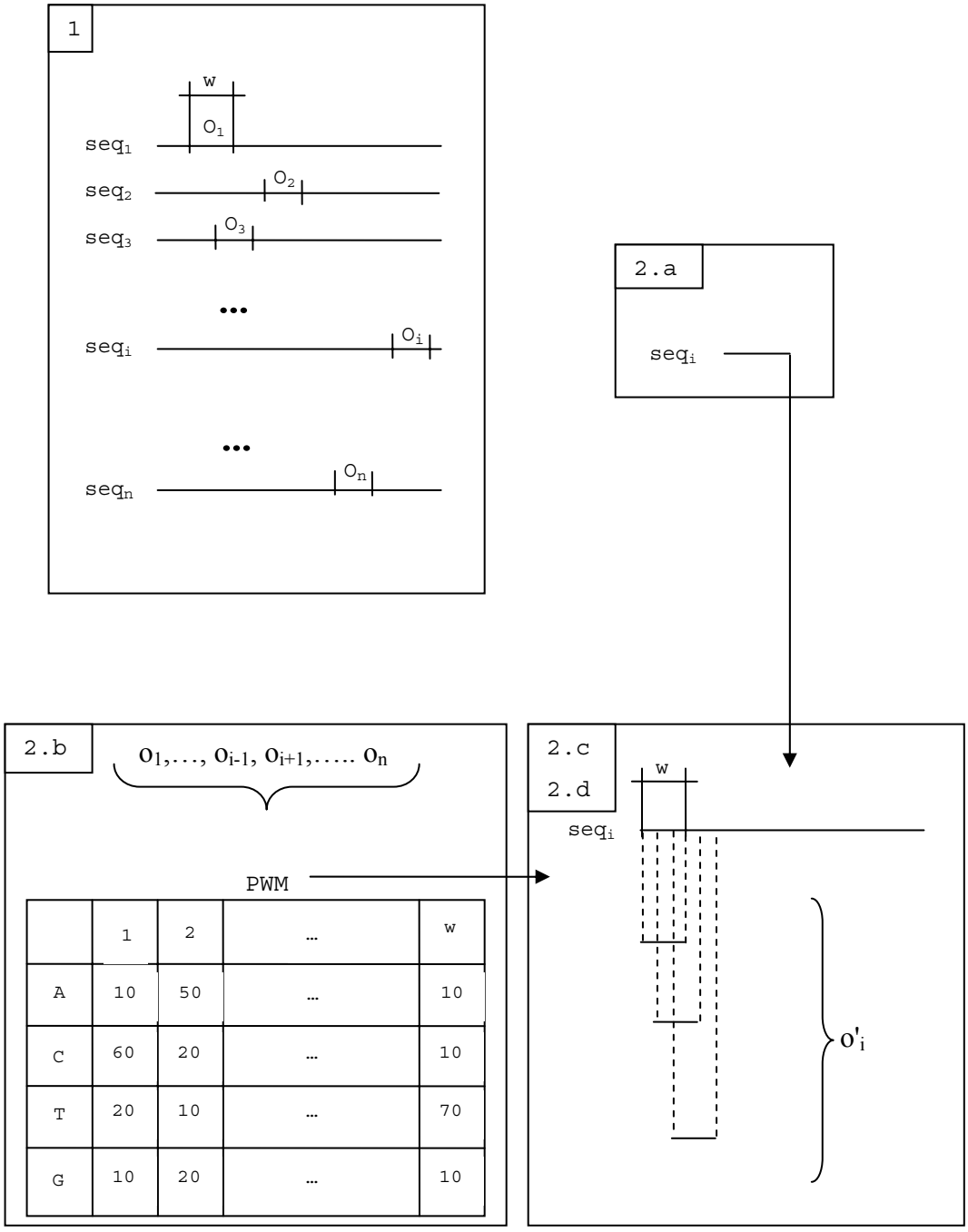


Figura 9. Esquema do algoritmo Gibbs.

Na maioria das vezes isto não é razoável, especialmente no tratamento de padrões múltiplos separados por buracos com tamanhos variáveis.

Um algoritmo genético é usado em PROBE [NLL+97] para determinar os parâmetros dos padrões, ou seja, o número de subsequências e seus comprimentos. Quando se tem dois conjuntos de parâmetros, é possível combiná-los e, em alguns casos, formar um conjunto melhor. Um conjunto de padrões é escolhido randomicamente com distribuição proporcional às suas pontuações [BDV+00].

3.1.4 Padrões com Buracos

Nem todas as posições dentro da subsequência de comprimento W são necessariamente importante. Em alguns casos é interessante considerar padrões com buracos, ou seja, somente J posições das W que formam a subsequência são relevantes (sendo J<W).

Este caso foi tratado em [LNL95]. Os autores sugeriram introduzir outro passo no procedimento Gibbs onde é substituída uma das J posições por uma das W-J+1 posições, que não estão incluídas no padrão. Novamente a escolha é feita randomicamente com distribuição de probabilidades proporcional às pontuações correspondentes [BDV+00].

3.2 Outros Métodos Iterativos

Existem diversos outros métodos que usam métodos iterativos similares ao Gibbs. Um método típico é começar com algum padrão, encontrar ocorrências deste padrão nas sequências, e construir um novo padrão que se case melhor com as ocorrências. Este processo é repetido com o novo padrão até que nenhuma melhora seja obtida. A principal diferença entre este método e o de Gibbs é que todas as sequências são usadas para definição do novo padrão e então é refinada a posição do padrão em todas as sequências. Este processo é completamente determinístico [BDV+00].

Este tipo de método foi utilizado em [PS00] para descobrir padrões determinísticos sem buracos de um dado comprimento que se casasse com todas as sequências minimizando o número de descasamentos. Usando métodos diferentes eles obtiveram um conjunto de ocorrências de algum padrão candidato desconhecido e refinaram-o pelo método iterativo. Em cada passo, eles calcularam o novo padrão tomando o caracter mais frequente em cada posição. Posteriormente o método é melhorado, removendo as colunas não significativas e obtendo um padrão com buracos. Este

método também pode ser utilizado para descobrir padrões que não ocorrem em todas as sequências de entrada [BDV+00].

O objetivo de [SBK+98] foi detectar regiões em alça e em receptores da histidina cinase. As regiões em alça foram detectadas em outras famílias, mas apesar das propriedades estatísticas de tais regiões serem conhecidas, elas podem ser levemente diferentes nesta família. O objetivo foi encontrar a distribuição de resíduos e pares de resíduos separados por distâncias diferentes por uma janela deslizante de comprimento fixo. O processo inicia-se levando em consideração a distribuição de outras famílias. Baseado nisso, atribui-se pontuação a cada posição da janela deslizante e as melhores pontuações indicam posições onde estão localizadas as regiões candidatas a serem regiões em alça. Depois, calcula-se uma nova distribuição com amostras destas regiões candidatas. Este processo é iterativo: cada passo um pseudo-contador é adicionado a partir da distribuição conhecida de outras famílias. Um pseudo-contador é um contador de aminoácidos que finge ter mais aminoácidos do que realmente tem nos dados para evitar conclusões biológicas infundadas (posteriormente esta definição será explicada melhor). Diferentemente do outro método, este é randômico e também não pode divergir muito do padrão original devido aos pseudo-contadores [BDV+00].

Em geral parece que os métodos iterativos simples são úteis para melhora de padrões obtidos por outros métodos ou por dados diferentes. No entanto eles não são bons o suficiente para descobrir padrões sem nenhum conhecimento prévio [BDV+00].

3.3 PTAS

Alguns problemas associados à descoberta de padrões são NP-difíceis, isto é, não existe um algoritmo com tempo polinomial que o resolva. Um exemplo disso é o *Padrão de Consenso*, cujo objetivo é descobrir um padrão *P*, formado apenas por caracteres de $\sum$, e uma ocorrência deste padrão em cada sequência de entrada, minimizando o número total de descasamentos em todas as ocorrências [BDV+00].

Como não existe um algoritmo que garanta a descoberta do melhor padrão em um tempo razoável, deseja-se ter a garantia que o padrão encontrado tenha no máximo α descasamentos quando comparado ao padrão ótimo. O valor α é chamado de taxa de aproximação. Por exemplo, se $\alpha=2$, garante-se encontrar um padrão que tenha no máximo 2 descasamentos do melhor padrão possível para este conjunto de sequências de entrada [BDV+00].

Em alguns casos é possível construir um algoritmo que funcione para qualquer α (o usuário especifica a precisão desejada). No entanto quanto menor é a taxa de aproximação, mais demorado se torna a execução do algoritmo. Este tipo de algoritmo é chamado de esquema de aproximação polinomial ou PTAS (de *Polynomial-Time Approximation Schemes*, em inglês). [LMW99] mostra um PTAS para o problema de Padrão de Consenso, que é baseado em uma idéia repetida para diferentes padrões iniciais [BDV+00].

O PTAS recebe as sequências de entrada, o comprimento L desejado para o padrão e um parâmetro r. Ele encontra todas as combinações possíveis de r subcadeias de comprimento L obtidas das sequências de entrada. Cada combinação pode conter 0, 1 ou várias subcadeias de cada sequência, sendo que é permitido que algumas subcadeias se repitam por mais de uma vez. Se o comprimento total de todas as sequências é N, existem $O(N^r)$ combinações [BDV+00].

Para cada combinação de r subcadeias os seguintes passos são executados:

- Calcule o padrão principal P das r subcadeias. Este padrão tem em cada posição o caracter que ocorreu mais frequentemente nestas posições nas r subcadeias;
- Para cada sequência de entrada, encontre a melhor ocorrência de P;
- Baseado nestas ocorrências, calcule o novo padrão principal P';
- Encontre as melhores ocorrências de P'em todas as sequências e calcule o número de descasamentos (custo de P').

O resultado será o padrão P' que obtiver o menor custo. Note que o algoritmo executa um passo de iteração com o padrão obtido de cada combinação possível. O tempo de execução do algoritmo é $O(N^{r+1}L)$ e sua taxa de aproximação é $1+(4|\sum|A-4)/\sqrt{e}(\sqrt{4r+1}-3))$ para $r \geq 3$ [BDV+00].

O resultado é muito interessante no ponto de visa teórico de Ciência de Computação, mas o algoritmo não é muito prático no caso real [BDV+00].

Existe um programa chamado COPIA [LLM00] inspirado no PTAS, que é mais prático. No entanto é apenas uma heurística sem garantia de desempenho.

A enumeração de todas as possíveis combinações de r subcadeias é substituída por uma amostragem randômica e uma a iteração da melhora do padrão para cada combinação é substituída pela iteração até não existir mais melhoras (similar a [PS00]) [BDV+00].

4 Métodos de Machine Learning

Algumas vezes um padrão não consegue ser descrito através de um padrão determinístico simples. Quando isso acontece é possível expressá-lo através de modelos estocásticos, como por exemplo a PWM e o modelo de Hidden Markov (HMM).

Nesta seção, serão apresentados um método simples utilizando PWM e um método utilizando HMM.

4.1 Expectation Maximization

Em [LR90] foi apresentado um algoritmo de aprendizado simples chamado *Expectation Maximization* (EM). Seu objetivo é estimar parâmetros do modelo estocástico do padrão, que ocorre uma vez em uma posição desconhecida de cada sequência de uma dada família de sequências. A PWM é usada como um modelo estocástico de base em [LR90]. No entanto, o método pode ser facilmente estendido para modelos mais complicados, por exemplo um que permita buracos flexíveis. Em alguns casos o modelo pode ser escolhido de acordo com algum conhecimento prévio da família de sequências.

O algoritmo é iterativo. Os seguintes passos são repetidos:

- Passo E. Para todas as sequências s e para todas as posições em S, calcule a probabilidade do padrão em s começar nesta posição. Baseie o cálculo no modelo do padrão construído na iteração anterior ou nos parâmetros iniciais do modelo, caso seja a primeira iteração. Os parâmetros iniciais do modelo são geralmente determinados randomicamente.
- Passo M. Para todas as posições do padrão, calcule novas probabilidades dos caracteres em suas posições. Baseie o cálculo em todas as ocorrências do padrão calculadas no passo E. Estes valores formarão novos parâmetros do modelo padrão.

Note que, diferentemente dos métodos anteriores, este usa todas as possíveis ocorrências do padrão para obter uma matriz, em vez de apenas uma ocorrência escolhida em cada cadeia.

Como a maioria dos métodos iterativos, o principal problema do algoritmo EM é que, em vez de convergir para um máximo global, ele convergirá para um máximo local dependendo dos parâmetros iniciais do modelo. O outro problema, descrito em

[LR90], é assumir que todos os padrões ocorrem exatamente uma vez em todas as sequências.

[BE95] trata destes dois problemas com o algoritmo MEME, uma modificação do EM. O MEME baseia-se na suposição de que um padrão descoberto deve se parecer, aproximadamente, com pelo menos uma subsequencia descoberta no conjunto de dados. Além disso, as fórmulas dadas em [LR90] foram modificadas para que fosse possível considerar várias ou nenhuma ocorrência de um padrão em uma sequência.

4.2 Modelos Hidden Markov

O modelo Hidden Markov [Kr98] (HMM, de *Hidden Markov Model* em inglês) é um modelo estatístico aplicável a muitas tarefas em biologia molecular. O caso de uso mais popular é o perfil probabilístico de uma família de proteínas, chamado de perfil HMM. A partir de uma família de proteínas (ou DNA) um perfil HMM pode ser usado para buscar membros de uma família em um banco de dados.

4.2.1 De Expressões Regulares a HMMs

As expressões regulares podem ser usadas para caracterizar famílias de proteínas e são a base do banco de dados PROSITE.

O uso de expressões regulares é uma maneira muito elegante e eficiente de busca de algumas famílias de proteínas, mas é complicada para outras. A dificuldade está na flexibilidade dos caracteres que formam uma proteína que faz com que as expressões regulares se tornem abrangentes e complexas. Suponha o seguinte padrão:

1 A C A - - - A T G
2 T C A A C T A T C
3 A C A C - - A G C
4 A G A - - - A T C
5 A C C G - - A T C

(Este exemplo está sendo feito com DNA somente porque o número de caracteres é bem menor que o de aminoácidos.)

Uma expressão regular para este padrão é:

[AT][CG][AC][ACGT]*A[TG][GC],

que mostra que a primeira posição é A ou T, a segunda é C ou G, e assim por diante. O termo '[ACGT]*' indica que as 4 letras podem ocorrer qualquer número de vezes.
O problema com a expressão regular acima é que ela não distingue uma sequência que é completamente implausível como

T G C T - - A G G

que possui o caracter menos provável em cada posição, da sequência consenso

A C A C - - A T C

que possui os caracteres mais plausíveis em suas posições. A alternativa é dar pontuações às sequências de acordo com o seu alinhamento.
Para gerar uma pontuação é dito que existe uma probabilidade de 4/5=0,8 para um A na primeira posição e 1/5=0,2 para um T porque se observa que das 5 letras, 4 são As e 1 é T. Similarmente na segunda posição a probabilidade do C é 4/5 e do G 1/5, e assim por diante. Depois da terceira posição no alinhamento, 3 das 5 sequências têm 'inserções' de comprimentos diferentes, então é dito que a probabilidade de se ter uma inserção é de 3/5 e, consequentemente, 2/5 de não ter (que correspondem às sequências que possuem 3 buracos nas posições 3, 4 e 5). O diagrama seguinte mostra estas probabilidades.

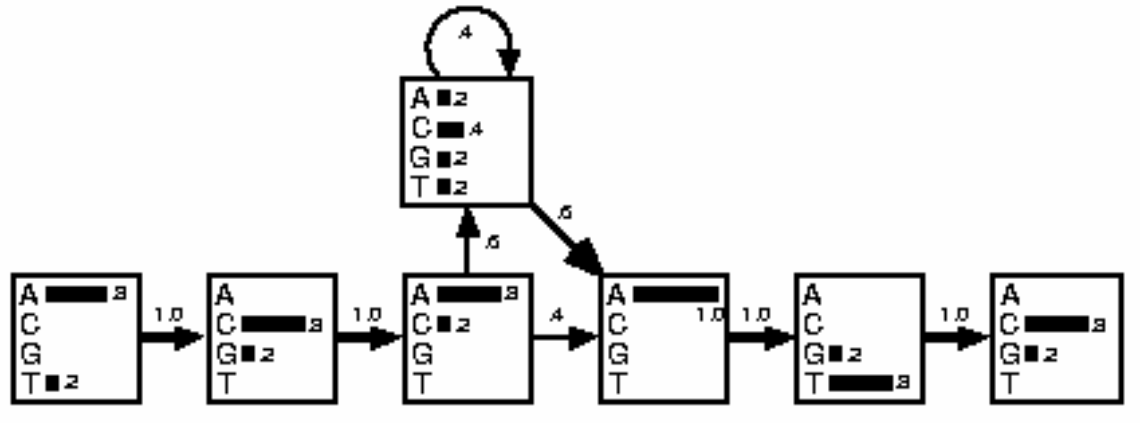


Figura 10. Expressão Regular do Modelo Hidden Markov.

Este é o modelo Hidden Markov. Cada box é chamado de estado e há um estado para cada termo da expressão regular. Todas as probabilidades são encontradas contando no alinhamento múltiplo quantas vezes cada evento ocorre, como descrito anteriormente. A única parte que aparentemente é complicada é a 'inserção' ou a parte

em que aparecem os buracos, que é representada por um estado acima dos outros. A probabilidade de cada letra é descoberta contando todas as ocorrências dos 4 nucleotídeos nesta região do alinhamento. No exemplo há um A, dois Cs, um G e um T com as probabilidades 1/5, 2/5, 1/5 e 1/5 respectivamente. Como as sequências 2, 3 e 5 têm uma inserção cada na posição 3, a sequência 2 tem ainda mais duas inserções nas posições 4 e 5. Consequentemente há 5 transições no estado de inserção e a probabilidade de se ter uma transição da inserção para ela mesma é de 2/5 e a probabilidade de se ter uma transição da inserção para o próximo estado é de 3/5.

Agora é simples gerar a pontuação da sequência consenso ACACATC. A probabilidade do primeiro A é 4/5. Isto é multiplicado pela probabilidade de transição do primeiro para o segundo estado, que é 1. Continuando esta idéia, a probabilidade total da sequência consenso é de:

$$P(ACACATC)=0,8 \times 1 \times 0,8 \times 1 \times 0,8 \times 0,6 \times 0,4 \times 0,6 \times 1 \times 1 \times 0,8 \times 1 \times 0,8$$
$$\approx 4,7 \times 10^{-2}$$

Fazendo o mesmo cálculo para a sequência implausível tem-se somente $0,0023 \times 10^{-2}$, que é 2000 vezes menor que a da consenso. Desta maneira se obtem o objetivo de atribuir uma pontuação a cada sequência, uma medida de quão bem ela se encaixa no padrão.

A mesma probabilidade pode ser calculada para as cinco sequências originais no alinhamento. Este resultado está na tabela da Figura 12. A probabilidade depende fortemente do comprimento da sequência. Consequentemente a própria probabilidade não é o número mais conveniente de ser usado como pontuação, e geralmente a pontuação *log-odds* apresentada na última coluna da tabela é melhor. A pontuação *log-odds* é o logaritmo da probabilidade da sequência dividido pela probabilidade de um modelo nulo. O modelo nulo utilizado aqui trata as sequências como cadeias randômicas de nucleotídeos e então a probabilidade da sequência de comprimento L é de $0,25^L$. Logo, a pontuação *log-odds* é

Log-odds para a sequência S = $log\ P(S) / 0,25^L = log\ P(S) - L log\ 0,25$.

O logaritmo natural foi utilizado na tabela, mas como os logaritmos são proporcionais, não importa qual será utilizado, mas é comum a base 2. É possível usar outros modelos nulos também, como, por exemplo, as frequências de todos os nucleotídeos no organismo estudado em vez de apenas 0,25.

Quando a sequência se encaixa bem no padrão o *log-odds* é alto. Quando ela se encaixa melhor com o modelo nulo, a pontuação *log-odds* é negativa. Note pela tabela que apesar da probabilidade da segunda sequência (a que tem 3 inserções) ser tão baixa quanto a da sequência implausível, a pontuação *log-odds* é muito maior. Infelizmente não é possível assumir sempre que qualquer sequência com pontuação *log-odds* positiva é um "acerto", porque existem casos de acertos randômicos quando se está fazendo uma busca em um banco de dados grande.

Em vez de se trabalhar com probabilidades é possível converter tudo para *log-odds*. Os números apresentados na tabela correspondem a aplicação do logaritmo na divisão da probabilidade do nucleotídeo pela probabilidade do modelo nulo (0,25 neste caso). As probabilidades de transição são também transformadas em logaritmos (veja Figura 11). Neste caso a pontuação *log-odds* pode ser calculada diretamente adicionando estes números em vez de multiplicando-os. Por exemplo, o cálculo do *log-odds* da sequência consenso é:

$$\textit{Log-odds}(\text{ACACATC}) = 1{,}16 + 0 + 1{,}16 + 0 + 1{,}16 - 0{,}51 + 0{,}47 - 0{,}51 + 1{,}39 + 1{,}16 + 0 + 1{,}16$$

$$= 6{,}64$$

(A precisão finita causa uma pequena diferença entre este número e o da tabela).

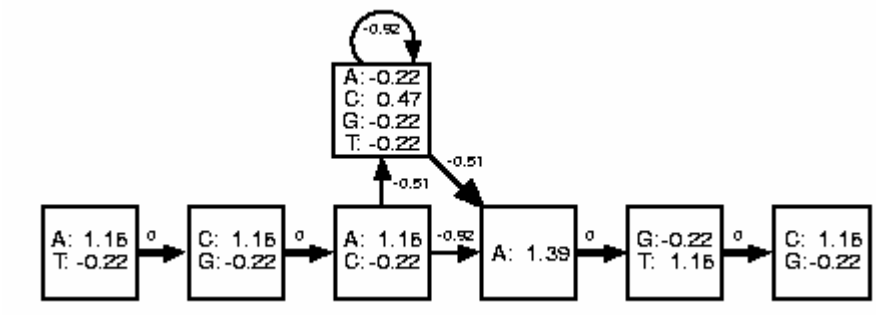


Figura 11. As probabilidades utilizando pontuação *log-odds*.

	Sequência	Probabilidade X 100	*Log-Odds*
Consenso	A C A C - - A T C	4.7	6.7
Sequências Originais	A C A - - - A T G	3.3	4.9
	T C A A C T A T C	0.0075	3.0
	A C A C - - A G C	1.2	5.3
	A G A - - - A T C	3.3	4.9
	A C C G - - A T C	0.59	4.6
Implausível	T G C T - - A G G	0.0023	-0.97

Figura 12. Probabilidades e pontuações *log-odds* para as 5 sequências, a sequência consenso e a sequência implausível.

Se o alinhamento não tivesse buracos ou inserções não existiria o estado de inserção, e então todas as probabilidades associadas com as linhas (as probabilidades de transição) seriam 1 e poderiam ser ignoradas completamente. Neste caso o HMM trabalharia exatamente como a PWM de pontuações *log-odds*, que é muito utilizada. É possível ver o HMM como uma generalização da PWM que incorpora inserções e remoções.

4.2.2 Perfis HMMs/Topologias de HMM

Um perfil HMM é um tipo de HMM cuja topologia representa penalidades a buracos de acordo com suas posições. Ele é construído através do alinhamento múltiplo e pode ser usado para busca em um banco de dados por membros de uma família de sequências. A topologia é apresentada na Figura 13. Ela especifica um layout geral do modelo usado para representar uma família de sequências.

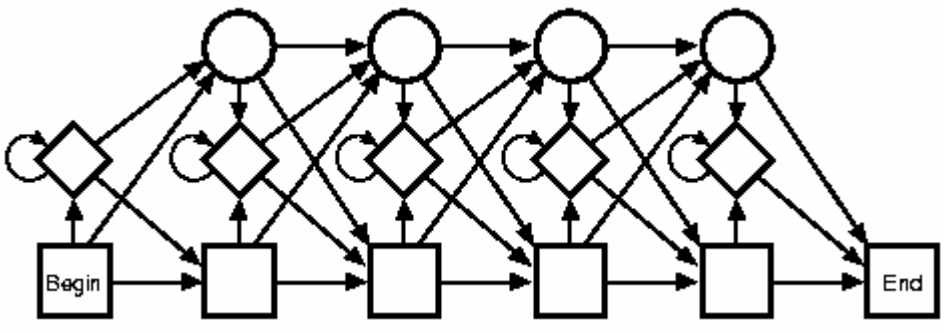


Figura 13. A estrutura da perfil HMM.

O modelo consiste de 3 tipos de estados: estados de casamento ou principais, estados de inserção e estados de remoção.

Os estados de casamento estão na linha de baixo e modelam as colunas conservadas do alinhamento e especificam a distribuição de probabilidade, que é apenas a frequência de aminoácidos ou nucleotídeos em uma determinada coluna do alinhamento como apresentada anteriormente.

A segunda linha de estados, com formato de diamante, representa os estados de inserção que são usados para modelar as regiões altamente variáveis no alinhamento, ou seja, para modelar possíveis buracos entre os estados de casamento. Elas funcionam exatamente como o estado do topo do modelo apresentado anteriormente (Figura 11).

E finalmente, os estados de remoção, com formato circular, permitem pular alguns estados de casamento para modelar a situação em que poucas sequências tem um buraco em uma posição do alinhamento múltiplo.

Veja o exemplo da Figura 14.

A Figura 14 apresenta um alinhamento múltiplo de 30 sequências de aminoácidos. As áreas cinzas são as mais conservadas e foram escolhidas para representar os estados de casamento do HMM. A área clara com letras minúsculas tem um alinhamento altamento incerto e foi escolhida para representar o estado de inserção. A Figura 15 representa o perfil HMM construído para este exemplo.

Para cada coluna da região conservada foi construído um estado e configuradas as probabilidades de transição iguais às frequências dos aminoácidos. Neste caso existem 14 estados de casamento. Existem duas transições de um estado de casamento para um estado de remoção apresentadas com linhas pontilhadas, do estado “begin” para o estado 1 de remoção e do estado de casamento 12 para o estado 13 de remoção. Ambos correspondem aos buracos nos alinhamentos. Em ambos os casos somente uma das 30 sequências contém buraco, e, portanto, a probabilidade destas transições de remoção é de 1/30. A quarta sequência continua no estado de remoção até o fim e, então, sua probabilidade de ir do estado 13 de remoção para o estado 14 de remoção é de 1, e do estado 14 de remoção para o estado “end” também é 1.

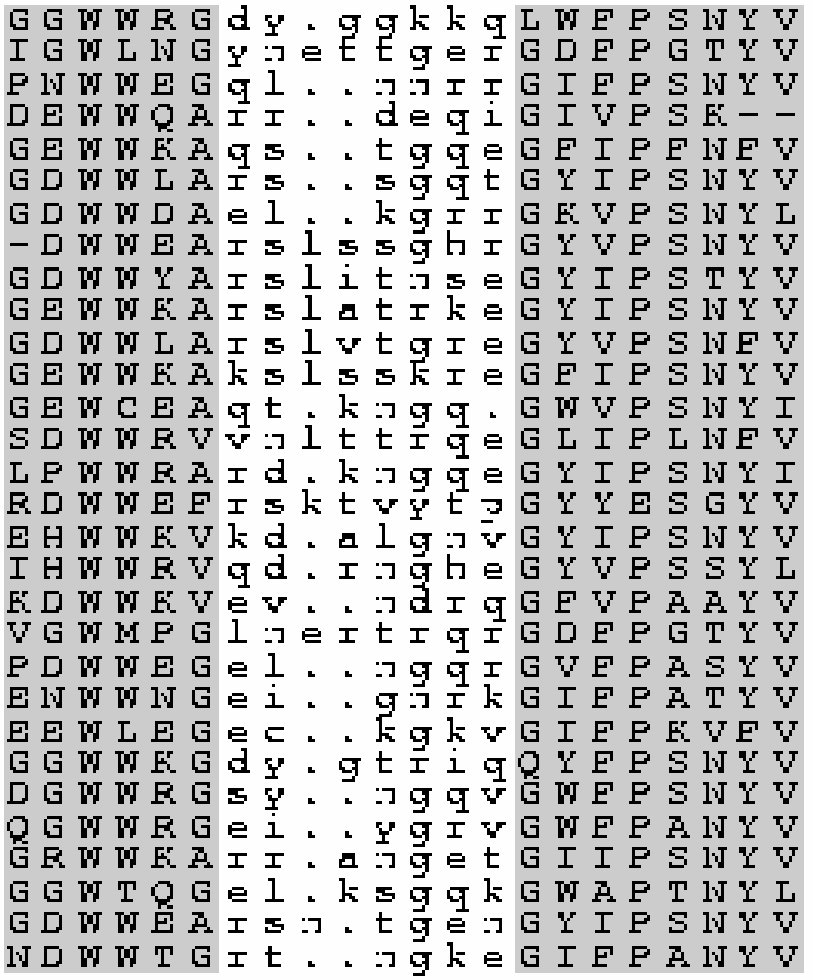


Figura 14. Alinhamento de 30 sequências de aminoácidos.

4.2.3 Pseudo-Contadores

A estimativa da distribuição de probabilidade a partir de poucos aminoácidos é perigosa. Se, por exemplo, existirem apenas duas sequências com leucinas em uma determinada posição, a probabilidade para leucina seria 1 e a probabilidade nesta posição para todos os outros aminoácidos seria 0, apesar de ser conhecido que a valina frequentemente substitui a leucina. Neste caso a probabilidade de uma sequência inteira facilmente se tornaria 0 se uma simples leucina fosse substituída por uma valina, ou equivalentemente, a pontuação *log-odds* tenderia a menos infinito.

Consequentemente é importante ter algum jeito de evitar estes tipos de conclusões fortes traçadas com pouca evidência. O método mais comum é o uso de pseudo-contadores que significa fingir ter mais aminoácidos do que realmente tem nos dados.

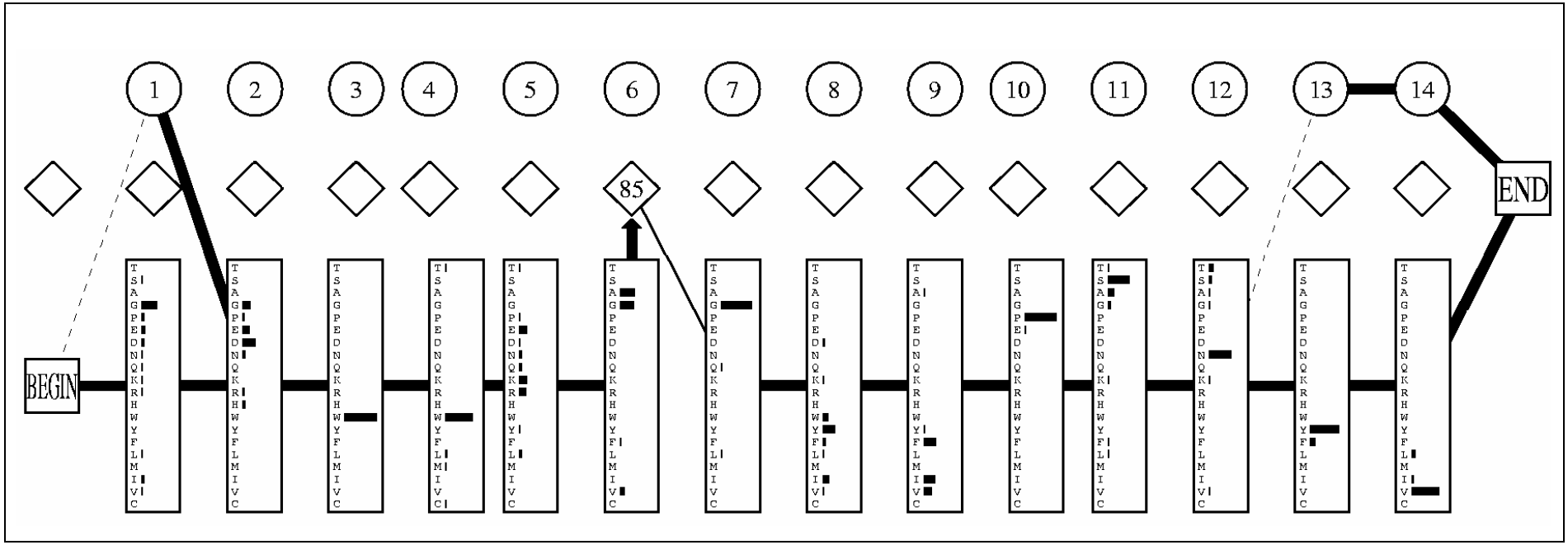


Figura 15. Um perfil HMM feito a partir do exemplo das 30 sequências.

A forma mais simples é adicionar 1 a todos os contadores de aminoácidos. No caso da leucina isto significa que a probabilidade de leucina seria estimada como 3/23 e outros 19 aminoácidos seriam 1/23. Na Figura 16 o modelo apresentado foi obtido do alinhamento da Figura 14 usando pseudo-contador igual a 1.

A adição de um a todos os contadores pode ser interpretada como uma suposição de que a priori todos os aminoácidos são equivalentemente iguais. No entanto, existem diferenças significativas na ocorrência dos 20 aminoácidos em sequências de proteínas conhecidas. Consequentemente o próximo passo é usar os pseudo-contadores proporcionalmente às frequências observadas dos aminoácidos. Este é o nível mínimo de pseudo-contadores que são usados em aplicações reais de HMMs.

Como a coluna de alinhamento pode conter informações sobre o tipo preferido de aminoácidos, é possível utilizar pseudo-contadores mais sofisticados. Se uma coluna consistir de leucinas predominantemente, pode-se esperar que substituições de aminoácidos hidrofóbicos sejam mais prováveis do que substituições de aminoácidos hidrofílicos. É possível, por exemplo, derivar pseudo-contadores para uma dada coluna a partir de matrizes de substituição. O método básico de utilização de pseudo-contadores é explicado em [KBM+94] e os mais avançados são discutidos em [BHK+93,TAK94,Kar95,HH96,SKB+96].

4.2.4 Busca em Bancos de Dados

Anteriormente foi apresentado como calcular a probabilidade de uma sequência em um alinhamento multiplicando todas as probabilidades (ou adicionando as pontuações *log-odds*) no modelo ao longo do caminho seguindo a sequência particular. No entanto, geralmente este caminho é desconhecido para outras sequências que não fazem parte do alinhamento original e se torna complicado a obtenção das pontuações de tais sequências. Se se encontrar um caminho através do modelo onde a nova sequência se encaixe bem seria possível gerar sua pontuação como anteriormente. Para tanto é necessário "alinhar" a sequência ao modelo. Isto se parece muito com o problema de alinhamento entre pares de sequências e de fato o mesmo tipo de algoritmo de programação dinâmica pode ser usado.

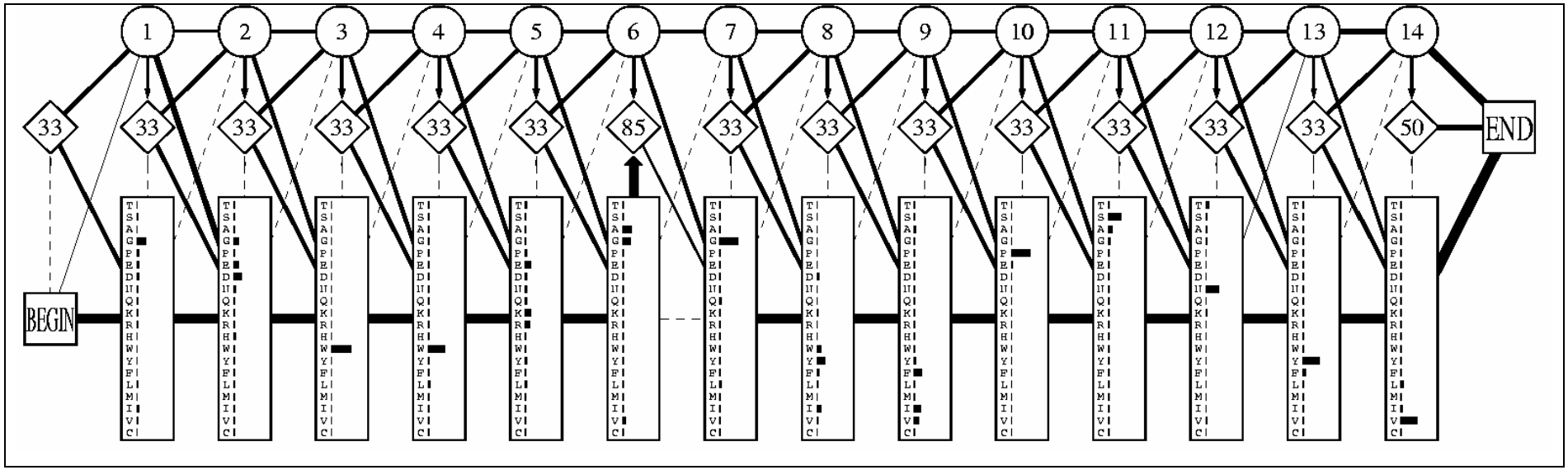


Figura 16. Um perfil HMM feito a partir do exemplo das 30 sequências utilizando pseudo-contadores.

Para uma sequência em particular, um alinhamento ao modelo (ou ao caminho) é uma atribuição de estados a cada caracter da sequência. Existem diversos alinhamentos possíveis para uma dada sequência. A idéia é obter todos estes alinhamentos e descobrir o "melhor" deles, ou seja, o de maior pontuação. Por exemplo, um alinhamento pode ser feito como se segue. Suponha que os aminoácidos sejam rotulados por A_1, A_2, A_3, etc. Analogamente suponha que os estados de casamento sejam M_1, M_2, M_3, etc e os estados de inserção sejam I_1, I_2, I_3, etc e assim por diante. Então um alinhamento poderia ter A_1 se casando com M_1, A_2 e A_3 se casando com I_1, A_4 se casando com M_2, A_5 com M_6 (depois passando por três estados de inserção) e assim por diante. Feito isso para cada caminho possível, calcula-se a probabilidade da sequência ou a pontuação *log-odds*, e depois descobre-se o "melhor" alinhamento. Embora o algoritmo de programação dinâmica descrito acima gere um número enorme de alinhamentos possíveis, ele pode ser feito eficientemente. Este algoritmo é conhecido como algoritmo de Viterbi [DEK+98] e ele também gera a probabilidade da sequência para o alinhamento e, consequentemente, a pontuação da sequência é obtida. Esta pontuação indica quão bem a sequência pode ser alinhada com o padrão.

4.2.5 Estimativa do Modelo HMM

Como foi visto o modelo HMM pode ser utilizado para representar uma família de proteínas. É possível treinar o modelo para que ele gere pontuações altas para as sequências similares às da família. Para tanto é necessário estimar parâmetros do modelo. O algoritmo Baum Welch [DEK+98] faz isso através de um método iterativo. Primeiro o método é iniciado com parâmetros de um modelo arbitrário. Caso se tenha um conhecimento prévio da família de sequências, ele pode ser utilizado. Por exemplo, pode ser que esteja disponível um alinhamento razoável de algumas sequências da família. Depois, em cada passo, calculam-se as probabilidades de todos os caminhos para todas as sequências, e reestimam-se os parâmetros do modelo a fim de maximizar a probabilidade das sequências de treinamento. Estas novas probabilidades podem levar a um alinhamento um pouco diferente. Se isto acontecer, o processo pode ser repetido e as probabilidades podem melhorar novamente. O processo é repetido até que nenhuma mudança ocorra. O alinhamento das sequências no modelo final gera um alinhamento múltiplo.

Apesar deste processo de estimativa soar simples, existem muitas questões que devem ser consideradas para que ele realmente funcione. Uma delas é a escolha do

comprimento do modelo apropriado, o que determina o número de inserções no alinhamento final. Outra questão complicada é que o processo iterativo pode convergir para soluções sub-ótimas. Não é garantido que ele descubra o alinhamento múltiplo ótimo, isto é, o mais provável.

Estimativas e alinhamento múltiplo são descritos em detalhe em [KBM+94], em [HK96] alguns métodos práticos são discutidos e em [BCH+94,Ed95] alguns métodos alternativos para estimativa de modelos são apresentados.

4.2.6 Melhorias ao Modelo HMM

Em [HK96] há uma idéia de ajustar o modelo HMM reduzindo seu número de parâmetros. Em particular os autores observaram os seguintes casos:

- Alguns estados de casamento são usados por pouquíssimas sequências. Neste caso, o estado pode ser removido e as sequências que o utilizavam são forçadas, neste ponto, a usar um estado de inserção ou a mudar seu alinhamento completamente.
- Alguns estados de inserção são usados por inúmeras sequências. Neste caso, o estado de inserção pode ser substituído por uma cadeia de estados de casamento. O número de estados de casamento inserido é igual ao número de inserções esperadas no estado de inserção substituído.

O Meta-MEME [GBE+97] usa um programa diferente para relatar padrões pequenos e simples. Estes padrões são transformados em estados de casamento no HMM. Os diferentes padrões são combinados usando estados de inserção como na Figura 17.

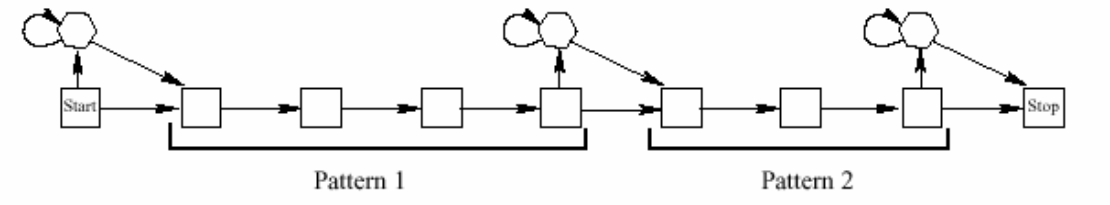


Figura 17. Topologia inicial do Meta-MEME. Os padrões encontrados por outros programas são conectados utilizando estados de inserção.

4.2.7 Descobrindo Sub-Famílias

Algumas vezes uma família de sequências pode consistir de várias sub-famílias. Isto significa que uma família é representada por vários padrões em vez de um só. Para

tratar deste caso, [KBM+94] combina vários HMMs em um único HMM como apresentado na Figura 18.

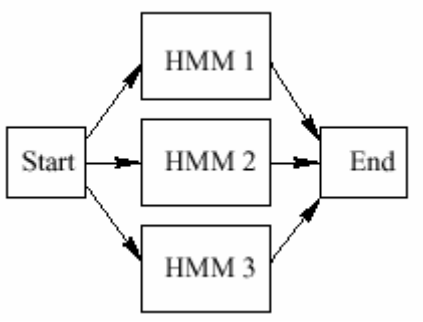


Figura 18. Topologia de HMM representando famílias de sequências com várias subfamílias.

5 Métodos que utilizam Informações Adicionais

A descoberta de padrões é uma tarefa complexa. Os métodos exaustivos não são suficientes para se descobrir um padrão flexível maior e, em muitos casos, não se conhece método que garanta um bom desempenho. Frequentemente o padrão que se quer descobrir é muito sutil, difícilmente distinguível em sequências randômicas.
Nesta seção serão discutidas outras origens de informação, além da própria sequência, que podem ser utilizadas para descoberta de padrões.

5.1 Descoberta de Padrões em Sequências Alinhadas

A descoberta de padrões e a descoberta de alinhamento local são tarefas relativamente próximas. Uma vez que há um alinhamento local entre múltiplas sequências é possível descobrir um padrão. O alinhamento local nos dá a posição em que um candidato a padrão está presente em uma determinada sequência. A partir disso é possível se obter um padrão na forma de uma sequência consenso, de uma PWM, etc. A questão fica mais interessante se for assumido que a entrada pode conter erros, i.e, que algumas sequências não estão alinhadas corretamente ou não são membros da família, etc. Neste caso é possível descobrir padrões que não se casam com todas as sequências e, talvez até, características de padrões para subfamílias. Esta tarefa é tratada pelo software EMOTIF [NWB98].
EMOTIF busca por padrões que contenham caracteres normais, ambíguos e buracos. O conjunto de caracteres ambíguos é fixo e é obtido por inspeção de frequências de substituição entre pares de aminoácidos em um banco de dados de famílias de proteínas alinhadas. Depois que o algoritmo de clusterização é aplicado, são produzidos conjuntos de clusters tais que aminoácidos dentro de um cluster são frequentemente substituídos uns pelos os outros e não são substituídos por outros aminoácidos fora do cluster. Os clusters resultantes são então usados como possíveis caracteres ambíguos.
Cada caracter (normal, ambíguo ou buraco) corresponde a uma coluna do alinhamento local. Para cada padrão é possível calcular sua especificidade (quão provável ele ocorre randomicamente) e sua cobertura (quantas sequências de treinamente ele cobre). O EMOTIF encontra padrões com diferentes valores de especificidade e cobertura. Para vários padrões com a mesma especificidade somente aquele com melhor cobertura é relatado e vice-versa. Os padrões são descobertos através de uma

busca exaustiva podada, tentando aplicar todos os caracteres ambíguos em cada coluna e calculando a especificidade e a cobertura.

5.2 Propriedades Globais de uma Sequência

Como apontado em [PBC+99], sinais de sítios de ligação de fatores de transcrição são frequentemente muito fracos para serem distinguíveis em sequências aleatórias. No entanto, a célula é capaz de reconhecer o início destes sítios corretamente. Consequentemente esta falta de informação local indica que o início destes sítios é algo representado globalmente, em fitas longas de DNA.

Em casamento de padrões é possível utilizar informação global para distinguir ocorrências randômicas daquelas que tem significado funcional. Em descoberta de padrões é possível usar este tipo de conhecimento prévio para escolher partes apropriadas do genoma como conjunto de entrada.

5.3 Árvore Filogenética

Uma das suposições na descoberta de padrões em sequências biológicas é que as regiões conservadas na evolução são funcionalmente importantes. Consequentemente é natural usar relações filogenéticas conhecidas entre as sequências para guiar a busca de padrões.

Suponha que se queira encontrar um elemento regulatório. Em vez de usar regiões regulatórias de vários genes correlacionados da mesma espécie, podem ser usados regiões regulatórias do mesmo gene de várias espécies relacionadas. Assumindo que a árvore evolutiva destas espécies é conhecida, é possível tentar descobrir um padrão pequeno melhor conservado na evolução. Este método é utilizado em [BST00].

5.4 Estruturas Secundária e Terciária

As posições importantes na estrutura secundária e terciária de proteínas são frequentemente conservadas. Se se conhece a estrutura de proteínas em questão, é possível tentar localizar as regiões importantes para obter tal estrutura. Este método foi utilizado em [IBB+00]. Os autores encontraram pontos de contato entre dois elementos da estrutura secundária como sítios de posições conservadas. Eles tentaram construir um padrão determinístico esparso contendo caracteres ambíguos e buracos flexíveis. Um padrão deve cobrir um comprimento inteiro da proteína. Eles encontraram seus padrões principais manualmente. Primeiro eles usaram um software para descobrir pontos de interação entre elementos de estrutura secundária (cada

ponto é dado como um par de resíduos que se interagem). Depois eles alinharam manualmente as sequências de tal maneira que os pontos de contato se alinharam juntos quando possível. Baseados neste alinhamento eles escolheram posições bem preservadas (entre as colunas que continham muitos pontos de contato). Este conjunto de posições constituem um padrão, que é, então, testado em um banco de dados de proteínas. Se o desempenho do padrão não for satisfatório, ele é melhorado manualmente posteriormente. Apesar dos autores em [IBB+00] terem criados os padrões manualmente, é possível implementar um processo similar em um programa. A principal dificuldade será escolher funções de pontuação para incorporar o conhecimento biológico e a intuição usada nos diversos estágios da descoberta dos padrões feita pelos autores.

A estrutura secundária e a pesquisa de padrões estão muito ligadas desde [GHS97]. Neste artigo os autores procuraram por padrões conservados em sequências de RNA.

5.5 Descoberta de Repetições Tandem

Coward e Drablos [CD98] apresentaram um método iterativo de descoberta de repetições que é baseada no alinhamento de uma dada sequência com ela mesma, enquanto Benson [Ben99] apresentou um método probabilístico para este mesmo problema.

A primeira fase do algoritmos de Coward e Drablos é um processo de *alinhamento em fases*: uma sequência é dividida em subsequências, cada uma comprimento p, e uma medida de 'concordância mútua' entre as subsequências é calculado. Se a sequência é p-periódica, esta medida é zero. Se a sequência é periódica exceto por poucas substituições, a medida da 'concordância mútua' continuará sendo pequeno. No entanto, uma inserção ou remoção na sequência terá um grande efeito na medida. Neste caso, o efeito da inserção ou remoção poderá ser compensado pelas subsequências seguintes por um deslocamento na direita ou esquerda [CD98]. O algoritmo de alinhamento iterará ate´que um alinhamento ótimo ou próximo do ótimo seja encontrado.

A segunda fase do algoritmo envolve a descoberta de um *padrão consenso*, um candidato a padrão periódico base. O terceiro estágio, que é opcional, trata do ajustamento do alinhamento em fases de acordo com um padrão consenso. A fase final trata do cálculo das pontuações de fase para diferentes valores de periodicidade. Os autores relatam que seu algoritmo funciona melhor para repetições tandem de pelo

menos 12 a 15 caracteres, e que quando existem poucas inserções e remoções, a fase de concordância é uma boa indicação da presença de repetição.

Em contraste, o método probabilístico de Benson [Ben99] para encontrar repetições tandem é mais complicado. Ele modela o alinhamento de duas cópias tandem de um padrão de comprimento n por uma sequência de n-independentes tentativas Bernoulli (lançamento de moedas). O algoritmo tem dois componentes separados de detecção e análise: o componente detecção usa um conjunto de critérios estatísticos para descobrir candidatos a repetições tandem; o componente análise tenta produzir um alinhamento potencial para cada candidato, e se for bem sucedido, tenta encontrar várias estatísticas sobre o alinhamento (por exemplo, percentual de identidade) e da sequência (por exemplo, composição e medida de entropia) .

6 Comentários Finais

Este trabalho apresentou um levantamento dos principais algoritmos para descoberta de padrões biológicos existentes atualmente.

A seguir está uma tabela que resume os principais métodos apresentados.

Métodos com Busca Exaustiva	
Idéia Principal	Método
Enumeração de todos os padrões	MOTIF [SAC90]
Enumeração e busca em profundidade em uma árvore	Pratt [Jon96] e ASSET [NG94]
Descoberta de large clicks em um grafo	Winnover [PS00]
Combinação de padrões pequenos para formar padrões maiores	TEIRESIAS [RF98b]
Métodos Heurísticos	
Idéia Principal	Método
Amostragem Gibbs baseada em uma matriz	Gibbs [LAB+93]
Esquema de aproximação polinomial	PTAS [LMW99]
Métodos de *Machine Learning*	
Idéia Principal	Método
Modelo estocástico baseado em uma matriz	EM [LR90]
Modelo de Hidden Markov baseado em um grafo	HMM [Kr98]
Métodos que utilizam Informações Adicionais	
Idéia Principal	Método
Alinhamento local	EMOTIF [NWB98]
Árvores Filogenéticas	[BST00]
Estruturas secundária ou terciária	[IBB+00]

Agradecimentos

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pelo apoio financeiro.

Ao Dr. Antônio Basílio da FioCruz pela contribuição com seu conhecimento em Biologia.

7 Referências

[ASS+86] M.D.Atkinson, R.R.Sack, N.Santoro, T.Strothotte. Min-max Heaps and Generalized Priority Queues. *Commun.ACM*, 29, 996-1000, 1986.

[BBH95] A.Bairoch, P.Bucher, K.Hofman. The PROSITE Database, its Status in 1995. *Nucleic Acids Research.*, 24:189-196, 1995.

[BCH+94] P.Baldi,Y.Chauvin,T.Hunkapiller,M.A.McClure. *Proc. of the Nat. Acad. of Sciences of the U.S.A*. 91:1059-1063,1994.

[BDV+00] B.Brejová, C.DiMarco, T.Vinar, S.R.Hidalgo, G.Hoguin, C.Patten. Project Report for CS798g, University of Waterloo, 2000. Disponível em http://citeseer.nj.nec.com/brejova00finding.html.

[BE95] T.L.Bailey, C.Elkan. Unsupervised learning of multiple motifs in biopolymers using expectation mazimization. *Machine Learning*, 21(1/2):51-80, 1990.

[Ben99] G.Genson. Tandem Repeats Finder: a Program to Analyze DNA Sequences. *Nucleic Acids Research*, 27(2): 573-580, 1999.]

[BHK+93] M.Brown, R.Hughey, A.Krogh, I.S.Mian, K.Sjölander, D. Haussler. Using Dirichlet mixture priors to derive hidden Markov models for protein families. In *Proc. of First Int. Conf. on Intelligent Systems for Molecular Biology*, 47-55, Editores L. Hunter, D. Searls, J. Shavlik. Menlo Park, CA. AAAI/MIT Press,1993.

[BJE+97] A.Brazma, I.Jonassen, I.Eidhammer, E.Ukkonen. Relation Patterns and their Automatic Discovery in Biosequences. Report n^{o}.135. Department of Informatics, University of Bergen, 1997.

[BJE+98] A.Brazma, I.Jonassen, I.Eidhammer, D.Gilbert. Approaches to the Authomatic Discovery of Patterns in Biosequences. *Journal of Computational Biology*, 5(2): 2740-2746, 1998.

[BPG96] W.C.Barker, F.Pfeiffer, D.G.George. Superfamily classification in PIR-International Protein Sequence Database. *Methods in Enzymology*, 266:59-71, 1996.

[BST00] M.Blanchette, B.Schwikowski, M.Tompa. An Exact Algorithm to Identify Motifs in Orthologous Sequences from Multiple Species. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology* (ISMB), 37-45, 2000.

[Cal00] A.Califano. SPLASH: Structural Pattern Localization Analysis by Sequential Histograms. *Bioinformatics*, 16(4):341-347, 2000.

[CD98] E.Coward, F.Drablos. Detecting Periodic Patterns in Biological Sequences. *Bioinformatics*, 14(6):498-507, 1998.

[CLR90] T.H.Cormen,C.E.Leiserson,R.L.Rivest. *Introduction to Algorithms*. McGraw-Hill, New York, 1990.

[Day79] Dayhoff, M.O.. Atlas of Protein Sequence and Structure, eds. Schwartz, R.M. & Dayhoff, M.O. (Natl. Biomed. Res. Found., Washington, DC), vol.5, suppl. 3, pp. 353-358, 1979.

[DEK+98] R.Durbin, S.R.Eddy, A.Krogh, G.Mitchison. *Biological Sequence Analysis*. Cambridge University Press, 1998.

[Ed95] S.R.Eddy. Multiple Alignmet using Hidden Markov Models. In *Proc.of the Third Int. Conf. on Intelligent Systems for Molecular Biology*, 3:114-120. Editores C. Rawlings, D.Clark, R.Altman, L.Hunter, T.Lengauer, S.Wodak. Menlo Park, CA. AAAI Press, 1995.

[EJT00] I.Eidhammer, I.Jonassen, W.R.Taylor. Structure Comparison and Structure Patterns. *Journal of Computational Biology*, 7(5): 685-716, 2000.

[Flo99] Aristidis Floratos. Pattern Discovery in Biology: Theory and Applications. PhD Thesis. Department of Computer Science, New York University, 1999.

[GB86] Gribskov M., Burgess R.R.. Sigma Factors from E.coli, B.substilis, phage SP01 and phage T4 are Homologous Proteins. *Nucleic Acids Research*, 14:6745-6763, 1986.

[GBE+97] W.N.Grundy, T.L.Bailey, C.P.Elkan, M.E.Baker. Meta-MEME: motif based hidden Markov models of protein families. *Computer Applications in the Biosciences*, 13(4):397-406, 1997.

[GHS97] J.Gorodkin, L.J.Heyer, G.D.Stormo. Finding the Most Significant Common Sequence and Structure Motifs in a Set of RNA Sequences. *Nucleic Acids Research*, 25(18):3724-3732, 1997.

[GJ79] M.Garey, D.Johnson. Computers and Itractability: A Guide of the Theory of NP-Completeness. New York:W.H.Freeman and Co, 1979.

[GME87] Gribskov M., McLachlan A.D., Eisenberg D..Profile Analysis: Detection of Distantly Related Proteins. Proc. Natl. Acad. Sci. U.S.A. 84:4355-4358, 1987.

[GLE90] Gribskov M., Luthy, R., Eisenberg, D..Profile Analysis. *Methods Enzymol.* 183, 146-159, 1990.

[HH92] Henikoff S., Henikoff, J.G.. Amino acid Substituition Matrices from Protein Blocks. *Proc. Nat.Acad.Sci.*, *U.S.A*. 89,10915-10919, 1992.

[HH96] J. G. Henikoff, S. Henikoff. *Computer Applications in the Bio-sciences,* 12, 135—143,1996.

[HK96] R.Hughey, A.Krogh. Hidden Markov Models for Sequence Analysis: Extension and Analysis of the Basic Method. *Computer Applications in the Biosciences,* 12(2):95-107, 1996.

[IBB+00] J.C.Ison, M.J.Blades, A.J.Bleasby, S.C.Daniel, J.H.Parish, J.B.Findlay. Key Residues Approach to the Definition of Protein Families and Analysis is of Sparse Family Signatures. *Proteins*, 40(2):330-331, 2000.

[Jon96] I. Jonassen. Efficient Discovery of Conserved Patterns using a Pattern Graph. Technical Report118, Department of Informatics, University of Bergen, Norway, 1996.

[JCH95] I.Jonassen, J.F.Collins, D.G.Higgins. Finding Flexible Patterns in Unaligned Protein Sequences. *Protein Science*, 4(8):1587-1595, 1995.

[Kar95] Karplus, K. Evaluating regularizers for estimating distributions of amino acids. In *Proc. of Third Int. Conf. on Intelligent Systems for Molecular Biology*, 3:188-196. Editores C. Rawlings, D.Clark, R.Altman, L.Hunter, T.Lengauer, S.Wodak. Menlo Park, CA. AAAI Press, 1995.

[KBM+94] A.Krogh, M.Brown, I.S.Mian, K.Sjolander, D.Hauslle r. Hidden Markov Models in Computational Biology. Applications to Protein Modeling. Journal of Molecular Biology, 235(5):1501-1531, 1994.

[Kr98] A.Krogh. An Introduction to Hidden Markov Models for Biological Sequences. Em Computational Methods in Molecular Biology. Editado por S.L.Salzberg, D.B.Searls, S.Kasif, 45-63. Elsevier, 1998.

[LAB+93] C.E.Lawrence, S.F. Altschul, M.S.Boguski, J.S.Liu, A.F.Neuwald, J.C.Wooton. Detecting Subtle Sequence Signals: a Gibb Sampling Strategy for Multiple Alignment. *Science*, 262 (5131): 208-214, 1993.

[LR90] C.E.Lawrence, A.A.Reilly. An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins*, 7(1):41-51, 1990.

[LXB94] Lüthy, R., Xenarios, I., Bucher, P.. Improving the Sensitivity of the Sequence Profile Method. *Protein Science*, 3: 139-146, 1994.

[Mou99] D.W.Mount. Pattern Searching in DNA and Protein Sequences, 1999. Disponível em http://www.blc.arizona.edu/courses/bioinformatics/patterns.html.

[NWB98] C.G.Nevil-Manning, T.D.Wu, D.L.Brutlag. Highly Specific Protein Sequence Motifs for Genome Analysis. *Proceedings of the National Academy of Sciences of the United States of America*, 95(11):5865-5871, 1998.

[NG94] A.F.Neuwald, P.Green. Detecting Patterns in Protein Sequences, *Journal of Molecular Biology*, 239:689-712, 1994.

[PBC+99] A.G.Pederson, P.Baldi, Y.Chauvin, S.Brunak. The Biological of Eukariotic Promoter Prediction – a Review. Computers and Chemistry, 23(3-4):191-207, 1999.

[PRF+00] L.Parida, I.Rigoutsos, A.Floratos, D.Platt, Y.Gao. Pattern Discovery on Character Sets and Real-valued Data: Linear Bound on Irredundant Motifs and an Efficient Polynomial Time Algorithm. *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 297-308, 2000.

[PRO02a] PROSITE - Database of Protein Families and Domains. Disponível em http://ca.expasy.org/prosite.

[PRO02b] ftp://www.expasy.org/databases/prosite/release/profile.txt.

[PS00] P.A.Pevzner, S.H.Sze. Combinatorial Approaches to Finding Subtle Signals in DNA Sequences. *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 269-278, 2000.

[Pri02] PRINTS: Protein Fingerprints Database. Bioinformatics Unit, University of Manchester, 2002. Em http://www.bioinf.man.ac.uk/dbbrowser/PRINTS/.

[Pro02] PROSITE. Swiss Institute of Bioinformatics, Expert Protein Analysis System (ExPASy), 2002. Em http://www.expasy.ch/prosite/.

[RF98a] I. Rigoutsos, A.Floratos. Combinatorial Pattern Discovery in Biological Sequences: The TEIRESIAS Algorithm. *Bioinformatics*, 14(1):55-67. Errata apareceu em *Bioinformatics*, 14(2):229, 1998.

[RF98b] I. Rigoutsos, A.Floratos. Motif Discovery without Alignment or Enumeration (extended abstract). *Proceedings of the Second Annual International Conference on Computational Molecular Biology (RECOMB)*, 221-227, 1998.

[RFP+00] I. Rigoutsos, A.Floratos, L.Parida, Y.Gao, D.Platt. The Emergence of Pattern Discovery Techniques in Computational Biology. *Metabolic Engineering*, 2(3):159-167, 2000.

[Rob85] Autores. *Titulo*, Local, Ano.

[SAC90] H.O. Smith, T.M.Annau, S.Chandrasegaran. Finding Sequence Motifs in Groups of Functionally related Proteins. *Proceedings of the National Academy of Sciences of the United States of America*, 87(2):826-830, 1990.

[SKB+96] K.Sjölander, K.Karplus, M.Brown, R Hughey, A.Krogh, I. S. Mian, D.Haussler, CABIOS 12:327-345,1996.

[TAK94] R. L. Tatusov, S. F. Altschul, E. V. Koonin. *Proc. of the Nat. Acad. of Sciences of the U.S.A.* 91: 12091-12095, 1994.

[Tom99] M.Tompa. An Exact Method for Finding Short Motifs in Sequences, with Application to the Ribosome Binding Site Problem. *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 262-271, 1999.

[VAC98] J.van Helden, B.Andre, J.Collado-Vides. Extracting Regulatory Sites from the Upstream Region on Yeast Genes by Computational Analysis of Oligonucleotide Frequencies. *Journal of Molecular Biology*, 281(5):827-832, 1998.