Identificação de Genes por Comparação de Seqüências

Said Sadique Adi

TESE APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO GRAU DE DOUTOR
EM
CIÊNCIA DA COMPUTAÇÃO

Área de Concentração: Ciência da Computação
Orientador: Prof. Dr. Carlos Eduardo Ferreira

– Durante o desenvolvimento deste trabalho, o autor recebeu apoio financeiro da FAPESP –

— São Paulo, abril de 2005 —

Identificação de Genes por Comparação de Seqüências

Este exemplar corresponde à redação
final da tese devidamente corrigida
e defendida por Said Sadique Adi
e aprovada pela comissão julgadora.

São Paulo, 25 de junho de 2005.

Banca examinadora:

- Prof. Dr. Carlos Eduardo Ferreira (orientador) (IME-USP)
- Prof. Dr. Alair Pereira do Lago (IME-USP)
- Prof. Dr. André Ponce de Leon Ferreira de Carvalho (ICMC-USP)
- Profa. Dra. Leila Ribeiro (UFRGS)
- Prof. Dr. Nalvo Franco de Almeida Junior (UFMS)

Aos meus verdadeiros amigos
Ashjan, Kaher e Sandro e aos
meus verdadeiros orientadores
Rahma Beirat e Sadique Adi

Agradecimentos

Impossível começar esta seção de agradecimentos sem um muito obrigado todo especial aos meus pais e irmãos. Sem as suas palavras de consolo, apoio constante e a confiança que sempre depositaram em mim, eu não teria chegado até aqui. Na realidade, sem essas pessoas que prezo muito, eu não teria chegado a lugar algum.

Falando em ensinamentos, deixo aqui os meus agradecimentos mais sinceros ao meu orientador, professor Carlos Eduardo Ferreira. Nesses quase sete anos de convivência acadêmica, vários foram as lições transmitidas a mim por essa pessoa. Com o Carlinhos aprendi o verdadeiro significado da palavra pesquisa, e espero ter retribuído toda a atenção dispensada por ele a mim. Se algum dia me perguntarem por um exemplo de orientador, não terei dúvida quanto a minha resposta. Além do Carlinhos, vários outros professores do IME-USP participaram de forma direta ou indireta de minha formação. Por isso, não posso deixar de agradecê-los. Muito obrigado ao Coelho, Cris, Siang, Yoshiko, Zé e Alair. Neste trabalho não cabe nem a metade de tudo que aprendi com vocês durante os meus anos de pós-graduando em nosso instituto. Agradecimentos com um gosto de saudade eu deixo aqui para a professora Marie-France Sagot. Muito obrigado por ter feito do meu sanduíche em Lyon uma experiência fascinante e extremamente enriquecedora. Muito mais que uma co-orientadora, a Marie foi uma amiga, cujas palavras e atitudes deveriam ser tomadas como exemplo por todos.

Mesmo rodeado pelos membros de uma excelente família e ótimos professores, eu nada seria sem os meus amigos. Por isso meus agradecimentos verdadeiros aos companheiros de república Fábio, Leonardo e Raphael. A este último, meu muito obrigado pelo papel de irmão que exerceu desde a minha chegada em São Paulo. Ao Fábio, meus agradecimentos pelos momentos de descontração e companheirismo no dia a dia. Ao Léo, pela calma e tranqüilidade que apenas ele, como um bom japonês, sabe passar. Um agradecimento especial aos amigos Charlie Brown, Claus, Emmanuel, Fábio Henrique, Marco Aurélio, Débora, Eliany e Valguima que, juntos, constituíram a minha família paulistana. Posso afirmar, sem receio nenhum, que, sem eles, tudo isso seria bem mais difícil. Aos amigos de longa data Amaury e Marcelo o meu valeu pela amizade constante. Sei que, mesmo longe, vocês dois sempre foram os que mais torceram por mim. Um obrigado intenso aos amigos mais recentes Andréia, Caio, Fernandinha, Fernanda, Gisela, Hilde, Lourdes, Rita, Patrícia e Maité por terem transformado a nossa sala de estudos. Sentirei muita falta de vocês. Aos amigos mais distantes Alexis, Camille, Celine, Emillie, Jeane, Laurent, Leonor, Manon, Rachid, Simon e Vincent pelos momentos intensos vividos em Lyon.

Finalmente, meus agradecimentos a todos os funcionários do IME. O que seria desse instituto sem eles? Um obrigado à dona Maria pelo café de todos os dias, ao Oswaldo pelas fotocópisa de todas as horas, à Margareth pelos reembolsos freqüentes, ao Marcos pela vigilância constante, à Selma pela administraçao da rede, ao Sérgio pelo apoio

técnico em todos seminários e ao Max pelos serviços da biblioteca. Agradecimento especial ao Feijão, Leca, Marilucia, Patrícia, Pinho e Rose que, de colegas de trabalho, transformaram-se em amigos com as quais sempre pude contar. É por essas e outras pessoas que tudo isso aqui vale a pena!

Said Sadique Adi

Resumo

Com os avanços das técnicas de seqüenciamento das moléculas de DNA, nos deparamos atualmente com um número expressivo de seqüências genômicas a serem analisadas. Dentre as várias informações de interesse guardadas em uma seqüência desse tipo está a localização de cada um de seus genes. O problema da identificação de genes em seqüências eucarióticas continua em aberto mesmo após os avanços obtidos com o desenvolvimento de métodos e ferramentas computacionais dedicadas a ele. Nesse trabalho propomos três heurísticas distintas para a tarefa de localização dos genes em uma seqüência de DNA eucariótica. Todas elas se baseiam na comparação de duas ou mais seqüências genômicas. A primeira dessas heurísticas trata a tarefa de identificar os genes em uma seqüência de interesse comparando-a com uma seqüência de cDNA. Isso é feito por meio da busca por um alinhamento especial entre duas seqüências denominado alinhamento *spliced*. Na segunda heurística, abordamos o problema do alinhamento sintênico entre duas seqüências e sua aplicação na tarefa de identificação de genes. Por último, temos uma heurística que trata o problema da identificação de genes seguindo uma nova abordagem. Nela, várias seqüências genômicas são comparadas na busca pelos seus genes. A precisão de cada um dos nossos programas compara-se àquelas de outros programas de predição descritos na literatura.

Abstract

With the advances in DNA sequencing technology, we now face the task of analyzing an expressive number of genomic sequences. Among the main information of interest encoded in a DNA sequence is the location of its genes. The gene prediction problem in eukaryotic sequences remains open even with recent advances achieved with the development of methods and tools dedicated to it. In this work we propose three heuristics to the task of finding the genes in a given eukaryotic DNA sequence. All of them are based on the comparison of two or more genomic sequences. The first one deals with the task of identifying the genes in a sequence of interest by comparing it with a related cDNA. This is accomplished by searching for a special type of alignment between these sequences called spliced alignment. To the development of the second heuristic, we discussed the syntenic alignment problem of and its application on the gene prediction task. Finally, we have developed a heuristic that deals with the gene prediction problem in an original way. In this approach, a number of sequences are compared when searching for their genes. The accuracy of each program compares with that achieved by another gene prediction tools in the literature.

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Um dos principais resultados dos avanços das técnicas de clonagem, clivagem e seqüenciamento das moléculas de DNA é o atual número de genomas total ou parcialmente seqüenciados. Analisar essas seqüências com o intuito de se obter informações úteis a respeito delas é o próximo passo na busca por soluções como o controle de pragas, prevenção de doenças, tratamento de endemias, etc. Dentre as várias informações de interesse guardadas em uma seqüência genômica está a localização e o papel desempenhado por cada uma de suas regiões funcionais. Ou seja, pelas regiões do genoma que desempenham alguma função durante o ciclo de vida celular. Dentre essas regiões, existe uma que merece atenção especial dada a sua ligação com várias das características que diferenciam os organismos de uma mesma classe, assim como os indivíduos de uma mesma espécie. Essas regiões são conhecidas como genes.

De forma simplificada, os genes correspondem às regiões das moléculas de DNA passíveis de serem transcritas em uma molécula de RNA. As moléculas de RNA desempenham uma série de funções dentro da célula e algumas delas podem ser traduzidas em uma ou mais proteínas por meio de um processo conhecido como Dogma Central da Biologia. Identificar os genes em uma seqüência genômica consiste em determinar os seus limites dentro dessa seqüência. Mesmo restringindo essa busca apenas aos genes que incluem a receita para a síntese de proteínas, temos aqui um problema de difícil solução. Isso se deve em grande parte ao fato de os genes não possuírem nenhuma propriedade específica que os distinguam das demais regiões constituintes de um genoma. Esse problema torna-se mais complicado ao focalizarmos nossas atenções sobre as seqüências genômicas dos organismos eucariotos. Os genes nelas codificados costumam incluir regiões distintas que participam de formas também distintas do processo de síntese de proteínas. Essas dificuldades levaram ao desenvolvimento de uma série de métodos computacionais que se propõem a determinar de maneira eficiente os limites dos genes em uma dada seqüência eucariótica de interesse.

Sobre os métodos existentes para identificação de genes, eles podem ser divididos em dois grandes grupos. No primeiro grupo, estão os que se baseiam apenas nas informações contidas na própria seqüência durante a tarefa de localização dos seus genes. No segundo grupo, encontramos os métodos que comparam a seqüência de interesse com outra seqüência relacionada no intuito de localizar os genes codificados na primeira. Os métodos de ambos os grupos foram desenvolvidos sobre a suposição biológica de que as regiões funcionais estão menos suscetíveis a mutações aleatórias que aquelas sem nenhum função aparente no genoma.

Apesar de grandes avanços na área, que se refletem no desenvolvimento de um número considerável de ferramentas de predição, o problema da identificação de genes ainda continua sem uma solução adequada. Análises das predições obtidas por várias dessas ferramentas indicam que seus resultados ainda estão bem longe daqueles esperados pelo usuário final. Além disso, esses estudos apontam para os métodos de identificação de genes por comparação de seqüências como os mais promissores na busca por uma solução adequada para o problema. Isso justifica pesquisas adicionais nesse campo, com o desenvolvimento de novas metodologias e programas de predição, baseados sobretudo na comparação da seqüência de interesse com outras.

Tendo em mente o fato de o problema da identificação de genes permanecer ainda em aberto, propomos neste trabalho três heurísticas distintas para ele, todas fundamentalmente baseadas na comparação de seqüências relacionadas. A primeira delas consiste em uma nova heurística para se identificar os genes em uma seqüência de interesse ao dispormos de uma seqüência de cDNA como base de comparação. Fundamentada sobre o problema de se determinar o caminho mínimo entre dois vértices de um grafo, essa heurística mostrou-se satisfatória quando implementada e comparada com outros programas de predição. O segundo resultado aqui apresentado consiste em uma nova heurística para o problema de identificar os genes em duas seqüências de DNA relacionadas evolutivamente. Ela possui como base a busca por um alinhamento dessas duas seqüências que bem reflete o fato delas incluírem regiões descontínuas de alta similaridade. Os testes realizados com a implementação dessa estratégia levaram a resultados comparáveis a de outros programas de predição. Finalmente, a terceira heurística aqui proposta corresponde a uma tentativa inicial de tratar o problema da identificação de genes por meio da comparação de várias seqüências de DNA. Essa nova heurística possui como base fundamental a tarefa de determinar um alinhamento múltiplo ótimo. Apesar dos poucos testes realizados com essa estratégia, os resultados obtidos mostram-se promissores. As publicações decorrentes do estudo, implementação e análise dos resultados de cada uma dessas heurísticas podem ser vistas na Tabela 1.1 ao final desta introdução.

Este trabalho encontra-se dividido em cinco capítulos distintos, além desta introdução e das conclusões finais. O segundo capítulo inclui as noções preliminares necessárias à compreensão do problema abordado e das estratégias que aqui propomos

na tentativa de solucioná-lo. O terceiro capítulo aborda o problema da identificação de genes propriamente dito, com suas dificuldades, importância e métodos relacionados. No Capítulo 4 descrevemos os detalhes de uma estratégia para identificação dos genes em uma seqüência de DNA por meio de sua comparação com um cDNA. Um pouco diferente, o Capítulo 5 aborda o problema da identificação de genes por meio da comparação de duas seqüências genômicas. Por último, tratamos no Capítulo 6 da estratégia de identificação de genes por comparação de várias seqüências de DNA. Todos esses três capítulos incluem, em suas primeiras seções, uma tentativa de formalização matemática do problema nele abordado e uma descrição genérica de algumas de suas soluções e heurísticas. O restante de cada um dos capítulos inclui uma descrição detalhada das heurísticas que desenvolvemos para tratamento do problema em questão e os resultados obtidos pelos programas que implementam as idéia descritas.

# Publicação	Informações sobre as publicações
[2]	Uma avaliação de ferramentas de predição de genes. *Proceedings* do Congresso da Sociedade Brasileira de Computação Ano: 2002 Páginas: 133-143.
[3]	*Gene prediction by spliced Alignment.* `http://www.vision.ime.usp.br/~icobicobi/` Ano: 2003
[6]	*A graph theoretical model for the gene prediction problem.* Ano: 2005 *status:* submetido.
[4]	*An experimental evaluation of similarity-based gene prediction tools* Ano: 2004 *status:* submetido.
[5]	*Gene prediction by syntenic Alignment.* Ano: 2005 *status:* submetido.
[88]	*An exact generic core algorithm for the prediction of the gene structure of close and distant gene family members.* Ano: 2005 *status:* submetido.

Tabela 1.1: Publicações decorrentes dos resultados apresentados neste trabalho.

Capítulo 2

Preliminares

Este trabalho possui como foco principal o desenvolvimento de soluções computacionais para um problema específico da Biologia. Daí, o conhecimento de alguns termos básicos de ambas essas áreas é imprescindível para uma melhor compreensão por parte do leitor das informações contidas neste documento. Com essa necessidade em mente, incluímos neste capítulo alguns dos principais conceitos ligados à Biologia Molecular e à Ciência da Computação.

Para uma melhor apresentação dos conceitos aqui descritos, decidimos por dividir este capítulo em duas partes principais. A primeira delas inclui alguns dos vários termos básicos pertencentes à Biologia Molecular. Nela daremos ênfase aos elementos das células diretamente ligados ao problema abordado neste trabalho. Já a segunda parte deste capítulo inclui os conceitos da Ciência da Computação utilizados no decorrer do documento.

2.1 Conceitos básicos de Biologia Molecular

A Biologia pode ser definida como o estudo dos seres vivos e das leis da vida[35]. Dentre os vários ramos dessa ciência está a Biologia Molecular, que estuda as funções, estruturas e mecanismos de atuação dos biopolímeros[1], especialmente das proteínas e dos ácidos nucléicos. No que segue, encontra-se uma definição precisa desses elementos, juntamente com a descrição de outras estruturas ligadas, de uma maneira ou de outra, à noção de proteínas e ácidos nucléicos. Várias das definições nesta seção foram retiradas dos livros *Genes VII*[55] e *Molecular Biology of the Cell*[7]

[1]Um biopolímero pode ser definido como qualquer composto formado no organismo pela aglomeração de grande número de moléculas fundamentais.

2.1.1 A célula

A **célula** é a unidade básica componente de todos os seres vivos. Com raras exceções, elas encontram-se divididas em duas partes principais: **núcleo** e **citoplasma**. Alguns tipos de células, denominadas **procariotas**, não possuem uma divisão nítida entre núcleo e citoplasma, em contraste com as células conhecidas como **eucariotas**, cujos núcleos estão separados do citoplasma por meio de uma membrana denominada **carioteca**. As células procariotas estão presentes em um número limitado de organismos, ditos **organismos procariotos** ou procariontes, como algas e bactérias unicelulares. Já as células eucariotas compõem a grande maioria dos organismos vivos, ditos **organismos eucariotos** ou eucariontes, e é exatamente sobre alguns dos elementos dessas células que focalizaremos as nossas atenções no decorrer deste trabalho.

O citoplasma inclui várias organelas constituintes das células eucariotas, como as mitocôndrias, o retículo endoplasmático e o Complexo de Golgi. Em conjunto com uma série de enzimas catalisadoras, essas organelas são responsáveis pela manutenção de algumas das atividades vitais da célula, como digestão, respiração, etc. Dentro do núcleo celular encontram-se outras estruturas celulares, como os cromossomos, que têm um papel fundamental durante o processo de reprodução da célula. Os cromossomos consistem basicamente em uma longa molécula de ácido desoxirribonucléico e várias moléculas de protéinas.

As moléculas de DNA e RNA

Além das organelas e enzimas catalisadoras, as células incluem também uma infinidade de outras moléculas. Dentre elas, existem duas de fundamental importância para o desenvolvimento celular: o ácido desoxirribonucléico e o ácido ribonucléico, comumente referenciadas como **ADN** e **ARN**, respectivamente, ou DNA (do inglês *desoxirribonucleic acid*) e RNA (do inglês *ribonucleic acid*)[2]. Essas moléculas são compostas por unidades denominadas **nucleotídeos**, que por sua vez constituem-se de um grupo fosfato, uma pentose e uma dentre quatro bases nitrogenadas: **adenina** (A), **citosina** (C), **guanina** (G), e **timina** (T), no caso do DNA ou **uracila** (U), no caso do RNA. Às bases adenina e guanina damos o nome de **purinas** (R), enquanto que às bases citosina, timina e uracila, o nome de **pirimidinas** (Y). Ao conjunto de moléculas de DNAs de um organismo damos o nome de **genoma**.

Em termos estruturais, os nucleotídeos componentes de um DNA encontram-se distribuídos em duas fitas anti-paralelas[3], dispostas helicoidalmente em torno de um eixo

[2]Apesar da existência de siglas correspondentes em português, no decorrer deste trabalho faremos referências a essas duas moléculas por meio das siglas em inglês (DNA e RNA) dado o uso freqüente delas na literatura.

[3]Chamamos duas fitas de anti-paralelas quando elas possuem a mesma direção mas sentidos opostos.

imaginário e ligadas uma a outra por meio de pontes de hidrogênio existentes entre bases específicas da molécula: adenina-timina e citosina-guanina. Essas bases são ditas **bases complementares**. A cada uma das fitas do DNA está associada uma orientação distinta, determinada pelos carbonos utilizados na ligação entre os seus nucleotídeos. Um nucleotídeo liga-se ao nucleotídeo seguinte da fita em que se encontra por meio da pentose, que inclui 5 átomos de carbono numerados 1, 2, 3, 4 e 5. Em uma das fitas, essa ligação envolve os carbonos de número 5 e 3, nessa ordem. Dizemos então que essa fita está orientada no sentido $5' \rightarrow 3'$. Na outra fita, os nucleotídeos encontram-se ligados por meio dos carbonos 3 e 5 da pentose. Essa fita está então orientada no sentido $3' \rightarrow 5'$. Na Figura 2.1 podemos ver um trecho das duas fitas de um DNA hipotético que inclui apenas dois pares de nucleotídeos. Note nessa figura a ligação entre os nucleotídeos por meio dos carbonos de números (5,3) e (3,5) da pentose e a respectiva orientação das fitas. A estrutura do DNA, como acabamos de descrever, foi elucidada em 1953 por Watson e Crick, e constitui-se em uma das maiores descobertas da área. Mais detalhes sobre esse trabalho podem ser encontrados em [93]. Com relação ao RNA, essa molécula possui uma estrutura mais simples, sendo formada por apenas uma fita de nucleotídeos. Além disso, vale notar que as moléculas de RNA costumam ser bem menores que as moléculas de DNA e podem desempenhar funções variadas dentro da célula.

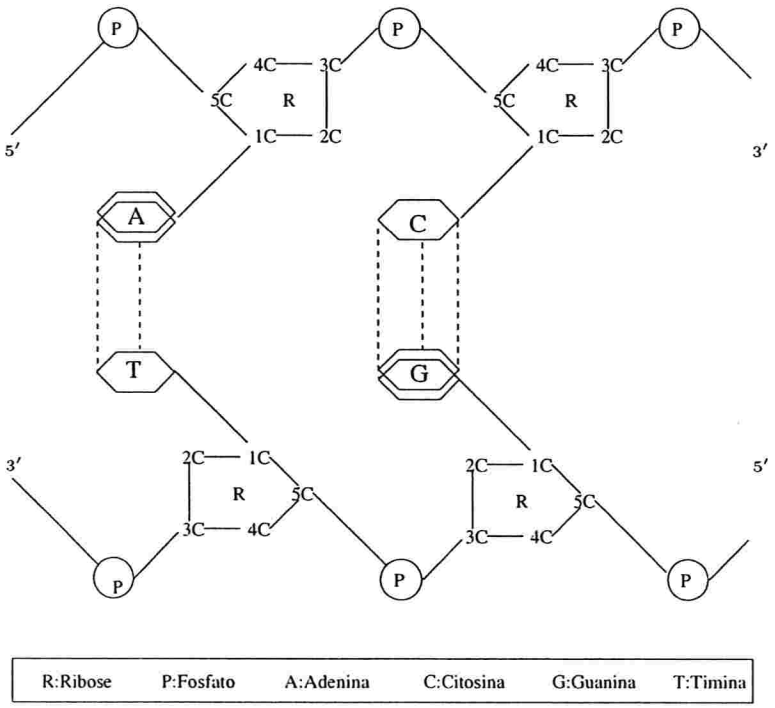

Figura 2.1: Dois pares de nucleotídeos das duas fitas de um DNA hipotético.

Em termos funcionais, as moléculas de DNA e RNA participam ativamente do processo de síntese de proteínas pelas células. Nas células eucariotas, esse processo, conhecido como **Dogma Central da Biologia**, inicia-se com uma transcrição de uma determinada região da molécula de DNA em uma molécula de RNA especial, conhecida como **pré-RNA mensageiro** (pré-RNAm). Vale observar que essa transcrição não é feita de forma direta. Cada uma das bases transcritas do DNA está associada à sua complementar no pré-RNAm, sempre lembrando que essa última molécula possui a base uracila no lugar da base timina. A molécula de pré-RNAm criada no processo de transcrição passa então por uma fase conhecida como fase de *splicing*. Nela, algumas das regiões componentes do pré-RNAm são eliminadas dessa molécula. As regiões mantidas compõem o que chamamos de **RNA mensageiro maduro**, ou simplesmente RNA mensageiro (RNAm). Na última fase do processo de síntese, a molécula de RNAm é finalmente traduzida para uma proteína. É importante notar que essa tradução é feita tomando-se três bases por vez. Cada tripla de bases traduzida recebe o nome de **códon**. A fase de transcrição e o processo de *splicing* do pré-RNAm, que conta com o auxílio de várias enzimas catalisadoras, como as RNA polimerases I, II e III, ocorrem no núcleo da célula, enquanto que a tradução dessa molécula acontece no citoplasma celular, onde outros elementos entram em ação, como o **RNA ribossomal** e o **RNA transportador**. Um esquema do processo de síntese de proteínas nas células eucariotas pode ser visto na Figura 2.2.

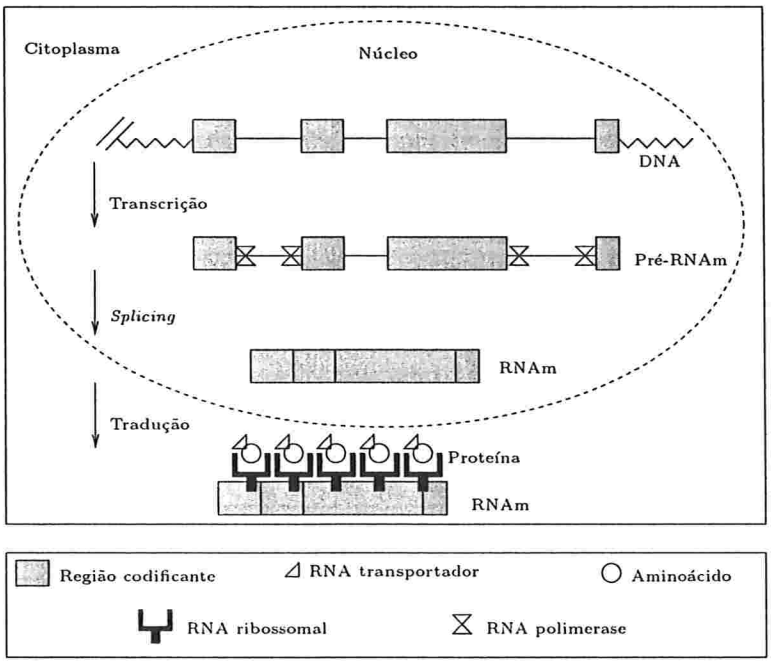

Figura 2.2: Esquema do processo de síntese de proteínas nas células eucariotas.

O produto final do processo de transcrição, *splicing* e tradução de um trecho do DNA corresponde a uma molécula de proteína. Diferentemente das moléculas de DNA e RNA, compostas de nucleotídeos, as proteínas possuem como componentes básicos elementos denominados **aminoácidos**. Existe um total de 20 aminoácidos conhecidos e cada um deles está associado a um tripla de bases de acordo com a Tabela 2.1, conhecida como Tabela do Código Genético[4]. Vale observar que algumas triplas de bases diferentes estão associadas ao mesmo aminoácido. Códons desse tipo são denominados **códons sinônimos**. Esse é o caso, por exemplo, das triplas `ATT`, `ATC` e `ATA`, todas associadas à Isoleucina. Note também que, diferentemente das outras triplas, os códons `TAA`, `TAG` e `TGA` não estão associados a nenhum aminoácido. Esses códons, conhecidos como **códons de parada**, determinam o final da fase de tradução. Informações mais detalhadas sobre os códons de parada serão fornecidas na seção seguinte.

Vale notar que nos organismos procariotos o processo de síntese de proteínas inclui apenas as fases de transcrição e tradução. Em outras palavras, toda a região de um DNA procariótico transcrita em uma molécula de RNAm é traduzida em uma proteína.

Os genes

As regiões do DNA passíveis de serem transcritas em uma molécula de RNA são conhecidas como **genes**. Chamamos de regiões intergênicas aquelas situadas entre os genes de uma molécula de DNA. Neste trabalho, estamos interessados apenas em genes que, em algum momento do ciclo de vida celular, podem ser transcritos em uma ou mais moléculas de RNA mensageiro. Ou seja, em genes que incluem a "receita" para a síntese de uma proteína qualquer.

Como mencionado anteriormente, em células eucariotas, nem todo o gene é traduzido em uma proteína, mas apenas as partes desse elemento que permanecem na molécula de RNAm após a fase de *splicing*. Essas regiões são chamadas de **éxons** ou regiões codificantes, enquanto que aquelas eliminadas na fase de *splicing*, de **íntrons**[5]. Apesar de essa definição de éxons e íntrons ser amplamente difundida, vale notar que, por um processo conhecido como *splicing alternativo*, alguns éxons podem ser eliminados da molécula de pré-RNAm durante a fase de *splicing*. Isso permite que várias proteínas sejam sintetizadas com base em um único gene. É importante observar também que determinados éxons incluem regiões que, apesar de transcritas e retidas após a fase de *splicing*, não passam pelo processo de tradução. O primeiro éxon de qualquer gene, por exemplo, inicia-se com uma região conhecida como **3′-UTR** (do inglês *untranslated ter-*

[4]Note nessa tabela que, seguindo a literatura, estamos utilizando a timina (T) ao invés da uracila (U) como uma das bases componentes dos códons, apesar de a síntese de proteína envolver a tradução de uma molécula de RNAm.

[5]Até hoje não se sabe exatamente a função dessas regiões dentro da célula, mas já foram encontrados indícios de que os íntrons incluem informações que auxiliam no processo de síntese de proteínas.

ATT	[I]	ACT	[T]	AAT	[N]	AGT	[S]	A	Alanina
ATC	[I]	ACC	[T]	AAC	[N]	AGC	[S]	R	Arginina
ATA	[I]	ACA	[T]	AAA	[K]	AGA	[R]	N	Asparagina
ATG	[M]	ACG	[T]	AAG	[K]	AGG	[R]	D	Aspartato
								C	Cisteína
								Q	Glutamina
CTT	[L]	CCT	[P]	CAT	[H]	CGT	[R]	E	Glutamato
CTC	[L]	CCC	[P]	CAC	[H]	CGC	[R]	G	Glicina
CTA	[L]	CCA	[P]	CAA	[Q]	CGA	[R]	H	Histidina
CTG	[L]	CCG	[P]	CAG	[Q]	CGG	[R]	I	Isoleucina
								L	Leucina
								K	Lisina
GTT	[V]	GCT	[A]	GAT	[D]	GGT	[G]	M	Metionina
GTC	[V]	GCC	[A]	GAC	[D]	GGC	[G]	F	Fenilalanina
GTA	[V]	GCA	[A]	GAA	[E]	GGA	[G]	P	Prolina
GTG	[V]	GCG	[A]	GAG	[E]	GGG	[G]	S	Serina
								T	Treonina
								W	Triptofano
TTT	[F]	TCT	[S]	TAT	[Y]	TGT	[C]	Y	Tirosina
TTC	[F]	TCC	[S]	TAC	[Y]	TGC	[C]	V	Valina
TTA	[F]	TCA	[S]	TAA	[*]	TGA	[*]	*	códon de
TTG	[F]	ACG	[S]	TAG	[*]	TGG	[W]		parada

Tabela 2.1: Tabela do código genético, juntamente com a sigla e nome de cada proteína e onde os códons associados ao símbolo "*" correspondem aos códons de parada.

minal region). Algo parecido acontece com o último éxon de qualquer gene, que termina com uma região conhecida como **5′-UTR**. Essas regiões, apesar de importantes durante o processo de síntese de proteínas pelas células, não são traduzidas em parte alguma desses biopolímeros. Apesar dessa diferença sutil entre éxons e regiões codificantes, iremos no decorrer desse trabalho utilizar esses dois termos de forma indiscriminada. Ou seja, estaremos sempre nos referindo à parte codificante dos éxons ao falarmos desses elementos.

Além dos éxons, íntrons e das 3′ e 5′-UTRs, os genes eucarióticos incluem outros elementos como os **sítios de aceitação**, os **sítios de doação**, o **códon de iniciação** e o **códon de parada**. Essas estruturas, denominadas genericamente de **sinais**, guardam informações úteis utilizadas pelo maquinário celular durante o processo de síntese de proteínas. Os códons de iniciação e parada, por exemplo, determinam o local onde se dará o início e o término, respectivamente, da tradução do gene em uma proteína. Normalmente, os códons de iniciação incluem as bases `ATG`, nessa ordem, enquanto

que os códons de parada, as bases `TAA`, `TAG` e `TGA`. Já os sítios de aceitação e doação determinam, respectivamente, o início e final dos éxons componentes do gene sendo processado. Essas estruturas, com raríssimas exceções, incluem, respectivamente, os dinucleotídeos `AG` e `GT` nas suas extremidades direita e esquerda[6]. Na Figura 2.3 podem ser vistos alguns dos principais sinais componentes da estrutura de um gene eucariótico:

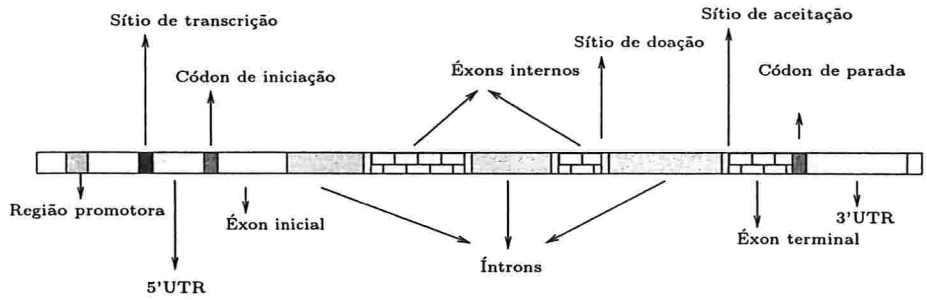

Figura 2.3: Estrutura simplificada de um gene eucariótico.

Apesar de não estar muito claro na Figura 2.3, gostaríamos de observar que os códons de iniciação e parada fazem parte dos éxons componentes do gene, enquanto que os sítios de aceitação e doação, de seus íntrons.

Uma outra região que tem uma importância fundamental no processo de síntese de proteínas é a **região promotora**. Essa região armazena informações que permitem ao maquinário celular determinar o local onde se dará início o processo de transcrição do gene. A região promotora, os éxons, e as $3'$ e $5'$-UTRs são denominadas genericamente de **regiões funcionais**[7].

Aos genes estão ligadas também as noções de cDNAS e ESTs, conhecidas genericamente como transcritos. Dado um gene qualquer, um cDNA corresponde à molécula composta dos éxons, e apenas dos éxons, desse gene. Elas são obtidas por meio da **transcrição reversa** de uma molécula de RNAm. Já os ESTs podem ser vistos como partes de um cDNA, normalmente obtidas de uma de suas pontas.

Evolução do genoma

O DNA contido nas células de qualquer organismo é repetidamente danificado por compostos químicos e radiação provenientes do ambiente ou de acidentes térmicos. Dada a importância desses biopolímeros para o desenvolvimento celular, esses danos precisam

[6]É comum a referência às extremidades esquerda e direita dos éxons, íntrons e dos próprios genes como extremidades $5'$ e $3'$, respectivamente.

[7]Na realidade, qualquer região da molécula de DNA que possui alguma função dentro da célula pode ser chamada de região funcional.

ser constantemente reparados. Isso é feito por enzimas denominadas enzimas de reparo. Infelizmente (ou felizmente), esse mecanismo de reparo não é perfeito. Quando as enzimas destinadas a recuperar uma alteração acidental no DNA não conseguem cumprir com êxito essa tarefa, ela se fixa na molécula. Tal alteração permanente da molécula de DNA é chamada de **mutação**. Além de fatores externos, falhas no processo de replicação do DNA podem também levar a alterações permanentes nessa molécula. Essas mutações vão desde mudanças (substituição, inserção e remoção) pontuais de algumas bases, incluindo aquelas componentes dos éxons, até a duplicação de partes ou de todo o gene em uma única célula.

Algumas mutações podem levar à destruição de um organismo. Essas mutações são conhecidas como **mutações deletérias** e costumam ocorrer nas regiões funcionais da molécula de DNA. Outras mutações, no entanto, não provocam nenhum dano aparente ao correto funcionamento do aparato celular. É o caso, por exemplo, das mutações que ocorrem dentro dos íntrons ou das que alteram certas bases pertencentes aos éxons de um gene mas não a proteína sintetizada com base nos seus códons. Tais mutações são denominadas **mutações silenciosas**. Por último, temos as mutações que trazem algum benefício ao organismo em cujo DNA ela ocorre. Ao contrário das mutações deletérias, que tendem a ser eliminadas por meio do processo de seleção natural, as mutações benéficas tendem a se espalhar entre os indivíduos de uma população. Em alguns casos, sucessivas mutações benéficas em uma das cópias de um gene duplicado podem dar origem a um gene com função distinta da original. Quando isso acontece, os genes duplicados são ditos genes **parálogos**. Ao contrário dos genes parálogos, que pertencem a organismos de uma mesma espécie mas possuem papéis diferentes dentro da célula, dois genes derivados de um único são ditos **ortólogos** quando pertencem a organismos de espécies distintas mas desempenham a mesma função dentro da célula. Genes derivados de um ancestral comum, pertencentes ou não a uma mesma espécie, com as mesmas funções ou funções distintas dentro da célula são chamados genericamente de genes **homólogos**.

Estudos mostram que as taxas de mutações nas regiões do genoma ditas funcionais são bem menores que as observadas nas regiões sem nenhuma função aparente. Esses resultados confirmam um princípio denominado **princípio da conservação das bases** que atesta uma tendência das regiões funcionais conservarem-se ao longo da evolução de um genoma. Uma explicação para esse fato está na relação direta entre as bases componentes das regiões funcionais e o papel que elas possuem dentro da célula. Qualquer mudança nessas porções das moléculas de DNA pode implicar danos sérios ao desenvolvimento celular.

Quando falamos em **seqüência genômica**, estamos nos referindo ao conjunto ordenado das bases de cada um dos nucleotídeos componentes de uma molécula de DNA. Dado o tamanho considerável de uma molécula de DNA, além de outros problemas, a determinação de sua seqüência é um processo lento e não trivial. Dentre outros pas-

sos, o processo de seqüenciamento de um DNA envolve a clonagem da molécula alvo, quebra desses clones em vários fragmentos sobrepostos, determinação das bases desses fragmentos e posterior montagem das partes de modo a recuperar a seqüência original. Esse método, conhecido como método *shotgun* e desenvolvido por Venter[91], tem sido amplamente utilizado no seqüenciamento das moléculas de DNA de várias espécies, dentre elas a espécie humana[89]. Um pouco diferente, chamamos de **seqüência transcrita** o conjunto ordenado das bases componentes de uma molécula de RNA (ou cDNA, proteína e EST). O seqüenciamento de uma molécula de RNA pode ser feito de forma direta dado o número pequeno de bases que ela inclui. Aqui, a dificuldade principal está na volatilidade dessa molécula e conseqüente dificuldade em localizá-la e extraí-la da célula.

Em determinadas situações, estamos interessados em comparar duas ou mais seqüências genômicas (ou uma seqüência genômica e uma seqüência de cDNA) no intuito de saber quão parecidas elas são. O grau de semelhança entre duas seqüências de DNA, por exemplo, pode nos fornecer informações úteis a respeito da proximidade evolutiva dessas moléculas, além de ser importante para determinação de algumas de suas regiões funcionais, como os éxons, regiões promotoras, etc. No que segue, abordaremos o problema computacional de comparar duas ou mais seqüências. Para uma melhor compreensão do assunto, alguns conceitos básicos da Ciência da Computação fazem-se necessários, e é com eles que damos início à seção seguinte.

2.2 Conceitos básicos de Ciência da Computação

A Ciência da Computação pode ser definida como a disciplina que estuda o projeto, a operação, o uso e a programação de computadores, aliando aspectos da matemática, lógica, engenharia e teoria da informação[35]. Dentre as inúmeras subáreas que essa ciência compreende, estamos interessados naquela que trata de problemas de otimização. Mais especificamente, de problemas envolvendo a comparação e processamento de seqüências.

Uma **seqüência** s é uma sucessão ordenada de caracteres ou símbolos de um certo conjunto Σ, comumente denominado **alfabeto**. Denotamos por Σ^* o conjunto incluindo todas as seqüências finitas construídas sobre o alfabeto Σ. O **tamanho** de uma seqüência s, denotado por $|s|$, é o número de símbolos componentes dessa seqüência. Nos exemplos

$$s = \texttt{AGATCAGTACATAGAA} \text{ e } t = \texttt{AACAGGTACAT},$$

temos duas seqüências distintas construídas sobre o alfabeto $\Sigma = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$, com $|s|$ igual a 16 e $|t|$ igual a 11. Representamos por $s[i]$ o símbolo presente na i-ésima posição de uma seqüência s. Nos exemplos acima, $s[3] = A$ e $t[5] = G$. Uma **subseqüência** de uma seqüência s corresponde a uma seqüência obtida a partir de s por meio da

remoção de alguns de seus caracteres. Por exemplo, ATTATA é uma subseqüência de s. Um pouco diferente, um **segmento** $s[i..j]$ de uma seqüência s, com $i \geq 1$ e $j \leq |s|$, é uma seqüência formada pelos símbolos consecutivos de s que se iniciam na i-ésima e terminam na j-ésima posição dessa seqüência. Por exemplo, $s[6..10]$ = AGTAC é um segmento de s. Dados dois segmentos $s[i..j]$ e $s[i'..j']$ de s, dizemos que $s[i..j]$ antecede $s[i'..j']$ em s se $j < i'$. Dizemos que um conjunto $S = \{s_1, s_2, \ldots, s_n\}$ de n segmentos de uma seqüência s está ordenado quando s_i antecede s_{i+1} em s para $1 \leq i \leq n$. A um conjunto de segmentos ordenados de uma seqüência s damos também o nome de cadeia de segmentos de s. Um **prefixo** $s[1..i]$, com $i \leq |s|$, de uma seqüência s é um segmento de s da forma $s[1]s[2]\ldots s[i]$. Ou seja, um segmento formado pelos i primeiros símbolos de s. Um **sufixo** $s[i..|s|]$, com $i \geq 1$, de uma seqüência s é um segmento de s da forma $s[i]s[i+1]\ldots s[|s|]$. Ou seja, um segmento formado pelos $|s| - i$ últimos símbolos de s. Finalmente, a concatenação de duas seqüências, denotada aqui por $s \bullet t$, corresponde à seqüência formada pelos símbolos de s seguidos dos símbolos de t, na mesma ordem em que eles aparecem nessas duas seqüências. Por exemplo, AGATCAGTACATAGAAAACAGGTACAT é a concatenação das seqüências s = AGATCAGTACATAGAA e t = AACAGGTACAT definidas acima. Boa parte dessas definições foram retiradas do livro intitulado *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*[46], que inclui uma descrição detalhada de vários problemas envolvendo seqüências e os seus principais algoritmos.

Um **grafo** é um par (V, E) em que V inclui elementos chamados **vértices** do grafo e E inclui elementos chamados **arestas** do grafo. Cada aresta corresponde a um par de vértices do grafo. Um grafo é denotado por letras maiúsculas, enquanto que os vértices, por letras minúsculas e arestas, por pares de vértices. É comum referenciarmos o conjunto de vértices V de um grafo G por V_G e o conjunto de suas arestas por E_G. Um grafo pode ser não-dirigido, quando suas arestas correspondem a pares não-ordenados de vértices, ou dirigido, quando suas arestas correspondem a pares ordenados de vértices. Nesse último caso, costumamos chamar as arestas do grafo de **arcos**. Em determinadas situações, torna-se conveniente associar um número qualquer às arestas de um grafo. Esses valores, de acordo com o contexto no qual o grafo está inserido, recebem o nome de **peso**, **custo** ou **distância** entre dois vértices do grafo. Grafos(dirigidos) podem ser representados por figuras compostas de retas não-orientadas(orientadas) e pontos, associados respectivamente, às arestas(arcos) e vértices do grafo. Na Figura 2.4, está a representação gráfica de dois grafos distintos G_1 e G_2, sendo o primeiro um grafo dirigido, tais que $V_{G_1} = \{v_1, v_2, v_3, v_4, v_5\}$, $V_{G_2} = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$, $E_{G_1} = \{(v_1, v_3), (v_3, v_4), (v_3, v_5), (v_4, v_5), (v_2, v_3), (v_2, v_1)\}$, e $E_{G_2} = \{(v_1, v_3), (v_3, v_5), (v_5, v_6), (v_5, v_7), (v_7, v_1), (v_8, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_6), (v_6, v_8)\}$.

Um **caminho**(dirigido) em um grafo G é uma lista de vértices distintos $v_1, v_2, \ldots, v_k$ em V_G tal que (v_i, v_{i+1}) é uma aresta(arco) do grafo para $1 \leq i < k$. Na Figura 2.4, v_2, v_1, v_3, v_5 é um caminho dirigido no grafo G_1. Um ciclo(dirigido) em um grafo G

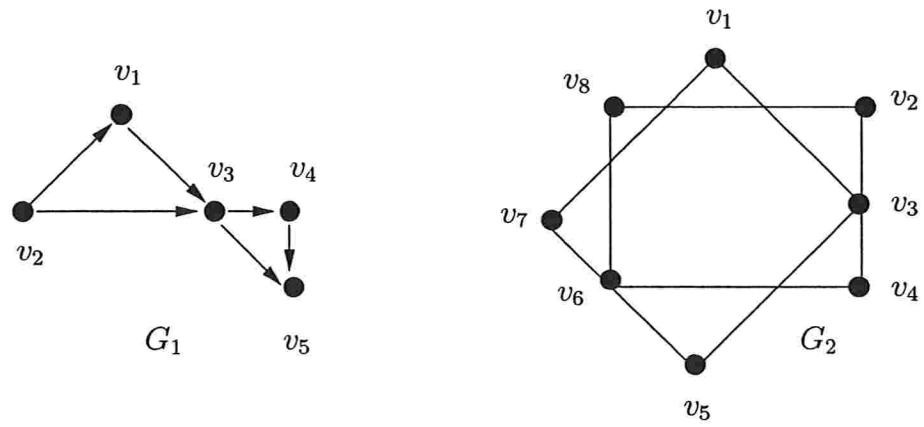

Figura 2.4: Exemplo de dois grafos distintos G_1 e G_2, sendo G_1 um grafo dirigido.

é uma lista de vértices distintos $v_1, v_2, \ldots, v_k$ em G tal que tal que (v_i, v_{i+1}) é uma aresta(arco) do grafo e $v_1 = v_k$. Na Figura 2.4, $v_1, v_3, v_5, v_6, v_7, v_1$ é um circuito no grafo G_2. O comprimento de um caminho ou ciclo é igual ao número de vértices desse caminho menos um. No caso em que as arestas de um grafo G possuem um peso associado, o peso de um caminho ou ciclo qualquer $(v_1, v_2, \ldots, v_k)$ em G é igual à soma dos pesos associados a cada um dos pares (v_i, v_{i+1}) tal que (v_i, v_{i+1}) é uma aresta do grafo para $1 \leq i < k$. Finalmente, dizemos que um grafo é acíclico quando ele não contém nenhum ciclo. O grafo G_1 da Figura 2.4, por exemplo, é um exemplo de grafo dirigido acíclico (GDA).

Dado um grafo G qualquer, várias são as perguntas que podemos fazer sobre ele. Um dos problemas clássicos envolvendo esse objeto matemático consiste em determinar o comprimento do menor caminho entre dois vértices distintos s e t de G e o caminho em si. No caso de um grafo com custo nas arestas, estamos interessado no caminho de menor peso entre s e t, também chamado de caminho mínimo entre s e t. Esse problema pode ser resolvidos em tempo $O(|E| \log |V|)$ por meio de um algoritmo conhecido como Algoritmo de Dijkstra[31].

Determinar o caminho mínimo entre dois vértices distintos de um grafo é um dos vários problemas envolvendo esses objetos que possui uma solução eficiente. Ou seja, um dos vários problemas para o qual já foi descoberto um algoritmo polinomial. Existem porém outros problemas para os quais ainda não foi encontrado nenhuma solução eficiente. Determinar o maior caminho, ou o caminho de maior peso entre dois vértices distintos de um grafo G qualquer é um exemplo de problema envolvendo esses objetos para o qual ainda não foi descoberto nenhum algoritmo polinomial. Problemas desse tipo costumam ser alvos da busca por soluções próximas da solução ótima. Ou seja, da busca por algoritmos eficientes que devolvem não necessariamente a melhor solução, mas

uma solução boa ou distante da ótima por algum fator α, normalmente pequeno. Em outras palavras, nesse caso busca-se por uma solução $S(I)$ tal que $val(S(I)) \leq \alpha opt(I)$ para toda instância I de um problema de minimização (ou $val(S(I)) \geq \alpha opt(I)$ para toda instância I de um problema de maximização), onde $val(S(I))$ é o valor associado à solução $S(I)$ e $opt(I)$ o valor associado à solução ótima do problema. Algoritmos do primeiro tipo costumam ser chamados de **heurísticas**, enquanto os que propõem uma solução próxima da ótima de **algoritmos de aproximação**. Essas noções serão amplamente utilizadas nos capítulos seguintes.

Apesar da dificuldade em se encontrar eficientemente um caminho máximo entre dois vértices distintos de um grafo G qualquer, esse problema torna-se fácil quando restrito a grafos dirigidos acíclicos. Mais do que isso, o próprio problema do caminho mínimo que, como já vimos, possui uma solução eficiente, torna-se mais fácil quando estamos tratando de grafos dirigidos acíclicos. Mais fácil no sentido de que ele pode ser resolvido em tempo $O(|E|)$ para grafos desse tipo por meio de uma ordenação topológica de seus vértices. O Algoritmo 1 mostra os passos a serem executados para determinação do peso de um caminho mínimo (e do próprio caminho) entre os vértices de um grafo dirigido acíclico G. Para determinação do caminho máximo entre esses dois vértices, basta utilizarmos uma variante simples desse algoritmo onde, além de alguns cuidados na inicialização de suas variáveis, o sinal de '>' na linha 9 do algoritmo é substituído pelo sinal de '<'.

Para confirmar a complexidade de tempo do Algoritmo 1, observe que a ordenação topológica dos vértices de G pode ser realizada em tempo $O(|V_G| + |E_G|)$ por meio de uma busca em profundidade nesse grafo. Uma vez ordenados e armazenados em l, cada um dos vértices dessa lista é processado uma única vez no laço que vai da linha 7 à linha 15 do Algoritmo 1. Dentro desse laço, cada arco do grafo é também processado uma única vez. Disso, podemos concluir que o laço em questão consome tempo igual a $O(|E_G|)$. Como os passos de inicialização (linhas 2 e 3 do algoritmo) e determinação do caminho mínimo (laço que vai da linha 17 à linha 19 do algoritmo) podem ser realizados em tempo $O(V_G)$, temos que a complexidade de tempo total do Algoritmo 1 é de $(|V_G| + |E_G|)$. Assumindo que nosso grafo possui um mínimo de $|V_G|$ arcos, temos a complexidade de $O(|E|)$ mencionada anteriormente.

Por último, vale falar também de um outro problema que, apesar de genericamente difícil, pode ser resolvido em tempo polinomial quando tratamos de grafos dirigidos acíclicos. Esse problema consiste em determinar o caminho de maior peso dentre todos os possíveis caminhos entre dois vértices quaisquer de um grafo G. Em um grafo dirigido acíclico, esse problema pode ser resolvido em tempo $O(|E_G|)$ seguindo os passos do Algoritmo 2. A confirmação de sua complexidade de tempo segue as mesmas idéias utilizadas acima para determinação da complexidade do Algoritmo 1.

Outros problemas e algoritmos relacionados envolvendo a busca por caminhos em

Algoritmo 1 CAMINHO_MINIMO_GDA(G, s, t, w)

Entrada: um grafo dirigido acíclico G, dois vértices s e t em V_G e uma função w que atribui um peso para cada arco em E_G.
Saída: caminho mínimo entre s e t e o peso desse caminho.

```
1: para cada v ∈ V_G faça
2:    d[v] ← ∞; {d[v]: peso do caminho mínimo entre s e v}
3:    p[v] ← ϵ; {p[v]: antecessor de v no caminho mínimo entre s e v}
4: fim-para
5: d[s] ← 0; p[s] ← s
6: l ← lista dos vértices de G ordenada topologicamente;
7: enquanto l ≠ ∅ faça
8:    u ← vértice na cabeça de l;
9:    para cada v adjacente u faça
10:      se d[v] > d[u] + w(u,v) então
11:         d[v] ← d[u] + w(u,v); p[v] ← u;
12:      fim-se
13:   fim-para
14:   l ← l − u
15: fim-enquanto
16: c[0] ← t; u ← p[t]; i ← 1;
17: enquanto u ≠ s faça
18:   c[i] ← u; u ← p[u]; i ← i + 1;
19: fim-enquanto
20: c[i] ← s;
21: Devolva d[t];
22: Devolva c invertido;
```

grafos dirigidos acíclico foram abordados por Azevedo *et al* em [12] e Martins e Santos em [57]. Além disso, vários são os trabalhos onde o leitor pode encontrar uma descrição detalhada dos problemas e algoritmos sobre grafos em geral. Aqui sugerimos o livro intitulado *Graph Theory with Applications*[20] como obra de referência sobre o assunto.

2.2.1 Alinhamento de (duas) seqüências

A forma mais comum de se realizar uma comparação entre duas seqüências é por meio de um processo conhecido como **alinhamento de seqüências**. De acordo com Meidanis e Setubal[60], alinhar duas seqüências s e t consiste em inserir espaços, representados aqui pelo símbolo '−', em locais arbitrários dessas seqüências de modo que elas fiquem do mesmo tamanho. Dessa forma, s e t podem ser dispostas uma em cima da outra, criando-se uma correspondência entre seus caracteres. Essa correspondência entre os

Algoritmo 2 CAMINHO_MAXIMO_DAG(G, w)

Entrada: um grafo dirigido acíclico G e uma função w que atribui um peso para cada arco em E_G.

Saída: caminho de maior peso dentre todos os possíveis caminhos entre dois pares de vértices distintos de G, e o peso desse caminho.

```
1: para cada v ∈ V_G faça
2:    d[v] ← 0; {d[v]: peso do caminho máximo terminando em v}
3:    p[v] ← ϵ; {p[v]: antecessor de v no caminho máximo que termina nesse vértice}
4: fim-para
5: l ← lista dos vértices de G ordenada topologicamente;
6: enquanto l ≠ ∅ faça
7:    u ← vértice na cabeça de l;
8:    para cada v incidente em u faça
9:       se d[u] < d[v] + w(v, u) então
10:         d[u] ← d[v] + w(v, u); p[u] ← v;
11:      fim-se
12:   fim-para
13:   l ← l − u
14: fim-enquanto
15: a ← max_{v∈V} d[v];
16: b ← v tal que d[v] é máximo;
17: c[0] ← b; u ← p[b]; i ← 1;
18: enquanto d[u] ≠ 0 faça
19:    c[i] ← u; u ← p[u]; i ← i + 1;
20: fim-enquanto
21: Devolva a;
22: Devolva c invertido;
```

caracteres de s e t pode ser melhor visualizada tomando-se uma matriz de duas linhas cujas colunas incluem os símbolos ocupando a mesma posição em s e t após a inserção de espaços nessas seqüências. Vale notar que espaços podem ser inseridos tanto no início quanto no final das seqüências. A única restrição imposta é que não pode haver uma ou mais colunas do alinhamento contendo apenas espaços. Um exemplo de alinhamento de duas seqüências s = `CAGCCACTGGATTCTCG` e t = `CAGCGTGCATTTC` pode ser visto na Figura 2.5.

Definindo-se um alinhamento formalmente, seja Σ um alfabeto qualquer tal que $\{-\} \notin \Sigma$. Chamamos de $\bar{\Sigma}$ o alfabeto contendo os símbolos de Σ mais o símbolo $\{-\}$. Ou seja, $\bar{\Sigma} = \Sigma \cup \{-\}$. Para toda seqüência $s \in \bar{\Sigma}^*$, seja $s_{|\Sigma}$ a seqüência s restrita ao alfabeto Σ (ou seja, a seqüência resultante da remoção de todos os espaços presentes em s). Um **alinhamento** A de duas seqüências s e $t \in \Sigma^*$ é um par $(\bar{s}, \bar{t})$, com $\bar{s}$ e $\bar{t}$

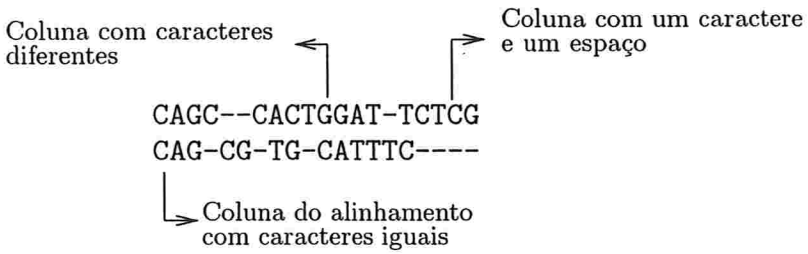

Figura 2.5: Exemplo de um alinhamento de duas seqüências

$\in \bar{\Sigma}^*$ tal que:

1. $|\bar{s}| = |\bar{t}|$;
2. $\bar{s}_{|\Sigma} = s$ e $\bar{t}_{|\Sigma} = t$;
3. Não existe i tal que $\bar{s}[i] = - = \bar{t}[i]$.

Dado um alinhamento de duas seqüências, uma pontuação pode ser associada a ele de acordo com o conteúdo de suas colunas. Pela Figura 2.5, podemos observar que as colunas de um alinhamento podem incluir dois caracteres iguais, dois caracteres diferentes ou um caractere e um espaço. Lembramos que, pelo terceiro item da definição acima, um alinhamento não inclui colunas com dois espaços emparelhados.

Seja $w : \bar{\Sigma} \times \bar{\Sigma} \rightarrow \mathbb{R}$ uma função que associa um certo valor real para cada par de símbolos (μ, ν) em $\bar{\Sigma}$. Essas funções, normalmente representadas por matrizes indexadas pelos elementos em $\bar{\Sigma}$, são chamadas de **funções de pontuação**. Considerando uma função de pontuação w qualquer, definimos a pontuação $\text{Score}_w(\bar{s}, \bar{t})$ de um alinhamento $A = (\bar{s}, \bar{t})$ de s e t por:

$$\text{Score}_w(\bar{s}, \bar{t}) = \sum_{i=1}^{|\bar{s}|} w(\bar{s}[i], \bar{t}[i]).$$

O alinhamento da Figura 2.5, por exemplo, possui pontuação total igual a -22 no caso de uma função de pontuação w tal que, para todo μ e $\nu \in \Sigma = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$, com $\mu \neq \nu$, $w(\mu, \mu) = 1$, $w(\mu, \nu) = -1$ e $w(\mu, -) = w(-, \nu) = -2$.

A **similaridade** $\text{sim}_w(s, t)$ entre duas seqüências s e t, dada uma função de pontuação w, é definida como a maior pontuação dentre as pontuações de todos os possíveis alinhamentos de s e t. Ou seja:

$$\text{sim}_w(s, t) = \max\{\text{Score}_w(\bar{s}, \bar{t})\},$$

tal que $(\bar{s}, \bar{t})$ é um alinhamento de s e t.

Um alinhamento $A = (\bar{s}, \bar{t})$ de s e t é dito **alinhamento ótimo** com respeito a uma função w se $\text{Score}_w(\bar{s}, \bar{t}) = \text{sim}_w(s, t)$.

Dadas essas definições, o problema inicial de medir quão parecidas são duas seqüências pode ser formulado como o seguinte problema de otimização:

Problema do Alinhamento de Duas Seqüências(PAS): *Dadas duas seqüências s e t construídas sobre um alfabeto Σ qualquer tal que $\{-\} \notin \Sigma$ e uma função de pontuação w, encontrar o valor da similaridade $\text{sim}_w(s, t)$ entre elas.*

Note que, da forma como a pontuação de um alinhamento foi definida, o PAS possui uma propriedade conhecida como **Propriedade das Subsoluções Ótimas**. De acordo com ela, se $\mathcal{A} = (A'A'')$ é um alinhamento ótimo de s e t (Figura 2.6(a)), e A' e A'' são dois pedaços desse alinhamento, com A' correspondendo a um alinhamento de um prefixo s' de s e um prefixo t' de t (Figura 2.6(b)) e A'' a um alinhamento de um sufixo s'' de s e um sufixo t'' de t (Figura 2.6(b)), então ambos os alinhamentos são ótimos. Isso é verdade pois, no caso da existência de um alinhamento A''' dos prefixos s' de s e t' de t alinhados em A' tal que $\text{Score}_w(A''')$ é maior que $\text{Score}_w(A')$ (Figura 2.6(c)), então o alinhamento $\mathcal{A}^* = (A'''A'')$ (Figura 2.6(d)) teria uma pontuação maior que aquela do alinhamento $\mathcal{A} = (A'A'')$, o que contraria o fato de esse último alinhamento ser ótimo. O mesmo acontece no caso da existência de um alinhamento A''' dos sufixos s'' de s e t'' de t alinhados em A'' tal que $\text{Score}_w(A''')$ é maior que $\text{Score}_w(A'')$ (Figura 2.6(c)).

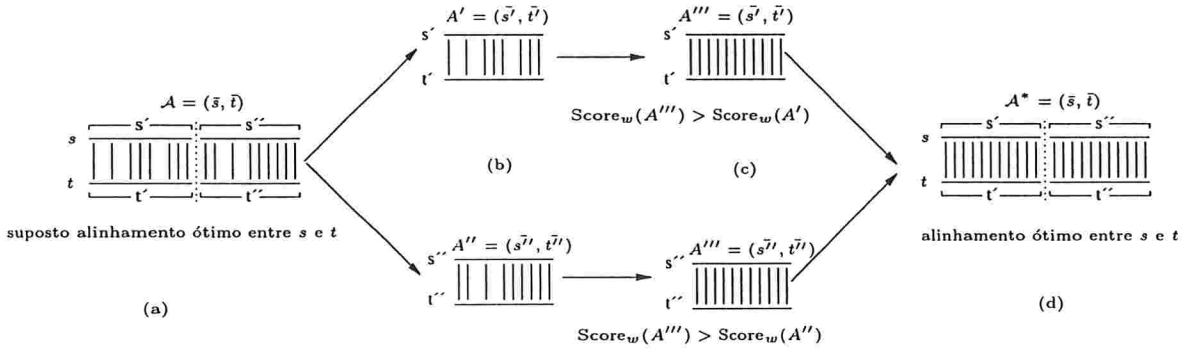

Figura 2.6: Esquema representativo da propriedade das subsoluções ótimas do problema do alinhamento.

Em particular, podemos considerar a última coluna do alinhamento $\mathcal{A}$ como A''. Nesse caso, observe que existem apenas três possibilidades para a escolha dos caracteres componentes dessa coluna: emparelhar um caractere de uma das duas seqüência com um espaço na outra ou emparelhar dois caracteres, um de cada seqüência. Disso, podemos pensar nos passos recursivos do Algoritmo 3 para o cálculo da similaridade entre duas seqüências s e t.

Algoritmo 3 CALCULA_SIM_RECURSIVO(s, t, w)

Entrada: duas seqüências s e t e uma função de pontuação w.
Saída: valor da similaridade entre s e t com respeito a w;

```
1: m ← |s|; n ← |t|;
2: Score ← 0;
3: se m = 0 então
4:    para i ← 0 até n faça
5:       Score ← Score + w(s[i], −);
6:    fim-para
7:    Devolva Score;
8: fim-se
9: se n = 0 então
10:   para i ← 0 até m faça
11:      Score ← Score + w(−, t[i]);
12:   fim-para
13:   Devolva Score;
14: fim-se
15: Score_1 ← CALCULA_SIM_RECURSIVO(s[1..m − 1], t[1..n], w) + w(s[|m|], −);
16: Score_2 ← CALCULA_SIM_RECURSIVO(s[1..m − 1], t[1..n − 1], w) + w(s[|m|], t[|n|]);
17: Score_3 ← CALCULA_SIM_RECURSIVO(s[1..m], t[1..n − 1], w) + w(−, t[|n|]);
18: Devolva max{Score_1, Score_2, Score_3};
```

Infelizmente, o Algoritmo 3 não possui uma aplicabilidade prática dada sua complexidade de tempo exponencial com relação ao tamanho das seqüências de entrada. O desenvolvimento da recorrência $T(m,n) = T(m-1,n)+T(m-1,n-1)+T(m,n-1)$, representativa do tempo consumido pelo algoritmo acima, possui como resultado $T(m,n) = \Omega(3^n)$, no caso em que $m = n$.

Uma solução eficiente do problema pode ser desenvolvida notando-se que o Algoritmo 3 realiza alguns cálculos redundantes. A similaridade entre $s[1..m-1]$ e $t[1..n-1]$, por exemplo, é calculada três vezes por CALCULA_SIM_RECURSIVO: uma delas na própria chamada a CALCULA_SIM_RECURSIVO($s[1..m-1]$, $t[1..n-1]$, w) e as outras duas nas chamadas a CALCULA_SIM_RECURSIVO($s[1..m]$, $t[1..n-1]$, w) e CALCULA_SIM_RECURSIVO($s[1..m-1]$, $t[1..n]$, w). Cálculos redundantes como os que acabamos de mostrar são realizados várias vezes durante a execução do Algoritmo 3. Se determinados em uma certa ordem e armazenados corretamente, as soluções dos subproblemas redundantes (similaridades entre partes das seqüências) podem ser reutilizadas para a solução eficiente do problema principal (similaridade entre as seqüência como um todo). Essa é a idéia principal de uma estratégia conhecida como **programação dinâmica**.

Por meio da estratégia de programação dinâmica, o valor da similaridade entre duas seqüências s e t, com $|s| = m$ e $|t| = n$, pode ser calculado eficientemente armazenando-se as soluções dos subproblemas em uma matriz M, de dimensão $(m + 1) \times (n + 1)$, indexada por $\{0 \ldots m\}$ e $\{0 \ldots n\}$, conhecida como **matriz de alinhamento**. Cada uma das células $M[i][j]$ dessa matriz armazena a similaridade entre o prefixos de s e t que incluem, respectivamente, os i e j primeiros caracteres dessas seqüências. Disso, após o preenchimento total da matriz, a similaridade procurada pode ser encontrada na última célula $M[|s|][|t|]$ de M. Para que todas as células de M possam ser calculadas eficientemente, essa matriz precisa ser preenchida de forma ordenada. A forma mais comum de se fazer isso é preencher a matriz M linha a linha, em ordem crescente de suas colunas.

Nas subseções seguintes são apresentados os principais algoritmos utilizados para o cálculo eficiente da similaridade entre duas seqüências com base na estratégia de programação dinâmica.

Alinhamento global e o algoritmo de Needleman-Wunch

Como descrito anteriormente, o PAS é comumente chamado de problema do alinhamento global no sentido de que a similaridade é calculada pensando-se nas duas seqüências como um todo.

Tomando-se duas seqüências quaisquer s e t, com $|s| = m$ e $|t| = n$, e uma função de pontuação w, a estratégia de programação dinâmica pode ser assim utilizada para cálculo do alinhamento global de s e t. Lembremos que a idéia principal é fazer uso de uma matriz M tal que $M[i][j]$ armazena a similaridade entre $s[1..i]$ e $t[1..j]$. Assim, a primeira linha e coluna de M podem ser preenchidas com múltiplos dos valores associados à coluna com um espaço. Em outras palavras $M[0][k] = w(-, t[k])$, para $0 < k \leq j$ e $M[k][0] = w(s[k], -)$, para $0 < k \leq i$. Essa inicialização decorre do fato de existir uma única possibilidade de alinhamento de uma seqüência qualquer com outra seqüência vazia: incluir nesta última um número de espaços igual ao tamanho da outra seqüência. Como esse alinhamento é único, ele é um alinhamento ótimo dessas seqüências.

Para o cálculo das outras posições da matriz M, é importante lembrar a propriedade das subsoluções ótimas do problema do alinhamento de duas seqüências. Por meio dela e atentando-se ao fato de existir apenas três possibilidades para a escolha da última coluna de um alinhamento, a similaridade entre $s[1..i]$ e $t[1..j]$ corresponde ao maior dentre os seguintes valores:

1. $\text{sim}_w(s[1..i], t[1..j-1]) + w(-, t[j])$;
2. $\text{sim}_w(s[1..i-1], t[1..j-1]) + w(s[i], t[j])$;

3. $\text{sim}_w(s[1..i-1], t[1..j]) + w(s[i], -)$.

Ou seja, $M[i][j]$ pode ser assim calculada:

$$M[i][j] = \max \begin{cases} M[i][j-1] + w(-, t[j]); \\ M[i-1][j-1] + w(s[i], t[j]); \\ M[i-1][j] + w(s[i], -). \end{cases} \tag{2.1}$$

Note que, preenchendo-se a matriz M linha a linha e em ordem crescente de coluna, os valores de $M[i][j-1]$, $M[i-1][j-1]$, $M[i-1][j]$ necessários para o cálculo de $M[i][j]$ já estarão disponíveis quando essa célula for alcançada.

Na Figura 2.7 pode-se ver uma matriz de alinhamento preenchida para duas seqüências $s =$ `ACGT` e $t =$ `ACC`, e uma função de pontuação w tal que $w(\mu, \mu) = 1$, $w(\mu, \nu) = -1$ e $w(\mu, -) = w(-, \nu) = -2$ para todo μ e $\nu \in \Sigma = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$ tal que $\mu \neq \nu$.

			A	C	C
			1	2	3
	0	0	← -2	← -4	← -6
A	1	↑ -2	↖ 1	← -1	← -3
C	2	↑ -4	↑ -1	↖ 2	← 0 ↖
G	3	↑ -6	↑ -3	↑ 0	↖ 1
T	4	↑ -8	↑ -5	↑ -2	↑ -1 ↖

Figura 2.7: Exemplo de matriz de alinhamento das seqüências $s =$`ACGT` e $t =$`ACC`.

Com base na Recorrência 2.1, o Algoritmo 4, devido a Needleman e Wunch[67], pode ser utilizado para o cálculo eficiente da similaridade entre duas seqüências s e t com respeito a uma função de pontuação w.

Como o Algoritmo 4 envolve apenas o preenchimento de uma matriz de dimensão $(m+1) \times (n+1)$, pode-se concluir que ele realiza sua tarefa em tempo e espaço $O(mn)$[8]. Na realidade, não necessitamos armazenar toda a matriz M para o cálculo da similaridade entre s e t. Observe que, para o preenchimento da posição $M[i][j]$ da matriz, precisamos apenas de alguns valores guardados em duas posições distintas da linha $i-1$ de M ($M[i-1][j-1]$ e $M[i-1][j]$), e de um outro valor guardado na linha i de M ($M[i][j-1]$). Dessa forma, podemos determinar o valor da similaridade entre s e t armazenando apenas a linha anterior e aquela incluindo a célula sendo calculada. Isso reduz a complexidade de espaço do Algoritmo CALCULA_SIMILARIDADE para $O(n)$.

[8]Vale notar que a complexidade de tempo do algoritmo descrito originalmente por Needleman e Wunch é cúbica (e não quadrática) no tamanho da entrada.

Algoritmo 4 CALCULA_SIMILARIDADE(s, t, w)

Entrada: duas seqüências s e t e uma função de pontuação w.
Saída: valor da similaridade entre s e t com respeito a w;

```
1: m ← |s|; n ← |t|;
2: para i ← 0 até n faça
3:    M[i][0] ← i × w(−, t[i]);
4: fim-para
5: para j ← 0 até m faça
6:    M[0][j] ← j × w(s[j], −);
7: fim-para
8: para i ← 1 até m faça
9:    para j ← 1 até n faça
10:      ins ← w(s[i], −); del ← w(−, t[j]); emp ← w(s[i], t[j]);
11:      M[i][j] ← max(M[i − 1][j] + ins, M[i − 1][j − 1] + emp, M[i][j − 1] + del);
12:   fim-para
13: fim-para
14: Devolva M[m][n];
```

Note que até o momento falamos apenas do cálculo da similaridade entre duas seqüências. Sobre o alinhamento ótimo, ele pode ser construído voltando-se na matriz de alinhamento pelo caminho inverso daquele utilizado para seu preenchimento. Dada uma posição qualquer $M[i][j]$ da matriz de alinhamento, sabemos que ela foi calculada de acordo com uma das três possibilidades abaixo:

1. utilizando o valor armazenado em $M[i][j-1]$. Nesse caso a coluna do alinhamento inclui um espaço em s emparelhado com o caractere $t[j]$;
2. ou por meio do valor em $M[i-1][j]$. Nesse caso a coluna do alinhamento inclui um espaço em t emparelhado com o caractere $s[i]$;
3. ou utilizando o valor em $M[i-1][j-1]$. Nesse caso a coluna do alinhamento inclui o caractere $s[i]$ emparelhado com o caractere $t[j]$.

A determinação de qual célula foi utilizada para o cálculo de $M[i][j]$ pode ser feita por meio de um cálculo simples. Uma vez realizado esse cálculo, ele nos fornece os caracteres componentes da última coluna do melhor alinhamento das seqüências $s[1..i]$ e $t[1..j]$.

Tomando-se como exemplo a matriz de alinhamento da Figura 2.7 para as seqüências s =ACGT e s =ACC, onde as setas em cada uma de suas células indicam qual das células vizinhas foi utilizada para seu preenchimento, o alinhamento ótimo de s e t pode ser

construído começando-se na última posição dessa matriz e seguindo as setas até que a primeira posição seja alcançada. Fazendo isso, encontraríamos o seguinte alinhamento ótimo de s e t:

```
ACGT
AC-C
```

Ainda sobre a matriz de alinhamento da Figura 2.7, observe que pode haver mais de uma seta em cada uma de suas células. Isso significa que podem haver mais de um alinhamento ótimo das seqüências s =ACGT e s =ACC. Voltando-se na matriz da Figura 2.7, podemos também encontrar o seguinte alinhamento ótimo das seqüências s e t, distinto daquele encontrado anteriormente.

```
ACGT
ACC-
```

O Algoritmo 5 recebe como entrada duas seqüências s e t, uma matriz de alinhamento já preenchida e devolve o alinhamento ótimo das duas seqüências.

Algoritmo 5 CONSTROI_ALINHAMENTO(s, t, M)

Entrada: duas seqüências s e t e uma matriz de alinhamento M.
Saída: um dos alinhamentos ótimos de s e t.

1: $i \leftarrow |s|$; $j \leftarrow |t|$;
2: $k \leftarrow 0$;
3: **enquanto** $i \neq 0$ e $j \neq 0$ **faça**
4: **se** $M[i][j] = M[i-1][j-1] + w(s[i], t[j])$ **então**
5: $\bar{s}[k] \leftarrow s[i]$; $\bar{t}[k] \leftarrow t[j]$;
6: $i \leftarrow i-1$; $j \leftarrow j-1$; $k \leftarrow k+1$;
7: **senão se** $M[i][j] = M[i][j-1] + w(-, t[j])$ **então**
8: $\bar{s}[k] \leftarrow$'-'; $\bar{t}[k] \leftarrow t[j]$;
9: $j \leftarrow j-1$; $k \leftarrow k+1$;
10: **senão se** $M[i][j] = M[i-1][j] + w(s[i], -)$ **então**
11: $\bar{s}[k] \leftarrow s[i]$; $\bar{t}[k] \leftarrow$ '-';
12: $i \leftarrow i-1$; $k \leftarrow k+1$;
13: **fim-se**
14: **fim-enquanto**
15: Devolva $(\bar{s}, \bar{t})$ invertidos;

Note que, a cada iteração do algoritmo acima, voltamos sempre uma posição na linha e/ou na coluna da matriz. Com isso, no pior caso, teremos um total de $m + n$

iterações do algoritmo até que a posição $M[0][0]$ seja alcançada. Disso, o alinhamento das duas seqüências de entrada pode ser construído em tempo $O(m+n)$.

Outra forma de abordar o problema do alinhamento de duas seqüências é através da busca pelo caminho de maior pontuação em um grafo conhecido como **grafo de alinhamento**. Dadas duas seqüências s e t de tamanho m e n, respectivamente, um grafo de alinhamento $G = (V, E)$ para s e t corresponde a um grafo dirigido acíclico com $(m+1)(n+1)$ vértices $v = (i, j)$, para $0 \leqslant i \leqslant m$ e $0 \leqslant j \leqslant n$. Com relação aos arcos desse grafo, eles podem ser de três tipos diferentes:

1. Arcos diagonais (representam as colunas do alinhamento contendo dois caracteres, distintos ou não): $\{((i-1, j-1)(i, j))|0 < i \leqslant m \text{ e } 0 < j \leqslant n\}$;
2. Arcos horizontais (representam as colunas do alinhamento contendo um espaço em t): $\{((i-1, j)(i, j))|0 < i \leq m \text{ e } 0 \leq j \leq n\}$;
3. Arcos verticais (representam as colunas do alinhamento contendo um espaço em s):
   $\{((i, j-1)(i, j))|0 \leq i \leqslant m \text{ e } 0 < j \leqslant n\}$.

Um exemplo desse grafo para as seqüências $s =$ `ACGT` e $t =$ `ACC` pode ser visto na Figura 2.8.

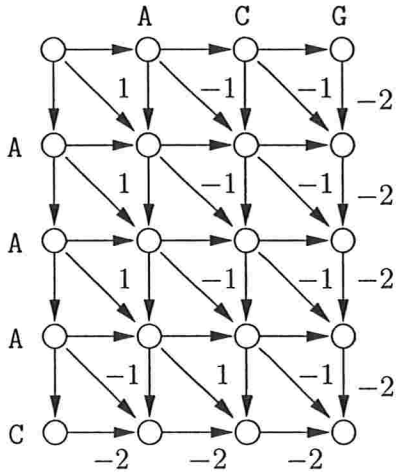

Figura 2.8: Exemplo de grafo de alinhamento para as seqüências $s =$ `ACGT` e $t =$ `ACC`

Observe que, associando um custo c para cada arco de G de acordo com uma função de pontuação w, existe uma relação direta entre um caminho dirigido de maior pontuação do vértice $(0, 0)$ ao vértice (m, n) de G e um alinhamento global ótimo de s e t com respeito a w. Também com esta abordagem, a similaridade entre s e t pode ser

encontrada em tempo $O(mn)$. Para isso, basta a utilização da variante do Algoritmo 1, descrito anteriormente, que busca por um caminho máximo entre dois vértices distintos de G.

Alinhamento local e o algoritmo de Smith-Waterman

Apesar da grande utilidade na tarefa de comparação de duas seqüências, um alinhamento global não nos diz muito sobre semelhanças entre duas seqüências compartilhando apenas alguns trechos similares. Ou seja, seqüências que apresentam apenas semelhanças locais. Tomando-se como exemplo as seqüência $s =$ `CTTCAGCACTTGGATTCTCGG` e $t =$ `AGCCACCTGCGGC`, e a mesma função de pontuação w da seção anterior, o seguinte alinhamento corresponde ao alinhamento global ótimo de s e t:

```
CTTCAGCACTTGGATTCTCGG
AGCCA-C-CT-G----CG-GC
```

Apesar de ser o melhor alinhamento global de s e t, existe outro alinhamento envolvendo essas seqüências que expressa de maneira mais clara as semelhanças entre elas. O alinhamento de s e t mostrado abaixo é conhecido como **alinhamento local** das seqüências.

```
------CACTTG--------
------CACCTG--------
```

Observe que, diferentemente do alinhamento global, o alinhamento local não leva em consideração as duas seqüências como um todo, mas apenas segmentos das seqüências sendo alinhadas na busca por semelhanças entre elas. Dessa observação, o seguinte problema de otimização, que corresponde a uma variante do problema do alinhamento global, pode ser formulado:

> **Problema do Alinhamento Local de Duas Seqüências**(PALS): *Dadas duas seqüências s e t construídas sobre um alfabeto Σ qualquer tal que $\{-\} \notin \Sigma$ e uma função de pontuação w, encontrar um segmento s' de s e um segmento t' de t tal que $sim_w(s', t')$ seja máxima.*

O algoritmo para determinação do alinhamento local ótimo de duas seqüências s e t, de tamanhos m e n respectivamente, é devido a Smith e Waterman[81] e corresponde a uma variante do algoritmo de Needleman-Wunch. Ele também está baseado na construção de uma matriz M de dimensão $(m+1) \times (n+1)$, mas cujas células armazenam

agora a pontuação do melhor alinhamento envolvendo um sufixo de $s[1..i]$ e um sufixo de $t[1..j]$. Essa matriz pode ser preenchida de acordo com a seguinte recorrência:

$$M[i][j] = \max \begin{cases} M[i][j-1] + w(-, t[j]); \\ M[i-1][j-1] + w(s[i], t[j]); \\ M[i-1][j] + w(s[i], -); \\ 0. \end{cases} \tag{2.2}$$

Note que a única diferença entre a Recorrência (2.2) e a Recorrência (2.1), usada no cálculo da similaridade global entre duas seqüências, está na possibilidade de se escolher o zero durante a determinação do máximo a preencher $M[i][j]$. Essa possibilidade adicional justifica-se pelo fato de que sempre existe um alinhamento de sufixos vazios de $s[1..i]$ e $t[i..j]$, cuja pontuação é zero.

Preenchida a matriz, a pontuação do alinhamento local ótimo de s e t corresponde ao maior valor que pode ser encontrado nas células de M. O alinhamento em si pode ser construído como no caso do alinhamento global, mas desta vez o processo de construção inicia-se na célula de maior valor e termina quando uma célula contendo o valor zero for encontrada.

Como a determinação da pontuação de um alinhamento local ótimo de s e t, com $s = m$ e $t = n$, envolve o preenchimento de uma matriz de dimensão $(m+1) \times (n+1)$ e uma busca pelo maior valor nesta matriz, ela também pode ser feita em tempo e espaço $O(mn)$. Pensando-se em termos de um grafo de alinhamento G, o alinhamento local ótimo de s e t equivale ao caminho de maior pontuação entre quaisquer dois vértices desse grafo. Lembramos que esse caminho pode ser determinado em tempo $O(|E_G|) = O(mn)$ através do Algoritmo 2 descrito anteriormente.

Alinhamento semiglobal

Além dos alinhamentos globais e locais, existe um outro tipo de alinhamento de duas seqüências s e t, conhecido como **alinhamento semiglobal**. Considerando-se novamente as seqüências s = `CTTCAGCACTTGGATTCTCGG` e t = `AGCCACCTGCGGC`, e a função de pontuação w tal que $w(\mu, \mu) = 1$, $w(\mu, \nu) = -1$ e $w(\mu, -) = w(-, \nu) = -2$, um alinhamento semiglobal dessas seqüências é mostrado abaixo

```
CTTCAG-CACTTG-GATTTCTCGG
----AGCCACCTGCGGC-------
```

Apesar de o alinhamento global ótimo possuir uma pontuação (-13) maior que aquela do alinhamento acima (-19), há de se convir que esse último mostra de maneira mais clara quão parecidas são as duas seqüências em questão. Isso reflete-se na

pontuação dos alinhamentos quando os espaços nas pontas não são levados em consideração. Nesse caso, o alinhamento global continua com sua pontuação igual a -13, enquanto que o alinhamento semiglobal tem sua pontuação melhorada de -19 para 1.

Essa outra variante do problema de alinhamento, que ignora os espaços nas pontas, também pode ser resolvida em tempo $O(mn)$. Para se encontrar o melhor alinhamento de duas seqüências, sem cobrar por espaços nas suas pontas, basta proceder como no caso do alinhamento global de duas seqüências, usando a Recorrência (2.1) para preencher a matriz de alinhamento. A única diferença está na inicialização dessa matriz, cuja primeira linha e coluna serão agora preenchidas com zero. Além disso, no caso do alinhamento semiglobal, o valor da similaridade não corresponderá àquele encontrado na última posição da matriz, mas sim ao maior valor presente em sua última linha ou coluna. Aqui vale observar que essa variante do PAS faz sentido apenas no caso em que a função de pontuação w utilizada associa um valor negativo aos pares de símbolos incluindo um espaço. Ou seja, ao "pagarmos" algo quando da escolha de uma coluna em branco durante a construção de um alinhamento.

Alinhamentos com funções genéricas e afins de penalização

Os algoritmos vistos até o momento têm tratado as colunas com espaços no alinhamento como eventos independentes. Isto é, cada uma dessas colunas contribui com um dado valor para a pontuação total do alinhamento sendo construído. Algumas vezes porém, podemos estar interessados em suposições como a de que a presença de vários espaços consecutivos é mais provável que a presença de vários espaços isolados em um alinhamento envolvendo tipos específicos de seqüências.

Uma série maximal de espaços consecutivos dentro das seqüências $\bar{s}$ e $\bar{t}$ de um alinhamento é chamada de **buraco**. Para um tratamento mais adequado dessas regiões, de acordo com a suposição acima, exige-se um algoritmo onde k espaços consecutivos não sejam pontuados de acordo com uma função linear $p(k) = \alpha k$, onde $\alpha = w(\mu, -)$ ou $\alpha = w(-, \mu)$ para um símbolo μ de um alfabeto qualquer Σ tal que $\{-\} \notin \Sigma$. As idéias descritas a seguir permitem a implementação de um outro algoritmo que calcula a similaridade de duas seqüências com respeito a uma função arbitrária p de pontuação de buracos.

A principal diferença deste algoritmo com relação aos vistos até agora está no fato de seu esquema de pontuação não ser aditivo com relação às colunas do alinhamento, mas sim com relação aos seguintes blocos nos quais um alinhamento pode ser dividido:

1. Dois caracteres emparelhados;
2. Uma série maximal de caracteres consecutivos em t alinhados com espaços em s;

3. Uma série maximal de caracteres consecutivos em s alinhados com espaços em t.

Agora, a pontuação do alinhamento não é mais calculada com base no conteúdo de cada coluna do alinhamento, e sim com base no conteúdo dos blocos descritos acima. Por isso, o que precisa ser armazenado agora, para cada par (i, j), é a maior pontuação dos alinhamentos envolvendo os prefixos $s[1..i]$ e $t[1..j]$ de s e t que terminam em um tipo particular de bloco. Isso pode ser feito utilizando três matrizes distintas, M_S, M_I e M_D, de dimensões $(m+1) \times (n+1)$. Cada uma delas está associada a um tipo de bloco e é inicializada da seguinte maneira:

1. $M_S[0][0] = 0$: matriz associada aos blocos com dois caracteres emparelhados;
2. $M_I[0][j] = -p(j)$: matriz associada aos blocos com buracos em s;
3. $M_D[i][0] = -p(i)$: matriz associada aos blocos com buracos em t.

O preenchimento das matrizes M_S, M_D e M_I pode ser feito por meio das seguintes recorrências, e o valor do alinhamento ótimo corresponde ao maior valor dentre aqueles encontrados nas últimas posições ($M_S[m][n]$, $M_D[m][n]$, e $M_I[m][n]$) das matrizes.

$$M_S[i][j] = w(s[i], t[j]) + \max \begin{cases} M_S[i-1][j-1]; \\ M_I[i-1][j-1]; \\ M_D[i-1][j-1]. \end{cases}$$

$$M_I[i][j] = \max \begin{cases} M_S[i][j-k] - p(k), \text{ para } 1 \leqslant k \leqslant j; \\ M_D[i][j-k] - p(k), \text{ para } 1 \leqslant k \leqslant j. \end{cases}$$

$$M_D[i][j] = \max \begin{cases} M_S[i-k][j] - p(k), \text{ para } 1 \leqslant k \leqslant i; \\ M_I[i-k][j] - p(k), \text{ para } 1 \leqslant k \leqslant i. \end{cases}$$

Para o preenchimento completo da matriz M_I, um total de $2m^2$ células das outras duas matrizes precisam ser analisadas, enquanto que o preenchimento completo da matriz M_D necessita que um total $2n^2$ células das matrizes M_S e M_I sejam verificadas. Disso, podemos concluir que a complexidade de tempo do algoritmo baseado na recorrência acima é $O(mn^2 + m^2n)$. Para a construção do alinhamento em si, basta proceder como no caso dos algoritmos anteriores, mantendo também a informação de qual matriz foi usada para o preenchimento de cada célula.

Considerando-se uma função de pontuação de buracos um pouco mais específica, como $p = g + rx$ (conhecida como função de pontuação afim), onde g corresponde ao valor da pontuação associada à abertura de um buraco e r, à pontuação para extensão desse buraco, um algoritmo com a mesma complexidade $O(mn)$ dos algoritmos básicos pode ser construído para busca da similaridade entre duas seqüências. Esse algoritmo, devido a Goth[43], está baseado nas seguintes recorrências que correspondem, na realidade, a um caso particular das recorrências descritas na seção anterior.

$$M_S[i][j] = w(s[i], t[j]) + \max \begin{cases} M_S[i-1][j-1] \\ M_I[i-1][j-1] \\ M_D[i-1][j-1] \end{cases}$$

$$M_I[i][j] = \max \begin{cases} M_S[i][j-1] + (h+g) \\ M_I[i][j-1] + g \\ M_D[i][j-1] + (h+g) \end{cases}$$

$$M_D[i][j] = \max \begin{cases} M_S[i-1][j] + (h+g) \\ M_I[i-1][j] + (h+g) \\ M_D[i-1][j] + g, \end{cases}$$

Para cada posição (i, j) das matrizes, $M_S[i][j]$ armazena a pontuação de um alinhamento ótimo de $s[1..i]$ e $t[1..j]$ que termina com $s[i]$ emparelhado com $t[j]$, $M_D[i][j]$ armazena a pontuação de um alinhamento ótimo de $s[1..i]$ e $t[1..j]$ que termina com um espaço emparelhado com $t[j]$ e $M_I[i][j]$ armazena a pontuação de um alinhamento ótimo de $s[1..i]$ e $t[1..j]$ que termina com $s[i]$ emparelhado com um espaço.

Além da similaridade, existe uma outra medida, denominada **distância de edição** que também pode ser utilizada na comparação de duas seqüências. Diferente da primeira, que determina quão parecidas são duas seqüências, a distância de edição mede quão próximas (evolutivamente) estão as seqüências sendo comparadas. Dadas as operações de edição do tipo inserção, remoção e substituição de um caractere, a distância de edição $\text{dis}(s, t)$ entre dua seqüências s e t pode ser definida como o número mínimo de operações de edição na seqüência s necessárias para transformá-la na seqüência t. Note que existe uma correspondência direta entre essas operações e as colunas de um alinhamento de s e t que incluem um espaço em s (inserção em s), um espaço em t (remoção em s) e dois caracteres diferentes (substituição em s). Dessa correspondência, determinar a distância de edicão entre duas seqüências equivale a calcular a pontuação de um de seus alinhamentos ótimos utilizando uma função de pontuação w tal que $w(\mu, \mu) = 0$, $w(\mu, \nu) = 1$, e $w(-, \mu) = 1$, $w(\mu, -) = 1$.

Uma observação importante a ser feita aqui é que, quando estamos tratando de distância de edição, um alinhamento ótimo de s e t corresponde ao alinhamento de pontuação mínima dentre todos os possíveis alinhamentos envolvendo essas seqüências. Isso decorre do fato de estarmos em busca do número mínimo (e não máximo) de operações que convertem uma seqüência na outra. A distância de edição entre duas seqüências pode ser determinada utilizando o Algoritmo 4 descrito anteriormente, tomando-se apenas o cuidado de escolher o mínimo entre $(M[i-1][j]+ins, M[i-1][j-1]+emp, M[i][j-1]+del)$ em cada uma de suas iterações.

Apenas para finalizar esta seção, gostaríamos de observar que o problema do alinhamento global admite uma série de variantes. Uma delas, que será abordada como parte do Capítulo 6, trata da comparação de várias seqüências. Para mais informações sobre o problema do alinhamento de seqüências e algumas de suas variantes, sugerimos novamente o livro escrito por Gusfield[46].

2.3 Conceitos básicos de Biologia Computacional

Até aqui temos falado dos conceitos ligados à Biologia e à Ciência da Computação como se essas fossem áreas totalmente isoladas uma da outra. Uma observação mais detalhada dos conceitos abordados até aqui mostra, no entanto, que essas duas áreas possuem muito em comum. A ligação entre elas fica clara ao considerarmos que uma dada seqüência genômica pode ser vista como uma seqüência de caracteres sobre o alfabeto $\Sigma = \{A, C, G, T\}$, cujos símbolos correspondem às iniciais das bases adenina, citosina, guanina e timina, respectivamente. Essa observação constitui a base de um ramo da Ciência da Computação conhecido como **Biologia Computacional**, cujas origens talvez remontem à década de 50, com o trecho que transcrevemos abaixo de uma carta enviada pelo físico George Gamow(1904-1968) a Watson(1928-) e Crick(1916-2004).

> *... But I am very much excited by your article in May 30th [1953] Nature, and think that bring Biology over into the group of "exact"sciences. I plan to be in England through most of September, and hope to have a chance to talk to you about all that, but I would like to ask a few questions now. If your point of view is correct [,] each organism will be characteristized by a long number written in quadrucal (?) system with figures 1, 2, 3, 4 standing for different bases ... This would open a very exciting possibility of theoretical research based on combinatorix [sic] and the theory of numbers! ... I have a feeling this can be done. What do you think?*

Dentre os problemas abordados pela Biologia Computacional está o do alinhamento de seqüências (e suas variantes) estudado anteriormente. Além dele, temos uma série

de outros problemas da área da Biologia, em especial da Biologia Molecular, que sofrem intervenção direta dos conceitos da Ciência da Computação. O seqüenciamento de uma molécula de DNA, por exemplo, exige um esforço computacional considerável durante a montagem dos fragmentos obtidos após a clivagem dos seus clones. Uma vez determinada a seqüência de um DNA de interesse, ela precisa ser processada na tentativa de se obter informações úteis a seu respeito. Dentre essas informações estão, por exemplo, a localização de suas regiões funcionais e posterior determinação do papel de cada uma delas. Além disso, podemos também estar interessados em outros problemas como a determinação da estrutura secundária e terciária dessa molécula. Mais informações sobre a Biologia Computacional e os problemas que ela aborda podem ser encontradas no livro [60], escrito por Meidanis e Setubal. Além dele, existem várias outras obras de referência sobre o assunto, como os livros escritos por Mount[66] e Pevzner[71].

Uma outra aplicação direta dos conceitos da Ciência da Computação na Biologia compreende o desenvolvimento de bases adequadas ao armazenamento eficiente dos dados provenientes do seqüenciamento das moléculas de DNA. O GenBank[15] é uma das várias bases de dados construídas para esse fim. Dentro do GenBank, as seqüências encontram-se armazenadas de várias formas diferentes. Um dos formatos mais conhecidos denomina-se FASTA, que inclui, basicamente, a identificação da seqüência na primeira linha, normalmente precedida do sinal '>' e, nas linhas seguintes, as bases que a constituem. A Figura 2.9 mostra um exemplo de seqüência no formato FASTA cuja identificação é AB010874.

Um outro formato no qual as seqüências armazenadas no GenBank podem ser encontradas é aquele mostrado na Figura 2.10. Aqui, além da seqüência em si, temos uma série de outras informações como a sua identificação (campo ACCESSION), referências aos artigos que dela tratam (campo REFERENCE), limites de suas regiões codificantes (campo CDS), etc.

Muito mais do que um simples repositório de seqüências genômicas, o GenBank inclui ferramentas que permitem a busca eficiente pelas seqüências nele contidas. Essa busca pode ser feita de várias maneiras diferentes. Uma delas exige apenas o conhecimento por parte do usuário da identificação da seqüência desejada. Além disso, podemos estar também interessados em seqüências dessa base parecidas com alguma outra que temos em mãos. Para recuperação dessas seqüências, o GenBank disponibiliza um conjunto de ferramentas que, para uma busca eficiente do seu conteúdo, foram construídas sobre heurísticas desenvolvidas especialmente para o problema de comparação de uma seqüência com várias outras. Uma dessas ferramentas denomina-se BLAST[11].

O funcionamento do BLAST está baseado nas noções de MSP (do inglês *maximal segment pair*) e LMSP (do inglês *locally maximal segment pair*). Dadas duas seqüências s e t, um MSP corresponde a um par de segmentos s' de s e t' de t tal que $|s|' = |t'|$ e cujo

```
>AB010874 Homo sapiens gene for ribosomal protein L41, complete cds
TTCGCCTTTCTCTCGGCCTTAGCGCCATTTTTTTGGGTGAGTGTTTTTTGGTTCCTGCGTTGGGATTCCG
TGTACAATCCATAGACATCTGACCTCGGCACTTAGCATCATCACAGCAAACTAACTGTAGCCTTTCTCTC
TTTCCCTGTATAAACCTCTGCGCCATGAGAGCCAAGGTGAGCGGTTCCTGGTAGTAAGCTTGGGAGGTAG
GAGTTGGCGAGTAGTAGCGGGGAGACGAAGGCAAGTCCGCCATACCTCCTGAACTACTGGGTTTCAAGGG
TGCCCAAGAGCTGGTGGGAGAGAGAAGGTAGTTTGTGAGAGAGCTAGCGGTTAAGTGCTATGGGTAGAGA
GGGTGGGCTTAGAAAAGGGTGGAATTCTGATCTTATGTTGGGAGGGTGTCCAAGTTACTGATGTAGTTGT
TACGACCAATCTTTCATACTTCTTGGTTAAGAATCTGTCCGGTTCTAAAGAGTGCATTTCATATCCTTGC
TAAGCCTACTAATAAGCTTCATCCCTTTTTTTTTTTTTTTTTTAATTATCTGCTGCTTGTGATCGGTT
GCTAGTGGAGGAAGAAGCGAATGCGCAGGTACGTTGAGACTTTGCCAGCCCAGGAAGAAGGAAAGTTCCC
TTGGACAAAAACTTTAGGAGAAACATTTGGTTTGGAATCTTAAAAGATCTTTAGGAGAAAAACGGTTTGA
GTGTTTTCTCCCTGGAGCCAGGATTTAACAGAACAGAGAACGATAGAACCGTAGTGCTTGTTTCATTTTA
CCACCTCATTCTTTATGTGGACGTTTGATTTAATGTGGGAGGGAAAGGCAACTCTGGTTTGAGGTGTATT
CCATTCCTGTGTCTGCTTTTCAGGCTGAAGCGCAAAAGAAGAAAGATGAGGCAGAGGTCCAAGTAAACCG
CTAGCTTGTTGCACCGTGGAGGCCACAGGAGCAGAAACATGGAATGCCAGACGCTGGGGATGCTGGCTAC
AAGTTGTGGGACTGCATGCTACTGTCTAGAGCTTGTCTCAATGGATCTAGAACTTCATCGCCCTCTGATC
GCCGATCACCTCTGAGACCCACCTTGCTCATAAACAAAATGCCCATGTTGGTCCTCTGCCCTGGTCCTGT
GACATTCTGGACTATTTCTGTGTTTATTTGTGGCCGAGTGTAACAACCATATAATAAATCACCTCTTCCG
CTGTTTTAGCTGAAGAATT
```

Figura 2.9: Exemplo de seqüência no formato FASTA

alinhamento (formado apenas por caracteres emparelhados das duas seqüências) possui pontuação máxima dentre todos os possíveis segmentos de mesmo tamanho de s e t. Um pouco diferente, um LMSP é um par de segmentos s' de s e t' de t tal que $|s|' = |t'|$ e cuja pontuação do alinhamento não pode ser melhorada pela expansão ou contração dos segmentos. Dadas essas definições, o que o BLAST propõe é, dada uma seqüência genômica s, encontrar todas as seqüências de um conjunto $T = \{t_1, t_2, \ldots, t_k\}$ que, juntamente com s, incluem um MSP cuja pontuação do alinhamento relacionado é maior que um certo valor V. Para isso, o BLAST utiliza uma heurística onde, primeiramente, são encontrados todos os segmentos de tamanho w da seqüência s e das seqüências em T cuja pontuação do alinhamento associado é maior que um certo valor v. Os pares encontrados nesse passo são denominados de *hits*. Cada um desses *hits* são então estendidos na busca por um LMSP cujo alinhamento possui pontuação maior que V. Para determinação dos *hits*, o que o BLAST faz é, para cada segmento α de s, de tamanho w, construir uma lista de palavras, também de tamanho w, cuja similaridade com α seja pelo menos u. Essas palavras são então armazenadas em uma estrutura de dados que permite a busca eficiente por todas as ocorrências exatas de uma delas no conjunto T.

```
LOCUS       AB010874            1209 bp    DNA     linear   PRI 25-DEC-2002
DEFINITION  Homo sapiens gene for ribosomal protein L41, complete cds.
ACCESSION   AB010874
VERSION     AB010874.1  GI:3298434
KEYWORDS    ribosomal protein L41.
SOURCE      Homo sapiens (human)
ORGANISM    Homo sapiens
REFERENCE   1
  AUTHORS   Go,H., Miyado,K., Hasuwa,H. and Taniguchi,S.
  TITLE     Characterization of human ribosomal protein L41 genomic structure
  JOURNAL   Unpublished
FEATURES             Location/Qualifiers
     source          1..1209
                     /organism="Homo sapiens"
                     /mol_type="genomic DNA"
                     /cell_type="melanoma"
     gene            join(165..176,566..588,864..906)
                     /gene="L41"
     CDS             join(165..176,566..588,864..906)
                     /gene="L41"
                     /codon_start=1
                     /product="ribosomal protein L41"
                     /protein_id="BAA31508.1"
                     /translation="MRAKWRKKRMRRLKRKRRKMRQRSK"
ORIGIN
        1 ttcgcctttc tctcggcctt agcgccattt ttttgggtga gtgtttttttg gttcctgcgt
       61 tgggattccg tgtacaatcc atagacatct gacctcggca cttagcatca tcacagcaaa
      121 ctaactgtag cctttctctc tttccctgta taacctctg cgccatgaga gccaaggtga
      181 gcggttcctg gtagtaagct tgggaggtag gagttggcga gtagtagcgg ggagacgaag
      241 gcaagtccgc catacctcct gaactactgg gtttcaaggg tgcccaagag ctggtgggag
      301 agagaaggta gtttgtgaga gagctagcgg ttaagtgcta tgggtagaga gggtgggctt
      361 agaaaagggt ggaattctga tcttatgttg ggagggtgtc caagttactg atgtagttgt
      421 tacgaccaat ctttcatact tcttggttaa gaatctgtcc ggttctaaag agtgcatttc
      481 atatccttgc taagcctact aataagcttc atccctttt ttttttttt tttttaatta
      541 tctgctgctt gtgatcggtt gctagtggag gaagaagcga atgcgcaggt acgttgagac
      601 tttgccagcc caggaagaag gaaagttccc ttggacaaaa actttaggag aaacatttgg
      661 tttggaatct taaaagatct ttaggagaaa aacggtttga gtgttttctc cctggagcca
      721 ggatttaaca gaacagagaa cgatagaacc gtagtgcttg tttcatttta ccacctcatt
      781 ctttatgtgg acgtttgatt taatgtggga gggaaaggca actctggttt gaggtgtatt
      841 ccattcctgt gtctgctttt caggctgaag cgcaaaagaa gaaagatgag gcagaggtcc
      901 aagtaaaccg ctagcttgtt gcaccgtgga ggccacagga gcagaaacat ggaatgccag
      961 acgctgggga tgctggctac aagttgtggg actgcatgct actgtctaga gcttgtctca
     1021 atggatctag aacttcatcg ccctctgatc gccgatcacc tctgagaccc accttgctca
     1081 taaacaaaat gcccatgttg gtcctctgcc ctggtcctgt gacattctgg actatttctg
     1141 tgtttatttg tggccgagtg taacaaccat ataataaatc acctcttccg ctgttttagc
     1201 tgaagaatt
//
```

Figura 2.10: Outro formato no qual as seqüências armazenadas no GenBank podem ser encontradas.

Capítulo 3

O Problema da Identificação de Genes

O problema da identificação, ou predição de genes consiste em, dada uma seqüência de DNA, determinar a posição inicial e final dentro dela de cada um dos seus genes, assim como a posição inicial e final dos éxons que os constituem. Vale lembrar que estamos sempre falando de seqüências eucarióticas.

É fato que existe uma relação direta entre as proteínas e as características estruturais e comportamentais de um dado organismo. Tomando-se como exemplo a espécie humana, sabemos que o nível de melanina em nossa pele determina sua tonalidade. Ou ainda, que a má formação de algumas proteínas ou sua produção em quantidade insatisfatória ocasiona o aparecimento de doenças específicas, como é o caso do diabetes. Desse fato e da ligação direta entre as proteínas e os genes presentes em um DNA, a identificação dessas estruturas torna-se de inquestionável importância prática. Se determinarmos a localização de um gene e posteriormente sua função no organismo poderemos, por exemplo, vir a desenvolver drogas que ajudem no combate de doenças ou pragas específicas.

Determinar os limites dos genes codificados em uma dada seqüência de DNA não é uma tarefa fácil. Isso se deve a uma série de fatores. O primeiro, e talvez o principal deles, está no fato de os genes eucarióticos incluírem tanto regiões codificantes, os éxons, como regiões não-codificantes, os íntrons. Além disso, os genes normalmente correspondem somente a uma pequena porção da seqüência sendo analisada. Apenas para citar um exemplo, Duret *et al.* em [33] estimam que os genes constituem apenas 3% do DNA humano, que inclui um total de aproximadamente 3×10^9 de bases. Além disso, os genes não possuem nenhum padrão de bases específico que permite distingüí-los das regiões intergênicas. Essas dificuldades podem ser estendidas diretamente aos éxons de um gene. Na grande maioria das vezes, as regiões codificantes são bem menores

que os íntrons entre os quais estão situadas. Os éxons também não possuem nenhum padrão bem definido e, mais do que isso, não existe uma seqüência de bases que pode ser encontrada única e exclusivamente nas fronteiras dessas regiões com as regiões não-codificantes.

Todas essas características ligadas aos genes (e também aos seus éxons) tornam o problema de localizar essas estruturas dentro do genoma algo difícil e interessante. A conseqüência principal desse fato está nos diversos métodos desenvolvidos para o problema da identificação de genes.

3.1 Métodos para identificação de genes

As motivações e dificuldades acima levaram ao desenvolvimento de métodos diversos que têm como objetivo principal a localização das regiões codificantes em uma dada seqüência de DNA eucariótico. Esses métodos costumam ser divididos em duas categorias principais: **métodos intrínsecos** ou *ab initio* e **métodos extrínsecos**.

3.1.1 Métodos intrínsecos

Os métodos intrínsecos utilizam-se apenas de informações presentes na própria seqüência (informações intrínsecas à seqüência) sendo analisada durante a busca pelos seus genes. Eles podem ser divididos em **métodos estatísticos** e **métodos de busca por sinais**, de acordo com o tipo de informação utilizada para distinção das regiões codificantes.

Métodos Estatísticos

Uma análise das bases constituintes dos éxons presentes em várias seqüências de DNA revela determinadas características estatísticas que podem ser úteis na distinção das regiões codificantes de uma seqüência qualquer. Dentre essas características está, por exemplo, o número de Gs e Cs, que costuma ser bem maior nos éxons que nos íntrons. Essa não-aleatoriedade da composição dos éxons pode ser utilizada na busca pelos genes de uma dada seqüência genômica.

A idéia geral dos métodos estatísticos é identificar porções da seqüência de interesse cujas características estatísticas, como a freqüência de cada um dos códons, assemelham-se àquelas de éxons já conhecidos. Para a determinação de quão parecida é, do ponto de vista estatístico, uma determinada região da seqüência com éxons previamente determinados, uma série de métricas estatísticas foram desenvolvidas. Dentre essas métricas, está o uso diferenciado dos códons nas regiões codificantes da seqüência genômica de

uma dada espécie. A Tabela 3.1 mostra, por exemplo, a freqüência (relativa) com que os diversos códons costumam ser encontrados nos éxons constituintes dos genes humanos.

TTT	0.0171	TCT	0.0149	TAT	0.0121	TGT	0.0103
TTC	0.0205	TCC	0.0177	TAC	0.0155	TGC	0.0126
TTA	0.0074	TCA	0.0120	TAA	0.0007	TGA	0.0012
TTG	0.0127	TCG	0.0045	TAG	0.0006	TGG	0.0131
CTT	0.0130	CCT	0.0174	CAT	0.0106	CGT	0.0046
CTC	0.0197	CCC	0.0201	CAC	0.0151	CGC	0.0107
CTA	0.0070	CCA	0.0168	CAA	0.0120	CGA	0.0062
CTG	0.0402	CCG	0.0071	CAG	0.0344	CGG	0.0117
ATT	0.0158	ACT	0.0130	AAT	0.0167	AGT	0.0120
ATC	0.0211	ACC	0.0192	AAC	0.0192	AGC	0.0195
ATA	0.0072	ACA	0.0149	AAA	0.0239	AGA	0.0117
ATG	0.0222	ACG	0.0062	AAG	0.0322	AGG	0.0117
GTT	0.0109	GCT	0.0185	GAT	0.0219	GGT	0.0108
GTC	0.0147	GCC	0.0284	GAC	0.0255	GGC	0.0226
GTA	0.0070	GCA	0.0159	GAA	0.0288	GGA	0.0164
GTG	0.0286	GCG	0.0076	GAG	0.0400	GGG	0.0165

Tabela 3.1: Freqüências dos códons constituintes dos genes humanos obtidas em `http://www.kazusa.or.jp/codon/`.

Dada a freqüência f de cada um dos códons e uma janela da seqüência sendo analisada contendo as triplas $c_1 c_2 \ldots c_n$, a probabilidade P da região limitada por ela ser codificante pode ser calculada de várias maneiras. Uma das mais simples é por meio da fórmula $\sum_{i=1}^{n} \log(\frac{f(c_i)}{f'(c_i)})$, conhecida como *log-likelihood ratio*, onde $f'(c_i)$ é a freqüência (relativa) do códon c_i calculada tomando-se várias regiões não-codificantes[1].

Tomando-se uma janela de tamanho 12 e deslizando-a sobre a seqüência AB010874 mostrada no capítulo anterior, podemos ver que o *log-likelihood ratio* da região entre as posições 165 e 176 dessa seqüência ser codificante (0.59) é maior, por exemplo, que o *log-likelihood ratio* daquelas de mesmo tamanho imediatamente anterior (0.36) e posterior (0.25) à ela. Exemplos mais detalhados da aplicação da freqüência diferenciada dos códons no problema da identificação de genes podem ser encontrados em [87] e [59].

Além do uso diferenciado de códons, uma série de outras métricas foram desenvolvidas para o tratamento do problema da identificação de regiões codificantes. Dentre as mais utilizadas estão:

[1] Assume-se normalmente que a distribuição dos códons nessas regiões segue um modelo aleatório. Ou seja, $f'(c_i) = \frac{1}{64}$.

1. Uso de aminoácidos: como conseqüência direta do uso diferenciado de códons, observa-se também uma freqüência diferenciada dos 20 aminoácidos componentes das proteínas. Traduzindo-se uma determinada seqüência de códons em uma seqüência de aminoácidos e utilizando os números mostrados na Tabela 3.1, pode-se calcular a probabilidade da seqüência resultante corresponder a uma seqüência de oligopeptídeos derivada de uma região codificante. Em [59], McCaldon e Argos baseiam-se nessa estratégia para determinar os possíveis genes em várias seqüências de DNA;

2. Uso de hexamer: cálculos baseados na freqüência de outros tipos de oligonucleotídeos (cadeia contendo de 2 a 20 nucleotídeos), que não sejam códons, também podem ser usados na determinação das regiões codificantes dentro de seqüências de DNA. Claverie *et al.*[29] foram os primeiros a usar uma métrica estatística baseada na freqüência de hexamers (cadeia de seis nucleotídeos). De acordo com Ficket e Tung em [36], e provavelmente devido à dependência existente entre aminoácidos adjacentes nas proteínas, essa corresponde a uma das métricas mais adequadas para discriminação entre regiões codificantes e não-codificantes de uma seqüência genômica;

3. Protótipo de códon: através da freqüência dos códons, pode-se derivar a probabilidade de cada uma das quatro bases ocorrer nas três posições dos códons componentes das regiões codificantes. Em [78], Sheperd notou que os códons mais freqüentes são da forma RNY(R = A ou G, Y = C ou T e N qualquer nucleotídeo), sugerindo um método para determinar regiões codificantes através da medição do número de diferenças entre a região sendo analisada e o padrão RNYRNY...RNY.

Todos os métodos estatísticos que se utilizam das métricas vistas acima necessitam de uma amostra das regiões codificantes de uma ou mais seqüências de DNA para estimar as freqüências relacionadas à métrica utilizada (códons, aminoácidos, hexamers etc.). Entretanto, para determinadas espécies, uma amostra significativa dos seus genes pode não estar disponível, tornando-se necessário o desenvolvimento de métricas que não dependam, a priori, de nenhum modelo representativo das regiões codificantes de uma seqüência. Uma série de métricas desse tipo tem sido proposta nos últimos anos (Assimetria de Posição[37, 86], Índice de Assimetria Periódica[52], Espectro de Fourier[56, 80], Entropia[10], etc.) e todas elas possuem como princípio a suposição biológica de que mudanças aleatórias são menos prováveis de ocorrer nas regiões codificantes de uma seqüência de DNA do que nas suas regiões não-codificantes. Uma descrição geral dessas métricas e de todas as outras citadas nesta seção pode ser encontrada em [36].

Métodos de busca por sinais

Os sinais dos genes codificados em várias seqüências de DNA costumam apresentar um padrão de bases específico. Isso deve-se a uma forte ligação entre a estrutura desses elementos e, conseqüentemente, de suas bases, com o maquinário celular envolvido no processo de síntese de proteínas. A região promotora dos genes humanos, por exemplo, costuma incluir as seqüências `TATAAA` e `CCAAT` aproximadamente 20 e 80 bases, respectivamente, anterior ao início do sítio de transcrição. Essa padronização dos sinais também pode ser utilizada como informação útil na localização dos genes presentes em uma dada seqüência de DNA.

A idéia dos métodos de busca por sinais é localizar dentro da seqüência de interesse porções dela que se assemelham à seqüência representativa de um sinal específico. Nos primeiros estudos envolvendo identificação de genes pela busca por sinais, a seqüência representativa de um sinal específico era determinada por meio do alinhamento de várias seqüências desse tipo e determinação da sua seqüência consenso[2]. As regiões do DNA sendo analisado mais parecidas com o consenso corresponderiam ao sinal procurado.

Uma maneira mais sofisticada de resumir a semelhança existente entre seqüências de um sinal específico é através do registro das freqüências dos nucleotídeos em cada uma das posições dessas seqüências. Isto é, as seqüências individuais são alinhadas e a freqüência de cada base b na posição i do sinal é registrada como $f(b, i)$. Então, uma **matriz de peso** m, também chamada PWM (do inglês *positional weight matrices*), pode ser derivada dos valores de f. Mais especificamente, $m[b][i] = log[f(b,i)/f'(b)]$, onde $f'(b)$ corresponde à freqüência genômica da base b. A Tabela 3.2 mostra a freqüência de cada uma das bases `A`, `C`, `G` e `T` nas 15 posições de um conjunto de 389 regiões promotoras distintas.

	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11
A	61	16	352	3	354	268	360	222	155	56	83	82	82	68	77
C	145	46	0	10	0	0	3	2	44	135	147	127	118	107	101
G	152	18	2	2	5	0	20	44	157	150	128	128	128	139	140
T	31	309	35	374	30	121	6	121	33	48	31	52	61	75	71

Tabela 3.2: Freqüência com que cada uma das bases são encontradas nas 15 posições de um conjunto de regiões promotoras. Esses valores foram retirados de [23].

Dada uma matriz de peso que representa de maneira adequada um certo tipo de sinal, a probabilidade de uma seqüência s qualquer, de tamanho n, incluir um sinal desse tipo pode também ser determinada por meio do cálculo do seu *log-likelihood ratio*

[2]Seqüência tal que a base em sua i-ésima posição é aquela que aparece mais vezes na i-ésima coluna do alinhamento

$\sum_{i=1}^{n} m[s[i]][i]$, onde $s[i]$ é a base ocupando a i-ésima posição da seqüência s. Tomando-se como exemplo a região da seqüência AB010874 mostrada no capítulo anterior, que vai de sua posição 148 até sua posição 162, temos um *log-likelihood ratio* de 1.0. O modelo que acabamos de descrever é conhecido na literatura como *Weight Matrix Model* e foi originalmente proposto por Staden em [85]. Um exemplo mais detalhado do uso de matrizes de peso na busca por regiões promotoras de genes eucarióticos pode ser encontrado em [23].

Note que, da forma como as matrizes de peso são calculadas, assume-se uma independência entre as bases componentes do sinal em questão. De forma a capturar uma possível dependência entre nucleotídeos vizinhos, um outro tipo de matriz pode ser utilizado, cujos elementos são calculados como $log[f(ab,i)/p(ab)]$, onde a é a base na posição $i-1$ do sinal e $p(ab)$ à freqüência genômica do dinucleotídeo ab. Esse método, conhecido como *Weight Array Matrix*, foi proposto por Zhang e Marr em [97], Nesse trabalho, eles o aplicam na identificação de sítios de doação em seqüências de pré-RNAm do fungo *Schizosaccharomyces pombe*. Nada impede que sejam observadas também dependências entre bases não-adjacentes de um sinal específico. No intuito de determiná-las, métodos mais recentes utilizam-se de modelos de máxima entropia[96] e redes Bayesianas[26]. Uma revisão dos métodos para identificação de genes por busca de sinais pode ser encontrada em [41].

Uma outra forma de identificar as regiões codificantes em uma dada seqüência de DNA utilizando informações a respeito das freqüências dos códons e conservação dos sinais é através de um modelo conhecido como **Modelo Oculto de Markov**, comumente referenciado como HMM (do inglês *Hidden Markov Model*). HMMs são amplamente utilizados para a modelagem de uma sucessão de eventos onde alguns fatos são observáveis enquanto outros mantêm-se ocultos. Um exemplo típico de sucessão de eventos com essas características é o lançamento, um número arbitrário de vezes, de um dado que pode ser honesto ou viciado. Em cada lance sabemos apenas qual número foi obtido, mas não podemos determinar qual dado foi lançado. Utilizaremos esse exemplo para uma melhor descrição do Modelo Oculto de Markov. Formalmente, um HMM pode ser definido como uma quíntupla $\mathcal{H} = \{S, V, A, B, C\}$, onde:

1. $S = \{S_1, \ldots, S_n\}$ é um conjunto finito de elementos denominados estados. Em conjunto, os estados correspondem à parte oculta do modelo e cada um deles está associado a um tipo de evento do sistema sendo modelado. No nosso exemplo, temos dois estados distintos S_h e S_v, cada qual associado a um dos dois tipos de dado que pode ser lançado: honesto(h) ou viciado(v);

2. $V = \{v_1, \ldots, v_m\}$ é um conjunto finito de elementos denominados símbolos. Esses símbolos representam os fatos que podem ser observados durante a sucessão de eventos sendo modelada. No nosso exemplo, temos os símbolos 1,2,3,4,5,6, correspondentes aos valores que podem ser obtidos durante o lançamento do dado;

3. $A = \{\alpha_{ij}\}$ $(1 \leq i \leq n$ e $1 \leq j \leq n)$ é um conjunto também finito de elementos que determinam a probabilidade do modelo estar no estado S_j, no instante $t+1$, dado que ele esteve no estado S_i, no instante t. Em nosso exemplo, isso pode ser visto como as probabilidades, definidas a priori, de se trocar ou não de dado durante uma seqüência de lançamentos;

4. $B = \{\beta_{kj}\}$ $(1 \leq i \leq m$ e $1 \leq j \leq n)$ é um conjunto de elementos que determinam a probabilidade de o modelo observar o símbolo v_k, no instante t, dado o estado S_j nesse mesmo instante. Em nosso exemplo, temos uma probabilidade de 1/6 para cada um dos números que podem ser observados em cada lançamento do dado honesto;

5. $C = \{\pi_j\}$ $(1 \leq j \leq n)$é um conjunto de elementos que determinam a probabilidade de a sucessão de eventos iniciar-se no estado S_j. Em nosso exemplo, isso pode ser visto como as probabilidades de se começar a seqüência de lançamentos com um dado viciado ou um dado honesto.

Uma forma amplamente utilizada para representação de um HMM é um diagrama como o mostrado na Figura 3.1. Nesse diagrama, os elementos do conjunto de estados S encontram-se representados pelos retângulos, os símbolos de observação pelos números 1, 2, 3, 4, 5 e 6 e as probabilidades pelos números localizados dentro dos retângulos (probabilidade dos símbolos de observação), nas setas ligando esses retângulos (probabilidade de transições entre os estados) e naquelas entrando nos estados (probabilidade dos estados iniciais).

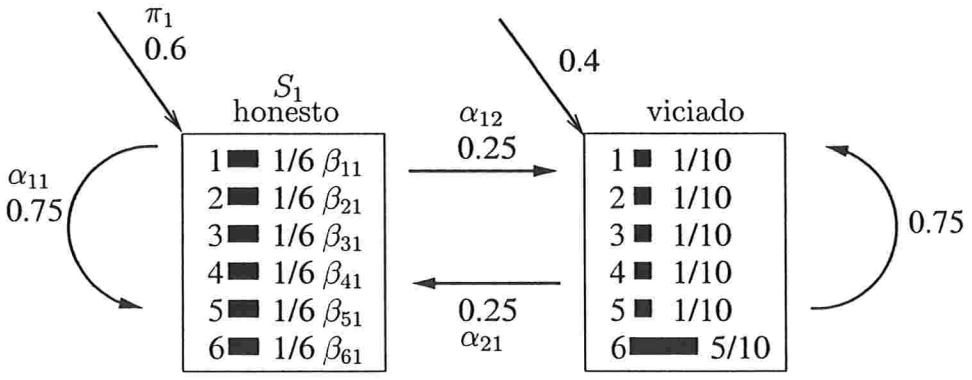

Figura 3.1: Diagrama representativo de um Modelo Oculto de Markov que modela a sucessão de lançamento de dados utilizada como exemplo.

Dada uma sucessão de eventos qualquer $\mathcal{E}$ e um HMM $\mathcal{H}$ que a represente de forma adequada, várias são as questões que podemos fazer tomando esses dois elementos em conjunto. Uma delas diz respeito à maior probabilidade da sucessão de eventos em

questão dado o modelo. Tomando-se com exemplo a seqüência de números 1, 4, 5, 6, a maior probabilidade dessa sucessão de eventos dado o modelo mostrado na Figura 3.1 é $0.6*1/6*0.75*1/6*0.75*1/6*0.25*5/10$. Esse valor pode ser calculado determinando a probabilidade associada a cada um dos caminhos[3] em β que dão origem à sucessão de eventos $\mathcal{E}$ e tomando como resposta a maior dentre todas as probabilidades calculadas. O problema com essa estratégia está no fato de o número de caminhos possíveis crescer exponencialmente com o tamanho da sucessão de eventos. Uma forma mais eficiente de realizar esta tarefa é através da utilização de um algoritmo denominado Algoritmo de *Viterbi*[92]. Por meio da estratégia de programação dinâmica, este algoritmo determina a maior probabilidade (e o caminho ótimo associado) de uma sucessão de eventos qualquer em tempo $O(N^2M)$, onde N corresponde ao número de estados do modelo e M ao número de eventos sucessivos.

Um HMM pode ser utilizado para a modelagem de um conjunto de seqüências representativas de um certo sinal. Tomando-se um alinhamento das seqüências determinantes de um sinal específico, um HMM para esse alinhamento poderia ser construído criando-se uma estado S_i para cada uma de suas colunas. A probabilidade de transição de um estado S_i para um estado S_{i+1} é igual a um. O conjunto de símbolos de observações incluem os caracteres `A`, `C`, `G` e `T`. A cada um desses símbolos está associada uma probabilidade f' calculada por meio da freqüência com que eles aparecem nas colunas do alinhamento. Na necessidade de se capturar possíveis dependências entre dois nucleotídeos vizinhos, teríamos um modelo mais genérico, onde o conjunto de símbolos incluiria os dinucleotídeos e a freqüência com que eles aparecem nas colunas do alinhamento seriam utilizadas para cálculo das probabilidades condicionais. Um HMM também pode ser utilizado para modelagem de certas características intrínsecas às regiões codificantes, como por exemplo a freqüência diferenciada dos seus códons. Aqui, teríamos um estado para cada uma das três posições de um códon. As probabilidades de ocorrência dos símbolos `A`, `C`, `G` e `T` em cada uma dessas três posições (protótipo de códon) podem ser calculadas com base em uma amostra de éxons já conhecidos. Em termos de diagrama, teríamos algo como o mostrado na Figura 3.2, onde assumimos uma distribuição uniforme das triplas de bases.

Dado um HMM $\mathcal{H}$ cujos estados e parâmetros associados modelam de forma adequada um conjunto de seqüências representativas de uma certa região de interesse - seja ela um sinal ou uma região codificante - e uma seqüência s qualquer, poderíamos nos questionar, por exemplo, a respeito da probabilidade de s corresponder a uma região daquele tipo ou não. Para isso, basta fazermos uso do algoritmo de Viterbi mencionado acima. Caso a probabilidade encontrada seja maior que um certo limiar, podemos afirmar que a seqüência em questão é interessante para nossos propósitos. Informações adicionais sobre Modelos Ocultos de Markov e suas possíveis aplicações em problemas envolvendo seqüências genômicas podem ser encontradas em [76].

[3]Nesse contexto, um caminho corresponde a uma sucessão de estados do modelo.

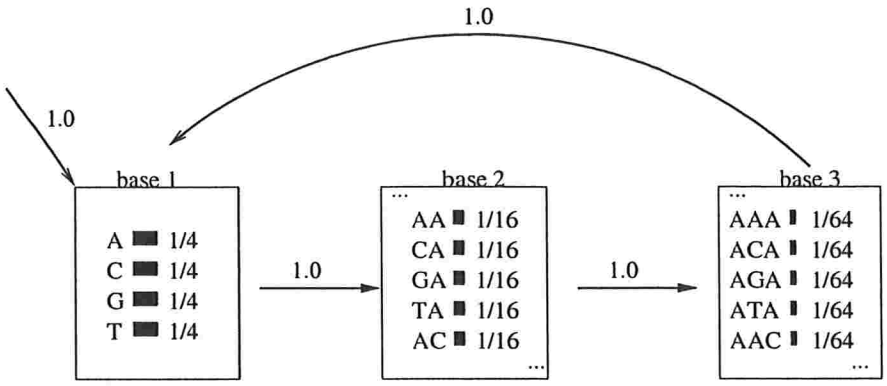

Figura 3.2: Diagrama representando um Modelo Oculto de Markov para regiões codificantes

Os métodos intrínsecos em geral possuem algumas desvantagens. No que diz respeito aos métodos de busca por sinais, precisamos levar em conta o fato de que as seqüências dos sinais envolvidos na determinação dos genes podem estar degeneradas ou mal definidas, impedindo uma distinção clara entre as porções da seqüência que realmente participam do processo de síntese de proteínas daquelas aparentemente não funcionais. Sobre os métodos estatísticos, temos a desvantagem de suas aplicações estarem restritas a seqüências parecidas com aquelas utilizadas no cálculo do valor da métrica utilizada como base da função discriminante. Esses problemas podem ser amenizados pela combinação de ambas as informações durante a busca pelas regiões codificantes em um dado DNA.

3.1.2 Métodos extrínsecos

Ao contrário dos métodos intrínsecos, os métodos extrínsecos utilizam-se de informações presentes em outras seqüências (informações extrínsecas à seqüência) que não aquela sendo analisada na busca pelos seus genes.

Também conhecidos como métodos de busca por similaridades[4] entre seqüências, a idéia geral dos métodos extrínsecos é identificar regiões dentro da seqüência de interesse significantemente parecidas com seqüências representativas de genes já conhecidos. Essas últimas referem-se tanto a seqüências de transcritos (RNAm, cDNA, proteínas, ESTs) quanto a outras seqüências genômicas cujos genes encontram-se já determinados. Esses métodos possuem como fundamento o princípio da conservação das bases mencionado no Capítulo 2. Ou seja, a tendência de as bases componentes das regiões codificantes conservarem-se ao longo da evolução da seqüência dada a relação existente entre a funcionalidade delas e das bases que as constituem.

[4]Nesse capítulo, a palavra similaridade está sendo utilizada com o sentido de semelhança.

Em determinadas ocasiões, a busca por regiões semelhantes em duas seqüências pode ser realizada utilizando os algoritmos de alinhamento descritos no capítulo anterior. Note pelo alinhamento local da Figura 3.3 a semelhança entre o último éxon da seqüência AB010874 e um segmento de uma molécula de RNAm transcrita a partir de um gene homólogo àquele ao qual o éxon pertence.

```
861 CAGGCTGAAGCGCAAAAGAAGAAAGATGAGGCAGAGGTCCAAGTAA  906
    ||||||||||||||.||||||||||||||||||||||||||||||
33  caggctgaagcgcaagagaagaaagatgaggcagaggtccaagtaa   78
```

Figura 3.3: Alinhamento local obtido por meio da execução do programaWATER (`http://bioweb.pasteur.fr/seqanal/interfaces/water.html`).

Na grande maioria das vezes, porém, os algoritmos tradicionais de alinhamento não podem ser aplicados de forma direta para a comparação de duas seqüências na busca por seus genes. No caso dessa comparação envolver duas seqüências de DNA, por exemplo, precisamos fazer uso de variantes daqueles algoritmos que tratam de forma adequada as regiões mal conservadas dessas seqüências, como os íntrons e as regiões-intergênicas. Algumas dessas variantes serão abordadas nos capítulos seguintes.

Uma das vantagens dos métodos extrínsecos é o fato de que, caso uma similaridade relevante seja encontrada entre uma determinada região da seqüência sendo analisada e um gene cuja função já tenha sido determinada, informações relacionadas a essa função podem nos dar boas indicações sobre a funcionalidade do novo gene encontrado. A desvantagem desse método é que, no caso de não existir nenhuma semelhança entre o novo gene a ser descoberto e genes já conhecidos, pouca informação útil pode ser obtida através da busca por similaridades entre seqüências. Estudos realizados por Claverie e descritos em [28] mostram que, em 1997, 50% dos genes ainda não determinados poderiam ser localizados por meio da comparação de suas seqüências com as de genes já conhecidos. É provável que essa porcentagem seja maior nos dias de hoje dado o crescente aumento das seqüências genômicas totalmente determinadas[58].

Note que a aplicação direta dos métodos acima permite apenas a identificação de componentes isolados de um possível gene na seqüência sendo analisada. Programas que apenas implementam um dos métodos de busca por sinais, como por exemplo o NETGENE[22], conseguem determinar somente possíveis sítios de aceitação/doação ou promotores em uma dada seqüência de interesse. Já aqueles desenvolvidos com base em um dos métodos estatísticos, como é o caso do TESTCODE[37], apenas identificam possíveis éxons da seqüência sendo analisada, sem se preocupar com a estrutura dos genes onde eles possam estar inseridos. Em outras palavras, a aplicação direta dos métodos intrínsecos e extrínsecos não leva a uma solução do problema sendo abordado:

a elucidação da estrutura completa dos genes em uma seqüência. Para isso, um pós-processamento faz-se necessário. Após a determinação dos possíveis éxons em uma seqüência, eles precisam ser montados de forma a compor um gene nela existente. Em linhas gerais, é isso que as ferramentas de predição, de uma forma ou de outra, propõem-se a fazer.

3.1.3 Ferramentas de predição de genes

Apesar de os métodos para identificação de regiões codificantes existirem desde o início da década de 80, nenhum programa de predição gênica havia sido implementado até o começo da década de 90. Provavelmente, a primeira ferramenta para predição de genes corresponda ao programa conhecido como *GM*[38], inicialmente desenvolvido para localização de genes na seqüência da *Caenorhabditis elegans*. A partir daí, uma série de outros programas com esse fim começaram a surgir. De acordo com as informações utilizadas por elas na determinação dos possíveis éxons, as ferramentas de predição costumam ser divididas em dois grupos principais.

No primeiro grupo, encontram-se as ferramentas de predição que se utilizam de informações intrínsecas à seqüência de interesse na busca por seus éxons. Uma das ferramentas que melhor representa esse conjunto de programas denomina-se GENSCAN. Desenvolvida pro Chris e Burge e descrita em [24], esse programa utiliza-se tanto de métodos estatísticos quanto de métodos de busca por sinais na tentativa de encontrar os genes codificados em um dada seqüência de entrada. As informações provenientes da aplicação de ambos os métodos são combinadas por meio de um HMM especial conhecido como HMM generalizado que permite a identificação dos possíveis éxons e a montagem eficiente desses elementos em um gene completo. Além do GENSCAN as ferramentas GENIE[54], FGENESH[74], e GRPL[47] utilizam-se também de um HMM generalizado na identificação dos genes em uma dada seqüência de DNA. Um pouco diferente, os programas FGENE[84], GAP3[95] e DAGGER[27], separam a tarefa de predição em duas fases bem distintas de determinação e posterior montagem dos éxons. Uma vez determinados, os possíveis éxons recebem uma certa pontuação de acordo com a probabilidade de corresponderem a um éxon real, e são então montados por meio de um algoritmo baseado na estratégia de programação dinâmica ou na busca de um caminho ótimo em um grafo cujos vértices representam os éxons e as arestas pares de éxons compatíveis.

No outro grupo, encontram-se as ferramentas que se utilizam basicamente de informações sobre as semelhanças entre uma seqüência e um gene já conhecido durante a tarefa de predição. Um dos primeiros programas desse tipo foi desenvolvido por Gelfand *et al.* e encontra-se descrito em [42]. Conhecida como PROCRUSTES, essa ferramenta busca em uma dada seqüência de entrada os éxons cuja concatenação e posterior tradução mais se assemelham a uma determinada proteína, também dada como

entrada. Isso é realizado eficientemente por meio de um algoritmo que se utiliza da estratégia de programação dinâmica. Além de proteínas, seqüências de cDNA também podem ser utilizadas como seqüências de comparação. É desse tipo de seqüências que ferramentas como GENEWISE[18], SIM4[39], GENESEQER[90], EST_GENOME[65] e PROJECTOR[61] utilizam-se na busca pelos éxons da seqüência principal. Ultimamente, seqüências genômicas também vêm sendo utilizadas na comparação de uma seqüência cujos genes estão sendo procurados. É o caso por exemplo das ferramentas ROSETTA[14], CEM[13], UTOPIA[19], PROGEN[68] e, mais recentemente, AGENDA[72], SLAM[8] e SGP[94, 69].

Além das ferramentas implementadas com base nas características estatísticas das regiões codificantes ou nas suas semelhanças com outras seqüências, existem também aquelas que se utilizam de ambas essas informações durante a tarefa de predição. É o caso por exemplo do EUGÈNE'HOM[40] e do TWINSCAN[53]. Essa última é uma versão melhorada do GENSCAN onde os genes são confirmados pela busca, em várias bases de dados, de seqüências transcritas semelhantes a eles. Por fim, temos um outro grupo de programas cujas predições baseiam-se em informações sobre os genes identificados por várias ferramentas distintas. Um exemplo de ferramenta desse grupo é o programa conhecido como COMBINER, desenvolvido por Allen *et al.* e descrito em [9].

A grande maioria desses programas recebe como entrada uma seqüência de DNA no formato FASTA e devolve como saída uma série de informações sobre os éxon preditos, como suas coordenadas dentro da seqüência, tipo (inicial, final, ou interno) e, em alguns casos, a probabilidade de acerto associada a cada um deles[5]. Uma revisão de várias das ferramentas existentes assim como dos métodos de predição implementadas por elas foi realizada por Mathé *et al.* e pode ser encontrada em [58]. Além dele, recomendamos também a leitura de [21], escrito por Brent e Guigó, para informações sobre avanços recentes na área. Finalmente, uma extensa bibliografia sobre o assunto pode ser consultada em `http://www.nslij-genetics.org/gene/`, sítio este mantido por W. Li.

3.1.4 Estimando a qualidade das ferramentas

Apesar de grandes avanços na área, ainda não podemos afirmar que o problema da identificação de genes esteja totalmente resolvido. Vários estudos sobre a precisão de alguns desses programas foram feitos nos últimos anos e as conclusões desses trabalhos mostram que ainda existe um longo caminho a ser percorrido até a elucidação completa do problema.

[5]Ainda não existe um consenso quanto ao formato da saída desses programas, mas muitos são os esforços na busca pela generalização de um formato conhecido como **GFF** (do inglês *Gene-Finding Format*), cujas especificações podem ser encontradas em `http://www.sanger.ac.uk/Software/formats/GFF/`.

No intuito de determinar quão precisa é uma certa ferramenta na tarefa de localização dos genes em uma dada seqüência, necessitamos de medidas objetivas que nos permitam avaliar a qualidade de sua predição. Isso pode ser realizado em dois níveis distintos: de éxons e de nucleotídeos. Em ambos os níveis, a qualidade de uma predição é determinada utilizando medidas comumente chamadas de especificidade e sensibilidade. Essas medidas foram originalmente descritas por Burset e Guigó em [25] e são amplamente utilizadas para avaliação dos resultados gerados por várias das ferramentas de predição existentes.

Para uma melhor compreensão das medidas de especificidade e sensibilidade no nível de nucleotídeos, considere as seguintes definições. Um nucleotídeo é dito realmente (ou predito como) codificante quando se localiza dentro de um éxon real (ou predito) da seqüência de interesse. Por éxon real queremos dizer um éxon cujos limites dentro da seqüência já foram confirmados experimentalmente e devidamente anotados. Ao contrário, um nucleotídeo é dito realmente (ou predito como) não-codificante quando localiza-se fora de um éxon real (ou predito) da seqüência. Sejam FN, FP, VN, VP o número de núcleotídeos erroneamente identificados como não-codificantes (falso-negativo), erroneamente identificados como codificantes (falso-positivo), corretamente identificados como não-codificantes (verdadeiros-negativos) e corretamente identificados como codificantes (verdadeiros-positivos), respectivamente (Figura 3.4). Dadas essas definições, os valores de especificidade e sensibilidade no nível de nucleotídeo podem ser calculados por meio das seguintes fórmulas:

- Especificidade no nível de nucleotídeos ($Sp_n = \frac{VN}{VN+FP}$): proporção de nucleotídeos realmente não-codificantes na seqüência de interesse corretamente identificados como não-codificantes pelo programa de predição;
- Sensibilidade no nível de nucleotídeos ($Sn_n = \frac{VP}{VP+FN}$): proporção de nucleotídeos realmente codificantes na seqüência original corretamente identificados como codificantes pelo programa de predição.

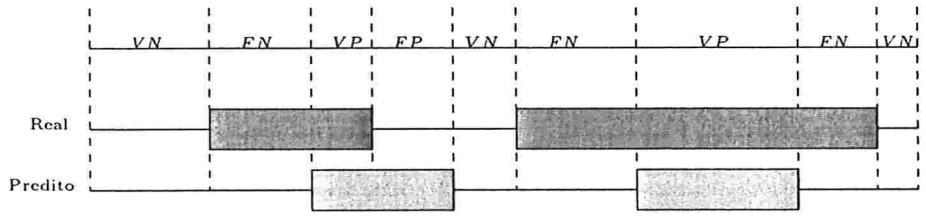

Figura 3.4: Uma representação de nucleotídeos realmente codificantes e aqueles identificados como tal por algum programa de predição.

Como o número de nucleotídeos não-codificantes dentro das seqüências de DNA é normalmente bem maior que o de nucleotídeos codificantes, TN costuma ser bem maior

que FP, implicando um alto valor de especificidade (Sp_n), nem sempre informativo da qualidade da predição obtida. Com isso, a seguinte fórmula alternativa tem sido utilizada na literatura [82, 32] para cálculo de Sp_n.

- Especificidade no nível de nucleotídeos ($Sp_n = \frac{VP}{VP+FP}$): proporção de nucleotídeos identificados como codificantes pelo programa de predição realmente codificantes.

Tanto a especificidade quanto a sensibilidade de uma predição podem ser resumidas em uma única medida conhecida como **Coeficiente de Correlação**, definida da seguinte forma:

$$CC = \frac{(TP \times TN) - (FN \times FP)}{\sqrt{(TP+FN) \times (TN+FP) \times (TP+FP) \times (TN+FN)}}$$

Essa medida, apesar de amplamente utilizada, possui a indesejável propriedade de não estar definida para aquelas seqüências que, por exemplo, não incluem nenhum gene. Note que, nesse caso, $(TN + FP) = 0$. Uma medida parecida ao CC e que pode ser utilizada em quaisquer circunstâncias é a **Correlação Aproximada** AC. Introduzida por Burset e Guigó em [25], ela pode ser assim calculada:

$$AC = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TP}{TP+FP} + \frac{TN}{TN+FP} + \frac{TN}{TN+FN}\right) - 1$$

Para o cálculo da qualidade de uma predição no nível de éxon, dizemos que um éxon encontra-se corretamente identificado quando seus limites (posição de seu primeiro e último nucleotídeo na seqüência original) são idênticos aos limites de algum éxon real da seqüência. Sejam NEC o número de éxons corretamente identificados, NEP o número de éxons preditos e NER o número de éxons reais. Com essas definições em mente, podemos calcular a especificidade e a sensibilidade no nível de éxon por meio das seguintes fórmulas:

- Especificidade no nível de éxon ($Sp_e = \frac{NCE}{NPE}$): proporção de éxons corretamente identificados pela ferramenta, com relação ao total de éxons (corretos e não corretos) preditos;
- Sensibilidade no nível de éxon ($Sn_e = \frac{NCE}{NRE}$): proporção de éxons corretamente identificados pela ferramenta, com relação ao total de éxons existentes na seqüência.

Os valores de especificidade e sensibilidade no nível de éxons costumam ser resumidos através da média $Av = \frac{Sp_e + Sn_e}{2}$ desses valores

A avaliação no nível de éxons provê informações complementares àquelas providas pela avaliação no nível de nucleotídeos. Enquanto que nessa última medimos quão bem as regiões codificantes são localizadas, no nível de éxons, medimos quão bem os sinais são identificados pelos programas.

Por meio das medidas descritas acima, Burset e Guigó em [25] e, mais recentemente, Rogic *et al.* em [73], realizaram um estudo extensivo de várias ferramentas de predição existentes. No primeiro trabalho, foram testadas tanto ferramentas implementadas com base em métodos intrínsecos quanto aquelas incluindo informações de similaridades entre a seqüência de interesse e genes já conhecidos. Um total de 570 seqüências foram utilizadas por Burset e Guigó durante o referido trabalho. Seus resultados mostram que a precisão das ferramentas citadas está bem abaixo do esperado pelo usuário. Em números, os valores de AC apresentados pelas ferramentas avaliadas variam entre 0.67 e 0.78. No nível de éxons, menos da metade de um total de 2649 regiões codificantes foram corretamente identificadas pelos programas. A má qualidade dos resultados apresentados pelas ferramentas torna-se mais evidente ao serem testadas sobre um conjunto menor de seqüências incluindo apenas aquelas que, supostamente, foram determinadas após a publicação dos referidos programas. Nesse caso, os valores de AC variam entre 0.62 e 0.71. Isso supõe uma forte dependência entre a precisão das ferramentas e as seqüências em que foram treinadas.

Uma outra conclusão importante do trabalho desenvolvido por Burset e Guigó diz respeito à precisão das ferramentas que se utilizam de informações extrínsecas à seqüência sendo analisada. Ou seja, de informações sobre o grau de semelhança de regiões dessas seqüências com genes já determinados. No trabalho de Burset e Guigó, essas ferramentas apresentaram valores de especificidade e sensibilidade um pouco maiores que as que utilizam somente informações da própria seqüência. Mais do que isso, mesmo após a retirada de algumas seqüências do conjunto inicial que pudessem superestimar a qualidade dos resultados apresentados por essas ferramentas, elas continuaram mostrando a mesma precisão. Em outro trabalho de avaliação [44], realizado por Guigó *et al.*, as ferramentas GENSCAN[24], PROCRUSTES[42] e GENEWISE[17] foram testadas utilizando uma seqüência genômica de aproximadamente 200kb. Essa seqüência foi construída artificialmente por meio da concatenação de outras seqüências compostas basicamente de genes já conhecidos situados entre bases geradas aleatoriamente. Os resultados dos testes realizados mostram que em situações como essas, um pouco mais realistas, a qualidade do GENSCAN diminui significativamente enquanto que PROCRUSTES e o GENEWISE mantêm seus resultados.

A maior precisão das ferramentas extrínsecas confirma-se também em um outro trabalho de comparação realizado por nós e descrito com detalhes em [2]. Nesse trabalho, escolhemos cinco ferramentas de predição distintas daquelas utilizadas em trabalhos anteriores. Três delas, descritas abaixo, foram implementadas com base em diferentes métodos de identificação:

1. FGENE[83, 84]: O Fgene pertence a uma família de programas de predição que inclui, além do Fgene, as ferramentas Fgenes e FgeneH (ambas já analisadas em [25] e [73] respectivamente), Fgenesh e Fgenesh+. De forma genérica, esses programas realizam a predição dos genes em uma dada seqüência de DNA seguindo os passos abaixo:

   (a) Primeiramente, todos os possíveis éxons internos, assim como possíveis éxons iniciais e finais são encontrados utilizando funções lineares discriminantes que combinam várias características relacionadas aos éxons encontrados nos genes;

   (b) Encontrados os éxons, eles são ordenados de acordo com as posições (na seqüência original) dos seus nucleotídeos da extremidade 3′;

   (c) Finalmente, por meio de um algoritmo que utiliza a técnica de programação dinâmica, encontra-se uma "montagem" ótima destes éxons e constrói-se o gene a ela relacionado.

2. AAT[49]: O servidor AAT identifica genes em uma seqüência de DNA através da comparação da seqüência de entrada com cDNAs e proteínas existentes nas seguintes bases de dados:

   - *Human Gene Index*: uma base de dados contendo seqüências de cDNAs humanos;
   - dbEst: uma base de dados incluída no GENBANK contendo apenas ESTs;
   - SwissProt: uma base de dados contendo seqüências de proteínas.

   Para realizar as comparações mencionadas, o AAT faz uso de dois conjuntos de programas: um deles contem os programas **DPS** e **NAP**, utilizados para comparação da seqüência de entrada com proteínas e o outro, os programas **DDS** e **GAP2**, utilizados na comparação da seqüência de entrada com cDNAs. Cada um dos conjuntos contém um programa eficiente de busca em base de dados e um programa rigoroso para alinhamento de duas seqüências. Os programas de busca têm a finalidade de encontrar, rapidamente, regiões da seqüência de entrada similares a uma seqüência da base de dados. Encontradas essas regiões, o programa de alinhamento constrói um alinhamento ótimo para cada uma dessas regiões e a seqüencia da base de dados.

3. GENVIEW[63, 62]: O GENVIEW é uma ferramenta de predição que se utiliza de métodos estatísticos e busca por sinais para identificação de genes em longas seqüências de DNA. Assim como outras ferramentas, o GENVIEW monta os possíveis genes por meio de um algoritmo baseado em programação dinâmica, e seu principal objetivo é minimizar o número de falsos éxons encontrados na seqüência.

As ferramentas descritas foram executadas em um conjunto contendo 25 seqüências de teste, provenientes dos organismos *Homo sapiens*, *Mus musculus* e *Rattus norvegicus* e inseridas no GenBank após agosto de 1997, ano de publicação da ferramenta mais recente dentre as cinco avaliadas. Os valores de especificidade e sensibilidade da Tabela 3.3 mostram que o AAT apresenta uma maior precisão no nível de éxons que as outras duas. Lembramos que essa é a ferramenta que se utiliza de informações extrínsecas à seqüência analisada na busca por seus genes. Vale observar que a baixa qualidade apresentada pelo AAT no nível de nucleotídeos deve-se ao fato de essa ferramenta não ter predito nenhum éxon em 15 das 25 seqüências analisadas (repare o baixo número de nucleotídeos preditos como codificantes por essa ferramenta). Ao considerarmos apenas as seqüências onde algum éxon foi encontrado para o cálculo da média dos valores de AC, temos uma elevação desse número para 0.75.

Ferramentas	# de nucleotídeos	Sp_n	Sn_n	AC	# de Éxons	Sp_e	Sn_e	Av
AAT	11614	0.35	0.29	0.30	81	0.70	0.38	0.54
Fgene	22026	0.79	0.67	0.67	134	0.34	0.31	0.33
GenView	16130	0.76	0.63	0.65	84	0.31	0.18	0.25

Tabela 3.3: Resultados obtidos

Os resultados acima sugerem que as ferramentas cujas predições envolvem a busca por regiões parecidas de uma seqüência e um gene já conhecido constituem as melhores opções para o problema da identificação de genes. Além disso, com o crescente aumento do número de seqüências determinadas, essas ferramentas mostram-se ainda mais promissoras na busca pelos genes ainda não descobertos.

Considerando o fato de o problema da identificação de genes permanecer ainda em aberto, propomos neste trabalho três heurísticas distintas para ele. Todas elas baseiam-se na comparação de duas ou mais seqüências relacionadas. Os detalhes de cada uma das heurísticas e os resultados obtidos com os programas implementados encontram-se nos próximos capítulos.

Capítulo 4

Identificação de genes por comparação de DNA com cDNA

Neste capítulo, abordaremos o problema da identificação dos genes em uma dada seqüência genômica por meio de sua comparação com a seqüência de um transcrito. Sendo um pouco mais específico, por meio da comparação de uma seqüência genômica com uma seqüência de cDNA.

4.1 Breve introdução

Pelo fato de os transcritos estarem diretamente ligados aos genes das moléculas de DNA, suas seqüências podem fornecer informações valiosas a respeito da localização dos éxons em uma dada seqüência genômica. Os cDNAs, em especial, por incluírem a totalidade dos éxons de um gene, podem revelar com bastante exatidão os limites das regiões codificantes de uma dada seqüência em análise. A importância das seqüências de cDNAs torna-se mais evidente ao observarmos que um dos objetivos principais do Projeto Genoma Humano é a obtenção e posterior seqüenciamento de moléculas de RNAm para construção de uma biblioteca de cDNAs representativas de nosso genoma [77].

Neste capítulo, focaremos nossas atenções na tarefa de identificação dos genes em uma dada seqüência de DNA por meio de sua comparação com uma seqüência de cDNA relacionada. Apresentamos inicialmente a descrição formal de um problema combinatório que bem modela a relação entre essas duas seqüências. Abordamos então uma das soluções e algumas das várias heurísticas existentes para ele. Por último, descrevemos uma abordagem própria e alternativa ao problema, cujos resultados mostraram-se satisfatórios quando da sua implementação e aplicação na tarefa de localização das

regiões codificantes em várias seqüências genômicas.

4.2 O problema do alinhamento *spliced*

Em linhas gerais, estamos aqui interessados em encontrar um alinhamento entre duas seqüências que represente de forma adequada o fato de uma delas ser composta de partes distintas e não-sobrepostas da outra. Em outras palavras, estamos em busca de alinhamentos como o representado na Figura 4.1, conhecido como alinhamento *spliced*.

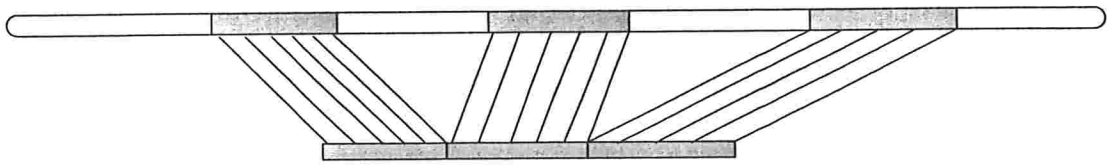

Figura 4.1: Representação de um alinhamento *spliced*

Em termos matemáticos, dadas duas seqüências s e t, com $|s| = m$, $|t| = n$ e $m > n$, estamos interessados em encontrar uma cadeia de segmentos de s[1] tal que a similaridade da concatenação desses segmentos com a seqüência t seja máxima. É dessa forma que esse problema, conhecido na literatura como **Problema do Alinhamento *spliced*** foi originalmente formulado por Gelfand *et al.* em [42]. Definindo-o formalmente, temos:

> **Problema do Alinhamento *Spliced*** (PASp): *Dadas duas seqüências s e t (com $|s| = m$, $|t| = n$ e $m > n$), construídas sobre um alfabeto Σ qualquer tal que $\{-\} \notin \Sigma$, uma função de pontuação w, e um conjunto $\mathcal{B} = \{b_1, \ldots, b_B\}$ de blocos (segmentos) de s, encontrar uma cadeia $\Gamma = (b_j, b_k, \ldots, b_l)$ formada por blocos de $\mathcal{B}$ tal que sim$(\Gamma^\bullet, t)$ sejá máxima, onde $\Gamma^\bullet = b_j \bullet b_k \bullet \ldots \bullet b_l$*[2].

Note a estreita relação entre esse problema e o problema da identificação de um gene em uma seqüência genômica quando uma seqüência de cDNA é utilizada como base de comparação. Tomando-se uma instância do problema acima onde s é uma seqüência de DNA qualquer que inclui um gene específico, t uma seqüência de cDNA relacionada e $\mathcal{B}$ um conjunto incluindo possíveis éxons em s, os blocos de uma possível solução do problema em questão possuem boas chances de corresponder aos éxons componentes do gene em s, sempre levando em conta o princípio da conservação das bases. Para o correto funcionamento da estratégia acima, precisamos garantir de alguma forma que

[1] Lembre-se que uma cadeia de segmentos de s corresponde a um conjunto de segmentos ordenados dessa seqüência.

[2] Note que a seqüência s desempenha um papel secundário nesse problema. Poderíamos utilizar perfeitamente apenas os blocos e a seqüência t como entrada do PASp.

o conjunto de blocos inclua todos os exóns componentes do gene sendo procurado. Isso pode ser feito considerando-se todos os possíveis éxons como parte do conjunto $\mathcal{B}$. Pensando na possível aplicação do PASp na tarefa de identificação de genes, Gelfand *et al.* propõem em [42] uma solução eficiente para esse problema.

4.2.1 Soluções e heurísticas para o problema do alinhamento *spliced*

Em [42], Gelfand *et al.* propõem uma solução para o problema do alinhamento *spliced* com base na estratégia de programação dinâmica. Em linhas gerais, dado o conjunto $\mathcal{B}$ de blocos de s, o que o algoritmo proposto Gelfand *et al.* faz é explorar o espaço de todas as montagens possíveis desses blocos e escolher como solução do problema aquela de maior similaridade com a seqüência t. Apesar de o número de possíveis montagens ser grande, o algoritmo localiza a melhor delas em tempo polinomial por meio da estratégia de programação dinâmica. A recorrência base desse algoritmo, denominado de *Spliced Alignment Algorithm*, é descrita a seguir com mais detalhes.

Para um melhor entendimento da recorrência proposta por Gelfand *et al.*, considere uma função de pontuação w qualquer e as seguintes definições retiradas de [42]. Dado um segmento $b_k = s[l..i..m]$ de s que inclui a $i-$ésima posição dessa seqüência, defina o $i-$prefixo de b_k, denotado por $b_k(i)$, como o segmento incluindo os $i-l+1$ símbolos de s. Ou seja, $b_k(i) = s[l..i]$. Dado um bloco $b_k = s[l..m]$, chamamos a primeira posição de b_k de $first(k) = l$ e a última posição desse bloco de $last(k) = m$. O tamanho do bloco é definido por $size(k) = m - l + 1$. Seja $\mathcal{B}(i) = \{k : last(k) < i\}$ o conjunto de todos os blocos terminando antes da posição i em s e $\Gamma = (b_1, \ldots, b_k, \ldots, b_t)$ uma cadeia tal que algum bloco b_k inclui a posição i de s. Considere $\Gamma_i^\bullet$ como sendo a seqüência $b_1 \bullet b_2 \bullet \ldots \bullet b_k(i)$ e defina Γ^k como o conjunto de todas as cadeias contendo o bloco b_k. Seja então

$$S(i,j,k) = \max_{\Gamma^k} \text{sim}(\Gamma^\bullet(i), t(j)).$$

$S(i,j,k)$ pode ser calculado por meio da seguinte recorrência:

$$S(i,j,k) = \max \begin{cases} S(i-1,j-1,k) + w(s[i],t[j]), & \text{se } i \neq first(k) \\ S(i-1,j,k) + w(s[i],-), & \text{se } i \neq first(k) \\ \max_{l \in \mathcal{B}(first(k))} S(last(l),j-1,l) + w(s[i],t[j]), & \text{se } i = first(k) \\ \max_{l \in \mathcal{B}(first(k))} S(last(l),j,l) + w(s[last(l)],-), & \text{se } i = first(k) \\ S(i,j-1,k) + w(-,t[j]). \end{cases} \quad (4.1)$$

Após o cálculo da matriz tridimensional S, a pontuação $\text{sim}(\Gamma^\bullet, t)$ do alinhamento

ótimo da melhor cadeia Γ de s e t pode ser calculada da seguinte forma:

$$\text{sim}(\Gamma^{\bullet}, t) = \max_{k} S(last(k), m, k).$$

Uma implementação ingênua da idéia acima leva a um algoritmo de complexidade de tempo igual a $O(mnB)$, onde $B = |\mathcal{B}|$. Observe no entanto que o número total de entradas da matriz S que precisa ser preenchida é $m \times \sum_{k=1}^{B} size(k) = mnc$, onde $c = \frac{1}{n}\sum_{k=1}^{B} size(k)$ corresponde à cobertura da seqüência genômica pelos blocos em $\mathcal{B}$. Isso reduz a complexidade de tempo da solução baseada na recorrência acima para $O(mnc + mB^2)$.

Com base na Recorrência 4.1, Gelfand *et al.* desenvolveram uma ferramenta para predição de genes conhecida como PROCRUSTES. Essa ferramenta recebe como entrada duas seqüências. Uma delas é a seqüência genômica cujos éxons precisam ser determinados e a outra uma seqüência de proteína que servirá como base de comparação. A saída do PROCRUSTES corresponde aos possíveis éxons da seqüência genômica cuja concatenação e posterior tradução mais se assemelha à proteína dada como entrada. Apesar de um número considerável de blocos ser importante para o correto funcionamento do PROCRUSTES, os autores dessa ferramenta não consideram todos os éxons como parte desse conjunto. Apenas os éxons limitados por potenciais sítios de aceitação e doação são considerados para processamento.

Além do PROCRUSTES, várias outras ferramentas para localização dos éxons em uma dada seqüência de DNA por meio de sua comparação com uma proteína ou um cDNA foram desenvolvidas. Dentre elas estão o SIM4, GENESEQER e EST_GENOME. Vale salientar que nem todas essas ferramentas propõem uma solução ao problema do alinhamento *spliced*, mas utilizam-se dos conceitos ligados a ele para a descoberta e posterior montagem dos éxons em uma dada seqüência de forma a reconstituir corretamente um de seus genes. Apresentamos abaixo, em linhas gerais, as idéias implementadas pelos programas que acabamos de mencionar:

- SIM4: o programa SIM4 foi desenvolvido por Zhang *et al.* e os detalhes do seu funcionamento podem ser vistos em [39]. Parecido com PROCRUSTES, o SIM4 realiza a busca por uma montagem adequada dos éxons utilizando um algoritmo também baseado na estratégia de programação dinâmica. A diferença principal entre esses dois programas está no fato de o SIM4 considerar como possíveis éxons apenas os segmentos da seqüência genômica parecidos com alguma porção da seqüência de comparação t. Esses segmentos são obtidos por meio do BLAST e, antes de ser montados, passam por um pré-processamento onde suas extremidades são aparadas ou estendidas de forma a coincidir com possíveis sítios de aceitação e doação na seqüência principal;

- EST_GENOME: desenvolvido e descrito por Mott em [65], o EST_GENOME utiliza-se de uma versão modificada do algoritmo de Smith-Waterman para solução do problema do alinhamento *spliced*. Nessa versão, além das penalidades usuais para as colunas com caracteres iguais e diferentes, têm-se também penalidades associadas a inserções e remoções fora dos íntrons ($-gap$), penalidades associadas aos íntrons iniciando-se com `GT` e terminando com `AG`($-splice$) e penalidades associadas aos íntrons sem essas características ($-$ *intron*). Para o EST_GENOME, um buraco de tamanho L custa $L \times gap$ no cDNA e $\min\{L \times gap, intron\}$ ou $\min\{L \times gap, splice\}$ na seqüência genômica;

- GENESEQER: o programa GENESEQER faz uso de uma rotina de alinhamento denominada SAHMTD (do inglês *Spliced Alignment Hiden Markov Tool for cDNA*). Descrita por Usuka *et al.* em [90], essa rotina alinha uma seqüência de DNA com uma seqüência de cDNA por meio de um HMM construído de forma a representar todos os possíveis alinhamentos dessas duas seqüências. De acordo com esse modelo, um alinhamento ótimo é definido como uma seqüência de estados $Q = q_1 q_2 \ldots q_l$, associada a uma saída S_n^m, tal que a probabilidade conjunta $P(Q, S_n^m)$ seja máxima.

4.3 Identificação de genes por alinhamento *spliced*

Propomos em [3, 6] uma abordagem alternativa para o problema do alinhamento *spliced*. A idéia principal de nossa estratégia é encontrar uma montagem de blocos satisfatória com base na busca por um caminho de custo mínimo em um grafo dirigido acíclico. A seguir descrevemos com mais detalhes os passos dessa heurística. Antes disso porém, gostaríamos de salientar que, aqui, falamos sempre em distância de edição (e não em similaridade) ao tratarmos da comparação de duas seqüências.

Dadas duas seqüências s e t, com $|s| = m$ e $|t| = n$, uma função de pontuação w, e um conjunto $\mathcal{B} = \{b_1, \ldots, b_B\}$ de segmentos de s, denominamos de **Grafo** *Spliced* $G_s = (V, A)$ um grafo dirigido acíclico com $(m+1)(n+1)$ vértices, cada um deles identificado por um par ordenado (i, j) ($0 \leqslant i \leqslant m$ e $0 \leqslant j \leqslant n$) e, no máximo, $2(m+1)(n+1) + nb$ arcos, que podem ser de três tipos diferentes:

1. Arcos horizontais $(i, j) \rightarrow (i, j+1)$, para $0 \leq i \leq n$ e $0 \leq j \leq m$: esses arcos possuem todos custo 1 e representam possíveis espaços na seqüência s.

2. Arcos verticais $(i, j) \rightarrow (i+1, j)$, para $0 \leq i \leq n$ e $0 \leq j \leq m$: esses arcos possuem todos custo 0 e representam possíveis espaços na seqüência t.

3. Arcos diagonais $(i, i') \rightarrow (j, j')$, para todo i e j de s determinantes da posição inicial e final de algum bloco $b \in \mathcal{B}$ e i' e j' de t determinantes da posicão

inicial e final do melhor alinhamento desse bloco com um sufixo de t: esses arcos possuem custo igual à distância de edição entre o bloco b iniciando-se na posição i e terminando na posição j de s e um sufixo de t. Elas representam possíveis "atalhos" a serem tomados durante a busca por um caminho mínimo em G_s.

Para uma melhor explicação das noções ligadas aos arcos diagonais, tomemos como exemplo uma seqüência s = CATGCGTCAGCTAGC, uma seqüência t = ATGCCTAG e o bloco b = CTAG que se inicia na posição 11 e termina na posição 14 de s. Note que um dos melhores alinhamentos de b com o sufixo t' =CCTAG de t que inclui os cinco últimos caracteres dessa seqüência é tal que $\bar{b}$ = -CTAG e $\bar{t'}$ = CCTAG. Quando dispostas uma em cima da outra $\begin{smallmatrix}\texttt{CCTAG}\\ \texttt{-CTAG}\end{smallmatrix}$, essas seqüências mostram um emparelhamento exato dos símbolos de b e t' a menos de um espaço inicial na seqüência $\bar{b}$ emparelhado com o primeiro 'C' de $\bar{t'}$.

Aqui estamos sempre falando de alinhamentos que não penalizam buracos na extremidade da seqüência representativa dos blocos. Dado o alinhamento $\begin{smallmatrix}\texttt{CCTAG}\\ \texttt{-CTAG}\end{smallmatrix}$, dizemos que as posições dois e cinco de t' são as posições iniciais e finais onde b melhor se alinha com essa seqüência. Note que elas correspondem às posições cinco e oito quando pensamos na seqüência t como um todo. Dadas essas posições e aquelas de início(11) e fim(14) do bloco na seqüência s, uma aresta $(11, 5) \rightarrow (14, 8)$, de peso 1, corresponde a uma das arestas diagonais do nosso grafo *spliced* G_s.

O grafo *spliced* para as seqüências s = CATGCGTCAGCTAGC e t = ATGCCTAG que acabamos de utilizar como exemplo pode ser visto na Figura 4.2. Aqui, consideramos um conjunto de blocos $\mathcal{B} = \{b_1, b_2\}$ com b_1 = ATGC e b_2 = CTAG, e uma função de pontuação w que atribui zero para os pares incluindo caracteres iguais e um para aqueles incluindo caracteres diferentes ou um espaço (distância de edição). Os alinhamentos de b com cada um dos sufixos de t podem ser vistos à direita na figura.

Dado um grafo *spliced* G_s construído sobre os alinhamentos entre os blocos em $\mathcal{B}$ e todos os sufixos de t, observe que a seqüência de arcos diagonais no melhor caminho entre o vértice $(0, 0)$ e o vértice $(m + 1, n + 1)$ representa uma cadeia $\Gamma = (b_1, \ldots, b_p)$ de blocos em $\mathcal{B}$ tal que $\Gamma^\bullet$ está próxima de t. As idéias acima encontram-se resumidas nos passos do Algoritmo 6.

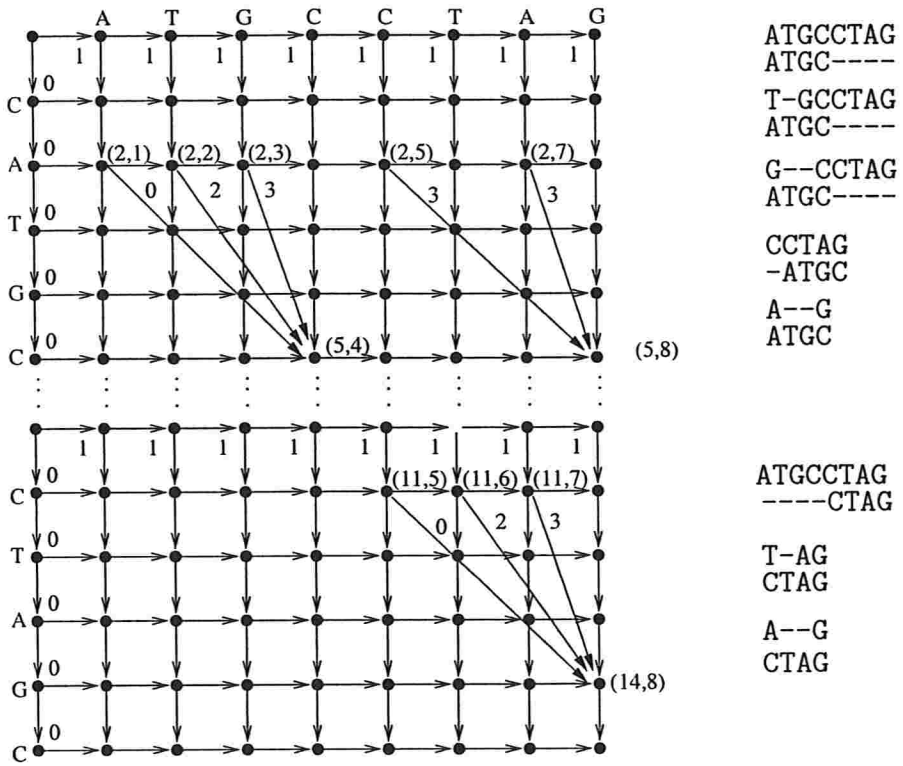

Figura 4.2: Um exemplo de grafo *spliced* para as seqüências s = CATGCGTCAGCTAGC e t = ATGCCTAG

Algoritmo 6 ENCONTRA_BLOCOS(s, t, $\mathcal{B}$, w)

Entrada: duas seqüências s e t, um conjunto de blocos (segmentos de s) $\mathcal{B}$ e uma função de pontuação w.
Saída: cadeia de blocos Γ tal que a distância entre $\Gamma^{\bullet}$ e t é pequena;

1: $G_s \leftarrow$ CONSTROI_GRAFO($|s|$, $|t|$) {sem arcos diagonais};
2: **para** $b \in \mathcal{B}$ **faça**
3: $A \leftarrow$ ALINHA_B_SUFIXOS_T(b, t, w);
4: INSERE_ARCOS_DIAGONAIS(G_s, A);
5: **fim-para**
6: $c \leftarrow$ CAMINHO_MINIMO_DAG(G_s, $(0,0)$, $(|s|+1, |t|+1)$, w);
7: Devolva arcos diagonais em c;

A respeito da complexidade de tempo do Algoritmo 6, ela é dominada pelos passos necessários para determinação dos arcos diagonais componentes de G_s e busca de um caminho mínimo entre os vértices $(0,0)$ e $(m+1, n+1)$ desse grafo. Note que o cálculo da distância de edição entre um bloco b e todos os sufixos t' de t pode ser realizada em tempo $O(|b||t|)$ por meio do algoritmo de Needleman e Wunch. No que

diz respeito às posições onde b melhor se alinha com respeito aos sufixos t' de t, elas podem ser determinadas construindo-se o alinhamento relacionado, passo esse que pode ser realizado em tempo $O(\sum_{i=1}^{|t|}(|b|+i) = |b||t| + |t|^2)$. Considerando todos os blocos em $\mathcal{B}$, temos a seguinte complexidade relacionada à determinação de todas as arestas diagonais do nosso grafo:

$$O(\sum_{i=1}^{B}|b_i||t| + \sum_{i=1}^{B}|b_i||t| + \sum_{i=1}^{B}|t|^2) = O(cmn + Bn^2).$$

Consideremos agora o problema de se encontrar um caminho mínimo entre os vértices $(0,0)$ e $(m+1, n+1)$ de G_s. Como G_s é um grafo dirigido acíclico, esse caminho pode ser encontrado em tempo $O(E_{G_s}) = O(nm + nB)$. Com isso, a complexidade de tempo total de nossa estratégia é igual a $O(cmn + Bn^2 + nm + nB)$. No caso em que a cobertura é suficientemente grande, podemos reescrever essa complexidade como $O(cmn + Bn^2)$.

4.3.1 Testes

No intuito de analisarmos o desempenho de nossa estratégia na tarefa de identificação de genes, a implementamos, utilizando a linguagem C, em um programa denominado EXON_FINDER1. Uma série de testes foram então realizados com esse programa onde uma seqüência genômica e uma seqüência de cDNA são utilizadas como entradas. Seus resultados foram então comparados com os obtidos por outros programas de predição que se utilizam também de um cDNA como seqüência de comparação. Informações adicionais sobre esse estudo comparativo podem ser encontradas em [4].

Seqüências utilizadas e ferramentas comparadas

Para a realização dos testes com nosso programa e avaliação dos resultados obtidos, utilizamos um total de 138 seqüências genômicas, escolhidas de três conjuntos de testes amplamente utilizadas na literatura. São eles: HMR195, construído por Rogic *et al.* e descrito com detalhes em [73], BG570, utilizado por Burset e Guigó em [25] e KR285, usado por Kulp *et al.* em [54] para treinamento da ferramenta GENIE. Todas as seqüências desses conjuntos incluem apenas um gene e seus éxons encontram-se verificados experimentalmente. Detalhes sobre cada uma das 138 seqüências escolhidas podem ser vistas na Tabela A.1 do Apêndice.

Nosso programa foi comparado às quatro ferramentas para predição de genes descritas na seção anterior. O único detalhe está no fato de termos utilizado uma outra versão do PROCRUSTES, denominada PRO-EST[64] que se utiliza de uma seqüência de

cDNA (e não proteína) para comparação e identificação dos genes em uma seqüência genômica.

Após a execução dos programas sobre as seqüências de teste escolhidas, as coordenadas dos éxons preditos foram comparadas com as coordenadas dos éxons reais, como anotadas no campo CDS de cada uma das seqüências no GENBANK. Dessa comparação, derivamos os números necessários para o cálculo da especificidade e sensibilidade descritos no Capítulo 2 para cada uma das seqüências. A média desses valores foi então utilizada para análise e comparação da qualidade dos resultados.

Resultados obtidos

Os primeiros testes que realizamos com nosso programa envolveu apenas 18 das 138 seqüências escolhidas. Todas elas fazem parte do conjunto utilizado por Rogic *et al.* em [73] e, na Tabela A.1, encontram-se separadas das restantes por meio de retângulos. Como seqüências de comparação, utilizamos aquelas associadas ao campo CDS de cada uma das seqüências no GENBANK. Ou seja, exatamente a seqüência de cDNA associada aos genes em cada uma das seqüências de DNA.

Como pode ser visto pelo resultado da Tabela 4.1, nosso programa foi o único que conseguiu identificar corretamente as bordas de todos os éxons das seqüências de entrada. O SIM4 também apresentou excelentes resultados. Apenas o segundo e terceiro éxon da seqüência AB010281 foram identificados erroneamente por essa ferramenta. No que diz respeito ao GENESEQER, esse programa deixou de identificar os três éxons da seqüências AB010874, o que justifica em parte a baixa qualidade de seus resultados no nível de nucleotídeos. O PRO-EST apresentou os piores resultados das cinco ferramentas avaliadas. Isso se justifica pela predição por parte dessa ferramenta de algumas bases adicionais na extremidade 3′ da maioria dos éxons reais.

Ferramentas	Sp_n	Sn_n	AC	Sp_e	Sn_e	Av
EXON_FINDER1	1.00	1.00	1.00	1.00	1.00	1.00
EST_GENOME	1.00	0.99	0.99	0.89	0.83	0.86
GENESEQER	0.94	0.94	0.94	0.92	0.83	0.88
PRO-EST	0.99	0.96	0.94	0.35	0.33	0.34
SIM4	1.00	1.00	0.99	0.96	0.93	0.95

Tabela 4.1: Qualidade das predições obtidas por cada uma das cinco ferramentas tomando-se o próprio CDS como seqüência alvo.

No intuito de determinar como os programas se comportam no caso da existência de alguns erros (substituição, inserção e remoção de bases) nas seqüências de entrada,

realizamos alguns testes adicionais, ainda com as 18 seqüências do conjunto inicial. A Tabela 4.2 mostra os valores de especificidade e sensibilidade obtidos quando um total de 3% de erros foi inseridos nas 18 seqüências genômicas tomadas como entrada pelos programas. Note que, pelo menos no nível de nucleotídeos, as ferramentas continuam apresentando bons resultados. No nível de éxons, porém, seus resultados encontram-se bem abaixo do esperado. Isso fica mais evidente quando olhamos para os resultados obtidos por nossa ferramenta. Uma análise detalhada das predições realizadas pelo EXON_FINDER1 mostra que alguns dos éxons anotados foram identificados como dois ou mais éxons menores por nosso programa. Isso se justifica pela possível mudança de distribuição dos AGs e GTs dentro das seqüências após a inserção dos erros. Tomando-se como exemplo a seqüência HSU76254, temos que o único éxon componente de seu gene inicia-se na posição 50 e termina na posição 1200 de HSU76254. No caso em que inserimos 3% de erros na seqüência, esse éxon foi predito como dois éxons distintos, que se iniciam e terminam nas posições 55..1194 e 1205..1207 de HSU76254. Essa predição fez-se possível dada a existência de um GT na posição 1195-1196 e de um AG na posição 1203-1204 da seqüência, dinucleotídeos esses não existentes na seqüência original.

Ferramentas	Sp_n	Sn_n	AC	Sp_e	Sn_e	Av
EXON_FINDER1	0.99	0.99	0.97	0.16	0.23	0.20
EST_GENOME	0.99	0.96	0.97	0.57	0.50	0.54
GENESEQER	0.94	0.94	0.93	0.51	0.46	0.49
PRO-EST	1.00	0.93	0.91	0.25	0.23	0.24
SIM4	0.99	0.99	0.98	0.44	0.43	0.44

Tabela 4.2: Qualidade das predições obtidas por cada uma das cinco ferramentas quando da inclusão de 3% de erros nas seqüências de DNA.

Apesar da importância dos resultados obtidos com o conjunto inicial de 18 seqüências, eles nos dão apenas uma vaga idéia da precisão das ferramentas avaliadas. Pensando nisso realizamos testes adicionais contando agora com um conjunto incluindo um total de 130 seqüências. Elas correspondem às seqüências da Tabela A.1 provenientes do genoma humano. Na tentativa de simular o uso das ferramentas de predição durante a tarefa de anotação de uma dada seqüência, escolhemos como alvos seqüências de RNAm que apresentam um bom nível de semelhança com as seqüências genômicas. Elas foram recuperadas por meio da execução do BLAST sobre as seqüências genômicas. O grau de semelhança, em porcentagem, entre as seqüências de RNAm e as seqüências de cDNA originais são mostradas na última coluna da Tabela A.1. Elas foram calculadas por meio de um programa de alinhamento global denominado GAP, descrito por Huang em [48].

Como pode ser visto pela Tabela 4.3, todas as ferramentas testadas apresentam quase a mesma qualidade no nível de nucleotídeos. O melhor valor de AC foi o obtido

por nosso programa, enquanto que o GENESEQER e o SIM4 apresentaram os piores resultados nesse nível. Isso pode ser atribuído à existência de algumas seqüências genômicas onde nenhum éxon foi predito o que, por sua vez, pode estar relacionado à falta de semelhanças entre as seqüências tomadas como entrada para esses programas e aquelas utilizadas para o treinamento dos modelos sobre os quais elas foram construídas.

No nível de éxons, podemos notar que existem diferenças significativas entre as ferramentas. Com relação a nosso programa, ele apresentou os piores resultados no que diz respeito à especificidade no nível de éxons. Isso pode ser explicado pelo total de éxons pequenos preditos pelo programa. Do total de 957 éxons preditos, 178 incluem apenas uma ou duas bases. Esse elevado número de pequenos falsos-positivos constitui o que é chamado na literatura de *efeito mosaico*. Essa noção fica mais clara ao observarmos o gráfico da Figura 4.3[3], que inclui os éxons anotados e os preditos por nosso programa ao ser executado sobre as seqüências AB00728, AF015224 e AB019534. Note nessa figura o elevado número de éxons minúsculos sem intersecção nenhuma com os éxons reais. Essa falha apresentada pelo EXON_FINDER1 deve-se ao fato de estarmos considerando todos os segmentos da seqüência genômica que se localizam entre um AG e um GT como possíveis éxons dessa seqüência. Dado algum sistema de filtragem utilizado pelo PRO-EST, esse programa apresentou resultados inversos. O PRO-EST deixou de predizer uma série de pequenos éxons reais das seqüências. O mesmo aconteceu com o SIM4, o que pode estar relacionado ao fato desse programa ter desconsiderado pequenos segmentos conservados durante a fase de determinação dos possíveis éxons.

Ferramentas	Sp_n	Sn_n	AC	Sp_e	Sn_e	Av
EXON-FINDER1	0.78	0.80	0.76	0.35	0.58	0.47
EST_GENOME	0.78	0.79	0.76	0.62	0.60	0.61
GENESEQER	0.74	0.74	0.71	0.64	0.61	0.63
PRO-EST	0.78	0.77	0.74	0.52	0.51	0.52
SIM4	0.78	0.76	0.73	0.51	0.52	0.52

Tabela 4.3: Qualidade das predições obtidas por cada uma das cinco ferramentas sobre o conjunto de dados principal.

4.4 Comentários gerais

Neste capítulo abordamos a tarefa de identificação dos genes em uma seqüência de DNA por meio da sua comparação com uma seqüência de cDNA. Inicialmente, apresentamos

[3]Esse gráfico, assim como aqueles dos capítulos seguintes, foram gerados via um programa denominado GFF2PS, desenvolvido por Abril e Guigó e descrito em [1].

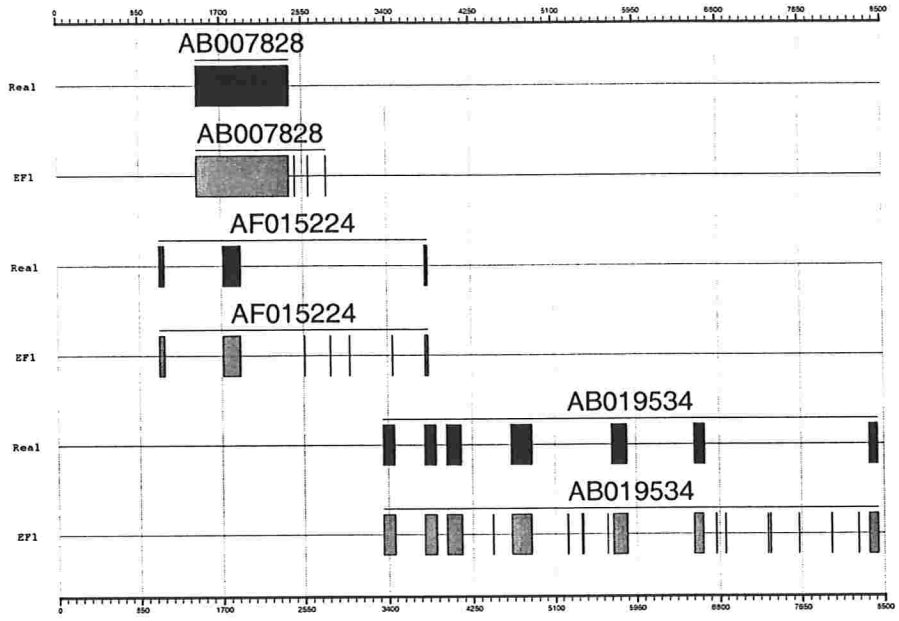

Figura 4.3: Representação gráfica de algumas das predições realizadas pelo programa EXON_FINDER1 (EF1) onde o efeito mosaico está evidente.

uma possível formalização do problema e então algumas de suas possíveis soluções e heurísticas. Na segunda parte do capítulo, descrevemos as idéias principais de uma abordagem alternativa para o PASp. Nessa abordagem, uma boa montagem dos blocos é obtida através da busca por um caminho mínimo em um grafo especial que chamamos de grafo *spliced*. Essa abordagem foi então implementada e testada para comprovação de sua eficácia no tratamento do problema da identificação de genes.

Os resultados apresentados por nosso programa comparam-se, ao menos no nível de nucleotídeos, aos obtidos por outras ferramentas que também se utilizam de um cDNA como seqüência base de comparação. No nível de éxons, porém, os resultados obtidos pelo EXON_FINDER1 mostraram-se apenas regulares, o que é devido ao fato de estarmos considerando todos os possíveis éxons como blocos de entrada. A conseqüência direta disso é o número elevado de pequenos blocos preditos como regiões codificantes, mas sem nenhuma intersecção com algum éxon anotado. Apesar dessa falha, a estratégia aqui descrita possui algumas vantagens ao pensarmos em algumas das variantes do problema do alinhamento *spliced*. A busca por várias soluções ótimas, ou por soluções subótimas para o PASp, capazes de revelar possíveis *splicings* alternativos do gene procurado, podem ser tratadas diretamente por nossa estratégia. Para isso, basta realizarmos a busca por variantes do caminho ótimo no grafo *spliced*. Tais possibilidades surgem como possíveis trabalhos futuros envolvendo as idéias aqui discutidas.

Capítulo 5

Identificação de genes por comparação DNA/DNA

Neste capítulo abordaremos o problema da identificação de genes por meio da comparação de duas seqüências genômicas evolutivamente relacionadas. Ou seja, duas seqüências contendo um ou mais genes homólogos.

5.1 Breve introdução

Mais uma vez lembramos que as regiões funcionais de um DNA possuem a tendência de se conservar ao longo da evolução da molécula. Esse fato, somado à funcionalidade já conhecida das regiões codificantes, torna a comparação de duas seqüências genômicas evolutivamente relacionadas extremamente útil na tarefa de identificação de seus genes. Esse tipo de comparação passou a ser comum nos últimos tempos dado o grande número de genomas disponíveis e em fase de seqüenciamento[1]. Isso se reflete no expressivo número de ferramentas para predição de genes desenvolvidas nos últimos anos que se utilizam da comparação de duas seqüências genômicas para realização de suas tarefas.

Neste capítulo abordamos o problema da identificação de genes por comparação de duas seqüências genômicas. Apresentamos inicialmente uma formalização matemática para o problema e algumas de suas heurísticas. Detalhamos então o funcionamento de um programa desenvolvido por nós que se propõe a comparar duas seqüências genômicas e determinar a localização de seus genes. Finalmente, descrevemos alguns dos resultados obtidos e concluímos o capítulo com alguns comentários gerais.

[1] Informações sobre os projetos de seqüenciamento já concluídos ou em andamento podem ser encontradas na base de dados conhecida como GOLD, descrita em [16]

5.2 O problema do alinhamento sintênico

Um pouco diferente do alinhamento *spliced* descrito no capítulo anterior, aqui estamos interessados em um alinhamento que represente de forma adequada o fato de as duas seqüências comparadas incluirem regiões semelhantes separadas por regiões não tão parecidas. Em outras palavras, estamos interessados em alinhamentos como o mostrado na Figura 5.1, onde os retângulos representam as regiões bem conservadas das duas seqüências.

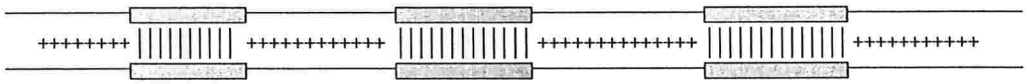

Figura 5.1: Representação de um alinhamento sintênico

Esse tipo de alinhamento é conhecido como **alinhamento sintênico**, e o problema de encontrá-lo pode ser formalmente definido como:

> **Problema do Alinhamento Sintênico** (PASi): *Dadas duas seqüências s e t, construídas sobre um alfabeto Σ qualquer tal que $\{-\} \notin \Sigma$, e uma função de pontuação w, encontrar uma cadeia $S = \{s'_1, s'_2, \ldots, s'_n\}$ de segmentos de s, e uma cadeia $T = \{t'_1, t'_2, \ldots, t'_m\}$ de segmentos de t tal que $sim_w(S^\bullet, T^\bullet)$ seja máxima, onde $S^\bullet = \{s'_1 \bullet s'_2 \bullet \ldots \bullet s'_n\}$ e $T^\bullet = \{t'_1 \bullet t'_2, \bullet \ldots \bullet t'_n\}$.*

Note a estreita relação entre o PASi e o problema da identificação de genes por meio da comparação de duas seqüências genômicas. Nesse caso, também estamos interessados na busca por regiões semelhantes das duas seqüências, os éxons, normalmente separadas por regiões não tão parecidas, os íntrons. Tomando-se uma instância do problema do alinhamento sintênico onde s e t são duas seqüências genômicas com um gene em comum, os segmentos de uma possível solução desse problema são as regiões parecidas nas duas seqüências. De acordo com o princípio da conservação de bases, essas regiões possuem boas chances de incluir os éxons reais de s e t.

5.2.1 Algumas heurísticas para o problema do alinhamento sintênico

Em [14], Batzoglou *et al.* descrevem uma heurística para o problema do alinhamento sintênico implementada em um programa denominado GLASS. Esse programa realiza os seguintes passos na busca por um alinhamento de duas seqüências s e t que bem reflete o fato de elas incluírem regiões descontínuas com alto grau de semelhança:

1. Primeiramente, todos os segmentos altamente conservados de tamanho k (k-mers) nas duas seqüências são encontrados;

2. Cada k-mer é então tratado como um caractere distinto e as seqüências de entrada convertidas em seqüências compostas única e exclusivamente por caracteres desse tipo;

3. As seqüências modificadas são então alinhadas de forma tradicional, mas por meio de um esquema de pontuação onde apenas o valor da similaridade das bordas dos k-mers - e não os k-mers propriamente ditos - são levadas em consideração. k-mers emparelhados recebem uma pontuação igual à pontuação do alinhamento de suas regiões periféricas;

4. Pares de k-mers consistentes e com pontuações acima de um dado valor V são fixados e as regiões das seqüências localizadas entre eles são alinhadas repetindo-se os passos acima com uma valor menor de k;

5. Realizado o último alinhamento, todos os pares são estendidos alinhando-se localmente os extremos de suas seqüências;

6. As regiões não processadas nos passos anteriores são então alinhadas por meio do algoritmo de Smith-Waterman.

Uma vez alinhadas, as seqüências s e t passam por um pós-processamento onde as regiões repetidas e aquelas com um número significativo de buracos e bases diferentes são identificadas. As regiões remanescentes constituem os segmentos procurados.

Utilizando a heurística acima, Batozglou *et al.* desenvolveram uma ferramenta para identificação de genes denominada ROSETTA. Pioneira na tarefa de identificação de genes por comparação de duas seqüências genômicas, essa ferramenta utiliza-se da noção de *parse* para a montagem adequada dos éxons de forma a constituir os genes procurados. Um *parse* de uma certa região genômica corresponde a uma seqüência de triplas $[(a_1, b_1, t_1) \ldots (a_n, b_n, t_n)]$ onde $e_i = (a_i, b_i, t_i)$ denota um éxon de tipo t_i (inicial, interno, final) começando na posição a_i e terminando na posição b_i da seqüência. Um *parse* é dito válido se as seguintes propriedades são observadas para cada par de éxons consecutivos: (i) se e_i é um éxon terminal, então e_{i+1} é um éxon inicial, e vice-versa; (ii) se e_i é um éxon interno, então $b_{i-1} < a_i$ e $b_i < a_{i+1}$.

Por meio do alinhamento construído pelo GLASS, podemos associar a cada *parse* válido em s um *parse* válido em t. Dessa associação, uma pontuação para cada *parse* válido pode ser calculada de acordo com a soma da pontuação das regiões conservadas, além de outras características desses segmentos (tamanho, potencialidade de seus sítios de aceitação e doação, etc). Por meio da estratégia de programação dinâmica, o que o ROSETTA faz é encontrar um *parse* válido nas duas seqüências de pontuação máxima.

Note que a estratégia utilizada pelo ROSETTA na busca pelos genes das duas seqüências sendo comparadas inclui dois passos distintos. No primeiro deles, as duas seqüências são alinhadas utilizando o GLASS e esse alinhamento é então processado na busca por um *parse* válido de peso máximo. Uma abordagem um pouco diferente dessa consiste em determinar os genes das duas seqüências dadas como entrada já durante a fase de alinhamento. Essa é, de forma bem simplificada, a abordagem utilizada pelo UTOPIA e PROGEN, dois outros programas de predição baseados na comparação de duas seqüências genômicas. Detalhando um pouco mais a estratégia utilizada por essas duas ferramentas, ela consiste basicamente no preenchimento de uma matriz de programação dinâmica padrão (ou seja, cuja células armazenam o valor do melhor alinhamento de prefixos das duas seqüências) tendo sempre em mente a possibilidade de se estar entrando em um éxon de uma ou ambas as seqüências sendo comparadas. Isso determina a possibilidade adicional de se estender o alinhamento construído tomando-se o máximo relacionado à melhor montagem terminando em um sítio de doação (final de um íntron) nas seqüências. A recorrência mostrada abaixo corresponde à utilizada pelo PROGEN na busca do melhor alinhamento sintênico de duas seqüências s e t. No que segue, a matriz I armazena a pontuação do melhor alinhamento terminando em um íntron em uma das seqüências (I_s e I_t) ou em ambas (I_{st}), e é atualizada a cada ocorrência de um sítio de doação nas seqüências.

$$M[i][j] = \max \begin{cases} M[i-1][j-1] + w(s[i], t[j]); \\ M[i][j-1] + w(-, t[j]); \\ M[i-1][j] + w(s[i], -); \\ \text{se } s[i-2]s[i-1] \text{ é um sítio de aceitação} \\ I_s[i][j-1] - \lambda; \\ I_s[i][j] + w(s[i], -) - \lambda; \\ \text{se } t[i-2]t[i-1] \text{ é um sítio de aceitação} \\ I_t[i-1][j] - \lambda; \\ I_t[i][j] + w(-, t[j]) - \lambda, ; \\ \text{se } s[i-2]s[i-1] \text{ e } t[j-2]t[j-1] \text{ são sítios de aceitação} \\ I_{st}(i,j) - 2 * \lambda. \end{cases} \tag{5.1}$$

A recorrência utilizada pelo UTOPIA é muito parecida com a utilizada pelo PROGEN. Na realidade, a principal diferença entre esses dois programas está no fato de o UTOPIA utilizar um alfabeto diferente daquele utilizado pelo PROGEN durante a tarefa de comparação das duas seqüências de entrada. Enquanto o PROGEN faz isso alinhando triplas de bases, o UTOPIA utiliza-se de um alfabeto mais genérico, que permite o alinhamento de uma, duas ou três bases. Isso acarreta certas modificações na função de pontuação, que torna esse programa mais lento. Em contrapartida, possíveis erros de

seqüenciamento, como os que incluem a remoção ou inserção de uma base (*frameshifts*), podem ser formalmente tratados.

Em [50], Huang e Chao desenvolveram um algoritmo para a determinação de um alinhamento de duas seqüências conhecido como **alinhamento generalizado**. Parecido com o alinhamento sintênico, o alinhamento generalizado também procura evidenciar a relação entre regiões parecidas de duas seqüências. Para o cálculo eficiente de um alinhamento generalizado entre duas seqüências, os autores utilizam-se de um novo modelo para o problema em questão onde, além das colunas usuais com dois caracteres emparelhados ou um caractere e um espaço, o alinhamento procurado pode incluir também uma série de colunas que, em conjunto, constituem o que Huang e Chao chamam de blocos de diferença. Um bloco de diferença consiste em símbolos de uma ou das duas seqüências que, se alinhados de forma global, apresentariam um baixo valor de similaridade. Em outras palavras, esse alinhamento incluiria um grande número de colunas com espaços e caracteres diferentes emparelhados. Para a determinação da similaridade relacionada a um alinhamento generalizado ótimo, precisamos então nos permitir a escolha de uma coluna constituinte de um possível bloco de diferença, além das colunas usuais, na construção desse alinhamento.

Vale notar que, da forma como proposto por Huang e Chao, apenas a entrada em um bloco de diferença (e não a entrada e a extensão) é penalizada durante a construção do alinhamento generalizado ótimo. Assim, durante o cálculo da similaridade, é preciso saber se a coluna relacionada ao máximo escolhido corresponde à primeira coluna de um bloco de diferença ou não. Como no caso da busca pelo melhor alinhamento utilizando funções afins de penalização, isso pode ser feito armazenando os valores dos melhores alinhamentos terminando em um bloco de diferença. É armazenando esses valores em uma matriz adicional que Huang e Chao desenvolvem uma solução para o problema de busca por um alinhamento considerando-se a possível existência de regiões pouco similares dentro delas. Esse alinhamento é dito generalizado por permitir o uso de um tipo de coluna a mais na construção de um alinhamento de duas seqüências. Em outras palavras, temos ao final do processo um alinhamento mais genérico que aquele podendo ser obtido apenas com o uso das colunas convencionais.

As recorrências utilizadas pelo algoritmo proposto por Huang e Chao são mostradas a seguir. As matrizes S, D e I são as mesmas utilizadas no cálculo do melhor alinhamento com funções afins de penalização, enquanto que a matriz H corresponde à matriz adicional onde os valores dos melhores alinhamentos terminando em um bloco de diferença encontram-se guardados. Nessas recorrências, a função de pontuação w, de forma usual, associa um valor real para cada par de símbolos do alinhamento, enquanto que h e g correspondem às penalidades para abertura e extensão de um buraco. A única diferença está no uso de uma penalidade adicional d, para entrada em um bloco de diferença.

$$H[i,j] = \max \begin{cases} \text{extensão de um bloco de diferença:} \\ H[i-1][j], H[i][j-1], \\ \text{início de um bloco de diferença:} \\ S[i-1][j]-d, D[i-1][j]-d, I[i-1][j]-d, \\ S[i][j-1]-d, D[i][j-1]-d, I[i][j-1]-d, \end{cases}$$

$$S[i][j] = w(s[i],t[j]) + \max\{\ S[i-1][j-1], I[i-1][j-1], D[i-1][j-1], H[i-1][j-1]$$

$$I[i,j] = \max \begin{cases} S[i][j-1]-(h+g), D[i][j-1]-g, \\ I[i][j-1]-(h+g), H[i][j-1]-(h+g) \end{cases}$$

$$D[i,j] = \max \begin{cases} S[i-1][j]-(h+g), D[i-1][j]-g, \\ I[i-1][j]-(h+g), H[i][j-1]-(h+g) \end{cases}$$

Outra forma de tratar o problema do alinhamento sintênico é por meio do uso de duas funções distintas de pontuação durante a construção do alinhamento procurado. Essa idéia, proposta originalmente por Almeida *et al.* em [51], parte do princípio que alinhamentos envolvendo regiões não parecidas em duas seqüências incluem um número maior de substituições, inserções e remoções do que aqueles envolvendo regiões parecidas. Daí, torna-se mais adequado penalizar diferentemente esses eventos de acordo com as partes das seqüências onde eles ocorrem. Um pouco diferente da estratégia anterior, onde um novo tipo de coluna é proposto, aqui continuamos com as três colunas usuais. A principal diferença em relação às idéias tradicionais está no fato de essas colunas poderem contribuir com valores distintos à pontuação total do alinhamento de acordo com as regiões das seqüências onde elas se encontram. Daí, temos seis escolhas distintas na determinação do máximo durante a busca por um alinhamento ótimo: quatro delas relacionadas a colunas com espaços e duas, a colunas com caracteres emparelhados. Mais uma vez, os valores intermediários de similaridade, necessários para o cálculo correto da similaridade final, podem ser armazenados em matrizes bidimensionais. De acordo com essa estratégia, as recorrências 5.2, 5.3, 5.4, 5.5, 5.6 e 5.7 podem ser utilizadas para determinação de um alinhamento sintênico de duas seqüências.

$$S_e[i][j] = w_e(s[i],t[j]) + \max \begin{cases} S_e[i-1][j-1], D_e[i-1][j-1], I_e[i-1][j-1] \\ S_i[i-1][j-1], D_i[i-1][j-1], I_i[i-1][j-1]. \end{cases} \quad (5.2)$$

$$S_i[i][j] = w_i(s[i], t[j]) + \max \begin{cases} S_e[i-1][j-1], D_e[i-1][j-1], I_e[i-1][j-1] \\ S_i[i-1][j-1], D_i[i-1][j-1], I_i[i-1][j-1]. \end{cases} \quad (5.3)$$

$$I_e[i][j] = \max \begin{cases} S_e[i][j-1] - (h_e + g_e), S_i[i][j-1] - (h_e + g_e) \\ D_e[i][j-1] - (h_e + g_e), D_i[i][j-1] - (h_e + g_e) \\ I_e[i][j-1] - g_e, I_i[i][j-1] - (h_e + g_e). \end{cases} \quad (5.4)$$

$$I_i[i][j] = \max \begin{cases} S_e[i][j-1] - (h_i + g_i), S_i[i][j-1] - (h_i + g_i) \\ D_e[i][j-1] - (h_i + g_i), D_i[i][j-1] - (h_i + g_i) \\ I_e[i][j-1] - (h_i + g_i), I_i[i][j-1] - g_i, \end{cases} \quad (5.5)$$

$$D_e[i][j] = \max \begin{cases} S_e[i-1][j] - (h_e + g_e), S_i[i-1][j] - (h_e + g_e) \\ D_e[i-1][j] - g_e, D_i[i-1][j] - (h_e + g_e) \\ I_e[i-1][j] - (h_e + g_e), I_i[i-1][j] - (h_e + g_e). \end{cases} \quad (5.6)$$

$$D_i[i][j] = \max \begin{cases} S_e[i-1][j] - (k + h_i + g_i), S_i[i-1][j] - (h_i + g_i) \\ D_e[i-1][j] - (h_i + g_i), D_i[i-1][j] - g_i \\ I_e[i-1][j] - (h_i + g_i), I_i[i-1][j] - (h_i + g_i). \end{cases} \quad (5.7)$$

Nas recorrências acima, $w_{e/i}$ são duas funções que atribuem valores distintos para emparelhamentos que ocorrem dentro de regiões parecidas/não parecidas das duas seqüências, enquanto que $h_{e/i}$ e $g_{e/i}$ são as penalidades para abertura e extensão de buracos nesses dois tipos de regiões.

5.3 Identificação de genes por alinhamento sintênico

Com base nas idéias acima, e tendo sempre em mente que os éxons possuem um grau de conservação maior que aquele apresentado pelos íntrons, desenvolvemos outro programa para predição de genes que, dessa vez, utiliza-se de duas seqüências genômicas na realização dessa tarefa. Os detalhes dessa estratégia encontram-se também descritas em [5]. A idéia geral da heurística implementada é comparar duas seqüências s e t considerando três blocos distintos para composição do alinhamento sendo construído:

1. bloco referente às regiões exônicas das seqüências;

2. bloco referente às regiões intrônicas das seqüências;

3. bloco referente às regiões intergênicas.

Para o tratamento dos dois primeiros casos fazemos uso de dois conjuntos distintos de matrizes de programação dinâmica. O primeiro deles inclui as matrizes que armazenam o valor da similaridade entre prefixos das seqüências cujo alinhamento termina com um emparelhamento (S_e), inserção (I_e) ou remoção (D_e) dentro de um éxon de s e t. O segundo inclui as matrizes utilizadas para o cálculo da similaridade entre partes das seqüências cujo alinhamento termina com um emparelhamento (S_i), inserção (I_i) ou remoção (D_i) dentro de um íntron de s e t. A diferença principal entre esses dois conjuntos de matrizes está na função de pontuação utilizada e penalidades associadas à abertura e extensão de buraco usadas em cada caso. Como no trabalho proposto por Almeida *et al.*, aqui usamos duas funções de pontuações distintas, sendo uma delas (aquelas associadas às matrizes S_e, D_e e I_e) bem mais severa que a outra (associadas às matrizes S_i, D_i e I_i) no tratamento dos buracos e caracteres diferentes emparelhados. Com relação aos blocos associados às regiões intergênicas, uma única matriz H é utilizada para armazenamento do valor da similaridade dos alinhamentos terminando nessas regiões.

Se observarmos a estrutura dos genes eucarióticos mostrada na Figura 2.3, podemos notar que eles sempre começam e terminam com um éxon[2]. Isso implica que o valor de um alinhamento ótimo terminando em uma região intergênica só pode ser calculado adicionando-se a pontuação da coluna em questão a um ótimo relacionado a uma região intergênica ou exônica. Da mesma forma, o valor de um alinhamento ótimo terminando em uma região intrônica pode ser calculado apenas tendo como opção um ótimo terminando em uma região exônica. Nunca por meio da extensão de uma região intergênica. Também de acordo com o modelo mostrado na Figura 2.3, um alinhamento ótimo terminando em uma região exônica/intrônica só pode ser calculado utilizando um máximo ligado a um alinhamento finalizado em uma região intrônica/exônica quando da existência de um sítio de aceitação/doação nas duas seqüências [3]. Mais do que isso, quando da existência de um sítio de *splicing* conservado nas duas seqüências. Essa restrição um pouco mais forte deve-se ao fato desses sinais geralmente apresentarem um elevado grau de conservação, dada a importância que possuem durante o processo de síntese de proteínas pelas células. As restrições descritas até aqui para o preenchimento das matrizes $S_{e/i}$, $D_{e/i}$, $I_{e/i}$ e H estão representadas esquematicamente na Figura 5.2.

[2]Em alguns casos, as regiões 3′ e 5′ encontram-se interrompidas por regiões não-codificantes. Por constituírem raras exceções, resolvemos não tratar desses casos neste trabalho.

[3]Note que essa restrição implica a suposição de que os dois genes possuem a mesma estrutura. Em outras palavras, estamos utilizando uma versão um pouco mais restrita do problema do alinhamento sintênico onde o número de segmentos procurados precisa ser igual.

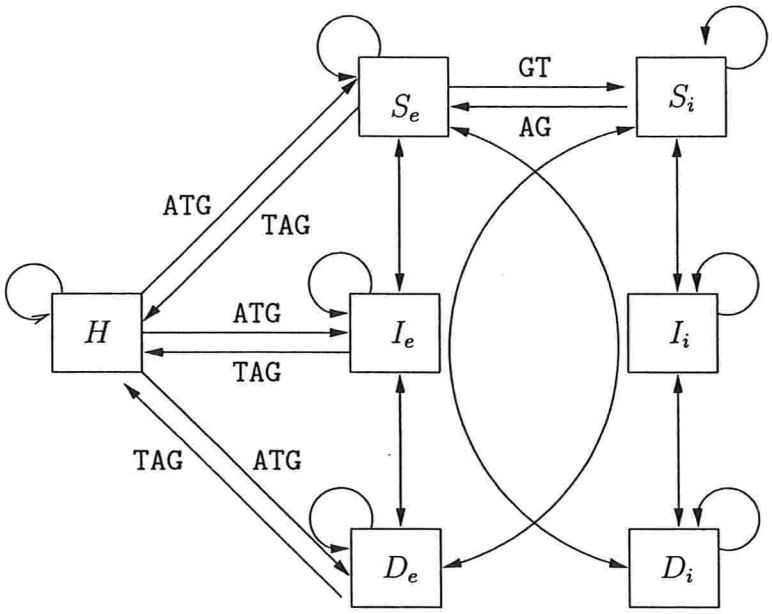

Figura 5.2: Representação esquemática das restrições para preenchimento das matrizes de programação dinâmica

A respeito dos sítios de *splicing*, a idéia mais simples é considerar todos os dinucleotídeos do tipo `AG/GT` como possíveis sítios de aceitação e doação, respectivamente. O problema com essa abordagem está no elevado número de dinucleotídeos desse tipo que não possuem a função específica de determinar o início e final dos éxons componentes das seqüências genômicas em consideração. Em outras palavras, ao considerarmos qualquer `AG/GT` nas seqüências durante o preenchimento das matrizes de programação dinâmica, corremos o risco de encontrar vários falsos-positivos. Ou seja, várias regiões que, apesar de conservadas e limitadas por possíveis sítios de *splicing*, não correspondem a nenhum éxon já anotado das seqüências de entrada. Com isso, nos decidimos por considerar apenas aqueles dinucleotídeos com altas chances de ser verdadeiros. Para isso, fizemos uso do *log-likelihood ratio* associado a cada ocorrência de `AG/GT` nas seqüências, calculado em uma fase de pré-processamento. A determinação dessas taxas é realizada com bases nas matrizes de peso descritas por Salzberg em [75]. Como descrito no Capítulo 3, as matrizes de peso incluem a probabilidade de se encontrar uma determinada base μ na posição i dada a base ν na posição $i-1$ de um seqüência representativa de um sinal específico. No trabalho desenvolvido por Salzberg, três matrizes de peso foram construídas: duas relacionadas aos sítios de aceitação e doação e uma terceira representativa das regiões incluindo verdadeiros códons de início. Os valores de todas essas matrizes foram calculados com base em um total de 2079 sítios de *splicing* e 562 códons de início, provenientes das seqüências de 570 genes distintos. Dadas as matrizes de peso em [75], o que fizemos foi, em uma fase de pré-processamento, calcular o valor

do *log-likelihood ratio*[4] para cada um dos AG e GT que aparecem nas duas seqüências de entrada[5]. Calculados esses valores, eles são utilizados de duas formas distintas em nossa abordagem. A passagem de uma matriz associada aos blocos intrônicos para outra associada aos blocos exônicos (e vice-versa) é realizada apenas quando da existência de dinucleotídeos AG/GT nas duas seqüências com valores elevados de *log-likelihood ratio*. Além disso, uma vez presentes esses dinucleotídeos, o valor da taxa P associado a cada um deles é utilizado como parte da pontuação do alinhamento sendo construído.

As idéias acima implicam as seguintes recorrências (5.8, 5.9, 5.10, 5.11, 5.12, 5.13, 5.14), utilizadas pelo Algoritmo 7 na busca pelos genes em duas seqüências.

$$H[i,j] = \max \begin{cases} H[i-1][j], H[i][j-1], \\ \text{Se existe um códon de parada em } s \text{ e } t \\ S_e[i-1][j] - d, S_e[i][j-1] - d, \\ D_e[i-1][j] - d, D_e[i][j-1] - d, \\ I_e[i-1][j] - d, I_e[i][j-1] - d. \end{cases} \tag{5.8}$$

$$S_e[i,j] = w_e + \max \begin{cases} S_e[i-1][j-1], \\ D_e[i-1][j-1], \\ I_e[i-1][j-1], \\ \text{Se existe um códon de início ATG em } s \text{ e } t \\ \text{(P(s[i..i+2]) } > 1.0 \text{ e P(t[j..j+2]) } > 1.0\text{):} \\ H[i-i][j-1], \\ \text{Se existe um sítio de aceitação AG em } s \text{ e } t \\ \text{(P(s[i-2..i-1]) } > 1.0 \text{ e P(t[j-2..j-1]) } > 1.0\text{):} \\ S_i[i-1][j-1]. \end{cases} \tag{5.9}$$

$$I_e[i,j] = \max \begin{cases} S_e[i][j-1] - (h+g), \\ D_e[i][j-1] - (h+g), \\ I_e[i][j-1] - g, \\ \text{Se existe um códon de início ATG em } s \text{ e } t \\ \text{(P(s[i+1..i+3]) } > 1.0 \text{ e P(t[j..j+2]) } > 1.0\text{):} \\ H[i][j-1], \\ \text{Se existe um sítio de aceitação AG em } s \text{ e } t \\ \text{(P(s[i-1..i]) } > 1.0 \text{ e P(t[j-2..j-1]) } > 1.0\text{):} \\ S_i[i][j-1] - (h+g). \end{cases} \tag{5.10}$$

[4]denotado aqui por P, como a probabilidade mencionada no Capítulo 3

[5]Lembre-se que, dada uma matriz de peso m, o *log-likelihood ratio* de um segmento qualquer de uma seqüência genômica s de tamanho n pode ser calculado por meio da fórmula $\sum_{i=1}^{n} m[s[i]][i]$.

$$D_e[i,j] = \max \begin{cases} S_e[i-1][j] - (h+g), \\ D_e[i-1][j] - g, \\ I_e[i-1][j] - (h+g), \\ \text{Se existe um códon de início } \texttt{ATG} \text{ em } s \text{ e } t \\ (\text{P(s[i..i+2])} > 1.0 \text{ e P(t[j+1..j+3])} > 1.0): \\ H[i-1][j], \\ \text{Se existe um sítio de aceitação } \texttt{AG} \text{ em } s \text{ e } t \\ (\text{P(s[i-2..i-1])} > 1.0 \text{ e P(t[j-1..j])} > 1.0): \\ S_i[i-1][j] - (h+g). \end{cases} \tag{5.11}$$

$$S_i[i,j] = \max \begin{cases} S_i[i-1][j-1], \\ D_i[i-1][j-1], \\ I_i[i-1][j-1]. \\ \text{Se existe um sítio de doação } \texttt{GT} \text{ em } s \text{ e } t \\ (\text{P(s[i-2..i-1])} > 1.0 \text{ e P(t[j-2..j-1])} > 1.0): \\ S_e[i-1][j-1] - k. \end{cases} \tag{5.12}$$

$$I_i[i,j] = \max \begin{cases} S_i[i][j-1] - (h+g), \\ D_i[i][j-1] - (h+g), \\ I_i[i][j-1] - g. \\ \text{Se existe um sítio de doação } \texttt{GT} \text{ em } s \text{ e } t \\ (\text{P(s[i-1..i])} > 1.0 \text{ e P(t[j-2..j-1])} > 1.0): \\ S_e[i][j-1] - (k+h+g). \end{cases} \tag{5.13}$$

$$D_i[i,j] = \max \begin{cases} S_i[i-1][j] - (h+g), \\ D_i[i-1][j] - g, \\ I_i[i-1][j] - (h+g). \\ \text{Se existe um sítio de doação } \texttt{GT} \text{ em } s \text{ e } t \\ (\text{P(s[i-2..i-1])} > 1.0 \text{ e P(t[j-1..j])} > 1.0): \\ S_e[i-1][j] - (k+h+g). \end{cases} \tag{5.14}$$

Nas recorrências acima d e k correspondem a penalidades adicionais para entrada nos íntrons e regiões-intergênicas. Essas penalidades fazem-se necessárias para diminuir o efeito da quebra de éxons decorrente de entradas inadequadas nessas regiões.

Algoritmo 7 ENCONTRA_EXONS(s, t, w)

Entrada: duas seqüências genômicas s e t relacionadas evolutivamente e uma função de pontuação w.

Saída: éxons componentes dos genes nas duas seqüências;

1: PREENCHE_MATRIZES(s, t, w, S, D, I, H) {Preenche as matrizes por meio das recorrências 5.8, 5.9, 5.10, 5.11, 5.12, 5.13, 5.14}
2: $A \leftarrow$ CONSTROI_ALINHAMENTO(s, t, S, D, I, H)
3: DEVOLVE_EXONS(A); {regiões conservadas no alinhamento}

Uma vez realizado o alinhamento, os éxons procurados correspondem às regiões maximais das seqüências alinhadas por meio das matrizes S_e, D_e e I_e.

Note que o algoritmo acima possui complexidade de tempo e espaço igual a $O(mn)$. Ele realiza apenas o preenchimento de um número constante de matrizes de dimensão $m \times n$ para a determinação da similaridade entre duas seqüências e um percurso inverso nessa matriz para construção do alinhamento e determinação dos éxons propriamente ditos. Como já vimos, esse último passo pode ser realizado em tempo $O(m + n)$.

5.3.1 Testes

No intuito de avaliarmos nossa estratégia, a implementamos, utilizando a linguagem C, em um programa denominado EXON_FINDER2 e então o testamos por meio de um conjunto incluindo vários pares distintos de seqüências genômicas. Aqui também comparamos os resultados obtidos por nosso programa com os de outros programas de predição que comparam duas seqüências genômicas na busca pelos seus genes.

Seqüências utilizadas e ferramentas comparadas

Para a realização dos testes com nosso programa, utilizamos um total de 50 pares de seqüências. Cada um desses pares, que incluem seqüências do *Homo sapiens*, *Rattus norvegicus* e *Mus musculus*, foram obtidos do *benchmark* utilizado por Jareborg *et al.* em [69] para teste de uma ferramenta de predição denominada SGP-2. Informações detalhadas sobre os pares utilizados podem ser vistas na Tabela A.2 localizada no Apêndice. Novamente, a qualidade da predição foi acessada por meio das medidas de especificidade e sensibilidade descritas no Capítulo 2.

Comparamos nosso programa com o UTOPIA e o PROGEN, descritas na seção anterior. Decidimos pela escolha dessas ferramentas por serem elas as únicas que não se utilizam de nenhuma heurística para comparação das seqüências dadas como entrada.

Resultados obtidos

As médias dos valores de especificidade e sensibilidade obtidas após a execução dos programas sobre os 50 pares utilizados para teste podem ser vistas na Tabela 5.1.

Ferramenta	Sp_n	Sn_n	AC	Sp_e	Sn_e	Av
EXON_FINDER2	0.87	0.94	0.87	0.43	0.45	0.44
UTOPIA	0.86	0.98	0.89	0.38	0.52	0.45
PROGEN	0.95	0.98	0.95	0.69	0.78	0.74

Tabela 5.1: Qualidade das predições obtidas por cada ferramenta sobre os 50 pares de teste.

Como pode ser visto pelos números apresentados na Tabela 5.1, nosso programa apresenta bons resultados no nível de nucleotídeos, comparáveis aos apresentados pelo UTOPIA. No nível de éxons, os resultados apresentados pelo EXON_FINDER2, assim como os obtidos pelo UTOPIA, mostraram-se apenas regulares. Isso se deve a uma série de fatores. O principal deles está associado à existência de regiões conservadas nas duas seqüências fora dos limites dos seus genes, o que se torna mais evidente nas fronteiras dos íntrons e regiões intergênicas com os éxons. Um exemplo dessa falha pode ser visto na Figura 5.3, que mostra parte do alinhamento sintênico das seqüências HSHIS4 e MMHIST4 (par H4) construído por nosso programa.

Falhas como essas determinam a baixa qualidade de nosso programa (assim como a do UTOPIA) no nível de éxons, que acabam sendo preditos com algumas bases adicionais em suas fronteiras. Isso fica mais claro ao observarmos a Figura 5.4, que inclui a representação gráfica de alguns éxons preditos pelo EXON_FINDER2 (inclusive o do exemplo acima) que extrapolam os limites dos éxons anotados. No caso do PROGEN, essas falhas não aparecem de forma tão freqüente, o que pode ser devido a uma melhor discriminação por parte desse programa dos verdadeiros sítios de aceitação e doação.

5.4 Uma análise adicional do Utopia e Progen

Outro trabalho que realizamos em torno do problema da identificação de genes por comparação de duas seqüências genômicas consistiu na análise do comportamento do UTOPIA e PROGEN em um cenário mais próximo ao da realidade. Em outras palavras, quando utilizamos pares de seqüências que, além de pertencer a uma variedade de espécies distintas, incluem um ou mais genes de estruturas bem diferentes (números diferentes de éxons) e com possíveis erros de seqüenciamento. Mais detalhes sobre esse trabalho podem ser encontrados em [88].

```
      PIP:477
aaatgacgaaATG T CGA    GAGGGCGGGGACAATTGAGAACGCTTCCCGCCGGCGCGCTTTCGGTT
**********EEE-E-EEE---- || ||  ||| |||| |||| | ||||||||| |||||||||| |||
         ATGATGCAACATCCAGAGCCCGGATAATTTAGAAAGGTTCCCGCCCGCGCGCTTTCAGTT
      PIP:115

TTCAATCTGGTCCGATACTCTTGTATATCAGGGGAAGACGGTGCTCGCCTTG  ACAGAAGCT GTCTAT
|||||||||||||||| ||||  ||||| || || |  |-  |||-|  ||-- ||  ||||-|| | |
TTCAATCTGGTCCGATCCTCTCATATATTAGTGGCACTC CACCTC CAATGCCTCACCAGCTGGTGTTT

            PIV:613
CGGGCTCCAGCGGTCATGTCCGGCAGAGGAAAGGGCGGAAAAGGCTTAGGCAAAGGGGGCGCTAAGCGCC
| |  | ||   |  ||||| |||||||||||||| ||||||||  |||||||  || ||||| |||||||
CAGATTACATTAGCTATGTCTGGCAGAGGAAAGGGTGGAAAGGGTCTAGGCAAGGGTGGCGCCAAGCGCC
            PIV:258

ACCGCAAGGTCTTGAGAGACAACATTCAGGGCATCACCAAGCCTGCCATTCGGCGTCTAGCTCGGCGTGG
| ||||| |||||| | |||||||| ||||| |||||||||| ||||| || || || |||||||| ||
ATCGCAAAGTCTTGCGTGACAACATCCAGGGTATCACCAAGCCCGCCATCCGCCGCCTGGCTCGGCGCGG

CGGCGTTAAGCGGATCTCTGGCCTCATTTACGAGGAGACCCGCGGTGTGCTGAAAGTGTTCTTGGAGAAT
 || || ||||| ||||| |||||||| |||||||||||||| |||||||||||||||||| |||||||
TGGGGTCAAGCGCATCTCCGGCCTCATCTACGAGGAGACCCGTGGTGTGCTGAAGGTGTTCCTGGAGAAC

GTGATTCGGGACGCAGTCACCTACACCGAGCACGCCAAGCGCAAGACCGTCACAGCCATGGATGTGGTGT
|| || || |||||||||||||||||||||||||||||||||||||||||||||||| || ||||||||||||
GTCATCCGCGACGCAGTCACCTACACCGAGCACGCCAAGCGCAAGACCGTCACCGCTATGGATGTGGTGT

ACGCGCTCAAGCGCCAGGGGAGAACCCTCTACGGCTTCGGAGGCTAG
|||| |||||||||||||  | |||||||||||||||||||||||**********************
ACGCTCTCAAGCGCCAGGGCCGCACCCTCTACGGCTTCGGAGGCTAGacgccgccgcttcaattcccccc
```

Figura 5.3: Parte do alinhamento das seqüências HSHIS4 e MMHIST4 construído pelo EXON_FINDER2 (PIP: posição inicial predita; PIV: posição inicial verdadeira.)

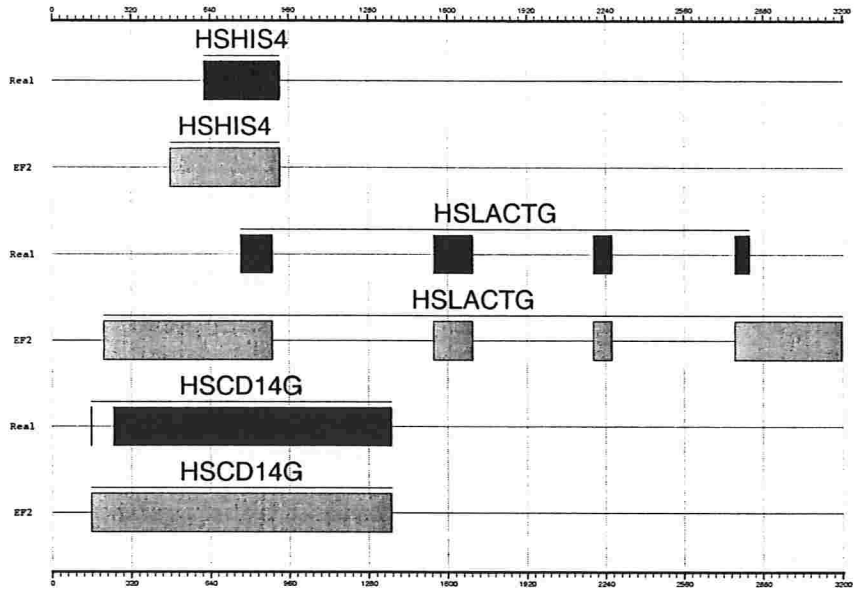

Figura 5.4: Representação gráfica de algumas das predições realizadas pelo programa EXON_FINDER2 (EF2) que incluem as falhas decorrentes da conservação nas regiões limítrofes dos éxons.

Para construção dos pares de testes utilizados neste trabalho, recuperamos inicialmente um total de 165 seqüências genômicas, pertencentes a 63 famílias distintas, de uma base denominada HOGENOM[70]. Dessas seqüências, escolhemos para análise aqueles pares compostos de seqüências compartilhando um identidade maior que 30% (no nível de proteínas) e cujos genes incluem, no mínimo, 900 nucleotídeos. O resultado dessa filtragem consiste em um conjunto de 28 pares de seqüências homólogas pertencentes a seis espécies distintas (*Arabidopsis thaliana*, *Drosophila melanogaster*, *Homo sapiens*, *Mus musculus*, *Saccharomyces cerevisae* e *Schizosaccharomyces pombe*). Desses 28 pares, 17 incluem genes parálogos (pertencentes a mesma espécie) e 11 incluem genes ortólogos (pertencentes a espécies distintas). Informações adicionais sobre cada um desses pares podem ser encontradas na Tabela A.3 do Apêndice.

Escolhidos os pares a serem utilizados como entrada para os programas, três baterias de teste foram então realizadas com cada um deles. A primeira delas consistiu na execução dos dois programas sobre os 28 pares de testes construídos inicialmente. Ou seja, utilizando as seqüências da forma como elas foram recuperadas da base de dados. Os valores de especificidade e sensibilidade, tanto no nível de éxons quanto no nível de nucleotídeos, obtidos para cada uma das predições podem ser vistas na Tabela 5.2. Os valores de AC apresentados pelo UTOPIA variam de 0.36 a 1. As piores predições (valores de AC abaixo de 0.5) foram obtidas sobre os pares de seqüências cujos íntrons

e regiões intergênicas são tão similares quanto os éxons componentes dos seus genes. Olhando-se agora para os resultados apresentados pelo PROGEN, podemos ver que eles são bem parecidos com os do UTOPIA, com esse último apresentando melhores valores de AC em quinze dos 28 pares de testes utilizados.

Ferramentas:	UTOPIA					PROGEN				
Medidas:	Sp_n	Sn_n	AC	Sp_e	Sn_e	Sp_n	Sn_n	AC	Sp_e	Sn_e
Pares										
1.	98	94	0.95	27	64	98	87	0.90	30	42
2.	100	100	1.00	100	100	99	99	0.99	66	80
3.	98	97	0.97	57	90	98	97	0.97	57	80
4.	100	100	1.00	100	100	100	100	1.00	100	100
5.	99	98	0.97	80	85	86	98	0.97	80	85
6.	45	94	0.60	31	57	37	97	0.58	25	61
7.	61	96	0.70	32	50	44	96	0.69	34	56
8.	96	62	0.73	46	39	66	61	0.72	51	43
9.	58	46	0.46	12	13	21	61	0.56	18	33
10.	98	95	0.96	0	0	33	91	0.94	0	0
11.	99	96	0.96	33	50	50	86	0.84	0	0
12.	36	93	0.44	0	0	0	86	0.38	0	0
13.	37	92	0.44	0	0	0	86	0.41	0	0
14.	31	93	0.36	0	0	0	93	0.36	0	0
15.	38	93	0.47	0	0	0	86	0.40	0	0
16.	32	93	0.37	0	0	0	93	0.36	0	0
17.	35	93	0.42	0	0	0	93	0.41	0	0
18.	65	98	0.80	16	35	31	98	0.76	20	64
19.	99	99	0.99	85	92	89	100	1.00	100	100
20.	99	99	0.99	85	92	89	100	1.00	100	100
21.	97	83	0.85	0	0	10	83	0.85	0	0
22.	97	85	0.85	0	0	8	87	0.87	0	0
23.	97	58	0.64	0	0	7	58	0.64	0	0
24.	97	68	0.75	0	0	14	61	0.70	0	0
25.	93	62	0.68	0	0	10	77	0.79	0	0
26.	98	75	0.77	55	52	66	75	0.76	50	52
27.	99	84	0.68	0	0	12	84	0.68	0	0
28.	97	73	0.70	0	0	12	73	0.70	0	0

Tabela 5.2: Qualidade das predições obtidas pelo UTOPIA e PROGEN sobre os pares originais.

Outro detalhe importante está no fato de as duas ferramentas terem se comportado bem em face aos pares incluindo genes com estruturas diferentes. A Figura 5.5 mostra as predições realizadas pelo UTOPIA e PROGEN sobre alguns dos pares com essa característica.

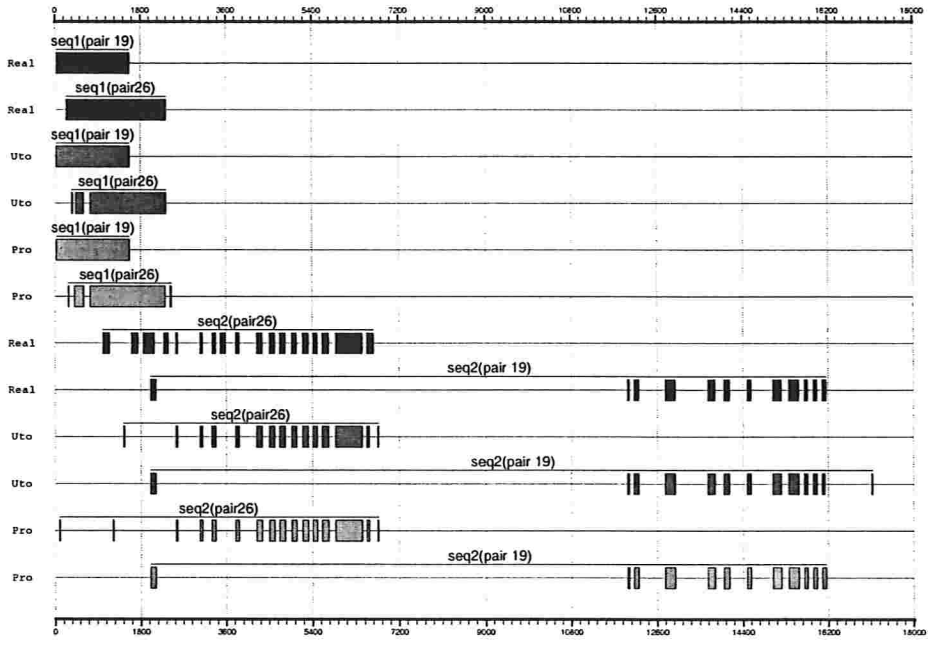

Figura 5.5: Representação gráfica de algumas das predições realizadas pelo UTOPIA(Uto) e PROGEN(Pro) sobre pares de seqüências cujos genes possuem diferentes estruturas.

A segunda rodada de testes foi realizada utilizando alguns pares cujas seqüências incluem um número variado de erros de anotação. Esses pares foram criados artificialmente por meio da inserção ou remoção manual de uma, duas e/ou três bases das seqüências componentes dos pares 5, 8 ,10 e 19. Um total de 52 testes distintos foram criados dessa forma, e os resultados da predição sobre alguns deles podem ser vistos na Tabela 5.3.

Ferramentas:	UTOPIA					PROGEN				
	Par 5									
Medidas:	Sp_n	Sn_n	cc	Sp_e	Sn_e	Sp_n	Sn_n	cc	Sp_e	Sn_e
Original:	98	95	0.96	0	0	99	91	0.94	0	0
Bas/pos:										
-G/1943	98	95	0.96	0	0	99	83	0.88	0	0
-A/2037	98	95	0.96	0	0	99	84	0.89	0	0
-G/2137	98	95	0.96	0	0	99	70	0.79	0	0
	Par 8									
Medidas:	Sp_n	Sn_n	cc	Sp_e	Sn_e	Sp_n	Sn_n	cc	Sp_e	Sn_e
Original:	99	99	0.99	85	92	100	100	1.00	100	100
Bas/pos:										
+G/12000	98	98	0.99	73	84	99	99	0.99	85	92
+A/12136	98	98	0.99	73	84	100	92	0.95	69	69
+T/14060	98	98	0.99	73	84	100	87	0.92	50	53
	Par 10									
Medidas:	Sp_n	Sn_n	cc	Sp_e	Sn_e	Sp_n	Sn_n	cc	Sp_e	Sn_e
Original:	99	98	0.97	80	85	99	98	0.97	80	85
Bas/pos:										
-C/222	99	98	0.97	80	85	99	93	0.91	56	64
-T/455	99	97	0.95	66	71	99	93	0.89	50	57
-A/977	99	97	0.95	66	71	98	86	0.80	43	50
	Par 19									
Medidas:	Sp_n	Sn_n	cc	Sp_e	Sn_e	Sp_n	Sn_n	cc	Sp_e	Sn_e
Original:	96	62	0.73	46	39	96	61	0.72	51	43
Bas/pos:										
-T/7201	96	61	0.72	45	37	96	60	0.71	50	41
-C/10357	96	61	0.72	42	35	96	60	0.71	48	40
-C/20202	96	61	0.72	40	34	96	60	0.71	46	39

Tabela 5.3: Qualidade das predições obtidas pelo UTOPIA e PROGEN sobre alguns dos testes contendo erros de seqüenciamento.

Uma análise detalhada dessas predições mostra que o UTOPIA foi capaz de identificar praticamente todos os erros inseridos nas seqüências. O PROGEN, ao contrário, não identificou nenhum deles. Essa falha no tratamento desse problema por parte do PROGEN reflete-se na predição errônea de um éxon como dois éxons distintos ou deslocados de algumas bases de sua correta posição, dependendo da posição onde as bases foram inseridas ou removidas.

Finalmente, o UTOPIA e o PROGEN foram testados sobre pares de seqüências incluindo mais de um gene. Aqui, como no caso do pares com erros de seqüenciamento, os testes foram criados artificialmente por meio da concatenação das seqüências correspondentes dos pares 1 e 3, 5 e 20, 10 e 11, e 27 e 28. A qualidade das predições realizadas pelos dois programas pode ser vista na Tabela 5.4. Como esperado, o PROGEN não foi capaz de tratar os casos em que mais de um gene está presente nas duas seqüências. Nos quatro pares de testes utilizados, somente um único gene foi predito por esse programa. Sobre o UTOPIA, ele identificou de forma correta apenas 6 do total de 16 éxons extremos e 20 do total de 24 éxons internos aos genes.

Ferramentas:	UTOPIA					PROGEN				
Medidas:	Sp_n	Sn_n	cc	Sp_e	Sn_e	Sp_n	Sn_n	cc	Sp_e	Sn_e
Casos:										
1	94	95	0.92	11	25	99	85	0.89	0	0
2	99	98	0.98	85	88	100	98	0.99	88	85
3	99	82	0.73	0	0	99	79	0.70	0	0
4	69	69	0.58	0	0	83	75	0.72	15	25
5	83	89	0.60	20	42	80	93	0.60	26	39

Tabela 5.4: Qualidade das predições obtidas pelo UTOPIA e PROGEN sobre os pares de testes contendo dois genes.

5.5 Comentários gerais

Neste capítulo abordamos o problema da identificação de genes por meio da comparação de duas seqüências de DNA. Apresentamos primeiramente uma possível formalização matemática do problema e, em seguida, algumas heurísticas que podem ser utilizadas na busca de suas soluções. Detalhamos então os passos de um algoritmo desenvolvido por nós que, com base em algumas das heurísticas apresentadas, se propõe a localizar os genes em duas seqüências de DNA relacionadas evolutivamente. O algoritmo proposto foi implementado em um programa denominado EXON_FINDER2. Esse programa recebe como entrada duas seqüências de DNA e devolve como saída os éxons componentes de cada um dos genes. Para o correto funcionamento de nossa estratégia, supomos que os genes codificados nas seqüências de entrada (que podem ser mais do que um) possuam a mesma estrutura. Apesar de parecer uma restrição forte, vale lembrar que em alguns pares de espécies, como aquele que inclui o *Homo sapiens* e o *Rattus norvegicus*, mais do que 80% dos seus genes ortólogos possuem o mesmo número de éxons[30].

Os resultados apresentados, especificamente no nível de núcleotídeos, mostram que a

precisão de nosso programa compara-se a de um outro programa de predição baseado na comparação de duas seqüências genômicas. Apesar de ainda não competitivo, é importante observar que nosso programa utiliza-se basicamente de informações a respeito das possíveis semelhanças que podem ser observadas entre as regiões codificantes de duas seqüências homólogas. Esse uso limitado de informação explica em partes os valores apenas regulares de especificidade no nível de éxons apresentado pelo EXON_FINDER2. As seqüências que utilizamos para teste pertencem a dois organismos não tão distantes evolutivamente e que, conseqüentemente, incluem algumas regiões não-codificantes extremamente conservadas. É provável que a comparação de organismos mais distantes levem a um melhora na precisão de nosso programa na busca pelos genes de suas seqüências. Vale lembrar também que estamos utilizando um conjunto de parâmetros ajustados de forma empírica. Acreditamos que outros conjuntos de parâmetros, a ser determinados de maneira mais adequada em um trabalho futuro, podem também levar a melhores resultados. Além disso, é também de nosso interesse implementar algumas mudanças em nosso programa de forma que ele possa ser utilizado em casos mais genéricos como aqueles onde os genes procurados não possuam o mesmo número de éxons.

Além da descrição e teste da heurística implementada no EXON_FINDER2, apresentamos também neste capítulo uma avaliação de outras duas ferramentas de predição baseadas na comparação de seqüências genômicas. Essa avaliação, realizada com outros pesquisadores durante o estágio do aluno no exterior, mostra que erros de seqüenciamento, a existência de vários genes e semelhanças entre as regiões não funcionais de dois genomas ainda precisam de um tratamento adequado por parte das ferramentas de predição.

Capítulo 6

Identificação de genes por comparação de vários DNAs

Neste capítulo abordaremos o problema da identificação de genes por meio da comparação de várias seqüências genômicas evolutivamente relacionadas. Novamente, necessitamos aqui que as seqüências incluam um ou mais genes homólogos.

6.1 Breve introdução

Como vimos no capítulo anterior, comparar duas seqüências pode ser extremamente útil na busca por seus genes. Em determinadas ocasiões, porém, dada a proximidade evolutiva das seqüências analisadas, algumas de suas regiões aparentemente não funcionais podem apresentar um grau elevado de semelhança. Isso pode sugerir, por exemplo, que elas fazem parte de algum gene das seqüências e ocasionar uma diminuição da especificidade dos resultados de algum programa de predição construído com base nos métodos comparativos. Uma forma de solucionar esse problema é por meio da busca por regiões conservadas em um número $k > 2$ de seqüências relacionadas evolutivamente. As vantagens por trás dessa possibilidade tornam-se mais claras ao considerarmos a idéia intuitiva de que uma região conservada em um número $k > 2$ seqüências possui uma probabilidade maior de desempenhar alguma função do que uma outra conservada em apenas duas seqüências.

Com isso em mente, descrevemos nesse capítulo os conceitos básicos relacionados ao problema do alinhamento múltiplo e a utilização desses conceitos em uma nova estratégia para identificação de genes por comparação de seqüências. Até onde sabemos, este é um dos primeiros trabalhos envolvendo a comparação de várias seqüências na

busca por melhores resultados na área [1]

6.2 Alinhamento múltiplo

Como vimos no Capítulo 2, alinhar duas seqüências s e t consiste na inserção de espaços em posições arbitrárias dessas seqüências de forma que elas fiquem do mesmo tamanho e que nenhum espaço em uma das seqüência emparelhe-se com um espaço da outra. Uma generalização natural dessa tarefa consiste em alinhar não apenas duas, mas um número qualquer $k > 2$ seqüências. Ou seja, estamos agora interessados na inserção de espaços em posições arbitrárias de um número k de seqüências de forma que todas elas fiquem do mesmo tamanho e que nenhuma coluna do alinhamento seja composta apenas de espaços. Várias são as motivações para essa generalização do problema do alinhamento. Uma das mais importantes é o fato de um alinhamento múltiplo ser capaz de elucidar regiões conservadas que podem passar despercebidas quando apenas duas seqüências são comparadas.

Em termos formais, um alinhamento múltiplo de k seqüências $s_1, s_2, \ldots, s_k$ construídas sobre um alfabeto Σ consiste em uma k-tupla $A = (\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_k)$, com $\bar{s_1}, \ldots, \bar{s_k} \in \bar{\Sigma}^*$ onde:

1. $|\bar{s_1}| = |\bar{s_2}| = \ldots = |\bar{s_k}|$;
2. $\bar{s}_{i|\Sigma} = s_i$ para todo $i = 1, \ldots, k$;
3. Não existe j tal que $\bar{s_i}[j] = -$ para todo $i = 1, \ldots, k$.

Da mesma forma que podemos atribuir uma pontuação para cada uma das colunas de um alinhamento envolvendo duas seqüências, podemos também atribuir uma pontuação para as colunas de uma alinhamento múltiplo de acordo com os caracteres que elas incluem. Generalizando o que foi dito anteriormente para o caso de alinhamento de pares de seqüências, aqui temos uma função de pontuação w definida como $w : (\bar{\Sigma})^k \rightarrow \mathbb{R}$. Ou seja, uma função que associa um certo valor real para cada k-tupla de símbolos $(\mu_1, \ldots, \mu_k)$ em $\bar{\Sigma}$.

Dada uma função de pontuação, a pontuação $\text{Score}_w(A)$ de um alinhamento qualquer $A = \bar{s_1}, \ldots, \bar{s_k}$ de k seqüências, como no caso do alinhamento de pares, é definida como a soma da pontuação de cada uma de suas colunas. Novamente, a similaridade $\text{sim}_w(s_1, \ldots, s_k)$ entre k seqüências $s_1, \ldots, s_k$, dada uma função de pontuação w,

[1] Um outro trabalho que envolve a comparação de várias seqüência na busca por seus genes foi recentemente descrito por Siepel e Haussler em [79].

é definida como a maior pontuação dentre as pontuações de todos os possíveis alinhamentos de $s_1, \ldots, s_k$. Ou seja: $\text{sim}_w(s_1, \ldots, s_k) = \max\{\text{Score}_w(\bar{s_1}, \ldots, \bar{s_k})\}$, onde $(\bar{s_1}, \ldots, \bar{s_k})$ é um alinhamento entre $s_1, \ldots, s_k$. Naturalmente, um alinhamento ótimo A^* de k seqüências $s_1, s_2, \ldots, s_k$ é definido como um alinhamento tal que $\text{Score}_w(A^*) = \text{sim}_w(s_1, \ldots, s_k)$. Dito isso, o problema do alinhamento múltiplo pode ser assim definido:

> **Problema do Alinhamento Múltiplo** (PAM): *Dadas k seqüências $s_1, \ldots, s_k$ construídas sobre um alfabeto Σ qualquer tal que $\{-\} \notin \Sigma$ e uma função de pontuação w, encontrar o valor da similaridade $\text{sim}_w(s_1, \ldots, s_k)$ entre elas e um de seus alinhamentos ótimos.*

Note que, da forma como a pontuação de um alinhamento múltiplo foi definida, o PAM também possui a propriedade das subsoluções ótimas discutida durante a busca por um algoritmo para o problema do alinhamento de duas seqüências. Com isso, o problema do alinhamento múltiplo também pode ser resolvido utilizando a estratégia de programação dinâmica. A única diferença em relação à recorrência apresentada na Seção 2.2 do Capítulo 2 está no número de escolhas a serem feitas para determinação da similaridade e de um alinhamento ótimo relacionado. No caso do alinhamento envolvendo apenas duas seqüências s e t, a similaridade entre $s[1..i]$ e $t[1..j]$ é calculada tomando-se o máximo dentre 3 opções, cada qual relacionada a uma das três possibilidades distintas de preenchimento da última coluna do alinhamento. No caso da busca pelo melhor alinhamento de $k > 2$ seqüências, o máximo dentre $2^k - 1$ opções precisa ser tomado durante o cálculo da similaridade associada. Isso se explica dado o fato de termos um total de $2^k - 1$ combinações possíveis de caracteres para uma determinada coluna do alinhamento. Note que isso torna a complexidade do algoritmo baseado na estratégia de programação dinâmica exponencial no tamanho da entrada nos casos em que k não é uma constante.

Na realidade, para uma função de pontuação especial, denominada função de pontuação de pares, ou pontuação SP, já foi provado o fato de o problema do alinhamento múltiplo ser NP-completo. Por ser uma das funções de pontuação mais utilizadas na literatura durante o tratamento do problema do alinhamento de várias seqüências, daremos uma atenção especial à pontuação SP no decorrer deste capítulo.

Dada uma função de pontuação qualquer w que associa um valor real para cada par de caracteres de um alfabeto Σ, a função de pontuação SP pode ser descrita como uma função que associa a cada coluna de um alinhamento múltiplo A um valor igual à soma da pontuação de todos os pares possíveis de caracteres dessa coluna. Ou seja,

$$\text{SP}_w(A) = \sum_{1 \leq i \leq i' \leq k} w(C[i], C[i']),$$

onde $C[i]$ denota o i-ésimo caractere da coluna C do alinhamento A. A função de pon-

tuação SP pode também ser vista como a soma da pontuação de todos os alinhamentos de duas seqüências induzidos por um alinhamento múltiplo. Um alinhamento induzido por A de duas seqüências s_i e s_j, denotado por $A_{|s_i s_j}$ corresponde ao alinhamento obtido pela remoção de todos os elementos da k-tupla $A = (s_1, \ldots, s_k)$, com exceção de $\bar{s}_i$ e $\bar{s}_j$, e então das possíveis posições i dessas seqüências tais que $\bar{s}[i] = - = \bar{t}[i]$. Disso, dado um alinhamento A de k seqüências $s_1, \ldots, s_k$, podemos também definir a pontuação SP de A como

$$\mathrm{SP}_w(A) = \sum_{i=1}^{k} \sum_{j=1}^{i-1} \mathrm{Score}_w(s_i s_j).$$

Um algoritmo de aproximação para o problema do alinhamento múltiplo com função de pontuação SP foi primeiramente desenvolvido por Gusfield e encontra-se descrito em [45]. As idéias desse algoritmo estão fortemente ligadas ao seguinte fato, descrito por Feng e Doolitlle em [34]. No que segue, dizemos que um alinhamento A de k seqüências $S = (s_1, \ldots, s_k)$ é compatível com um alinhamento A' de duas seqüências quaisquer s_i e s_j em S se a pontuação do alinhamento induzido por A dessas duas seqüências é igual à pontuação de A'

Teorema 6.2.1: *Dada uma árvore qualquer T onde cada nó v representa uma seqüência s_v e cada aresta uv um alinhamento qualquer das seqüências s_u e s_v, existe um alinhamento de todas as seqüências representadas pelos vértices de T que é compatível com os alinhamentos de pares representados pelas arestas dessa árvore.*

Demonstração: Iremos provar o teorema acima enumerando os passos de um algoritmo que recebe como entrada uma árvore qualquer T e, em tempo polinomial, devolve um alinhamento múltiplo das seqüências representadas pelos vértices de T compatível com os alinhamentos de pares representados pelas suas arestas. Para uma melhor compreensão desse algoritmo, considere as seguintes definições envolvendo uma seqüência qualquer s e um conjunto $\mathcal{A} = \{A_1, \ldots, A_n\}$ de alinhamentos de s e alguma outra seqüência. Seja $p_{s_{i,j}}$ a posição do j-ésimo caractere de s em um alinhamento $A_i \in \mathcal{A}$. Defina $z_{s_{i,j}}$ como o número de espaços imediatamente anteriores a $s[j]$ em A_i. Em outras palavras, $z_{s_{i,j}} = p_{s_{i,j}} - p_{s_{i,j-1}} - 1$. Por último, considere $z^*_{s_{1\ldots n,j}}$ como sendo o maior número de espaços imediatamente anteriores ao caractere $s[j]$ tomando-se o conjunto $\mathcal{A} = \{A_1, \ldots, A_n\}$ dos n alinhamentos envolvendo essa seqüências. Ou seja, $z^*_{s_{1\ldots n,j}} = \max z_{s_{i,j}}$ para $i = 1, \ldots, n$. Dadas essas definições, partamos agora para os passos do algoritmo propriamente dito.

Seja s uma seqüência de tamanho m envolvida em pelo menos dois alinhamentos

distintos $A_1 = \{\bar{s}, \bar{t}\}$ e $A_2 = \{\bar{s}, \bar{u}\}$[2] representados por duas arestas, st e su, de T[3]. Tomemos então um desses alinhamentos. Seja $A_1 = \{\bar{s}, \bar{t}\}$ o alinhamento escolhido. Consideremos agora a seqüência u ligada à seqüência s por meio da aresta su representativa do alinhamento A_2. O passo seguinte é determinar o valor de $z^*_{s_{1..2,j}}$ tomando-se o conjunto $A = \{A_1, A_2\}$, para todo $1 \leq j \leq m$. Ou seja, calcular o maior número de espaços imediatamente anteriores a todos os caracteres de s nos alinhamentos A_1 e A_2 que envolvem essa seqüência. Uma vez determinado esses valores, o próximo passo é inserir um total de $z^*_{s_{1..2,j}}$ colunas formadas apenas por espaços nas posições $p_{s_{i,j}}$ de s em ambos os alinhamentos A_1 e A_2, para todo $1 \leq j \leq m$. Denotemos por $\bar{A}_1$ e $\bar{A}_2$ os alinhamentos obtidos após a inserção dessas colunas contendo apenas espaços e por $\bar{\bar{s}}$, $\bar{\bar{t}}$ e $\bar{\bar{u}}$ as seqüências componentes de cada um desses alinhamentos. Note que, após esse último passo, todos os caracteres de s encontram-se agora separados da mesma forma em $\bar{A}_1$ e $\bar{A}_2$. Com isso, $\bar{A}_1$ e $\bar{A}_2$ possuem o mesmo tamanho, o que nos permite agrupá-los em um único alinhamento. Ou, em outras palavras, estender o alinhamento $\bar{A}_1$ pela adição da seqüência $\bar{\bar{u}}$. Ao final desse processo, temos então que $\bar{A}_1 = \bar{A}_1 \cup \{\bar{\bar{u}}\} = \{\bar{\bar{s}}, \bar{\bar{t}}, \bar{\bar{u}}\}$. Observe que a seqüência $\bar{\bar{s}}$ em $\bar{A}_1$ é uma cópia da seqüência $\bar{\bar{s}}$ em $\bar{A}_2$, podendo então ser desconsiderada em um desses alinhamentos. Uma vez agrupados os alinhamentos, o que fazemos então é eliminar de T a aresta su escolhida inicialmente e repetir os passos descritos acima até que não restem mais arestas em T. Em cada uma das iterações seguintes, tomamos A_1 como sendo o alinhamento resultante do agrupamento realizado no último passo da iteração e A_2 como um alinhamento qualquer envolvendo uma seqüência já alinhada em A_1. Considerando uma função de pontuação que associa o valor zero às colunas contendo apenas espaços, o alinhamento obtido ao final dessas iterações é compatível com os todos alinhamentos representados pelas arestas da árvore. ∎

A Figura 6.1 mostra um exemplo de alinhamento múltiplo compatível com os alinhamentos representados pelas arestas da árvore T e construído seguindo os passos acima. Esses passos encontram-se enumerados no Algoritmo 8.

Para determinação da complexidade do Algoritmo 8, assumimos que, na sua i-ésima iteração (laço que vai da linha 3 à linha 11 do algoritmo), as seqüências alinhadas em A_1 possuem tamanho li^{A_1} e aquelas alinhadas em A_2 tamanho li^{A_2}. Assim e tomando-se l_i como $\max\{li^{A_1}, li^{A_2}\}$, a linha 5 do algoritmo consome tempo $O(l_i)$ na sua i-ésima iteração. Como existe um total de $|E|$ iterações a ser realizada, o tempo total da linha 5 do algoritmo é de $O(\sum_{i=1}^{|E|} l_i)$. Sobre a linha 6 do algoritmo, ela consome tempo $O(|s|)$ em cada uma de suas iterações. Como $|s| \leq l_i$ em cada uma das i-ésima iterações, podemos dizer que a linha 6 do algoritmo pode ser executada em $O(l_i)$ e o tempo total

[2]Observe que as seqüências $\bar{s}$ em A_1 e A_2 não são necessariamente as mesmas.

[3]Note que uma seqüência desse tipo sempre existe dada a conexidade de nosso grafo e o fato de estarmos trabalhando com um número $k \geq 3$ de seqüências.

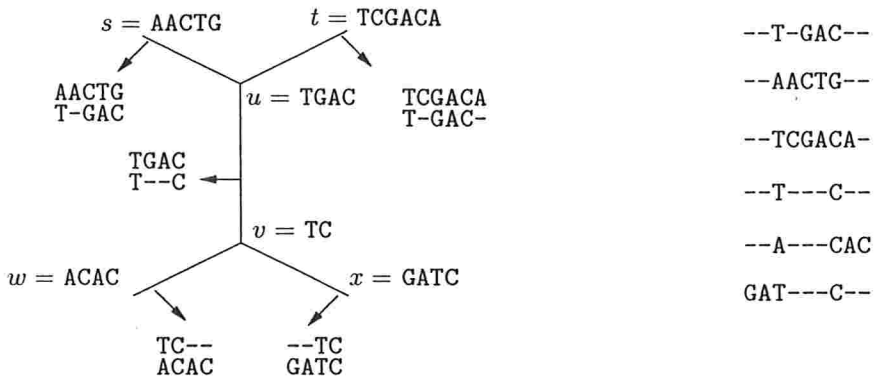

Figura 6.1: Árvore representando vários alinhamentos de duas seqüências e o respectivo alinhamento múltiplo compatível com eles

que ela consome é também de $O(\sum_{i=1}^{|E|} l_i)$. A tarefa de inserir um total de $z^*_{s_{1..2},j}$ colunas contendo espaços, para $j = 1, \ldots, |s|$, nos alinhamentos A_1 e A_2 pode ser realizada em tempo $O(\sum_{j=1}^{|s|} l_i)$. Disso, as linhas 7 e 8 do algoritmo consomem tempo total de $O(\sum_{i=1}^{|E|} \sum_{j=1}^{|s|} l_i)$. A linha 9 do algoritmo consiste, basicamente, em uma cópia dos caracteres de $\overline{\overline{u}}$, que pode também ser realizada em $O(l_i)$ em cada uma das iterações, consumindo tempo total de $O(\sum_{i=1}^{|E|} l_i)$. Tomando-se k como o número total de arestas da árvore e l como o tamanho das seqüências alinhadas em A_1 no final do Algoritmo 8, temos que o tempo total consumido por ele é de $O(kl) + O(kl) + O(kl^2) = O(kl^2)$.

Com base no fato acima, Gusfield desenvolveu um algoritmo de aproximação para o problema do alinhamento de k seqüências, com respeito a função de pontuação SP, cuja razão é de $2 - 2/k$, onde k é o número de seqüências alinhadas. A idéia geral desse algoritmo é construir um alinhamento múltiplo compatível com vários alinhamentos ótimos de uma seqüência s_g, denominada seqüência guia, e todas as outras seqüências a serem alinhadas. Dadas k seqüências de entrada, o que o algoritmo proposto por Gusfield faz é, primeiramente, tomar as seqüências duas a duas, alinhá-las e determinar aquela mais parecida com as outras. Ou seja, aquela seqüência cuja soma da similaridade (ou distância) com todas as outras seja máxima (ou mínima). Essa é a seqüência guia s_g citada anteriormente. Os alinhamentos ótimos de s_g com todas as outras seqüências são então agrupados, por meio do Algoritmo 8, em um alinhamento envolvendo as k seqüências de entrada. Esse algoritmo é conhecido pelo nome de **Algoritmo Estrela** pelo fato de a árvore T nesse caso incluir um vértice central, aquele representativo da seqüência guia, ligado a todos os outros vértices de T.

Para mostrarmos que a razão de aproximação do Algoritmo 9 é igual a $2 - 2/k$, precisamos assumir o uso de uma função de pontuação pertencente à classe das métricas. Ou seja, uma função f que, dentre outras propriedades, satisfaz $f(\mu, \nu) = 0$ se e somente

Algoritmo 8 AGRUPA_ALINHAMENTOS(T)

Entrada: uma árvore $T = (V, E)$ cujos vértices representam seqüências e cada aresta um alinhamento das seqüências ligadas por ela.

Saída: um alinhamento das seqüências em V compatível com os alinhamentos representados pelos elementos em E.

1: $A_1 \leftarrow$ alinhamento representado por uma aresta qualquer st em E.
2: $E \leftarrow E - st$;
3: **enquanto** $E \neq \emptyset$ **faça**
4: $A_2 \leftarrow$ alinhamento representado por uma aresta su que possui uma das pontas (seqüência) em A_1. Seja s essa seqüência;
5: Calcule $p_{s_{1,j}}$ e $p_{s_{2,j}}$ para $j = 1, \ldots, |s|$;
6: Calcule $z^*_{s_{1..2,j}} = \max\{z_{s_{1,j}}, z_{s_{2,j}}\}$ para $j = 1, \ldots, |s|$;
7: $\bar{A}_1 \leftarrow A_1$ inserido de um total de $z^*_{s_{1..2,j}}$ colunas contendo espaços na posição $p_{s_{1,j}}$, para todo $j = 1, \ldots, |s|$.
8: $\bar{A}_2 \leftarrow A_2$ inserido de um total de $z^*_{s_j}$ colunas contendo espaços na posição $p_{s_{2,j}}$, para todo $j = 1, \ldots, |s|$.
9: $A_1 \leftarrow \bar{A}_1 \cup \{\overline{u}\}$
10: $E \leftarrow E - su$;
11: **fim-enquanto**
12: Devolva A_1;

$\mu = \nu$ e $f(\mu, \nu) \leq f(\mu, \tau) + f(\tau, \nu)$, para todo μ, ν, $\tau \in \Sigma$. Esse é o caso por exemplo da função de pontuação zero-ou-um que nos permite determinar a distância de edição entre duas seqüências. É com uma função desse tipo que trabalharemos a prova da razão de aproximação do Algoritmo 9 a seguir.

No que segue, chamamos de A o alinhamento das seqüências $s_1, s_2, \ldots, s_k$ devolvido pelo Algoritmo 9 e de A^* um alinhamento ótimo dessas seqüências. Dadas duas seqüências s_i e s_j envolvidas nesses alinhamentos, denotamos por $d(s_i, s_j)$ o custo associado ao alinhamento de s_i e s_j induzido por A, por $d^*(s_i, s_j))$ o custo associado ao alinhamento dessas duas seqüências induzido por um alinhamento ótimo A^* e por $D(s_i, s_j)$ o custo de um alinhamento ótimo dessas seqüências. Além disso, usamos M_i para denotar a soma dos custos dos alinhamentos envolvendo a seqüência s_i e todas as outras seqüências dadas como entrada para o Algoritmo 9. Finalmente, chamamos de $c(A)$ e $c(A^*)$ o custo dos alinhamentos A e A^*. Dadas essas definições o seguinte teorema pode ser demonstrado:

Teorema 6.2.2: *o custo do alinhamento A devolvido pelo Algoritmo 9 é tal que $c(A) \leq (2 - 2/k)c(A^*)$.*

Algoritmo 9 ALGORITMO_ESTRELA($s_1, s2, \ldots, s_k$)

Entrada: um conjunto de k seqüências a serem alinhadas.
Saída: um alinhamento múltiplo A das k seqüências tal que $\text{Score}_{SP}(A) \geq (2 - 2/k)\text{Score}_{SP}(A^*)$.

1: **para** $i = 1$ até k **faça**
2: **para** $j = i + 1$ até k **faça**
3: $A_{ij} \leftarrow$ CALCULA_SIMILARIDADE(s_i, s_j, SP);
4: **fim-para**
5: **fim-para**
6: **para** $i = 1$ até k **faça**
7: $M_i \leftarrow \sum_{j=1(j\neq i)}^{k}(A_{ij})$;
8: **fim-para**
9: $g \leftarrow i$ tal que M_i é máximo;
10: $M_g \leftarrow \max_{i=1}^{k} M_i$;
11: $T \leftarrow A_{g1}, A_{g2} \ldots A_{gk}$;
12: $A \leftarrow$ AGRUPA_ALINHAMENTOS(T);
13: Devolva A;

Demonstração: Para demonstração do Teorema 6.2.2, lembremos que

$$c(A) = \sum_{i=1}^{k}\sum_{j=1}^{i-1} d(s_i, s_j).$$

Pela desigualdade triangular associada à função de pontuação f, temos que

$$c(A) \leq \sum_{i=1}^{k}\sum_{j=1}^{i-1}[d(s_i, s_g) + d(s_g, s_j)] = \sum_{i=1}^{k}\sum_{j=1}^{i-1} d(s_i, s_g) + \sum_{i=1}^{k}\sum_{j=1}^{i-1} d(s_g, s_j).$$

Note que, para um i fixo, o termo $d(s_i, s_g)$ (ou $d(s_g, s_i)$) aparece $i - 1$ vezes no primeiro somatório e $k - i$ vezes no segundo. Segue então que

$$c(A) \leq (k - 1)\sum_{i=1}^{k} d(s_i, s_g).$$

Como os alinhamentos induzidos por A que envolvem a seqüência guia s_g são ótimos, temos que $d(s_i, s_g) = D(s_i, s_g)$. Daí

$$c(A) \leq (k - 1)\sum_{i=1}^{k} D(s_i, s_g) = (k - 1)M_g.$$

Por outro lado, sabemos que $c(A^*) = \sum_{i=1}^{k} \sum_{j=1}^{i-1} d^*(s_i, s_j) = \frac{1}{2} \sum_{i=1}^{k} \sum_{j=1}^{k} d^*(s_i, s_j)$. Como $d^*(s_i, s_j) \geq D(s_i, s_j)$, então

$$c(A^*) \geq \frac{1}{2} \sum_{i=1}^{k} \sum_{j=1}^{k} D(s_i, s_j) \geq \frac{1}{2} \sum_{i=1}^{k} M_i \geq \frac{1}{2} \sum_{i=1}^{k} M_g = \frac{1}{2} k M_g$$

Daí, $c(A) \leq (k-1)M_g = (k-1)\frac{2}{k}c(A^*) = (2 - 2/k)c(A^*)$. ∎

Para determinação da complexidade do Algoritmo 9, assumiremos que as k seqüências a serem alinhadas possuem tamanho m. Dessa forma, o primeiro laço do algoritmo acima consome tempo $O(k^2m^2)$. Uma vez determinados os alinhamentos envolvendo todos os possíveis pares de seqüências, o cálculo de cada M_i pode ser realizado em tempo $O(k)$. Com isso, o segundo laço do algoritmo leva tempo $O(k^2)$. Dessa forma, a complexidade total de tempo Algoritmo 9 é $O(k^2m^2)$.

6.3 O Problema do alinhamento sintênico múltiplo

Parecido com o problema do alinhamento sintênico descrito no capítulo anterior, aqui também estamos interessados em um alinhamento que represente de forma adequada o fato de as seqüências sendo comparadas incluírem regiões altamente similares separadas por regiões não tão parecidas. A diferença principal está no fato de mais de duas seqüências estarem agora envolvidas na comparação. Em outras palavras, estamos interessados em alinhamentos como o mostrado na Figura 6.2, onde as caixas representam as regiões bem conservadas das seqüências.

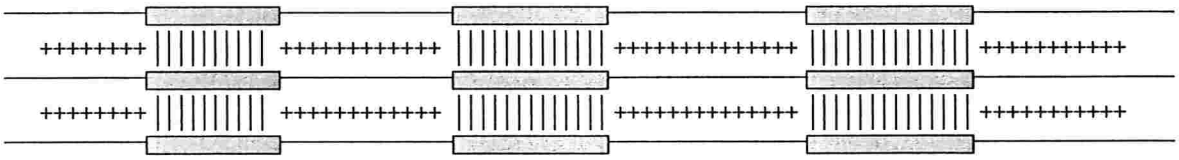

Figura 6.2: Representação de um alinhamento sintênico múltiplo (3 seqüências)

Esse tipo de alinhamento é conhecido como **alinhamento sintênico múltiplo**. O problema de se encontrar um alinhamento sintênico múltiplo entre várias seqüências pode ser assim formulado:

Problema do Alinhamento Sintênico Múltiplo (PASiM): *Dadas k seqüências $s_1, s_2, \ldots, s_k$ construídas sobre um alfabeto Σ qualquer tal que $\{-\} \notin \Sigma$, e uma função de pontuação w, encontrar uma cadeia $S_i = \{s'_{i1}, s'_{i2}, \ldots, s'_{in}\}$ de segmentos de s_i, para todo $i = 1, \ldots, k$, tal que $\sum_{i=1}^{k} \sum_{j=1}^{i-1} Score_w(S_i^\bullet, S_j^\bullet)$ seja máxima, onde $S_i^\bullet = \{s'_{i1} \bullet s'_{i2} \bullet \ldots \bullet s'_{in}\}$.*

Nossa idéia aqui é prover uma heurística ao problema descrito acima com base na aproximação desenvolvida por Gusfield para o problema do alinhamento múltiplo. A principal diferença de nossa estratégia com relação a essa aproximação está no fato de utilizarmos um algoritmo de alinhamento que leva em consideração a existência, nas seqüências comparadas, de regiões bem conservadas separadas por outras não tão parecidas. Como no capítulo anterior, aqui também fizemos uso do algoritmo desenvolvido por Almeida *et al.* que se utiliza de duas funções distintas de pontuação na determinação de um alinhamento sintênico de duas seqüências.

Em linhas gerais, a idéia de nossa heurística é tomar as seqüências de entrada duas a duas, alinhá-las sintenicamente e identificar aquela de maior semelhança com as demais. A partir dos alinhamentos envolvendo essas seqüências, o próximo passo é construir um alinhamento múltiplo compatível com eles. Isso é feito utilizando o Algoritmo 8 descrito anteriormente. As regiões conservadas nas k seqüências corresponderiam aos segmentos procurados.

Como a idéia inicial é encontrar os genes em cada uma das seqüências dada como entrada, o que fazemos após a construção do alinhamento é considerar as regiões conservadas em todas elas como núcleos dos possíveis éxons componentes desses genes. Assim, uma vez determinadas essas regiões, o que fazemos é estendê-las ou apará-las até que os sinais determinantes das fronteiras dos éxons sejam encontrados. O resultado desse processo consiste em um conjunto de possíveis éxons associado às regiões conservadas de cada uma das seqüências. Aos éxons assim determinados associamos então uma pontuação igual à soma da pontuação das partes dos alinhamentos de pares envolvendo as seqüências desses éxons, alinhamentos esses determinados pela projeção do alinhamento múltiplo construído inicialmente sobre todos os pares de seqüências. O último passo consiste então em montar esses éxons de forma a reconstituir da melhor maneira possível um ou mais genes de cada uma das seqüências.

Esse último passo é implementado por meio da construção de um grafo dirigido G representativo de todas as possíveis montagens dos éxons encontrados. Cada um dos vértices $v_i \in V_G$ representa um possível éxon e_i da seqüência sendo considerada. Sobre o conjunto E_G, ele inclui um arco para cada par ordenado de vértices compatíveis. Dados dois vértices quaisquer u e $v \in V_G$, dizemos que os vértices do par ordenado (u, v) são compatíveis se o éxon representado por u precede o éxon representado por v na seqüência de onde eles foram recuperados. Além dessa restrição de precedência, se u representa um éxon inicial, v representa necessariamente um éxon interno ou final. O mesmo acontece no caso em que u representa um vértice interno. Os arcos que conectam um par de vértices compatíveis recebem um peso igual à pontuação do éxon representado pelo vértice à direita no par. Além dos vértices representativos dos possíveis éxons nas seqüências, temos outros dois vértices adicionais o (origem) e d (destino), sendo que o está conectado a todos os vértices representativos dos éxons iniciais, enquanto que todos os vértices representativos dos éxons finais estão conectados ao vértice d. A

Figura 6.3 mostra um exemplo de grafo desse tipo para um conjunto de sete éxons $e_1 = \{15, 76, i, 256\}$, $e_2 = \{25, 87, i, 45\}$, $e_3 = \{13, 87, i, 156\}$, $e_4 = \{231, 245, m, 85\}$, $e_5 = \{356, 412, m, 90\}$, $e_6 = \{238, 300, m, 56\}$ e $e_7 = \{459, 578, f, 152\}$, onde os dois primeiros campos das quádruplas representam a posição inicial e final do éxon na seqüência, o terceiro campo o tipo do éxon (i - inicial, m - interno e f - final), e o quarto a pontuação relacionada.

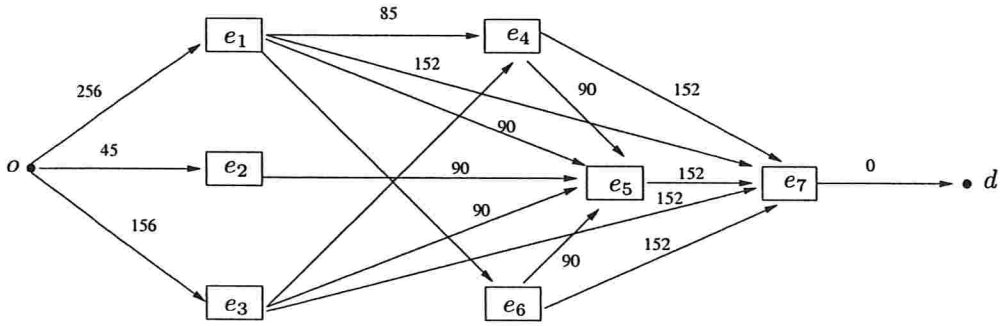

Figura 6.3: Exemplo de um grafo representativo de todas as possíveis montagens de um conjunto de éxons.

Uma vez construído o grafo representativo de todas as montagens dos possíveis éxons, o próximo passo é determinar um caminho de peso máximo em G que vai do vértice o ao vértice d. Os éxons representados pelos vértices desse caminho (a menos dos vértices extremos) correspondem aos éxons procurados. Os passos acima encontram-se detalhados no Algoritmo 10.

Com relação à complexidade de tempo da nossa heurística, note que a determinação do alinhamento múltiplo na primeira linha do Algoritmo 10 consome tempo igual a $O(k^2m^2)$, relembrando que todas as seqüências possuem tamanho m. A fase de determinação das posições de início e fim das regiões conservadas de cada uma das seqüências pode ser realizada em tempo $O(k\sum_{i=1}^{k}|s_i|)$. Com isso, o primeiro laço do Algoritmo 10 consome tempo igual a $O(k^2m)$. Sobre o segundo laço do algoritmo, os limites de cada éxon podem ser determinados em tempo $O(kb)$, onde b é o número total de blocos conservados. Seja b_i o número de possíveis éxons na seqüência s_i determinados no passo anterior. Os grafos representativos das possíveis montagens desses éxons pode ser construído em $O(\sum_{i=1}^{k} b_i^2)$. Note que os grafos construídos são dirigidos e acíclicos. Com isso, os caminhos de peso máximo determinantes dos éxons procurado podem ser encontrados em tempo $O(\sum_{i=1}^{k}|E_{G_i}|) = O(\sum_{i=1}^{k} b_i^2)$. Temos então que a complexidade de tempo total da estratégia acima é igual a $O(k^2m^2) + O(\sum_{i=1}^{k} e_i^2)$.

Algoritmo 10 ALGORITMO_MONTA_EXONS($s_1, s_2, \ldots, s_k$)

Entrada: um conjunto com k seqüências.
Saída: os possíveis éxons constituintes dos genes nas k seqüências de entrada.

1: $A \leftarrow$ ALGORITMO_ESTRELA($s_1, s_2, \ldots, s_k$);
2: $b \leftarrow$ número total de regiões conservadas em A;
3: **para** $i = 1$ até k **faça**
4: **para** $j = 1$ até b **faça**
5: Blocos[i][j] $\leftarrow$ posição inicial e final da j-ésima região conservada da seqüência s_i;
6: **fim-para**
7: **fim-para**
8: **para** $i = 1$ até k **faça**
9: **para** $j = 1$ até b **faça**
10: Exons[i] $\leftarrow$ regiões limitadas por sítios de início e final de éxons em torno do bloco determinado por Blocos[i][j] ;
11: **fim-para**
12: **fim-para**
13: **para** $i = 1$ até k **faça**
14: $G_i \leftarrow$ CONSTROI_GRAFO(EXONS[i]);
15: $c_i \leftarrow$ CAMINHO_MINIMO_DAG(G_i, o, d, w)
16: Devolva éxons representados pelos vértices em c_i;
17: **fim-para**

6.3.1 Testes

Como nos capítulos anteriores, no intuito de avaliarmos nossa estratégia, a implementamos em um programa denominado EXON_FINDER3 (utilizando novamente a linguagem C) e então o testamos sobre um conjunto incluindo algumas poucas triplas de seqüências pertencentes ao *Homo sapiens*, *Rattus norvegicus* e *Mus musculus*. Essas seqüências foram obtidas por meio de uma busca no GENBANK por seqüências homólogas àquelas componentes dos pares utilizados no capítulo anterior. Um total de 3 triplas foram recuperadas nesse processo, cujas informações a respeito do tamanho, número de éxons, coordenadas iniciais e finais, etc. podem ser encontradas na Tabela 6.1.

As predições obtidas por nosso programa para cada uma das três triplas podem ser vistas na Tabela 6.2.

Triplas:	Seqüências	Tamanho	Coordenadas
1.	HSCKBG	4200	join(1148..1340, 1464..1618, 1691..1823, 2208..2379, 3050..3173, 3331..3520, 3600..3778)
	MUSCRKNB	4521	join(1711..1903, 2090..2244, 2322..2454, 2624..2795, 3381..3504, 3652..3841, 3925..4103)
	RATCKBR	4360	join(1276..1468, 1656..1810, 1888..2020, 2197..2368, 2941..3064, 3174..3363, 3443..3621)
2.	HSU66875	1569	join(802..874, 976..1112, 1281..1364)
	MMU34801	1910	join(729..801, 988..1124, 1202..1285)
	RNCOX6B	2569	join(1487..1762, 1951..2087, 2172..2366)
3.	HUMTHY1A	2806	join(27..63, 547..882, 1410..1522)
	MUSTHY1GC	3257	join(555..591, 1182..1520, 1907..2019)
	RNTHY1G	2863	join(31..67,735..1070,1473..1585)

Tabela 6.1: Triplas utilizadas para teste do Exon_Finder3

Triplas:	Predição
1.	(1148..1340, 1464..1618, 1691..1754, 1783..1823, 2089..2379, 2957..3104, 3120..3219, 3331..3520, 3600..4081) (1581..1988, 2090..2109, 2120..2311, 2322..2454, 2624..2947 3381..3504, 3527..3584, 3594..3861, 3925..4412) e (790..1030, 1203..1557, 1588..1675, 1677..1833, 1888..2030 2085..2520, 2797..2917, 2941..3074, 3083..3123, 3140..3363, 3412..3992)
2.	(641..874, 931..1186, 1281..1312, 1325..1399), (625..801, 988..1031, 1036..1171, 1202..1286, 1294..1388, 1427..1500) e (1388..1762, 1871..1976, 1994..2107, 2172..2268, 2283..2535);
3.	(628..882, 1290..1424, 1451..1479, 1499..1624), (1263..1520, 1839..1941, 1948..2022, 2035..2171) e (630..1070, 1341..1453, 1473..1588, 1601..1656).

Tabela 6.2: Predições realizadas

Pela Tabela 6.2, podemos ver que, na primeira tripla, várias foram as extremidades preditas corretamente por nosso programa: 21 de um total de 42. Note também que, de um total de 34 éxons preditos, apenas 5 não possuem nenhuma sobreposição com um éxon conhecido. Sobre o segundo caso de teste, os resultados foram um pouco piores, com a identificação de alguns falsos-positivos por nosso programa. Sobre a terceira tripla, além de alguns falsos-positivos, os primeiros éxons de cada uma das seqüência acabaram não sendo identificados por nosso programa. Essa falha pode estar relacionada ao fato desses éxons incluírem apenas um pequeno número de bases.

Apesar dos resultados apenas regulares aqui mostrados, é importante notar que, ao menos no segundo caso de teste, o resultado obtido pelo EXON_FINDER3 parece melhor que aquele apresentado pelo programa EXON_FINDER2 quando ambos os programas são executados com os mesmos parâmetros. As seguintes coordenadas foram preditas por esse último quando executado sobre o par envolvendo as seqüências HSU66875 e MMU34801: (66..123, 276..878, 976..1488) e (89..137, 208..805, 988..1500). A diferença entre as duas predições fica evidente na Figura 6.4. Esse exemplo está de acordo com a intuição de que melhores resultados podem ser obtidos ao compararmos mais do que duas seqüências na busca por seus genes.

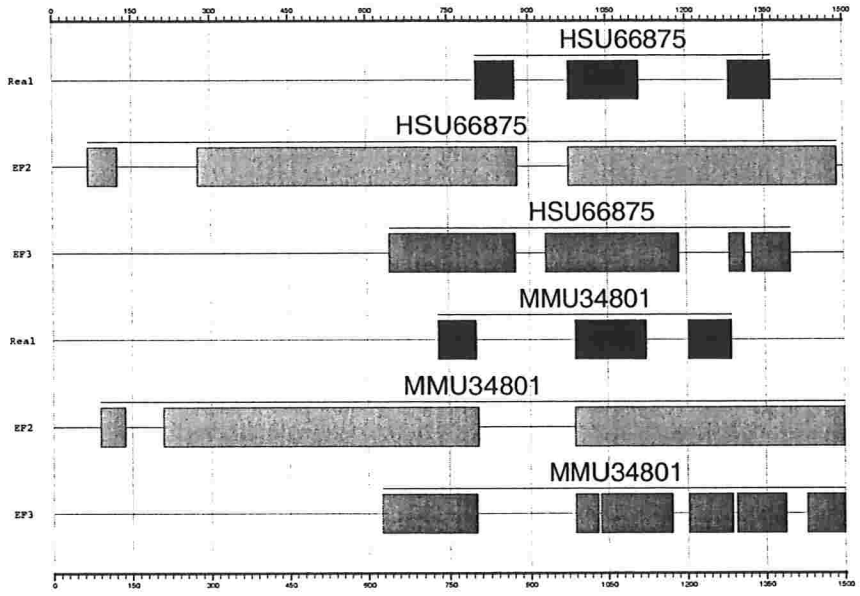

Figura 6.4: Representação gráfica das predições realizadas pelo EXON_FINDER2(EF2) e EXON_FINDER3(EF3) sobre as seqüências HSU66875 e MMU34801.

6.4 Comentários gerais

Neste capítulo apresentamos as idéias iniciais de uma nova estratégia para identificação de genes, baseda na comparação de várias seqüências. Essa estratégia foi implementada em um programa que chamamos de EXON_FINDER3 e possui como base uma tradicional aproximação para o problema do alinhamento múltiplo com pontuação SP. Apesar dos poucos testes realizados e dos resultados ainda insatisfatórios, pudemos observar que essa estratégia revela-se promissora como uma nova abordagem ao problema da identificação de genes. Aqui, como ocorrido no capítulo anterior, os testes foram realizados com seqüências bem próximas evolutivamente e, conseqüentemente, com alto

grau de conservação em regiões aparentemente não funcionais. Isso ocasiona a predição de alguns éxons adicionais pelo programa, o que determina sua baixa especificidade. Melhores resultados podem ser obtidos por meio do uso de informações adicionais na determinacão dos possíveis éxons e construção do grafo de montagem.

Capítulo 7

Conclusão

Neste trabalho abordamos um problema da biologia computacional ainda sem solução adequada. Dada uma seqüência genômica, estivemos interessados na identificação de seus genes. Dentre os vários métodos existentes para tratamento desse problema, focalizamos nossas atenções nos que comparam a seqüência de interesse com outra seqüência relacionada. Com base nos conceitos ligados a esses métodos, desenvolvemos três heurísticas distintas para o problema em questão e discutimos os resultados obtidos de suas implementações.

O Capítulo 4 inclui os detalhes de uma heurística que recebe como entrada uma seqüência genômica e um cDNA e, com base na busca por um caminho mínimo entre dois vértices de um grafo especial, identifica os possíveis éxons constituintes do gene da primeira seqüência. A implementação e testes com essa heurística demonstraram bons resultados no nível de nucleotídeos, que se justificam pelo uso de todos os possíveis éxons na busca pela cadeia de maior similaridade com o cDNA. Esse fato, porém, implica identificação de pequenos falsos éxons (efeito mosaico) em vários casos de testes, o que determina os resultados apenas regulares de nossa estratégia no nível de éxons. Apesar disso, a heurística proposta por nós no Capítulo 4 possui a vantagem de ter sido construída sobre em um problema tradicional da teoria dos grafos: o problema do caminho mínimo. Isso possibilita, dentre outras coisas, um tratamento direto de algumas das variantes do problema do alinhamento *spliced*, como a busca por todas as soluções ótimas para o problema ou por cadeias de blocos cuja similaridade com o cDNA esteja acima de um certo valor limite. Uma outra variante do problema do alinhamento *spliced* passível de ser atacada no futuro, e cujas soluções podem ser úteis na tarefa de identificação de genes, consiste em determinar uma cadeia de blocos da seqüência de interesse similar não apenas a uma, mas a várias seqüências de cDNAs.

No Capítulo 5 abordamos o problema da identificação de genes por comparação de duas seqüências genômicas evolutivamente relacionadas. A heurística que aqui desenvol-

vemos para esse problema possui como base um algoritmo de alinhamento que compara duas seqüências considerando o fato de elas serem constituídas de regiões bem e mal conservadas. Os testes realizados mostraram que nossa heurística é comparável com outro programa para identificação de genes que se utiliza do mesmo princípio. A principal dificuldade enfrentada por nossa estratégia está na distinção entre os verdadeiros éxons e as regiões aleatoriamente conservadas nas duas seqüências. Essa dificuldade torna-se mais evidente ao focalizarmos nossas atenções sobre as bordas das regiões codificantes. Nesse caso, implementações de melhoras na fase de pré-processamento, onde os possíveis sítios de aceitação e doação são determinados, surgem como possíveis trabalhos futuros sobre o tema. Além disso, a determinação de um conjunto de parâmetros adequados para as diversas classes de seqüências também está entre as possibilidades de pesquisas futuras. Por último, acreditamos que melhoras podem ser obtidas se considerarmos triplas de bases durante a comparação das seqüências e custos distintos para inserção e remoção de bases de acordo com a sua posição dentro dessas triplas.

Finalmente, no Capítulo 6 descrevemos as idéias principais de uma abordagem para o problema da identificação de genes baseada na comparação de várias seqüências. Esse é um dos primeiros trabalhos onde mais do que duas seqüências são comparadas na busca por seus genes. Aqui fizemos uso de uma aproximação bem conhecida para o problema do alinhamento múltiplo na busca pelas regiões conservadas das seqüências dadas como entrada. Essas regiões constituem-se no núcleo dos possíveis éxons a ser montados para determinação dos genes das seqüências. Apesar dos poucos testes realizados, os resultados obtidos mostraram-se promissores. Acreditamos que testes adicionais, envolvendo seqüências mais distantes, sejam necessários para a determinação e implementação de possíveis melhoras em nossa estratégia. De qualquer forma, a utilização de informações adicionais sobre a estrutura e composição dos genes sendo procurados é tida como passo certo na busca por melhores resultados.

Apesar dos resultados apresentados por nossas heurísticas mostrarem-se apenas comparáveis aos de outros programas de predição, acreditamos na importância de cada uma delas para a busca constante por soluções mais adequadas ao problema da identificação de genes. Como pode ser observado pelo que foi descrito nos parágrafos anteriores, várias são as melhoras possíveis de ser implementadas imediatamente em cada uma das heurísticas, sobretudo naquela que trata do problema da identificação por comparação de várias seqüências. Outras melhoras podem vir à tona com informações adicionais sobre novos genomas seqüenciados. Além disso, gostaríamos de salientar que nossas heurísticas acomodam de forma adequada a busca por outras regiões funcionais das seqüências. Elas podem ser extremamente úteis, por exemplo, na busca por genes que codificam outros tipos de elementos que não proteínas, além de possíveis regiões conservadas dentro dos íntrons. A busca por elementos desse tipo começa a se tornar importante e vislumbra como um problema potencial da Biologia Computacional.

Apêndice A

Tabelas com informações sobre as seqüências utilizadas para teste.

A.1 Identificação de genes por comparação DNA/cDNA

ID	TS	TA	NE	SM.
AB002059	28984	1359	12	100%
AB003730	2291	1830	1	96%
AB006987	6017	1515	9	86%
AB007546	11134	456	4	100%
AB007828	3300	978	1	84%
AB010281	1260	1173	3	100%
AB010874	1209	78	3	94%
AB012113	13236	270	3	100%
AB012668	3042	1029	2	78%
AB012922	16790	2007	18	100%
AB016243	14728	1014	7	84%
AB016492	4970	354	5	79%
AB018249	9163	363	3	100%
AB019534	8813	1005	7	84%
AB021866	3611	576	7	100%
AF001689	4111	393	5	77%
AF005058	5161	1059	2	97%
AF007189	1601	660	1	87%
AF007876	7894	873	7	90%
AF008216	5785	744	1	79%
AF009356	3653	609	5	87%
AF009962	7422	648	1	100%
AF013711	5388	606	4	91%
AF015224	4206	273	3	96%
AF015881	961	942	1	100%
AF015954	4031	1131	10	99%

ID	TS	TA	NE	SM
AF065988	11412	1059	2	89%
AF071216	4799	195	2	83%
AF076214	4002	681	3	100%
AF071552	1618	873	1	100%
AF071596	1693	582	1	100%
AF080237	4300	678	6	81%
AF084941	11822	846	4	99%
AF092047	4477	1014	2	94%
AF096303	5744	1257	10	89%
AF099730	2186	813	1	84%
AF099731	1810	816	1	80%
D38752	1020	615	1	90%
D67013	8748	1104	7	99%
D83195	4592	855	8	76%
D83956	3884	1089	7	95%
D84342	1822	606	4	91%
D88010	3270	456	6	99%
K02401	2301	654	5	88%
L13391	7345	636	5	89%
L24498	5378	498	4	90%
M15894	2740	654	5	97%
M30135	4379	435	5	100%
U01102	4995	276	3	100%
U07807	4839	158	3	82%
U08198	2344	609	7	82%
U17081	9009	380	4	86%

ID	TS	TA	NE	SM.	ID	TS	TA	NE	SM
AF016052	9700	1007	3	88%	Z25749	7513	585	6	99%
AF016697	1636	1005	2	100%	X14445	11608	738	3	80%
AF016898	17004	378	3	100%	X12671	5368	963	9	98%
AF017115	9219	510	4	100%	X14974	6325	1011	6	75%
AF019409	3828	930	6	89%	X15334	4200	1146	7	94%
AF019563	3877	342	5	100%	X52150	3637	1512	8	85%
AF022382	3997	1044	10	88%	X52851	6711	498	5	100%
AF026564	1581	957	1	100%	X55710	3840	1098	8	95%
AF027152	5116	1524	9	81%	X57152	5917	648	6	95%
AF028233	4575	864	3	91%	X62654	4370	717	7	100%
AF029081	10034	747	1	92%	X65921	2016	402	4	93%
AF031327	441	1059	1	98%	X66114	2518	945	7	90%
AF031426	1040	426	1	100%	X66133	2957	957	4	89%
AF035684	795	402	3	100%	X68677	4842	657	8	89%
AF032437	1968	1422	1	90%	X69907	9457	411	4	100%
AF036329	4498	345	3	87%	X76776	6933	774	6	82%
AF037062	6330	957	4	89%	X84707	3573	396	4	100%
AF037207	5491	372	5	83%	X86810	3339	780	6	78%
AF037372	2331	240	4	92%	Y00371	5408	1953	8	91%
AF037438	4079	786	6	100%	Y00477	5292	804	5	100%
AF039307	4322	942	2	94%	Y16791	5800	1215	7	76%
AF039704	7947	1692	13	85%	Z29491	995	816	4	86%
AF039954	6484	363	3	100%	U31468	2142	1047	1	91%
AF040714	2691	1200	2	86%	U44436	1164	1077	1	100%
AF042001	4034	807	3	90%	U47815	1232	744	1	100%
AF042782	3390	1119	2	84%	U53447	2282	1875	1	87%
AF043105	6566	678	8	100%	U55058	3371	336	3	81%
AF044311	4606	372	5	83%	U76254	1230	1146	1	86%
AF045999	5895	453	4	100%	U89486	923	396	3	100%
AF047383	3659	1104	10	90%	U96846	6965	477	4	98%
AF049259	5698	1314	7	81%	V00520	1964	654	5	90%
AF051160	11066	522	5	95%	V00536	5961	501	4	94%
AF052572	8747	1059	2	97%	V00695	3630	465	4	97%
AF053455	1408	807	1	91%	X02158	3398	579	5	93%
AF055080	2078	975	1	91%	X02619	1755	309	3	74%
AF055475	9531	354	4	100%	X02882	5691	753	5	82%
AF055903	5033	1131	10	100%	X03021	3043	435	4	97%
AF058293	8435	357	3	100%	X03656	2960	645	5	71%
AF058761	3607	420	2	83%	X03833	11970	816	6	93%
AF058762	3036	1119	2	84%	X03991	10050	543	5	91%
AF059734	2401	558	4	83%	X04143	1675	300	4	78%
AF060229	1377	615	2	100%	X07247	1282	534	4	100%
AF061327	1812	1299	2	82%	X07292	5198	1095	8	97%

Tabela A.1: Informações sobre as seqüências utilizadas para teste do EXON_FINDER1. (ID: identificação da seqüência genômica; TS: tamanho da seqüência genômica; TA: Tamanho da seqüência alvo; NE: # de éxons; SM: similaridade do RNAm utilizado com respeito ao CDS)

A.2 Identificação de genes por comparação DNA/DNA

Pair	Seq1	Seq2	Tam1	Tam2	NE
COXD	HSU66875	MM34801	1569	1901	3
ENOB	HSENO3	MMENO3G	7194	5472	11
H1T	HSHISH1T	MMHISTA	874	851	1
H4	HSHIS4	MMHIST4	1098	968	1
HS71	HSHSP70D	RNHSP70	2691	5918	1
KCRB	HSCKBG	MMCRKNB	4200	4521	7
MIF	HSMIF	MM20156	2167	920	3
MT3	S72043	S72046	2015	1845	3
PAP1	HSPAP1	MMD360	4497	4292	5
PSPA	HSSFTP1A	S48768	4732	4942	4
ROM1	HSROD1X	MMROM1X	2841	2787	3
RS7	HSRPS7	AF043285	7513	6637	6
SPEE	HSSPERSYN	MMSPERSYN	7623	3915	7
MIT1	HSCKIIBE	MMGMCK2B	5917	7874	6
MIT10	HSINT1G	MUSINT1A	4522	5607	4
MIT101	HSA6693	MUSSER1	3448	3366	1
MIT102	HUMIL2RGA	MMU21795	4038	5267	8
MIT103	HUMCRPGA	MMCRPG	2840	2140	2
MIT104	HSBCDIFFI	MMIL5G	3230	6727	4
MIT105	HUMTHY1A	MUSTHY1GC	2806	3257	3
MIT107	HUMSAP01	MUSSAPRB	1394	1350	2
MIT108	HUMPAP	D63360	4497	4292	5
MIT11	HUMSRI1A	MUSSRI1A	1634	1265	1
MIT110	HUMCAPG	MUSCATHG	3734	3438	5
MIT111	HUMCTLA1	MUSSPCTLS	4505	4348	5
MIT112	HSLACTG	MUSALCALB	3310	3045	4
MIT114	HSCD14G	MMCD14	1570	2404	2
MIT115	HSGAPIGNA	MMU60528	2609	5416	3
MIT116	HSBGPG	MUSOGC	1675	949	4
MIT12	HSMIMAR	MUSACHRM1	2100	1574	1
MIT13	HSFAU1	MUSFAUA	2016	2850	4
MIT14	HUMCRYABA	MUSALPBCRY	4206	4181	3
MIT17	HUMKCHN	MUSMK3A	2397	1994	1
MIT18	HUMPCNA	MMPCNAG	6340	4970	6
MIT19	HSU73304	MMU22948	5665	1654	1
MIT2	HUMSAACT	MUSACASA	3778	4007	7
MIT21	HUMNT3A	MMNT3	1029	1284	1
MIT23	HUMACHRM4	MMM4ACHR	2595	1707	1
MIT24	HUMMHHSP2	MUSHSP7A2	2876	3518	1
MIT25	HUMAPEXN	MUSAPEX	3730	4042	4
MIT26	HUMGAD45A	MUSGAD45	5378	3100	4
MIT3	HSH4EHIS	MMHIS412	859	637	1
MIT31	HSHIS10G	MMU18295	2530	2893	1
MIT32	HSCFOS	MMCFOS	3565	3967	4

Pair	Seq1	Seq2	Tam1	Tam2	NE
MIT33	HUMHGCR	MUS5HT1B	2635	2348	1
MIT34	HUMUDPCNA	MUSGLCNACT	4705	1894	1
MIT36	HUMGALTB	MMU41282	4286	4023	11
MIT39	HUMMIF	MMU20156	2167	920	3
MIT4	HSU12202	MMMRPS24	4942	5820	5
MIT40	AF049259	MMU13921	5698	4678	8

Tabela A.2: Informações sobre os pares utilizados para teste do EXON_FINDER2. (Par: identificação do par; Seq1: identificação da primeira seqüência; Seq2: identificação da segunda seqüência; Tam1: tamanho da primeira seqüência; Tam2: tamanho da segunda seqüência ; NE: # de éxons)

Par	Seq1	Seq2	coord1	coord2	NE1	NE2	Id total	Id CDS
1.	AE003408	AE003408	214956-221163	203720-210933	2	5	42.9%	54.2%
2.	AE003416	AE003416	100933-104500	104500-110062	3	2	34.6%	62.8%
3.	AE003416	AE003416	106794-112453	100933-106323	3	2	44.1%	56.3%
4.	AE003416	AE003416	108500-112453	103001-108500	3	3	29.6%	61.3%
5.	AJ010552	U02069	1-1610	1-2523	4	10	38.3%	61.5%
6.	U52111	AF133093	68058-80512	172101-183886	13	13	53.7%	87.9%
7.	U52111	AF133093	162492-170729	46755-54992	15	15	59.4%	76.6%
8.	AJ006216	AE003415	795-31501	126188-149227	48	31	42.2%	45.9%
9.	L44140	AC005203	141445-148305	219-6854	7	8	43.4%	47.3%
10.	U11869	U11870	1-2847	57-4452	1	1	40.9%	81.5%
11.	X65634	X65633	1-1633	1-1850	1	1	42.4%	51.8%
12.	U82671	U82670	3811-8755	105301-110230	1	1	69.2%	79.5%
13.	U82671	U82670	301590-306534	105301-110230	1	1	72.4%	80.8%
14.	U82671	U82671	301590-306534	3811-8755	1	1	75.1%	93.2%
15.	U82671	U82670	266231-271168	105301-110230	1	1	67.9%	80.7%
16.	U82671	U82671	266231-271168	3811-8755	1	1	67.5%	91.8%
17.	U82671	U82671	266231-271168	301590-306534	1	1	70.8%	92.5%
18.	AL021127	U82671	64412-99686	133084-159875	7	1	44.8%	82.3%
19.	Z84471	AF277315	1-1584	118330-136485	1	12	7.3%	84.8%
20.	Z84471	X55448	1-1584	1351-19503	1	12	7.3%	84.6%
21.	AJ222714	U11870	1-1674	57-4452	1	1	23.5%	52.8%
22.	AJ222714	U11869	1-1674	1-2847	1	1	33.5%	50.5%
23.	U26029	U26028	1-2091	1-2355	1	1	44.0%	49.3%
24.	U26030	U26028	1-3633	1-2355	1	1	37.1%	49.1%
25.	U26030	U26029	1-3633	1-2091	1	1	36.1%	48.7%
26.	M76739	AJ000497	1-2690	1-7020	1	18	24.9%	49.0%
27.	M89778	U06465	1-3046	1-3240	1	1	52.0%	55.5%
28.	U02280	L17040	1-1956	1-2333	2	1	46.8%	53.7%

Tabela A.3: Informações sobre os pares utilizados para teste do UTOPIA e PROGEN. (Par: identificação do par; Seq1: identificação da primeira seqüência; Seq2: identificação da segunda seqüência; coord1: coordenadas da primeira seqüência; coord1: coordenadas da segunda seqüência; NE1: # éxons na primeira seqüência; NE2: # éxons na segunda seqüência; Id total: identidade entre as seqüências; Id CDS: identidade dos respctivos CDS).

Referências Bibliográficas

[1] J. F. Abril and R. Guigó. gff2ps: visualizing genomic annotations. *Bioinformatics*, 16(8):743–744, 2000.

[2] S. S. Adi and C. E. Ferreira. Uma avaliação de ferramentas de predição de genes. In *Proceedings do Congresso da Sociedade Brasileira de Computação*, pages 133–143, Florianópolis-SC, 2002.

[3] S. S. Adi and C. E. Ferreira. Gene prediction by spliced alignment. http://www.vision.ime.usp.br/~cesar/programa/pdf/95.pdf, 2003.

[4] S. S. Adi and C. E. Ferreira. An experimental evaluation of similarity-based gene prediction tools. submitted, 2004.

[5] S. S. Adi and C. E. Ferreira. Gene prediction by syntenic alignment. submitted, 2005.

[6] S. S. Adi and C. E. Ferreira. A graph theoretical model for the gene prediction problem. submitted, 2005.

[7] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J. D. Watson. *Molecular Biology of the Cell.* Garland Publishing, Inc., 3 edition, 1994.

[8] M. Alexandersson, S. Cawley, and L. Pachter. SLAM: cross-species gene finding and alignment with a generalized pair hidden markov model. *Genome Research*, 13(3):496–502, 2003.

[9] J. E. Allen, M. Pertea, and S. L. Salzberg. Computational gene prediction using multiple sources of evidence. *Genome Research*, 14(1):142–148, 2004.

[10] H. Almagor. Nucleotide distribution and the recognition of coding regions in DNA sequences: an information theory approach. *Journal of Theoretical Biology*, 117:127–136, 1985.

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

[12] J. A. Azevedo, M. E. Costa, and J. J. Madeira. An algorithm for the ranking of shortest paths. *European Journal of Operational Research*, 69:97–106, 1993.

[13] V. Bafna and D. Huson. The conserved exon method for gene finding. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 3–12, 2000.

[14] S. Batzoglou, L. Pachter, J. Mesirov, B. Berger, and E. S. Lander. Human and mouse gene structure: comparative analysis and application to exon prediction. *Genome Research*, 10:950–958, 2000.

[15] D. A. Benson, I. K.-Mizrachi, D. J. Lipman, J. Ostell, B. A. Rapp, and D. L. Wheeler. GenBank. *Nucleic Acids Research*, 30(1):17–20, 2002.

[16] A. Bernal, U. Ear, and N. Kyrpides. Genomes online database (GOLD): a monitor of genome projects world-wide. *Nucleic Acids Research*, 29(1):126–127, 2001.

[17] E. Birney, M. Clampa, and R. Durbin. Genewise and genomewise. *Genome Research*, 14(5):988–995, 2004.

[18] E. Birney and R. Durbin. Dynamite: a flexible code generating language for dynamic programming methods used in sequence comparison. *Intell. Syst. Mol. Biol.*, 5:56–64, 1997.

[19] P. Blayo, P. Rouzé, and Marie-France Sagot. Orphan gene finding - an exon assembly approach. *Theoretical Computer Science*, 290(3):1407–1431, 2003.

[20] J. A. Bondy and U.S. R. Murty. *Graph Theory with Applications*. MacMillan, 1976.

[21] M. R. Brent and R. Guigó. Recent advances in gene structure prediction. *Current Opinion in Structural Biology*, 14(3):264–272, 2004.

[22] S. Brunak, J. Engelbrecht, and S. Knudsen. Prediction of human mRNA donor and acceptor sites from the DNA sequence. *Journal of Molecular Biology*, 220:49–65, 1991.

[23] P. Bucher. Weight matrix descriptions of four eukariotic RNA polymerase II promoter elements derived from 502 unrelated promoter sequences. *Journal of Molecular Biology*, 212:563–578, 1990.

[24] C. B. Burge and J. S. Karlin. Finding the genes in genomic DNA. *Current Opinion in Structural Biology*, 8(3):346–354, 1998.

[25] M. Burset and R. Guigó. Evaluation of gene structure prediction programs. *Genomics*, 34(3):353–367, 1996.

[26] R. Castelo and R. Guigó. Splice site identification by idlBNs. *Bioinformatics*, 20:i69–i76, 2004.

[27] J. S. Chuang and D. Roth. Gene recognition based on dag shortest paths. *Bioinformatics*, 1:1–9, 2001.

[28] J. M. Claverie. Computational methods for the identification of genes in vertebrate genomic sequences. *Human Molecular Genetics*, 6(10):1735–1744, 1997.

[29] J.-M. Claverie, I. Sauvaget, and L. Bougueleret. K-tuple frequency analysis: from intron/exon discrimination to t-cell epitope mapping. *Methods in Enzymology*, 183:237–252, 1990.

[30] Mouse Genome Sequencing Consortium. Initial sequencing and comparative analysis of the mouse genome. *Nature*, 420:520–562, 2002.

[31] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[32] S. Dong and D.B. Searls. Gene structure prediction by linguistic methods. *Genomics*, 23:540–551, 1994.

[33] L. Duret, D. Mouchiroud, and C. Gautier. Statistical analysis of vertebrate sequences reveals that long genes are scarce in gc-rich isochores. *Journal of Molecular Evolution*, 40(3):308–317, 1995.

[34] D. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25:351–360, 1987.

[35] A. B. H. Ferreira. *Novo Aurélio Século XXI: o dicionário da língua portuguesa.* Editora Nova Fronteira, 1999.

[36] J. W. Ficket and C-S. Tung. Assessment of protein coding measures. *Nucleic Acids Research*, 20(24):6441–6450, 1992.

[37] J.W. Fickett. Recognition of protein coding regions in DNA sequences. *Nucleic Acids Research*, 10:5303–5318, 1982.

[38] C. A. Fields and C. A. Soderlund. GM: a practical tool for automating DNA sequence analysis. *Computer Applications in Biosciences*, 6:263–270, 1990.

[39] L. Florea, G. Hartzell, Z. Zhang, G. Rubin, and W. Miller. A computer program for aligning a cDNA sequence with a genomic sequence. *Genome Research*, pages 967–974, 1998.

[40] S. Foissac, P. Bardou, A. Moisan, M.-J. Cros, and T. Schiex. EugÈne'hom: a generic similarity-based gene finder using multiple homologous sequences. *Nucleic Acids Research*, 31(13):3742–3745, 2003.

[41] M. S. Gelfand. Prediction of function in DNA sequence analysis. *Journal of Computational Biology*, 2(1):87–117, 1995.

[42] M. S. Gelfand, A. A. Mironov, and P. A. Pevzner. Gene recognition via spliced sequence alignment. In *Proceedings of the National Academy of Sciences*, pages 9061–9066, USA,1993.

[43] O. Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162:705–708, 1982.

[44] R. Guigó, P. Agarwal, J. F. Abril, M. Burset, and J. W. Fickett. An assessment of gene prediction accuracy in large DNA sequences. *Genome Research*, 10(10):1631–1642, 2000.

[45] D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bulletin of Mathematical Biology*, 55(1):141–154, 1993.

[46] D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology.* Cambridge University Press, 1997.

[47] P. Hooper, H. Zhang, and D. Wishart. Prediction of genetic structure in eukaryotic DNA using reference point logistic regression and sequence alignment. *Bioinformatics*, 16:425–438, 2000.

[48] X. Huang. On global sequence alignment. *Computer Applications in the Biosciences*, 10:227–235, 1994.

[49] X. Huang, M. D. Adams, H. Zhou, and A. R. Kerlavage. A tool for analyzing and annotating genomic sequences. *Genomics*, 46(1):37–45, 1997.

[50] X. Huang and K. M. Chao. A generalized global alignment algorithm. *Bioinformatics*, 19(2):228–233, 2003.

[51] N. F. Almeida Jr., J. C. Setubal, and M. Tompa. On the use of don't care regions for protein sequence alignment. Technical Report 99-07, Instituto de Computação - Universidade de Campinas, março 1999.

[52] A. K. Knopka. *Structure and Methods: VI. Human Genome Initiative and DNA Recombination*, chapter Towards Mapping Functional Domains in Indiscriminantly Sequenced Nucleic Acids: A Computational Approach. Adenine Press, Guilderland, New York, 1994.

[53] I. Korf, P. Flicek, D. Duan, and M. R. Brent. Integrating genomic homology into gene structure prediction. *Bioinformatics*, 1(1):S1–S9, 2001.

[54] D. Kulp, D. Haussler, M. G. Reese, and F. H. Eeckman. A generalized hidden markov model for the recognition of human genes in DNA. In *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, volume 4, pages 134–142, 1996.

[55] B. Lewin. *Genes VII.* Oxford University Press, 1999.

[56] W. Li, T. G. Mart, and K. Kaneko. Understanding long-range correlations in DNA sequences. *Physica D*, 75:392–416, 1994.

[57] E. Q. Martins and J. L. Santos. A new shortest paths ranking algorithms. *Investigação Operacional*, 20:47–62, 2000.

[58] C. Mathé, M.-F. Sagot, T. Schiex, and P. Rouzé. Current methods of gene prediction, their strengths and weaknesses. *Nucleic Acids Research*, 30(19):4103–4117, 2002.

[59] P. McCaldon and P. Argos. Oligopeptide biases in protein sequences and their use in predicting protein coding regions in nucleotide sequences. *Proteins: Structure, Function, and Genetics*, 4:99–122, 1988.

[60] J. Meidanis and J. C. Setubal. *Introduction to Computational Molecular Biology.* PWS Publishing Co., Boston, 1997.

[61] I. M. Meyer and R. Durbin. Gene structure conservation aids similarity based gene prediction. *Nucleic Acids Research*, 32(2):776–783, 2004.

[62] L. Milanesi, N. A. Kolchanov, and I. B. Rogozin. A computing system for protein-coding regions prediction in diptera nucleotide sequences. *Drosophila Information Service*, 76:185–187, 1995.

[63] L. Milanesi, N. A. Kolchanov, I. B. Rogozin, I. V. Ischenko, A. E. Kel, Y.L. Orlov, M. P. Ponomarenko, and P. Vezzoni. GenViewer: a computing tool for protein-coding regions prediction in nucleotide sequences. In *Proceedings of the 2nd. International Conference on Bioinformatics, Supercomputing and Complex Genome Analysis*, pages 573–587, 1992.

[64] A. A. Mironov, J. W. Fickett, and M. S. Gelfand. Frequent alternative splicing of human genes. *Genome Research*, 9:1288–1293, 1999.

[65] R. Mott. Est_genome: a program to align spliced DNA sequences to unspliced genomic dna. *CABIOS*, 13(4):477–478, 1997.

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

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

[68] P. Novichkov, M. S. Gelfand, and A. A Mironov. Prediction of the exon-intron structure by comparison of genomic sequences. *Mol. Biol. (Mosk.)*, 34:230–236, 2000.

[69] G. Parra, P. Agarwal, J. F. Abril, T. Wiehe, J. W. Fickett, and R. Guigó. Comparative gene prediction in human and mouse. *Genome Research*, 13(1):108–117, 2003.

[70] G. Perrière, C. Combet, S. Penel, C. Blanchet, J. Thioulouse nd C. Geourjon, J. Grassot, C. Charavay, M. Gouy, L. Duret, and G. Deléage. Integrated databanks access and sequence/structure analysis services at the PBIL. *Nucleic Acids Research*, 31(13):3393–3399, 2003.

[71] P. A. Pevzner. *Computational Molecular Biology An Algorithmic Approach.* The Mit Press, 2000.

[72] O. Rinner and B. Morgenstern. AGenDA: gene prediction by comparative sequence analysis. *In Silico Biology*, 2(3):195–205, 2002.

[73] S. Rogic, A. K. Mackworth, and F. B. Ouellette. Evaluation of gene-finding programs on mammalian sequences. *Genome Research*, 11(5):817–832, 2001.

[74] A. A. Salamov and V. V. Solovyev. Ab initio gene finding in drosophila genomic DNA. *Genome Research*, 10:516–522, 2000.

[75] S. L. Salzberg. A method for identifying splice sites and translational start sites in eukaryotic mRNA. *Computer Applications in the Biosciences*, 13(4):365–376, 1997.

[76] S. L. Salzberg and D. B. Searls. *Computational Methods in Molecular Biology*, chapter An introduction to hidden Markov models for biological sequences, pages 46–63. Elsevier, Amsterdam, The Netherlands, 1998.

[77] G. D. Schuler, E. S. Lander, and T. J. Hudson. A gene map for the human genome. *Science*, 274:540–546, 1996.

[78] J. C. W. Shepherd. Method to determine the reading frame of a protein from the purine/pyrimidine genome sequence and its possible evolutionary justification. In *Proceedings of National Academy of Sciences*, volume 78, pages 1596–1600, 1981.

[79] A. Siepel and D. Haussler. Computational identification of evolutionarily conserved exons. In *Proceedings of the eighth annual international conference on Computational molecular biology*, pages 177–186, San Diego, California, USA, 2004.

[80] B. D. Silverman and R. Linsker. A measure of DNA periodicity. *Journal of Theoretical Biology*, 118:295–300, 1986.

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

[82] E.E. Snyder and G.D. Stormo. Identification of coding regions in genomic DNA sequences: an application of dynamic programming and neural networks. *Nucleic Acids Research*, 21:607–613, 1993.

[83] V. V. Solovyev, A. A. Salamov, and C.B. Lawrence. Predicting internal exons by oligonucleotide composition and discriminant analysis of spliceable open reading frames. *Nucleic Acids Research*, 22:5156–5163, 1994.

[84] V. V. Solovyev, A. A. Salamov, and C.B. Lawrence. Identification of human gene structure using linear discriminant functions and dynamic programming. In *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, pages 367–375, 1995.

[85] R. Staden. Computer methods to locate signals in nucleic acid sequences. *Nucleic Acids Research*, 12:505–519, 1984.

[86] R. Staden. Measurements of the effects that coding for a protein has on a DNA sequence and their use for finding genes. *Nucleic Acid Research*, 12:505–519, 1984.

[87] R. Staden and A. McLachlan. Codon preference and its use in identifying protein coding regions in long DNA sequences. *Nucleic Acids Research*, 10:141–156, 1982.

[88] R. Tavares, S. S. Adi, P. Blayo, and M.-F. Sagot. An exact generic core algorithm for the prediction of the gene structure of close and distant gene family members. submitted.

[89] International Human Genome Sequencing Consortium (E. S. Lander *et al.*). Initial sequencing and analysis of the human genome. *Nature*, 409:860–921, 2001.

[90] J. Usuka, W. Zhu, and V. Brendel. Optimal spliced alignment of homologous cDNA to a genomic DNA template. *Bioinformatics*, 16:203–211, 2000.

[91] J . C. Venter. A new strategy for genome sequencing. *Nature*, 381:364–366, 1996.

[92] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions Information Theory*, IT-13:260–269, 1967.

[93] J. D. Watson and F. H. C. Crick. Molecular structure of nucleic acids. *Nature*, 171:737–738, 1953.

[94] T. Wiehe, S. Gebauer-Jung, T. Mitchell-Olds, and R. Guigó. SGP-1: prediction and validation of homologous genes based on sequence alignments. *Genome Research*, 11:1574–1583, 2001.

[95] Y. Xu, R. J. Mural, and E. C. Uberbaker. Constructing gene models from accurately predicted exons: an application of dynamic programming. *Computer Applications in the Biosciences*, 10:613–623, 1994.

[96] G. Yeo and C. B. Burge. Maximum entropy modeling of short sequence motifs with applications to rna splicing signals. In *Proceedings of the seventh annual international conference on Computational molecular biology*, pages 322–331, 2003.

[97] M. Q. Zhang and T. G. Marr. A weight array model for splicing signal analysis. *Computer Applications in the Biosciences*, 9:499–509, 1993.