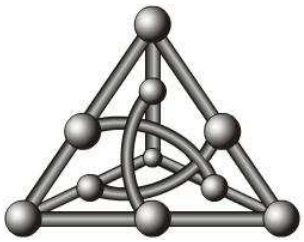
Identificação de Genes por Comparação de DNAs

Ronaldo Fiorilo dos Santos

DISSERTAÇÃO APRESENTADA
À
FACULDADE DE COMPUTAÇÃO
DA
UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
PARA
OBTENÇÃO DO GRAU DE MESTRE
EM
CIÊNCIA DA COMPUTAÇÃO



Área de Concentração: Ciência da Computação
Orientador: Prof. Dr. Said Sadique Adi

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

— Campo Grande, dezembro de 2010 —

Identificação de Genes por Comparação de DNAs

Este exemplar corresponde à redação final
da dissertação devidamente corrigida e
defendida por Ronaldo Fiorilo dos Santos
e aprovada pela banca examinadora.

Campo Grande, 15 de dezembro de 2010.

Banca examinadora:

- Prof. Dr. Said Sadique Adi (orientador) (FACOM-UFMS)
- Prof. Dr. Carlos Eduardo Ferreira (IME-USP)
- Prof. Dr. Fábio Henrique Viduani Martinez (FACOM-UFMS)

Aos meus verdadeiros mestres,
Antonio Ap. Mansilhas dos Santos
e Ap. de Fátima Fiorilo dos Santos.

Agradecimentos

Primeiramente agradeço a Deus por ter colocado em meu caminho todas as pessoas que contribuiram direta ou indiretamente no desenvolvimento desse trabalho e também por tudo que Ele vem realizando em minha vida.

Um agradecimento especial ao meu orientador, professor Dr. Said Sadique Adi, que me apresentou à Bioinformática ainda na graduação e por toda atenção, paciência e incentivos dedicados desde essa época até a conclusão deste trabalho de mestrado.

Aos meus colegas, dos mais antigos aos mais recentes, que proporcionaram momentos de descontração seja em Campo Grande ou em Rinópolis e região. Não há como listar todos aqui, correndo o risco de ser injusto esquecendo alguém ou colocar alguém não merecido por indução. Abro uma exceção para os companheiros Lucas Fermino (Frango) e Rodrigo Kishi (Zé) que, desde a graduação, fizeram companhia durante as aulas e durante as noites passadas em claro estudando.

Com um carinho imenso, agradeço meus tios Dorival e Terezinha por me acolherem logo quando cheguei à Campo Grande e pelo tratamento de filho que tive durante os anos que residi com eles. Assim também como não posso deixar de agradecer todos tios, primos e avós que torceram pelo meu sucesso.

Por fim, mas a quem mais devo agradecimentos nesse momento, as três pessoas essenciais na minha vida. Simples agradecimentos não são suficientes para essas pessoas que foram quem mais me apoiaram durante toda minha trajetória e, mesmo a quilômetros de distância, sempre se fizeram presentes com palavras de incentivo e conforto. Aos meus pais, que fazem o possível e o impossível pelos filhos, não agradeço por terem me dado a vida, mas sim por cada "puxão de orelha", por cada palavra de incentivo e por cada manifestação de amor que me fizeram chegar até aqui. Ao meu irmão por cada risada ao seu lado, por cada briga em que um ficava de "cara virada" pro outro (por no máximo meia hora :P), pela cumplicidade que deve haver entre irmãos e por cada momento que passamos juntos e que agora estão cada vez mais raros, devido aos caminhos que a vida nos levaram. Antonio Ap. Mansilhas dos Santos, Ap. de Fátima Fiorilo dos Santos e Rafael Fiorilo dos Santos: Amo vocês!

Ronaldo Fiorilo dos Santos

Resumo

O Problema da Identificação de Genes consiste na busca pelas regiões codificantes presentes em uma sequência de DNA. Uma das formas de abordar esse problema é através da comparação entre sequências de DNA próximas evolutivamente. Neste trabalho realizamos um estudo detalhado do problema neste contexto comparativo, propondo uma formulação matemática para ele. Com base nessa formulação, desenvolvemos um algoritmo baseado em programação dinâmica que foi implementado em uma nova ferramenta de identificação de genes. Ainda como parte deste trabalho, confrontamos o desempenho dessa ferramenta com o de outras ferramentas de predição, também baseadas no método de comparação de sequências. Nessa avaliação experimental, constatamos que a ferramenta desenvolvida por nós superou as outras em praticamente todos os níveis de avaliação.

Palavras-chave: Biologia Computacional, Otimização Combinatória, Predição de Genes, Comparação de Sequências.

Abstract

The Gene Prediction Problem can be defined as the search for the coding regions of a DNA sequence. One approach for this problem is based on the comparison of evolutionary related DNA sequences. In this work we performed a study of the problem in this comparative context, proposing a mathematical formulation for it. Based on this formulation, we developed a dynamic programming algorithm that was coded in a new gene prediction tool. Also as part of this work, we compared the performance of our tool with others comparison based gene prediction programs. In this experimental evaluation, the tool developed by us surpassed the others in almost all levels of evaluation.

Keywords: Computational Biology, Combinatorial Optimization, Gene Prediction, Sequence Comparison.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Considerações preliminares

A unidade básica de todo organismo vivo é a célula. É no núcleo das células que está presente a molécula de DNA, encarregada de coordenar o desenvolvimento e o funcionamento do organismo. Uma das regiões mais importantes do DNA são os genes, dado o fato de codificarem as informações necessárias para a síntese de proteínas. Dada a importância dos genes, há um grande interesse por parte da comunidade científica em localizá-los dentro de uma sequência de DNA. O problema relacionado a essa tarefa recebe o nome de Problema da Identificação de Genes e corresponde ao tema principal deste trabalho.

O Problema da Identificação de Genes consiste em, dada uma sequência de DNA, determinar a localização de cada um dos seus genes. Essa tarefa tem aplicação direta em áreas como a medicina (na análise do DNA do paciente para escolha do medicamento mais eficiente e que não cause reações adversas) e a agricultura (na identificação de genes que tornam uma planta imune a determinados insetos), por exemplo.

Existem hoje, na literatura, várias ferramentas desenvolvidas com o propósito de identificar genes. Os métodos nos quais elas se baseiam vão desde a observação de características intrínsecas à sequência de DNA de interesse até a comparação dessa sequência com outra(s) sequência(s) similar(es). Mesmo com um número considerável de métodos desenvolvidos, os resultados obtidos pelas ferramentas de identificação de genes ainda estão aquém do esperado devido a vários fatores. Primeiramente, os genes são compostos de várias regiões distintas. Além disso, os genes correspondem apenas a uma pequena porção da sequência analisada, além de não existir nenhum padrão de bases que os diferencie das outras regiões da sequência. Por último, vale salientar que as sequências de DNA são extensas o que demanda tempo considerável para serem processadas.

Uma informação que pode ser utilizada no processo de identificação de genes é que essas regiões tendem a se conservar ao longo do processo evolutivo das espécies devido à relação entre a funcionalidade dos genes e das bases que os constituem. Disso, um método bastante utilizado para a identificação de genes baseia-se na comparação entre sequências. A idéia geral desse método é identificar, dentro da sequência de DNA de interesse, regiões significativamente parecidas com sequências representativas de genes

já conhecidos. Uma variante desse método envolve a comparação da sequência de DNA sendo analisada com outra sequência homóloga a ela. Alguns estudos apontam o método de predição de genes por comparação de sequências como o mais promissor para obtenção de melhores resultados. Isso justifica estudos complementares nesse campo, com o desenvolvimento de novos métodos e ferramentas de predição baseadas em comparação de sequências.

1.2 Objetivos

Nosso principal objetivo neste trabalho é estudar o problema da identificação de genes no intuito de desenvolver soluções computacionais precisas e eficientes para ele. Mais especificamente, estamos interessados no desenvolvimento de uma ou mais ferramentas computacionais de predição de genes baseadas na comparação de um DNA com outro DNA.

1.3 Contribuições

As principais contribuições deste trabalho são:

- Formulação de um problema de Otimização Combinatória, denominado Problema do Alinhamento de Pares de Segmentos, que modela o problema de identificação de genes.
- Desenvolvimento de um algoritmo que soluciona o problema do alinhamento de pares de segmentos.
- Desenvolvimento de uma nova ferramenta de predição de genes por comparação de sequências, baseada no algoritmo para o problema do alinhamento de pares de segmentos.
- Criação de um conjunto de instâncias de testes para a avaliação de ferramentas de predição baseadas na comparação de sequências.
- Um estudo comparativo entre ferramentas de predição de genes existentes.

1.4 Organização do trabalho

Este texto está dividido em seis capítulos, incluindo esta introdução. No segundo capítulo apresentamos alguns conceitos básicos que devemos ter em mente para entender o problema em questão e a proposta de solução apresentada para ele. Esse capítulo abrange os conteúdos de Biologia Molecular, Ciência da Computação e Bioinformática. No terceiro capítulo, o problema da identificação de genes é formalmente definido e são apresentadas algumas técnicas desenvolvidas na tentativa de solucioná-lo. No quarto e no quinto capítulo apresentamos, respectivamente, a estratégia que utilizamos para

abordar o problema da identificação de genes e os resultados obtidos por nossa ferramenta, comparados com o resultado de outras ferramentas. Por fim apresentamos, no sexto capítulo, algumas considerações finais sobre nosso estudo.

Capítulo 2

Conceitos Básicos

Para um melhor entendimento do problema apresentado nesta dissertação e das técnicas utilizadas para resolvê-lo, é necessário compreender algum dos conceitos básicos das grandes áreas nas quais nosso trabalho está inserido. Sendo assim, este capítulo está dividido em três seções, sendo a primeira dedicada aos conceitos de Biologia Molecular, retirados basicamente de [49, 30, 16, 37, 35], a segunda dedicada aos conceitos de Ciência da Computação, retirados basicamente de [17, 27, 58, 52], e a terceira dedicada à Bioinformática, escrita com base em [9].

2.1 Conceitos básicos de Biologia Molecular

A Biologia é o ramo da ciência que estuda a vida. Ela examina a origem, estrutura, função e evolução dos seres vivos, classificando e descrevendo as várias formas de organismos e detalhando as interações entre eles e o ambiente em que vivem. Dentre as diversas áreas da Biologia está a Biologia Molecular, que procura explicar as interações entre os diversos sistemas celulares, partindo da relação entre DNA, RNA e síntese de proteínas.

2.1.1 A célula

A **célula** é a unidade estrutural e funcional dos seres vivos [49]. É possível agrupar as células em duas categorias principais: procariontes e eucariontes. As células **procariontes** são as mais simples, não apresentando um envoltório nuclear. Por outro lado, as células **eucariontes** possuem um núcleo, no qual está contido o material genético. Apesar das diferenças entre as duas categorias de células, elas apresentam grandes semelhanças funcionais e organizacionais.

A célula é composta basicamente por água, sais inorgânicos e alguns compostos orgânicos. Dentre estes compostos orgânicos encontramos os ácidos nucleicos, carboidratos, lipídios e proteínas. Na próxima seção discorreremos um pouco mais sobre os ácidos nucleicos e as proteínas.

2.1.2 Ácidos nucleicos e proteínas

Na natureza existem dois tipos de ácidos nucleicos: o ácido desoxirribonucleico e o ácido ribonucleico, comumente referenciados como DNA (do inglês, *DeoxyriboNucleic Acid*) e RNA (do inglês, *RiboNucleic Acid*), respectivamente. Os **ácidos nucleicos** são polímeros[1] de **nucleotídeos** que, por sua vez, são formados por um monossacarídeo[2] com 5 átomos de carbono (pentose), uma base nitrogenada e um grupo fosfato. A pentose pode ser de dois tipos: desoxirribose (no DNA) ou ribose (no RNA). A principal diferença entre esses dois monossacarídeos é que a desoxirribose tem um átomo de oxigênio a menos. As bases nitrogenadas presentes nos nucleotídeos são: adenina (A), citosina (C), guanina (G), timina (T) e uracila (U). No DNA estão presentes a adenina, citosina, guanina e timina, enquanto que no RNA estão presentes a adenina, citosina, guanina e uracila. Os fosfatos unem o carbono 3' da pentose de um nucleotídeo ao carbono 5' da pentose do nucleotídeo seguinte. A extremidade da molécula que contém a pentose com o carbono 3' livre (sem ligação com um fosfato) é chamada extremidade 3' e aquela que possui o carbono 5' livre recebe o nome de extremidade 5'. A ligação entre o carbono 3' de um nucleotídeo e o carbono 5' do próximo nucleotídeo induz uma orientação nas moléculas de ácido nucleico. Essa orientação é denotada por 5' → 3' e indica que o ácido começa na extremidade 5' e termina na extremidade 3'. Na Figura 2.1 podemos ver um trecho de uma molécula de DNA onde as pentoses são representadas por pentágonos, os fosfatos por losangos e as bases nitrogenadas por hexágonos com a inicial de cada base em seu interior.

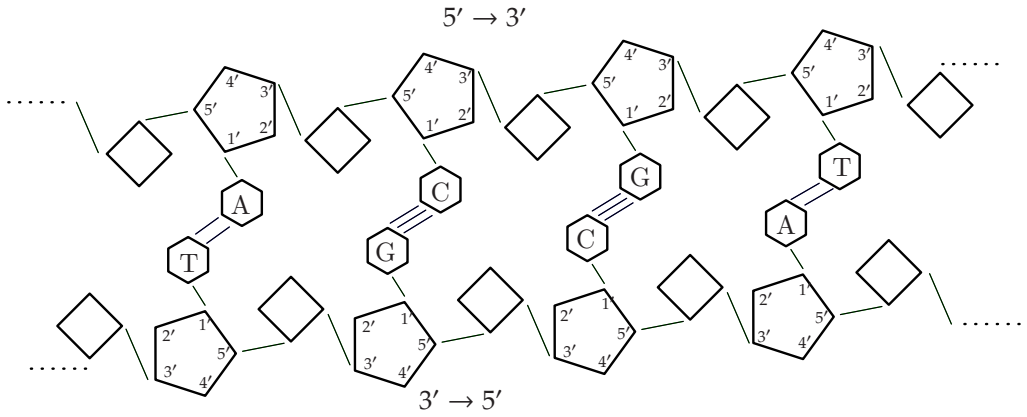


Figura 2.1: Trecho de uma molécula de DNA.

O DNA é formado por duas cadeias antiparalelas[3] de ácidos nucleicos, que formam uma dupla hélice em torno de um eixo imaginário central. Ambas as cadeias estão interligadas por meio de pontes de hidrogênio estabelecidas entre pares de bases específicas. Esses pares de bases, chamados de **bases complementares**, são: adenina-timina

[1] Polímeros são macromoléculas constituídas de várias (poli) unidades (meros = partes) estruturais menores.

[2] Monossacarídeos são os carboidratos mais simples e apresentam de 3 a 7 carbonos em sua estrutura.

[3] Cadeias com orientações opostas.

(A-T, T-A) e citosina-guanina (C-G, G-C). Devido a essa complementaridade entre as bases, podemos dizer que as duas cadeias do DNA também são complementares, o que implica que cada uma delas pode servir como um molde para uma nova cadeia complementar. Ao conjunto de moléculas de DNAs de um organismo damos o nome de **genoma**. Detalhes da estrutura de uma molécula de DNA podem ser vistos na Figura 2.2.

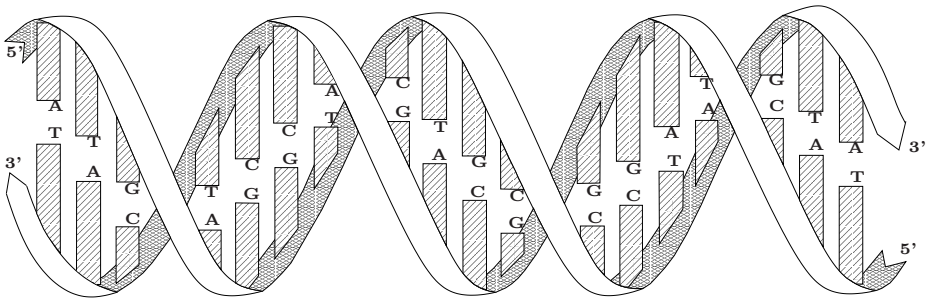


Figura 2.2: Dupla hélice do DNA, indicando as ligações entre as bases.

A estrutura do RNA é semelhante à do DNA, exceto pela presença da ribose ao invés da desoxirribose, da base uracila no lugar da timina e pelo fato da molécula de RNA ser formada por apenas uma cadeia de nucleotídeos. Existem três tipos principais de RNA: O RNA mensageiro (RNAm), o RNA ribossomal (RNAr) e o RNA transportador (RNAt). Os três atuam no processo de síntese de proteínas. De forma resumida, o RNAm é responsável por carregar a informação genética, copiada do DNA, que estabelecerá a sequência dos aminoácidos na proteína; o RNAr é o principal componente dos ribossomos, local onde ocorre de fato a síntese de proteínas; e, por fim, o RNAt identifica e carrega os aminoácidos para os ribossomos. O processo de síntese de proteínas está descrito em detalhes mais adiante.

As **proteínas** são cadeias de **aminoácidos** que, por sua vez, são ácidos orgânicos no qual o carbono, chamado de carbono alfa (C_α), está ligado a um grupo carboxila ($-COOH$), a um grupo amina ($-NH_2$) e a uma molécula de hidrogênio (H). Além disso, esse mesmo carbono está ligado a um radical (R), que varia de acordo com o tipo do aminoácido. Cada aminoácido é codificado, no RNAm, por uma tripla de bases que recebe o nome de **códon**. Tendo em vista que existem 4 tipos de bases, o número de códons possíveis é 64 ($4^3 = 64$). A relação entre cada um desses 64 códons com os aminoácidos que eles codificam recebe o nome de **código genético** e pode ser visto na Tabela 2.1.

Analisando a tabela do código genético, podemos verificar que, com exceção da metionina e do triptofano, todos os aminoácidos podem ser codificados por mais de um códon. Os códons que codificam um mesmo aminoácido são ditos **códons sinônimos**. Dentre os códons possíveis, podemos identificar três deles que não especificam um aminoácido: TAA, TAG e TGA. Esses códons são conhecidos como **códons de parada** e indicam a terminação da síntese de uma cadeia de aminoácidos.

				(Ala/A) Alanina
AAA (Lys/K)	ACA (Thr/T)	AGA (Arg/R)	ATA (Ile/I)	(Arg/R) Arginina
AAC (Asn/N)	ACC (Thr/T)	AGC (Ser/S)	ATC (Ile/I)	(Asn/N) Asparagina
AAG (Lys/K)	ACG (Thr/T)	AGG (Arg/R)	ATG (Met/M)	(Asp/D) Ácido aspártico
AAT (Asn/N)	ACT (Thr/T)	AGT (Ser/S)	ATT (Ile/I)	(Cys/C) Cisteína
				(Gln/Q) Glutamina
CAA (Gln/Q)	CCA (Pro/P)	CGA (Arg/R)	CTA (Leu/L)	(Glu/E) Ácido glutâmico
CAC (His/H)	CCC (Pro/P)	CGC (Arg/R)	CTC (Leu/L)	(Gly/G) Glicina
CAG (Gln/Q)	CCG (Pro/P)	CGG (Arg/R)	CTG (Leu/L)	(His/H) Histidina
CAT (His/H)	CCT (Pro/P)	CGT (Arg/R)	CTT (Leu/L)	(Ile/I) Isoleucina
				(Leu/L) Leucina
GAA (Glu/E)	GCA (Ala/A)	GGA (Gly/G)	GTA (Val/V)	(Lys/K) Lisina
GAC (Asp/D)	GCC (Ala/A)	GGC (Gly/G)	GTC (Val/V)	(Met/M) Metionina
GAG (Glu/E)	GCG (Ala/A)	GGG (Gly/G)	GTG (Val/V)	(Phe/F) Fenilalanina
GAT (Asp/D)	GCT (Ala/A)	GGT (Gly/G)	GTT (Val/V)	(Pro/P) Prolina
				(Ser/S) Serina
TAA (Stop)	TCA (Ser/S)	TGA (Stop)	TTA (Leu/L)	(Thr/T)Treonina
TAC (Tyr/Y)	TCC (Ser/S)	TGC (Cys/C)	TTC (Phe/F)	(Trp/W) Triptofano
TAG (Stop)	TCG (Ser/S)	TGG (Trp/W)	TTG (Leu/L)	(Tyr/Y) Tirosina
TAT (Tyr/Y)	TCT (Ser/S)	TGT (Cys/C)	TTT (Phe/F)	(Val/V) Valina
				(Stop) Códon de Parada

Tabela 2.1: Tabela do código genético, com a sigla, nome e símbolo de cada proteína, juntamente com os códons associados a cada uma delas.

2.1.3 Síntese de proteínas

Cada célula contém milhares de proteínas diferentes, que desempenham as mais diversas funções, orientando praticamente todas as suas atividades. O processo de síntese de proteínas, nas células eucariontes, possui três fases: transcrição, *splicing* e tradução.

As fases de **transcrição** e ***splicing*** são as responsáveis pela criação de uma molécula de RNAm. Ambas ocorrem no núcleo das células com o auxílio de várias enzimas catalisadoras, tais como as RNA polimerases I, II e III. A transcrição se resume na síntese de uma molécula especial de RNA, conhecida como **pré-RNA mensageiro (pré-RNAm)**, a partir das informações de um trecho do DNA. Nesse processo, a dupla hélice do DNA se separa em duas cadeias. O pré-RNAm será, então, uma cadeia de nucleotídeos complementar ao trecho de interesse do DNA, com a base complementar da adenina, nesse caso, sendo a uracila. Quando a transcrição termina, a molécula de pré-RNAm se separa da cadeia de DNA e são reestabelecidas as pontes de hidrogênio e, consequentemente, a dupla hélice do DNA. Uma ilustração da transcrição de um trecho de uma molécula de DNA pode ser vista na Figura 2.3.

A molécula de pré-RNAm transcrita possui regiões internas que não são traduzidas em proteínas. Esses trechos não utilizados na síntese proteica são removidos durante a fase de *splicing*, gerando assim uma molécula de **RNA mensageiro maduro** ou simplesmente RNAm. Após a fase de *splicing*, o RNAm é transportado através da membrana nuclear para o citoplasma, local onde ocorre a próxima fase da síntese proteica. Vale ressaltar que nos organismos procariontes não temos a fase de *splicing* no processo de síntese proteica. Nesses organismos, toda a região do DNA que é transcrita em uma molécula de RNAm é traduzida em proteínas. Dessa forma, nos procariontes, existem apenas as fases de transcrição e tradução.

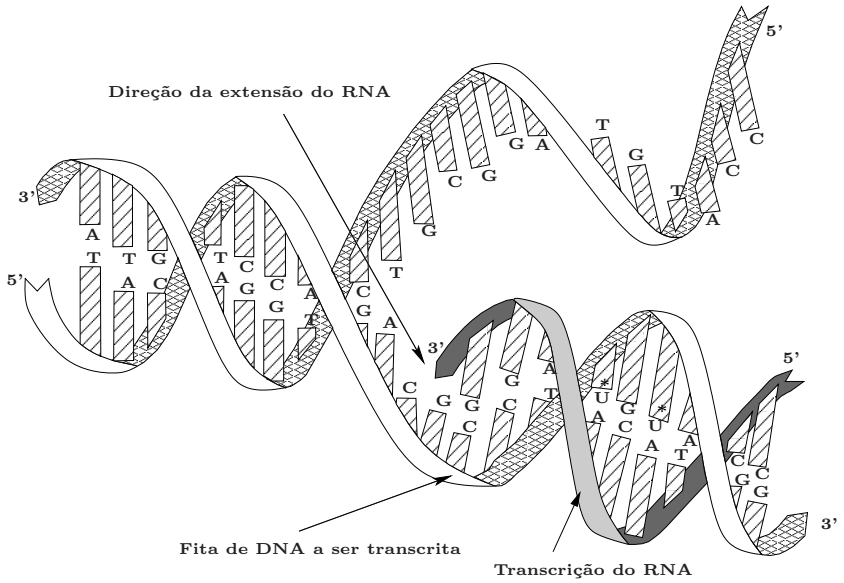


Figura 2.3: Transcrição de uma Molécula de DNA.

O caminho inverso da transcrição, onde uma molécula de RNA origina uma molécula de DNA, é denominado **transcrição reversa**. Esse processo, por utilizar uma molécula de RNAm como molde, dá origem a uma molécula que inclui somente os éxons do gene que codifca o RNA. Essa molécula é denominada **cDNA** (DNA complementar).

A **tradução** é o processo de construção da cadeia de aminoácidos que forma a proteína sendo sintetizada. Ela se inicia em um códon específico, denominado **códon de início**[4] e ocorre nos ribossomos, organelas responsáveis pela decodificação das moléculas de RNAm. Essa decodificação é feita com base no código genético e conta com o auxílio das moléculas de RNAt. As moléculas de RNAt agem como adaptadores entre os códons do RNAm e o aminoácido que é codificado. Uma das pontas do RNAt carrega o aminoácido e a outra ponta consiste de uma sequência de três nucleotídeos conhecida como **anticódon**, que tem a capacidade de reconhecer os códons do RNAm. Sendo assim, para cada aminoácido, existe pelo menos um RNAt.

De forma resumida, o processo de tradução é realizado da seguinte forma: o RNAm combina-se com os ribossomos e tem sua sequência de códons lida. Para cada códon lido do RNAm, seu respectivo RNAt é atraído até os ribossomos e, pela complementaridade de bases, é feita a ligação entre o códon e o anticódon. Com essa ligação o RNAt libera o aminoácido que está carregando. Este aminoácido é então ligado à cadeia crescente de aminoácidos. A síntese da proteína encerra-se quando os ribossomos encontram um códon de parada no RNAm.

2.1.4 Os genes

Como mencionado, apenas alguns trechos do DNA são transcritos em moléculas de RNA durante o processo de síntese de proteínas. Cada um desses trechos recebe o nome de **gene**. Observe que nem todos os códons presentes em um gene são, de

[4]Na maioria dos genes, o códon de início é o que especifica o aminoácido metionina (ATG).

fato, traduzidos em um aminoácido da proteína sendo sintetizada. Apenas a parte do gene que permanece no RNAm após a fase de *splicing* codifica proteína. Essas regiões, que compõem o RNAm (maduro), são conhecidas como **éxons** ou regiões codificantes, enquanto as regiões eliminadas do pré-RNAm na fase de *splicing* são conhecidas como **íntrons**. Os éxons podem ser classificados em quatro classes distintas: **éxon inicial** é o primeiro éxon que aparece no gene; **éxon final** é o último éxon do gene; **éxon interno** é qualquer éxon do gene que está entre o éxon inicial e o éxon final; por fim, **éxon único** é aquele que compõe um gene formado por um único éxon.

Os genes possuem algumas regiões, ditas **reguladoras**, que atuam no controle da transcrição e de outros processos relacionados. Nessas regiões encontramos elementos como os sítios de doação, os sítios de aceitação, o códon de início, o códon de parada, o promotor e o terminador. O **promotor** interage com a RNA polimerase no intuito de indicar o local onde se dará início o processo de transcrição. O **terminador**, encontrado no final do gene, determina o fim do processo de transcrição. Os **códons de início e de parada**, indicam, respectivamente, os locais no gene onde irá iniciar e terminar o processo de tradução. Normalmente, o códon de início é o ATG e o códon de parada pode ser o TAA, TAG ou TGA. Já os **sítios de doação e aceitação** determinam, respectivamente, o início e o fim de cada íntron dentro do gene. Com raríssimas exceções, o início de um íntron (fim de um éxon) é denotado pelos dinucleotídeos GT e o fim de um íntron (início de um éxon) é denotado pelos dinucleotídeos AG. Finalmente, nas extremidades do genes existem duas regiões reguladoras conhecidas como 3' e 5'-UTR (do inglês, *untranslated regions*) que são transcritas na molécula de RNAm mas não traduzidas. A Figura 2.4 ilustra as regiões de um gene eucarionte.

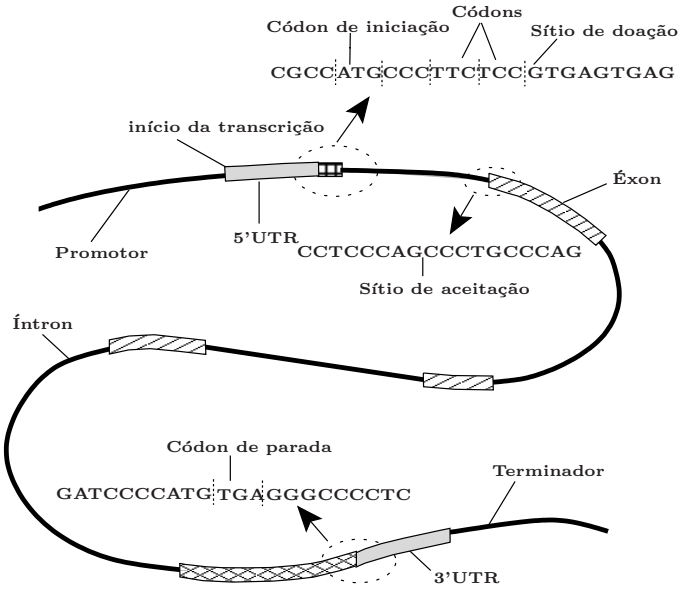


Figura 2.4: Estrutura de um gene eucarionte.

Por fim, vale observar que alguns genes podem dar origem a duas ou mais proteínas diferentes por meio de um processo conhecido como ***splicing* alternativo** [34]. Nesse processo, alguns éxons podem ser eliminados da molécula de pré-RNAm durante a

fase de *splicing*, permitindo que várias proteínas sejam sintetizadas a partir de um mesmo gene. Deve-se notar que os fatores que controlam esse processo ainda são pouco conhecidos.

2.1.5 Evolução do genoma

Uma célula se reproduz duplicando seu DNA e se dividindo em duas novas células-filhas, cada uma com uma cópia do DNA duplicado. Porém, nesse processo de duplicação, a cópia do DNA pode se danificar devido a fatores externos ou por falhas no próprio processo. Devido à importância do DNA para a sobrevivência da célula, grande parte dessas alterações são imediatamente corrigidas por processos denominados coletivamente de **reparo de DNA** [34]. Em algumas ocasiões o processo de reparo do DNA falha, causando assim uma alteração permanente no DNA. Essa alteração recebe o nome de **mutação**. As mutações podem variar de uma simples mudança de uma base do DNA (substituição, inserção, remoção), à duplicação de partes, ou de genes inteiros, dentro do DNA.

Uma mutação, por menor que seja, pode destruir um organismo dependendo da região em que ocorre. Mutações desse tipo são conhecidas como **mutações deletérias** e costumam ocorrer em regiões funcionais do DNA. As mutações deletérias geralmente causam alterações na sequência de bases que codificam certos aminoácidos de uma proteína, criando proteínas parcial ou totalmente não-funcionais. Outras mutações, porém, não resultam em nenhum dano aparente ao organismo. Essas mutações geralmente ocorrem nas regiões não-codificantes (nos íntrons ou nas regiões intergênicas). Tais mutações são conhecidas como **mutações silenciosas**, e, em outras palavras, são aquelas que não modificam a sequência de aminoácidos codificados nos genes de um DNA. Por fim, uma pequena porcentagem das mutações podem ser benéficas ao organismo. Essas mutações, diferentemente das deletérias, são passadas para as próximas gerações e ajudam o organismo a se adaptar melhor ao seu ambiente. Essas mutações são conhecidas como **mutações benéficas**.

A duplicação gênica ocorre em altas taxas durante a evolução das espécies, dando origem a famílias distintas de genes. Dizemos que dois ou mais genes são **homólogos** quando eles possuem um ancestral em comum, possuindo assim semelhanças na sua sequência nucleotídica. Genes homólogos que possuem a mesma função, em espécies distintas, são chamados de genes **ortólogos**. Quando os genes homólogos pertencem a uma mesma espécie e possuem como origem um evento de duplicação dentro do genoma, recebem o nome de genes **parálogos**. Nesse caso, eles podem exercer as mesmas funções dentro do genoma ou funções diferentes.

Nas regiões do genoma ditas funcionais, a incidência de mutações é bem menor que nas regiões sem nenhuma função aparente. Esse fato confirma um princípio que atesta que as regiões funcionais de um genoma tendem a se conservar durante a evolução. Esse princípio é denominado **princípio da conservação das bases**. Uma explicação para isso está no fato de que qualquer mudança nessas regiões do DNA pode acarretar sérios danos ao organismo. Tendo em mente este princípio, podemos, em determinadas situações, comparar duas ou mais sequências de modo a determinar quão parecidas elas são e obter informações sobre a localização de seus éxons e outras regiões funcionais.

2.2 Conceitos básicos de Ciência da Computação

Um **alfabeto** Σ é definido como um conjunto finito de símbolos. Um exemplo típico de alfabeto é o alfabeto latino $\{a, \ldots, z, A, \ldots, Z\}$, sobre o qual estamos construindo este texto. Dado um alfabeto Σ qualquer, uma **sequência** s construída sobre Σ é uma sucessão finita de símbolos pertencentes a esse alfabeto. As sucessões de símbolos $u = CATAGCA$ e $v = GCCATGAACGT$ são exemplos de sequências construídas sobre o alfabeto $\Sigma = \{A, C, G, T\}$. O conjunto de todas as sequências que podem ser construídas sobre um alfabeto Σ é denotado por Σ^*. O **tamanho** de uma sequência s, denotado por $|s|$, é a quantidade de símbolos que compõem a sequência. Uma sequência que possui tamanho zero é conhecida como **sequência vazia** e é denotada por ε. Para as sequências u e v dadas como exemplo, temos que $|u| = 7$ e $|v| = 11$. Dada uma sequência s e um determinado i, $1 \leq i \leq |s|$, $s[i]$ denota o i-ésimo símbolo de s. Considerando as sequências dadas como exemplo, temos que $u[2] = A$ e $v[10] = G$.

Uma **subsequência** de uma sequência s consiste de uma sequência obtida a partir da remoção de símbolos de s. Por exemplo, $CTACA$ é uma subsequência de $x = CCGTATTCAGA$. Um **segmento** de uma sequência s é um trecho contíguo de s que tem início em uma posição $i \geq 1$ e se estende até uma posição $j \leq |s|$ de s. Denotamos um segmento de s de i até j por $s[i..j]$. Por exemplo, $CATGA$ é um segmento de $w = TGCATGAAT$. Um **prefixo** de uma sequência s equivale a um segmento $s[1..i]$, com $i \leq |s|$. Ou seja, um prefixo de s é um segmento de s que inclui os i primeiros símbolos da sequência. Um **sufixo** de uma sequência s, por sua vez, equivale ao segmento $s[j..|s|]$, com $j \geq 1$. Dessa forma, um sufixo de s é um segmento que inclui os $|s| - j + 1$ últimos símbolos da sequência.

2.2.1 Programação dinâmica

A programação[5] dinâmica é uma técnica de projeto de algoritmos para a resolução de problemas computacionais, em especial os de Otimização Combinatória.

A idéia básica da programação dinâmica é obter uma resposta ótima para um problema combinando respostas já obtidas para subproblemas do problema original. Para isso, o problema é inicialmente decomposto em subproblemas mais simples de se resolver, para os quais são obtidas respostas diretas. Em cada etapa seguinte, os resultados dos subproblemas são combinados, obtendo assim, ao final da execução, a resposta para o problema inicial. Também em cada etapa, os resultados intermediários são guardados em uma tabela para que futuramente não seja necessário calculá-los novamente. Essa estratégia de armazenar o resultado de um subproblema em uma tabela na primeira vez em que ele é encontrado e resolvido, recebe o nome de **memoização** [17].

A programação dinâmica assemelha-se ao método de divisão e conquista no fato de resolver os problemas combinando as soluções de seus subproblemas. O que a difere do método de divisão e conquista é que, neste último, sempre que for encontrado um subproblema para o qual já foi calculada uma solução ele será resolvido novamente, realizando um processamento desnecessário. Um algoritmo de programação dinâmica

[5] A palavra "programação" nesse contexto refere-se a um método tabular (em forma matricial) e não à ação de escrever código de computador.

resolveria todos os subproblemas apenas uma vez, guardando os resultados em uma tabela, economizando assim o trabalho de recalcular a resposta toda vez que um subproblema já resolvido for encontrado.

Para que a programação dinâmica seja aplicável a um problema de otimização, ele deve possuir duas características: subestrutura ótima e sobreposição de problemas. Um problema apresenta um **subestrutura ótima** se uma solução ótima para o problema contém em seu interior soluções ótimas para seus subproblemas. Já a **sobreposição de problemas** diz respeito à necessidade de reexaminar o mesmo problema muitas vezes na busca pela solução do problema original.

2.2.2 Alinhamento de sequências

Conforme citado no fim da Seção 2.1.5, em determinadas situações, estamos interessados em comparar duas ou mais sequências de modo a determinar quão parecidas elas são. Uma forma de realizar essa comparação é através de um processo conhecido como **alinhamento de sequências**.

Informalmente, um alinhamento entre duas sequências (construídas sobre o mesmo alfabeto) consiste na inserção de espaços, aqui representados pelo símbolo '−', em pontos arbitrários ao longo delas de modo que fiquem do mesmo tamanho. Vale notar que os espaços podem ser inseridos inclusive no início ou no final das sequências e a única restrição é que um espaço em uma sequência nunca pode ter um espaço correspondente na outra sequência.

Formalmente, um alinhamento de duas sequências s e $t \in \Sigma$, com $|s| = n$ e $|t| = m$, é uma terceira sequência w, com $|w| \leq n + m$, construída sobre o alfabeto $\overline{\Sigma} = \{(\Sigma \cup \{-\}) \times (\Sigma \cup \{-\})\} \setminus \{(-,-)\}$ tal que o primeiro elemento do par é um símbolo de s ou $\{-\}$ e o segundo um símbolo de t ou $\{-\}$ [58]. Além disso, todos os símbolos de s e t devem estar presentes em w e devem aparecer nessa sequência na mesma ordem em que aparecem nas sequências s e t. Dadas as sequências $s = CATAGCA$ e $t = GCCATGAACGT$ um possível alinhamento entre elas é a sequência $w = (-,G)(-,C)(C,C)(A,A)(T,T)(-,G)(A,A)(G,A)(C,C)(-,G)(A,T)$. Dado um alinhamento w entre duas sequências s e t, chamamos de $\overline{s}$ e $\overline{t}$ as sequências formadas, respectivamente, pelos elementos que ocupam a primeira e segunda posição dos pares que compõem w. No exemplo dado, temos que $\overline{s} = --CAT-AGC-A$ e $\overline{t} = GCCATGAACGT$.

Um alinhamento $w = (\overline{s}[1], \overline{t}[1])(\overline{s}[2], \overline{t}[2]) \ldots (\overline{s}[p], \overline{t}[p])$ pode ser representado da seguinte forma:

$$\begin{array}{cccc} \overline{s}[1] & \overline{s}[2] & \ldots & \overline{s}[p] \\ \overline{t}[1] & \overline{t}[2] & \ldots & \overline{t}[p] \end{array}$$

Analisando o alinhamento acima, podemos observar três tipos distintos de colunas: colunas com o mesmo símbolo em s e em t ($s[i] = t[i]$); colunas com símbolos diferentes e não incluindo espaços ($s[i] \neq t[i]$, com $s[i] \neq -$ e $t[i] \neq -$); colunas incluindo um espaço ($s[i] \neq t[i]$, com $s[i] = -$ ou $t[i] = -$). Cada uma dessas colunas recebe o nome de ***match***, ***mismatch*** e ***space***, respectivamente.

Ainda considerando $s = CATAGCA$ e $t = GCCATGAACGT$, a Figura 2.5 mostra a representação do alinhamento w entre elas. Observe que esse alinhamento possui 5 *matches*, 1 *mismatch* e 4 *spaces*.

```
--CAT-AGC-A
GCCATGAACGT
```

Figura 2.5: Representação de um alinhamento w entre as sequências $s = CATAGCA$ e $t = GCCATGAACGT$.

Dado um alinhamento entre duas sequências s e t, podemos associar-lhe uma pontuação de acordo com o número de *matches*, *mismatches* e *spaces* que possui. Isso é feito associando um valor para cada uma dessas colunas através de uma **função de pontuação**. Somando-se esses valores o resultado final corresponde à pontuação total do alinhamento, que recebe o nome de ***score***. Um **alinhamento ótimo** para as duas sequências s e t é aquele que tiver o maior *score*. Esta pontuação máxima é chamada de **similaridade** entre s e t.

Formalmente, definimos uma função de pontuação ω como $\omega : \overline{\Sigma} \to \mathbb{R}$, que associa um valor real para cada par de símbolos $(\mu, \nu) \in \overline{\Sigma}$. Essas funções são comumente representadas por matrizes indexadas pelos símbolos em $\Sigma \cup \{-\}$ e, por isso, elas são também chamadas de **matrizes de pontuação**. O *score* de um alinhamento de duas sequências s e t, utilizando uma função de pontuação ω qualquer, é então dado por:

$$score_\omega(\overline{s}, \overline{t}) = \sum_{i=1}^{|\overline{s}|} \omega(\overline{s}[i], \overline{t}[i])$$

Sendo assim, a similaridade $sim_\omega(s, t)$ entre duas sequências s e t, utilizando a função de pontuação ω, é dada por:

$$sim_\omega(s, t) = \max\{score_\omega(\overline{s}, \overline{t})\},$$

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

Quando estamos interessados em verificar quão parecidas são duas sequências, a função de pontuação deve ser definida de forma a valorizar os *matches* e penalizar os *spaces*. Sobre os *mismatches*, geralmente, eles não são tão valorizados quanto os *matches* e nem tão penalizados quanto os *spaces*.

Com as definições e observações acima, podemos formular o problema de determinar quão semelhantes são duas sequências como um problema de otimização da seguinte forma:

Problema do Alinhamento de Duas Sequências*: Dadas duas sequências s e t construídas sobre o mesmo alfabeto Σ, tal que $\{-\} \notin \Sigma$, e uma função de pontuação ω, encontrar o valor da similaridade $sim_\omega(s, t)$ entre elas.*

Um algoritmo para resolver o problema do alinhamento de duas sequências foi introduzido por Needleman e Wunsch [43] em 1970. Ele utiliza o paradigma da programação dinâmica e devolve a similaridade entre as duas sequências, s e t, dadas como entrada. Os passos do algoritmo de Needleman-Wunsch podem ser vistos no Algoritmo 1. Esse algoritmo recebe como entrada duas sequências, s e t, de tamanhos n e m, respectivamente, e uma função de pontuação ω, e devolve o valor da similaridade entre as duas sequências. O algoritmo determina a similaridade entre as duas sequências comparando cada par de símbolos e computando os *matches*, *mismatches* e *spaces* nas duas sequências, de modo a obter o melhor *score* possível [42].

Algoritmo 1 Algoritmo baseado em programação dinâmica que calcula a similaridade entre duas sequências.

Algoritmo Similaridade(s, t, ω): Recebe como entrada duas sequências, s e t, construídas sobre o mesmo alfabeto, e uma função de pontuação ω. O algoritmo devolve o valor da similaridade entre as sequências s e t.

```
1: M[0,0] ← 0
2: n ← |s|;
3: m ← |t|;
4: para i ← 0 até n faça
5:     M[i,0] ← M[i−1,0] + ω(s[i],−);
6: para j ← 0 até m faça
7:     M[0,j] ← M[0,j−1] + ω(−,t[j]);
8: para i ← 1 até n faça
9:     para j ← 1 até m faça
10:        M[i,j] ← M[i−1,j] + ω(s[i],−);
11:        se M[i,j] ≤ M[i,j−1] + ω(−,t[j]) então
12:            M[i,j] ← M[i,j−1] + ω(−,t[j]);
13:        se M[i,j] ≤ M[i−1,j−1] + ω(s[i],t[j]) então
14:            M[i,j] ← M[i−1,j−1] + ω(s[i],t[j]);
15: Devolva M[n,m];
```

Uma matriz M, com dimensões $(n+1) \times (m+1)$, indexada por $\{0 \ldots n\}$ e $\{0 \ldots m\}$, é a estrutura base do algoritmo de Needleman-Wunsch. Essa matriz é conhecida como **matriz de alinhamento**. Cada posição $M[i, j]$ da matriz de alinhamento armazena o *score* do alinhamento ótimo entre os prefixos $s[1..i]$ e $t[1..j]$ de s e t, respectivamente. Sendo assim, a posição $M[n, m]$ possui, ao término do preenchimento da matriz, o valor da similaridade entre as duas sequências. Também de acordo com o conteúdo de M temos que $M[0, 0] = 0$ (similaridade entre duas palavras vazias), $M[0, j] = j \times \omega(-, t[j])$ (similaridade entre prefixos de t e a palavra vazia), para $1 \leq j \leq m$, $M[i, 0] = i \times \omega(s[i], -)$ (similaridade entre prefixos de s e a palavra vazia), para $1 \leq i \leq n$. Com a primeira linha e a primeira coluna preenchidas, cada posição $M[i, j]$, para $1 \leq i \leq n$ e $1 \leq j \leq m$, é computada de acordo com a Recorrência 2.1.

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

A Recorrência 2.1 considera que as similaridades entre os prefixos menores que $s[1..i]$ e $t[1..j]$ já estejam calculadas, ou seja, que as similaridades entre os prefixos $s[1..i-1]$ e $t[1..j]$, $s[1..i]$ e $t[1..j-1]$ e $s[1..i-1]$ e $t[1..j-1]$, chamados aqui de prefixos candidatos, já são conhecidas. Essas similaridades encontram-se, respectivamente, acima, à esquerda e na diagonal superior esquerda de $M[i, j]$, na matriz de alinhamento.

Uma vez que já sabemos as similaridades entre os prefixos candidatos, podemos estender o alinhamento relacionado entre eles inserindo um símbolo de uma das sequências (alinhado com um espaço) ou então de ambas as sequências. A inserção de um espaço se caracteriza quando estendemos o alinhamento entre os prefixos $s[1..i]$ e $t[1..j-1]$

ou $s[1..i-1]$ e $t[1..j]$. Como podemos notar nesses pares de prefixos, um deles já tem tamanho i ou j, dessa forma, para chegarmos ao alinhamento dos prefixos $s[1..i]$ e $t[1..j]$, basta inserir um espaço no prefixo de tamanho i ($M[i, j-1] + \omega(-, t[j])$) ou j ($M[i-1, j] + \omega(s[i], -)$) e um símbolo no outro prefixo. Já ao estendermos o alinhamento entre os prefixos $s[1..i-1]$ e $t[1..j-1]$ temos que inserir um símbolo em cada um deles de forma a obtermos o alinhamento dos prefixos $s[1..i]$ e $t[1..j]$, caracterizando assim um *match* ou um *mismatch* no alinhamento ($M[i-1, j-1] + \omega(s[i], t[j])$).

Um exemplo de matriz de alinhamento preenchida pelo Algoritmo 1, para as sequências $s = CATAGCA$ e $t = GCCATGAACGT$, e uma função de pontuação ω tal que $\omega(\mu, \nu) = 1$ se $\mu = \nu$, $\omega(\mu, \nu) = -1$ se $\mu \neq \nu$, $\omega(\mu, \nu) = -2$ se ou $\mu = -$ ou $\nu = -$, pode ser visto na Figura 2.6.

	0_-	1_G	2_C	3_C	4_A	5_T	6_G	7_A	8_A	9_C	10_G	11_T
0_-	**0**	**-2**	**-4**	-6	-8	-10	-12	-14	-16	-18	-20	-22
1_C	-2	-1	-1	**-3**	-5	-7	-9	-11	-13	-15	-17	-19
2_A	-4	-3	-2	-2	**-2**	-4	-6	-8	-10	-12	-14	-16
3_T	-6	-5	-4	-3	-3	**-1**	**-3**	-5	-7	-9	-11	-13
4_A	-8	-7	-6	-5	-2	-3	-2	**-2**	-4	-6	-8	-10
5_G	-10	-7	-8	-7	-4	-3	-2	-3	**-3**	-5	-5	-7
6_C	-12	-9	-6	-7	-6	-5	-4	-3	-4	**-2**	**-4**	-6
7_A	-14	-11	-8	-7	-6	-7	-6	-3	-2	-4	-3	**-5**

Figura 2.6: Matriz de alinhamento das sequências $s = CATAGCA$ e $t = GCCATGAACGT$, com a função de pontuação $\omega(\mu, \nu)$.

Como podemos observar, o Algoritmo 1 consiste basicamente no preenchimento de uma matriz de dimensões $(n+1)\times(m+1)$. O preenchimento de cada uma dessas posições envolve um número constante de operações: três operações de soma e duas operações de comparação para escolha do maior valor dentre as somas realizadas. Podemos, a partir disso, concluir que sua complexidade de tempo e espaço é $O(n \times m)$.

Falamos até agora de como calcular a similaridade entre duas sequências. Em algumas situações, além da similaridade, é interessante também construir um alinhamento que possui tal pontuação. Para obter um alinhamento ótimo, basta percorrermos a matriz de alinhamento na ordem inversa em que ela foi preenchida. Para isso, partimos da posição $M[n, m]$, que é a última posição da matriz, e a partir de um teste simples, conseguimos saber qual a próxima posição a ser visitada, até chegarmos na posição $M[0, 0]$. O teste a ser feito para descobrir qual próxima posição visitar leva em conta o modo como a matriz foi preenchida. Na recorrência utilizada no preenchimento da matriz, o valor de $M[i, j]$ é calculado a partir de $M[i-1, j]$, $M[i, j-1]$ ou $M[i-1, j-1]$. Sendo assim, há apenas três possibilidades para voltarmos pela matriz. Um algoritmo que constrói um alinhamento ótimo, dada uma matriz de pontuação, pode ser visto no Algoritmo 2.

Note que, a cada passo do Algoritmo 2, voltamos sempre uma posição na linha e/ou na coluna da matriz. Dessa forma, temos que o algoritmo realiza um total de $n + m$ passos até que a posição $M[0, 0]$ seja alcançada. Devido a isso, um alinhamento ótimo das duas sequências pode ser construído em tempo $O(n + m)$.

Algoritmo 2 Algoritmo que constrói um alinhamento ótimo entre duas sequências.

Algoritmo Constroi_Alinhamento(s, t, M, ω): Recebe como entrada duas sequências, s e t, construídas sobre o mesmo alfabeto, a matriz de alinhamento M utilizada para calcular a similaridade entre s e t e a função de pontuação ω utilizada para calcular a similaridade entre s e t. O algoritmo devolve um dos alinhamentos ótimos entre s e t.

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

Da forma como abordado até aqui, o alinhamento entre duas sequências leva em consideração todos os seus símbolos. Esse tipo de alinhamento é conhecido como **alinhamento global** [43]. Além dele, existem outras variantes de alinhamento, sendo as mais conhecidas o **alinhamento local** e o **alinhamento semiglobal**. Alinhar duas sequências localmente é procurar pelos segmentos das duas sequências com maior valor de similaridade global. Para a solução desse problema, existe um algoritmo baseado no algoritmo de Needleman-Wunsch. Esse algoritmo, proposto por Smith e Waterman [61], inicializa a primeira linha e a primeira coluna da matriz com o valor 0 (zero) e a recorrência utilizada para computar os valores da matriz também deve possuir o valor 0 (zero) como opção de pontuação [42]. Além disso, no alinhamento local, todas as posições da matriz indicam uma similaridade entre segmentos das sequências. Sendo assim, a similaridade máxima de um segmento de s e de t se encontra na posição da matriz que possui o maior valor após seu preenchimento. Em um alinhamento semiglobal, consideramos as duas sequências como um todo, porém, diferentemente do alinhamento global, os espaços inseridos nas extremidades das sequências não são contabilizados no *score* do alinhamento. Esse tipo de alinhamento também pode ser construído com uma variante do algoritmo de Needleman-Wunsch que inclui mudanças na inicialização da matriz e na localização do valor de similaridade.

2.2.3 Modelos ocultos de Markov

Um Modelo Oculto de Markov (HMM, do inglês, *Hidden Markov Model*) é um modelo estatístico bastante utilizado no reconhecimento de padrões temporais como a fala, a escrita, os gestos e também na Bioinformática [39].

Para entender melhor o funcionamento de um HMM, considere uma sala onde existe um palco cercado por uma cortina. No palco, existe um conjunto de n caixas, cada uma com m bolas de cores diferentes. A única pessoa que pode ver as caixas é a pessoa que as manuseia. Essa pessoa escolhe uma caixa de acordo com um processo aleatório inicial. Ao escolher a caixa, uma bola é retirada aleatoriamente e mostrada através da cortina. A cor da bola é a única observação para quem está na sala. Em seguida, a bola é devolvida à caixa de onde foi retirada e então é escolhida a próxima caixa de qual será retirada uma bola. A escolha da próxima caixa também é feita através de um processo aleatório, que leva em conta apenas a última caixa que foi escolhida. Nessa analogia, as caixas correspondem aos estados do HMM e as cores das bolas às observações geradas, sendo que a probabilidade de cada cor é definida diferentemente para cada caixa. Estamos interessados nesse caso em, dada uma sequência de cores (de bolas) mostradas pela cortina, determinar a probabilidade dela ter sido gerada ou então a mais provável sequência de caixas que a originou.

Formalmente, um HMM pode ser visto como um conjunto finito de estados onde, a cada unidade de tempo, ocorrem transições de um estado para outro (ou de um estado para ele mesmo). A probabilidade de transição para um próximo estado depende apenas do estado atual. Essa é a chamada suposição de Markov e o modelo resultante é dito um HMM de primeira ordem. No entanto, o próximo estado pode depender dos k estados anteriores. Assim, também é possível obter um HMM de ordem k. É importante frisar que quanto maior a ordem do HMM, maior será sua complexidade. Um HMM é habitualmente caracterizado pelos seguintes elementos:

1. um conjunto $S = \{S_1, \ldots, S_n,\}$ de n estados, onde cada estado está associado a um evento do sistema sendo modelado. O estado em que o modelo se encontra no instante de tempo t é denotado por q_t;

2. um conjunto $V = \{V_1, \ldots, V_m\}$ de m símbolos que representam as possíveis observações que podem ser geradas em cada estado;

3. um conjunto de elementos $\Pi = \{\pi_i\}$, para $1 \leq i \leq n$, que determinam a probabilidade de cada estado ser escolhido como estado inicial;

4. um conjunto finito de elementos $A = \{a_{i,j}\}$, para $1 \leq i, j \leq n$, que determinam a distribuição de probabilidades de transição entre os estados, ou seja, a probabilidade do modelo estar no estado S_j, no instante $t + 1$, sabendo que ele estava no estado S_i no instante t;

5. um conjunto finito de elementos $B = \{b_{i,j}\}$, para $1 \leq i \leq m$ e $1 \leq j \leq n$, que determinam a distribuição de probabilidade dos símbolos observáveis para cada estado S_j.

Dada uma quíntupla $\lambda(S, V, \Pi, A, B)$, que representa um modelo, pode-se gerar uma sequência de observações $O = \{o_1, \ldots, o_T\}$, onde cada observação o_t, para $1 \leq t \leq T$, é um símbolo de V. Essa sequência de observações é obtida através dos seguintes passos:

1. para $t = 1$ um estado inicial ($q_1 = S_i$) é escolhido de acordo com Π;
2. é gerada uma observação o_t de acordo com o conjunto B;
3. é realizada a transição para um novo estado $q_t = S_j$ de acordo com o conjunto A;
4. se $t > T$ a sequência está gerada. Senão, t é incrementado e retorna-se ao passo 2.

Pode-se notar que uma mesma sequência de observações O pode ser gerada a partir de diferentes transições entre os estados do modelo. Essa característica é a que torna o HMM um modelo oculto, pois dada uma sequência de observações O não se sabe ao certo qual a sequência de estados que lhe deu origem. Disso, três problemas básicos surgem ao tratarmos de HMMs. São eles:

1. calcular a probabilidade de uma dada sequência de observações O, dado um modelo λ;
2. descobrir qual a sequência de estados com maior probabilidade de ter gerado uma sequência de observações O, dado um modelo λ;
3. ajustar os parâmetros de um modelo de modo que a probabilidade de uma sequência de observações seja maximizada.

Como soluções para estes problemas são utilizados os algoritmos *forward/backward* [48], Viterbi [60] e Baum-Welch [8], respectivamente. Geralmente, uma solução para o terceiro problema é utilizada para treinar um modelo que possa ser usado nos outros dois problemas.

Além dos HMMs como descritos aqui, existem os chamados Modelos Ocultos de Markov Generalizados (GHMM, do inglês, *Generalized Hidden Markov Model*). Uma das diferenças entre um GHMM e um HMM é a possibilidade de modelar explicitamente a duração de cada estado. Em um GHMM não há o conceito de transição de um estado para ele mesmo, uma vez que um mesmo estado pode gerar mais do que uma observação. Dessa forma há um melhor controle da quantidade de observações geradas seguidamente por um mesmo estado.

2.3 Conceitos básicos de Bioinformática

Este trabalho enquadra-se em uma área recente de pesquisa denominada Bioinformática, que utiliza ferramentas e métodos computacionais para resolver problemas biológicos. A base da Bioinformática está na fusão dos conceitos da Ciência da Computação e da Biologia Molecular. Um exemplo dessa fusão pode facilmente ser observado ao considerarmos uma dada sequência genômica como uma sequência de caracteres construída sobre o alfabeto $\Sigma = \{A, C, G, T\}$, cujos símbolos são referências às iniciais

das bases adenina, citosina, guanina e timina, respectivamente. Vale observar que antes do surgimento da Bioinformática, havia somente duas formas de realizar experimentos biológicos: dentro de um organismo vivo (técnica *in vivo*) ou em um ambiente artificial (técnica *in vitro*). Porém, com o aparecimento da Bioinformática, podemos agora realizar experimentos no computador (técnica *in silico*[6]).

Com o auxílio da Bioinformática é possível apurar informações relevantes sobre sequências de DNA e proteínas, tais como a função que elas exercem e a qualidade de uma sequência. Também com o auxílio da Bioinformática podemos montar o mapa genético de um organismo e analisá-lo. Além dos problemas citados, existe uma série de outros problemas biológicos que podem ser resolvidos através dos conceitos de computação. Dentre eles destacam-se:

- Sequenciamento de DNA (montagem de fragmentos): determinar a sequência de nucleotídeos de uma molécula de DNA a partir da montagem de fragmentos sobrepostos dessa molécula, fornecidos por máquinas de sequenciamento;
- Predição da estrutura secundária do RNA: determinar a estrutura secundária de uma molécula de RNA baseando-se na sua sequência de nucleotídeos;
- Construção de árvores filogenéticas: construir árvores de filogenia a partir da comparação de características e sequências de DNAs de diferentes organismos, a fim de obter informações a respeito da distância evolutiva entre eles;
- Determinação da função das proteínas: determinar a função de uma proteína a partir de sua estrutura primária através da busca por padrões pré-determinados ou através de métodos comparativos utilizando informações de proteínas de estruturas conhecidas;
- Predição de genes: determinar em que lugar da sequência de DNA se encontram suas regiões codificantes.

A Bioinformática também tem um papel muito importante na organização dos dados devido ao aumento crescente de informações geradas pelo processamento de sequências de DNA disponíveis. Para organizar essa grande quantidade de informação, são mantidos bancos de dados públicos onde encontram-se armazenadas sequências de nucleotídeos, aminoácidos ou estruturas de proteínas. Dentre os principais bancos de dados públicos temos o GenBank [10], que é mantido pelo NCBI (*National Center for Biotechnology Information*). Além do Genbank, o NCBI possui um dos principais *websites* para pesquisas em Bioinformática do mundo, com um diversificado acervo de dados disponíveis ao público.

Neste trabalho, temos como objeto principal de estudo o problema da identificação de genes em sequências de DNA de organismos eucariontes através da comparação de sequências homólogas. Mais detalhes sobre esse problema podem ser vistos no capítulo a seguir.

[6] O termo *in silico* provém do material que os microprocessadores são feitos, o silício.

Capítulo 3

O Problema da Identificação de Genes

Este trabalho aborda um problema da Bioinformática denominado Problema da Identificação de Genes. Detalhes deste problema são dados neste capítulo, que encontra-se dividido em três seções. Na primeira delas definimos o problema e fazemos algumas considerações sobre ele. A segunda seção aborda os métodos desenvolvidos na tentativa de solucionar o problema da identificação de genes. Finalmente, a última seção descreve algumas ferramentas utilizadas para se localizar os genes em uma sequência de DNA.

3.1 Considerações sobre o problema da identificação de genes

O **Problema da Identificação de Genes**, ou **Problema da Predição de Genes**, é um dos vários problemas estudados no contexto da Bioinformática. Ele se resume em determinar a posição inicial e final de um ou mais genes codificados em uma sequência de DNA assim como as posições iniciais e finais dos éxons que os compõem. A identificação de genes em sequências de DNA pode ser aplicada em pesquisas objetivando o controle de pragas, tratamento de doenças, produção de novos medicamentos, etc.

Por mais simples que seja a sua descrição, o problema da identificação de genes não é um problema trivial. Predizer os genes em um sequência é um pouco mais simples ao tratarmos de organismos com genomas compactos (como bactérias, moscas e seres procariontes em geral) devido à tendência desses genomas incluírem éxons extensos e íntrons curtos (ou nem incluírem íntrons). Além disso, esse tipo de genoma tende a possuir uma grande concentração de genes, onde mais de 85% da sequência de DNA codifica proteínas [7, 45]. O desafio é muito maior quando genomas mais complexos precisam ser processados. Nesses casos, apenas uma pequena porção do genoma codifica proteínas. Os genes presentes no DNA humano, por exemplo, constituem apenas cerca de 3% de todo o genoma [29]. Nessa pequena porção do genoma, onde encontramos os genes, nos deparamos com a presença de longos íntrons entre os éxons e, devido a uma falta de padrões de bases específicos que permitam a distinção dos éxons das outras partes da sequência, nenhum algoritmo de predição de genes é totalmente confiável.

As considerações acima tornam o problema da identificação de genes um problema difícil e um campo fértil de pesquisa em Ciência da Computação. Os métodos utilizados na tentativa de solucioná-los assim como algumas ferramentas desenvolvidas para esse fim são descritas a seguir.

3.2 Métodos para identificação de genes

Os principais métodos utilizados para tentar solucionar o problema da identificação de genes estão baseados no princípio da conservação das bases, descrito no Capítulo 2, e podem ser divididos em duas categorias principais: métodos intrínsecos ou *ab initio* e métodos extrínsecos. O funcionamento desses métodos encontra-se detalhado a seguir.

3.2.1 Métodos instrínsecos

Os **métodos intrínsecos** utilizam-se de informações contidas somente na sequência que está sendo analisada em busca dos seus genes. Ou seja, de informações intrínsecas à sequência. Eles costumam ser divididos em métodos estatísticos e métodos de busca por sinais.

Métodos estatísticos

Os **métodos estatísticos** identificam porções da sequência de interesse que possuem características estatísticas semelhantes àquelas apresentadas por genes já conhecidos [41]. Isso é feito, basicamente, deslizando-se uma janela de tamanho fixo pela sequência de interesse e calculando-se, para cada uma dessas janelas, o valor de uma métrica específica. Esse cálculo é feito com base na composição da região interna à janela e o valor em si é utilizado para se determinar a probabilidade dessa região incluir um gene.

Dentre as várias métricas utilizadas pelos métodos estatísticos, a principal delas é a frequência com que os códons ocorrem nos genes de um certo organismo. Outra métrica bastante utilizada diz respeito à preferência, em certos organismos, por determinados códons sinônimos em detrimento a outros.

Uma descrição geral das métricas utilizadas nessa abordagem é apresentada por Ficket e Tung em [19]. Dentre os principais trabalhos de predição de genes que se utilizam dessa abordagem, encontram-se aqueles publicados por Staden e McLachlan em [55] e por Arquès *et al.* em [6].

Métodos de busca por sinais

Os **métodos de busca por sinais** procuram identificar sinais (regiões) associados ao processo de expressão gênica (promotores, códons de início e parada, sítios de doação e aceitação) no intuito de determinar a presença de genes na sequência. Assim como no caso dos métodos estatísticos, essa busca também é feita deslizando-se uma janela pela sequência de interesse e determinando-se, para cada uma dessas janelas, um valor específico. Esse valor corresponde à similaridade entre a região interna à janela e uma sequência ou modelo representativo do sinal sendo procurado.

O modelo representativo de um sinal pode ser determinado de várias formas diferentes com base em sequências de sinais identificados experimentalmente. Uma delas é através da geração de uma sequência consenso, construída alinhando várias sequências de determinado sinal e, para cada posição, escolhendo o nucleotídeo que aparece mais vezes nela. Uma maneira mais sofisticada de gerar o modelo representativo de um sinal faz uso da frequência dos nucleotídeos em cada uma das posições das sequências que representam o sinal. Nesse processo, as sequências dos sinais são inicialmente alinhadas. Depois disso, conta-se a frequência de cada base em cada uma das colunas do alinhamento construído no passo anterior. Essas frequências são registradas em uma matriz de peso, também chamada PWM (do inglês, *positional weight matrix*).

Exemplos de trabalhos que utilizam esse método encontram-se descritos por Brunak *et al.* em [12], por Staden em [54] e por Akhtar *et al.* em [4].

3.2.2 Métodos extrínsecos

Os **métodos extrínsecos** fazem uso de informações contidas em outras sequências, com as regiões codificantes já conhecidas ou não, para identificar os possíveis genes na sequência analisada. A idéia geral dos métodos extrínsecos é identificar regiões significativamente parecidas entre duas ou mais sequências.

A busca por regiões semelhantes entre duas sequências geralmente é feita através do processo de alinhamento. Na grande maioria das vezes porém, em se tratando da tarefa de predição de genes, os algoritmos de alinhamento descritos na Seção 2.2.2 não podem ser aplicados diretamente, sendo necessário o uso de algumas variantes desses algoritmos que levam em conta a existência de regiões mal conservadas, como os íntrons e regiões intergênicas dentro das sequências. Uma dessas variantes, implementadas pelo BLAST [5], identifica vários segmentos com alta similaridade entre as duas sequências. Ou seja, vários alinhamentos locais entre elas. Outra variante busca encontrar o melhor alinhamento entre a sequência de interesse e segmentos que, concatenados, formam a outra sequência (alinhamento *spliced*). Alguns exemplos de estudos utilizando essa abordagem foram feitos por Gish e States em [23], por He e Goldwasser em [28] e por Adi e Ferreira em [3].

Os métodos descritos servem como base para várias ferramentas desenvolvidas para localizar os genes codificados em uma sequência de DNA. Detalhes sobre algumas dessas ferramentas podem ser vistos a seguir.

3.3 Ferramentas para identificação de genes por comparação de DNAs

Os métodos para identificação de genes existem desde o início da década de 80 mas apesar disso, nenhuma ferramenta para predição de genes havia sido implementada até o início da década de 90. A partir desse período, uma série de ferramentas com essa finalidade começaram a ser implementadas. De acordo com os métodos utilizados pelas ferramentas para realizar a predição dos genes, elas também costumam ser divididas em dois grupos.

No primeiro grupo se enquadram as ferramentas que fazem uso de informações intrínsecas à sequência em que estamos procurando os genes. Uma das ferramentas mais conhecidas desse grupo denomina-se GENSCAN. Essa ferramenta, desenvolvida por Chris e Burge, e descrita em [13], baseia-se em métodos estatísticos e de busca por sinais dentro da sequência analisada à procura dos seus genes. Uma vez identificados, os sinais são então combinados por meio de um Modelo Oculto de Markov Generalizado (GHMM), que descreve a estrutura de um gene. Isso permite identificar possíveis éxons e montá-los em um gene completo. Exemplos de outras ferramentas presentes nesse grupo são GENEID [26], GENIE [33], TIGRSCAN [38], EVIGAN [36], SNAP [31], dentre outras.

No outro grupo enquadram-se as ferramentas que se utilizam de informações presentes em outra sequência, semelhante à sequência analisada, para realizar a predição dos genes nelas codificados. Uma das ferramentas pioneiras nessa linha é a PROCRUSTES, desenvolvida por Gelfand *et al.*, e descrita em [21]. Essa ferramenta recebe como entrada, além da sequência de DNA onde serão procurados os genes, uma outra correspondente a uma sequência de proteína ou a uma sequência de cDNA. Ela busca na sequência de DNA trechos que, depois de concatenados, mais se assemelham à proteína ou ao cDNA dado como entrada. Algumas outras ferramentas nesse grupo são AGENDA [57], GENESEQER [59], EXONFINDER3 [2], SPROCKET [47] e a GENOMETHREADER [24], desenvolvida a partir da GENESEQER.

Existem também algumas ferramentas que, visando melhorar a qualidade da identificação de genes, se utilizam tanto de informações intrínsecas à sequência quanto da sua comparação com sequências já conhecidas. Esse é o caso, por exemplo, da TWINSCAN, que é uma versão melhorada da GENSCAN onde os genes são confirmados através de uma comparação com sequências semelhantes à sequência onde o gene foi predito [32]. Outras ferramentas que se utilizam de ambos métodos são a EUGÈNE'HOM [20], SHORTHMM [62], AUGUSTUS+ [56] e SGP2 [46].

Neste trabalho, damos enfoque ao tratamento do problema da identificação de genes por meio da comparação de sequências. Por isso, descrevemos a seguir o funcionamento de quatro ferramentas que fazem uso da comparação de sequências homólogas de DNAs para a execução de suas tarefas.

3.3.1 AGenDA

A AGENDA (do inglês, *Alignment-based GENe Detection Algorithm*) é uma ferramenta de predição de genes baseada na comparação entre sequências homólogas, proposta por Rinner e Morgenstern em [57]. A AGENDA possui várias estapas de execução, onde cada etapa é realizada por uma ferramenta diferente. Primeiramente, uma ferramenta chamada REPEATMASKER [53] analisa as sequências de entrada a fim de mascarar as regiões onde há repetições de nucleotídeos. A ferramenta CHAOS, proposta por Brudno e Morgenstern em [11], é então usada para localizar regiões significativamente semelhantes das sequências de entrada. Essas regiões devolvidas pela CHAOS são utilizadas para reduzir o espaço de busca e, consequentemente, o tempo de execução de outra ferramenta, a DIALIGN, proposta por Morgenstern em [40], utilizada logo em seguida.

A DIALIGN determina um alinhamento local de segmentos nas regiões devolvidas pela CHAOS, sem a inserção de *spaces*. Cada um desses alinhamentos locais recebe o nome

de fragmento. A DIALIGN pontua esses fragmentos verificando a probabilidade da ocorrência de algum outro fragmento de mesmo tamanho e mesmo número de *matches*. A partir desses fragmentos, a AGENDA cria, para cada sequência, uma lista de éxons candidatos. Isso é feito agrupando-se os fragmentos de maior pontuação (em cada sequência) devolvidos pela DIALIGN e alongando ou encurtando esses agrupamentos até que sinais limitantes de éxons (sítios de aceitação/doação ou códons de início/parada) sejam encontrados. Cada um dos éxons candidatos possui uma pontuação calculada com base no grau de similaridade dos fragmentos devolvidos pela DIALIGN, na porcentagem de nucleotídeos que pertencem ao éxon candidato mas não ao agrupamento dos fragmentos correspondente (ou vice-versa), e pela qualidade dos sítios de aceitação/doação.

Por fim, com os éxons candidatos gerados e devidamente pontuados, a AGENDA escolhe os melhores éxons de forma a montar o gene que é devolvido ao usuário. Essa escolha é feita de modo a maximizar a soma de pontuações dos éxons escolhidos e dar origem a um gene biologicamente consistente. Em outras palavras, a um gene que tenha o primeiro éxon começando com um códon de início e terminando com um sítio de doação; o último éxon iniciando com um sítio de aceitação e terminando com um códon de parada e todos os éxons restantes iniciando com um sítio de aceitação e terminando com um sítio de doação. Em adição a essas regras, o tamanho total do gene deve ser múltiplo de três, não pode haver códons de parada internos ao gene e as lacunas entre os éxons candidatos devem obedecer a algumas restrições de tamanho. Além do gene, outras informações também são geradas pela AGENDA, tais como a lista completa de éxons candidatos, representação gráfica do modelo de saída, etc.

3.3.2 Progen

Assim como a AGENDA, a PROGEN, descrita por Novichkov *et al.* em [44], também é baseada na comparação entre duas sequências genômicas homólogas. Um pouco diferente da AGENDA, a PROGEN identifica os éxons componentes do gene sendo procurado à medida em que alinha as sequências, não havendo um passo exclusivo para a montagem do gene.

O primeiro passo realizado pela PROGEN é determinar os códons de início e de parada, assim como predizer os candidatos a sítios de aceitação e doação em ambas as sequências. Após essa etapa, um alinhamento ótimo entre as duas sequências é construído. Isso é feito preenchendo-se uma matriz de alinhamento com base na Recorrência 3.1. Nessa recorrência, Γ é a penalidade de se inserir um *space* em uma das sequências, Δ corresponde à penalidade para inserção de um íntron e $\Phi(A, B)$ é o valor da comparação do códon A com o códon B. Os vetores μ, ν e λ mantêm o *score* dos melhores alinhamentos encontrados até o momento que terminam, respectivamente, em um sítio de doação em s, em um sítio de doação em t ou em um sítio de doação em ambas as sequências. Podemos ver em nas fórmulas em 3.2 como os valores desses vetores são calculados.

$$
M[m,n] = \max \begin{cases}
M[m-3,n-3] & \\
\quad +\Phi(s_m,t_n) & (\textit{match} \text{ ou } \textit{mismatch} \text{ dos códons} \\
 & \quad (s[m]s[m+1]s[m+2]) \text{ e } (t[n]t[n+1]t[n+2])) \\
M[m,n-3]-\Gamma & (\textit{space} \text{ em } s \text{ e o códon} \\
 & \quad (t[n]t[n+1]t[n+2]) \text{ em } t) \\
M[m-3,n]-\Gamma & (\textit{space} \text{ em } t \text{ e o códon} \\
 & \quad (s[m]s[m+1]s[m+2]) \text{ em } s) \\
\mu_m[n-3]-\Delta & \text{se } m \text{ é um sítio de aceitação} \\
 & \quad (\text{íntron em } s) \\
\mu_m[n]-\Delta-\Gamma & \text{se } m \text{ é um sítio de aceitação} \\
 & \quad (\text{íntron em } s, \textit{space} \text{ em } t) \\
\nu_n[m-3]-\Delta & \text{se } n \text{ é um sítio de aceitação} \\
 & \quad (\text{íntron em } t) \\
\nu_n[m]-\Delta-\Gamma & \text{se } n \text{ é um sítio de aceitação} \\
 & \quad (\text{íntron em } t, \textit{space} \text{ em } s) \\
\lambda_{mn}-2\Delta & \text{se } m,n \text{ são sítios de aceitação} \\
 & \quad (\text{íntrons em } s \text{ e } t)
\end{cases} \tag{3.1}
$$

$$
\begin{aligned}
\mu_m[n] &= \max\{M[i,n] \mid i<m, i \text{ é um sítio de doação em } S\}, \\
\nu_n[m] &= \max\{M[m,j] \mid j<n, j \text{ é um sítio de doação em } T\}, \\
\lambda_{mn} &= \max\{M[i,j] \mid i<m, i \text{ é um sítio de doação em } S, \\
&\qquad\qquad j<n, j \text{ é um sítio de doação em } T\}.
\end{aligned} \tag{3.2}
$$

A Figura 3.1 representa parte da matriz de alinhamento preenchida pela PROGEN, onde cada interseção entre as retas representa uma posição na matriz. As setas que incidem na posição $M[m,n]$ indicam todas as possibilidades de preenchimento dessa posição, de acordo com a Recorrência 3.1. As setas com rótulos $-\Gamma$ e Φ são consideradas a todo momento e representam a continuação do alinhamento a partir do códon anterior de ambas sequências. Já as outras cinco setas representam a possibilidade de preencher a posição $M[m,n]$ da matriz considerando-se outras posições além das imediatamente anteriores. Essas possibilidades só são avaliadas quando a posição que está sendo analisada corresponde a um possível sítio de aceitação (em apenas uma ou em ambas as sequências), de tal forma que o alinhamento possa ser estendido a partir de um possível sítio de doação (em uma ou em ambas as sequências), resultando em um melhor *score* para o alinhamento global das sequências.

As setas com rótulos $-\Delta$ e $-\Delta-\Gamma$ só são consideradas se a posição $m(n)$ na sequência $s(t)$ corresponde a um possível sítio de aceitação. Nesse caso, essas duas setas representam a extensão do alinhamento a partir da posição $\nu_n[m-3](\mu_m[n-3])$ ou $\nu_n[m](\mu_m[n])$, respectivamente, que correspondem a um possível sítio de doação em $s(t)$. Essa escolha equivale à inserção de um íntron em $s(t)$, no caso da extensão ser feita a partir da posição $\nu_n[m-3](\mu_m[n-3])$, e à inserção de um íntron em $s(t)$ e um *space* em $t(s)$ no caso da extensão ser feita a partir da posição $\nu_n[m](\mu_m[n])$. A seta de rótulo -2Δ só é considerada se m e n corresponderem a possíveis sítios de aceitação, em s e t, respectivamente. Nesse caso, essa seta representa uma extensão do alinhamento a partir da posição λ_{mn}, o que representa a inserção de um íntron em s e outro em t.

A área sombreada na Figura 3.1 representa a área da matriz de alinhamento onde os valores de μ, ν e λ permanecem inalterados.

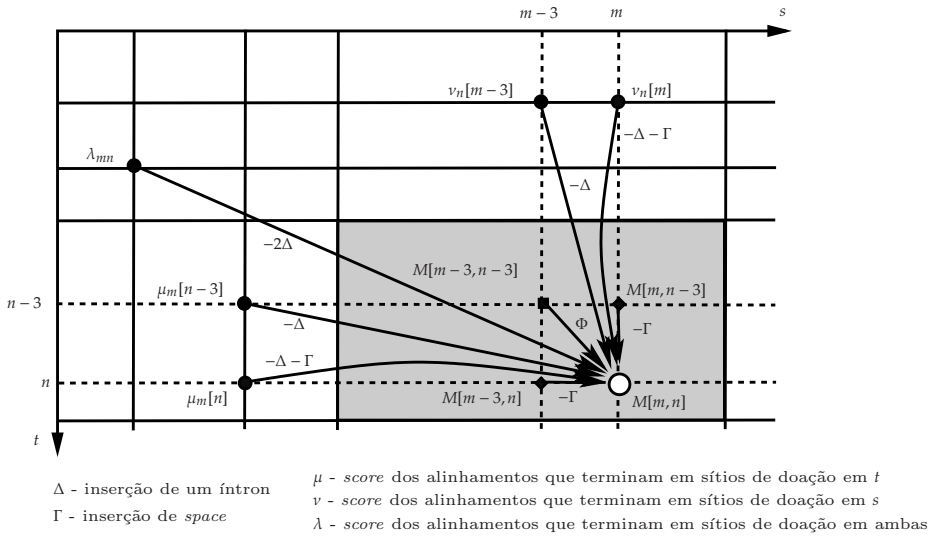


Figura 3.1: Possibilidades de preenchimento da posição $M[m,n]$ da matriz de alinhamento do PROGEN.

A PROGEN analisa os sinais (códons de início e de parada e candidatos a sítios de aceitação e doação) no nível de nucleotídeos. Porém, o alinhamento das duas sequências é realizado no nível de aminoácidos. Sendo assim, o alinhamento é realizado com a comparação de códons, ou seja, de três em três nucleotídeos. Uma outra característica da PROGEN é que ela faz uso de dois valores diferentes para penalizar os *spaces* inseridos nas sequências. Quando é inserido o primeiro *space* (após um *match* ou *mismatch*) temos a penalidade $\Gamma = \Gamma_{gap}$. Para dar sequência a um *space* já inserido (inserir um *space* logo em seguida a outro) temos a penalidade $\Gamma = \Gamma_{del}$. Para um bom alinhamento, Novichkov *et al.* definem esses valores de tal forma que $\Gamma_{gap} > \Gamma_{del}$. Além disso, $\Gamma_{gap} > \Delta$ e $\Gamma_{gap} < 2 \times \Delta$. Vemos então que é preferível inserir um íntron no alinhamento a inserir um *space* inicial, porém a inserção de dois íntrons seguidos já não é tão interessante quanto a inserção de um *space*. De acordo com os autores da PROGEN, este balanço nas penalidades garante, por um lado, a não predição de pequenos éxons espúrios e, por outro, que até uma modesta similaridade entre éxons corretos seja suficiente para incorporá-los na predição.

3.3.3 Sgp2

A SGP2 (do inglês, *Syntenic Gene Prediction*) também é uma ferramenta de predição de genes baseada na comparação de sequências. Os genes são procurados na sequência de interesse com o auxílio de uma ou mais sequências utilizadas como informante ou referência. Resumidamente, a SGP2 é uma integração da ferramenta de predição GENEID, desenvolvida por Guigó *et al.* e descrita em [26], que é baseada em informações intrínsecas à sequência de interesse, com a ferramenta de busca por similaridade TBLASTX, desenvolvida por Gish em [22].

Essencialmente, a GENEID é utilizada para predizer todos os potenciais éxons na sequência de interesse, observando que essa ferramenta recebe apenas uma sequência como entrada. O primeiro passo da GENEID é procurar por sinais (códons de início/parada e sítios de aceitação/doação) ao longo da sequência de interesse, atribuindo uma pontuação para eles de acordo com uma matriz de pesos. Após esse passo, um conjunto de éxons candidatos é criado a partir dos sinais encontrados. Cada éxon candidato recebe uma pontuação considerando a qualidade dos sinais que o definem e um modelo de Markov de ordem cinco que analisa a probabilidade desse éxon corresponder a uma região codificante. Com o conjunto de éxons candidatos devidamente criado, a GENEID escolhe os éxons que melhor representam a estrutura de um gene eucarionte e cuja soma de suas pontuações seja máxima.

A SGP2 utiliza o resultado da comparação entre as sequências de interesse e informante realizada pela TBLASTX para modificar a pontuação dos éxons candidatos gerados pela GENEID. A idéia, em síntese, é incorporar à pontuação de cada éxon candidato, atribuída pela GENEID, a pontuação do melhor HSP[1] que se sobrepõe a ele. É a TBLASTX que provê esses HSPs e infere a pontuação correspondente. Uma vez que a pontuação de cada éxon candidato é atualizada com as informações obtidas pela TBLASTX, a predição de genes prossegue como de costume na GENEID: alguns éxons são escolhidos do conjunto de éxons candidatos, de forma a maximizar a soma das pontuações desses éxons escolhidos para fazer parte da estrutura do gene.

3.3.4 Twinscan

A TWINSCAN, apresentada por Korf *et al.* em [32], assim como a SGP2, é uma ferramenta que se utiliza de informações obtidas através da comparação entre sequências genômicas para melhorar os resultados de uma ferramenta baseada em métodos intrínsecos denominada GENSCAN. Para entender o funcionamento da TWINSCAN, primeiramente precisamos compreender como funciona a ferramenta GENSCAN.

A GENSCAN, desenvolvida por Burge e Karlin, e descrita em [13], associa cada nucleotídeo da sequência de interesse a uma das sete categorias disponiveis: promotor, 5' UTR, éxon, íntron, 3' UTR, sinal de poliadenilação (poly-A), intergênico. Essa associação é feita de acordo com um GHMM que modela a estrutura de um gene.

Na GENSCAN, uma sequência que possua todos os seus nucleotídeos devidamente associados a um estado do GHMM recebe o nome de sequência rotulada. Dada uma sequência de DNA, a GENSCAN calcula os possíveis rótulos dessa sequência e para cada rótulo gerado é atribuído uma probabilidade. O rótulo com maior probabilidade da sequência é devolvido, por padrão, como resposta. O conjunto de éxons desse rótulo é chamado de conjunto de éxons ótimos. Além do conjunto de éxons ótimos, a GENSCAN também pode devolver um conjunto de éxons subótimos (próximo do ótimo), que possui éxons com alta probabilidade de serem reais mas não presentes no conjunto de éxons ótimos.

A TWINSCAN, por sua vez, recebe como entrada duas sequências genômicas, a sequência de interesse e a informante. O primeiro passo é utilizar a REPEATMASKER a fim de

[1] Um HSP (do inglês, *high-scoring segment pair*) corresponde a um alinhamento local com pontuação alta.

encontrar regiões que se repetem na sequência de interesse e mascará-las. Logo após é feito um alinhamento da sequência de interesse com a sequência informante utilizando a BLASTN [22], dando origem a um conjunto de HSPs. A similaridade entre as duas sequências é representada por uma sequência de conservação. Uma sequência de conservação associa a cada nucleotídeo da sequência alvo um símbolo que representa *match*, *mismatch* ou caractere não alinhado. A sequência de conservação é gerada por um algoritmo que mescla os HSPs devolvidos pela execução da BLASTN.

O modelo utilizado pela TWINSCAN é o mesmo usado pela GENSCAN, porém com algumas mudanças. A TWINSCAN atribui uma probabilidade para uma sequência rotulada de DNA juntamente com uma sequência de conservação, com probabilidades de emissão independentes em cada estado. Dadas uma sequência de DNA e uma de conservação, pode-se calcular a probabilidade desse par de sequências ter sido gerado pelo modelo da TWINSCAN utilizando-se o algoritmo de Viterbi.

Capítulo 4

Uma Proposta para o Problema da Identificação de Genes

Apesar da existência de vários métodos e ferramentas para o problema da identificação de genes, ainda não podemos considerá-lo resolvido. Vários estudos mostram que a exatidão de várias ferramentas está ainda muito aquém do esperado [1, 14, 25, 50], o que justifica pesquisas adicionais nessa área. Neste capítulo apresentamos a nossa abordagem para o problema da identificação de genes por comparação de DNAs. Descrevemos inicialmente uma modelagem que propomos para o problema, seguida dos detalhes teóricos e de implementação da solução proposta.

4.1 Uma modelagem para o problema da identificação de genes

Na tentativa de solucionar o problema da identificação de genes por meio da comparação de sequências, propusemos inicialmente uma formalização matemática para ele. Ou seja, modelamos o problema através de um problema da Otimização Combinatória que pode ser tratado utilizando-se conceitos teóricos da computação. Esse problema recebe o nome de Problema do Alinhamento de Pares de Segmentos e, para sua melhor compreensão, considere as seguintes definições. Seja s uma sequência qualquer. Dizemos que um segmento $s_1 = s[i..j]$ de s **antecede** um outro segmento $s_2 = s[k..l]$, também de s, se $j < k$, e denotamos esse tipo de relação por $s_1 \prec s_2$. Seja C um conjunto de segmentos de s. Um subconjunto $\Gamma_C = \{s_1, s_2, \ldots, s_p\}$ de C é dito uma **cadeia de segmentos** se $s_1 \prec s_2 \prec \ldots \prec s_p$. Dadas essas definições, o Problema do Alinhamento de Pares de Segmentos é definido como se segue:

Problema do Alinhamento de Pares de Segmentos *($PAPS(s, t, C, B, \omega)$): Dadas duas sequências s e t, de tamanhos n e m respectivamente, construídas sobre o mesmo alfabeto Σ, dois conjuntos B e C de segmentos de s e t, respectivamente, e uma função de pontuação ω, encontrar duas cadeias de segmentos $\Gamma_B = \{b_1, \ldots, b_q\}$ e $\Gamma_C = \{c_1, \ldots, c_q\}$ de B e C, respectivamente, tais que $\sum_{i=1}^{q} sim_\omega(b_i, c_i)$ seja a maior possível.*

Informalmente, com o PAPS estamos interessados em encontrar subconjuntos ordenados de B e C, de mesmo tamanho, cujos segmentos que ocupam a mesmo posição nesses subconjuntos sejam bem parecidos. A Figura 4.1 ilustra a definição do PAPS para duas sequências construídas sobre o alfabeto $\Sigma = \{A, \ldots, Z\}$. As cadeias de segmentos Γ_B = {UTILIZA, METODOS, COMPUTACIONAIS} e Γ_C = {UTILIZANDO, METODOS, COMPUTACIONAIS} são aquelas cuja soma das similaridades dos segmentos correspondentes é a maior dentre todos os pares de cadeias possíveis.

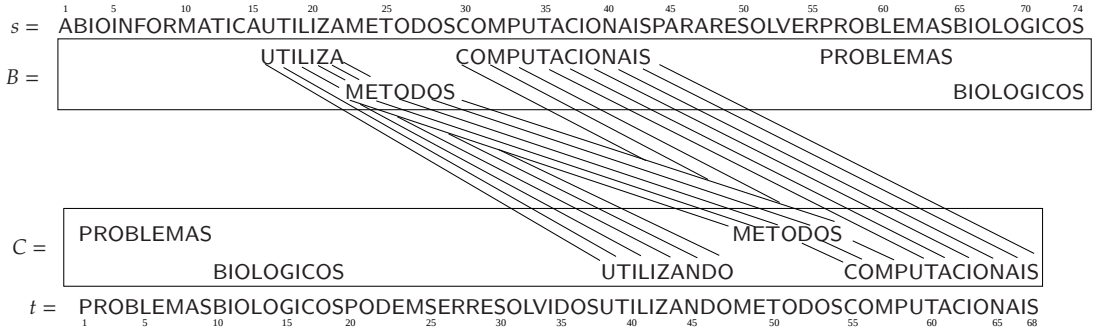


Figura 4.1: Ilustração do Problema do Alinhamento de Pares de Segmentos.

Observe que o PAPS pode ser aplicado na busca por uma solução para o problema da identificação de genes se considerarmos s e t como sendo duas sequências genômicas, nas quais estamos procurando os genes, e os conjuntos de segmentos B e C como sendo os possíveis éxons dessas sequências. Escolhendo-se uma função de pontuação adequada e considerando-se o princípio da conservação das bases, que atesta que os éxons tendem a ser mais conservados do que as outras regiões de uma sequência de DNA, é de se esperar que eles pertençam às cadeias Γ_B e Γ_C.

Uma solução trivial para o PAPS, baseada no método da força bruta, consiste em gerar todas as cadeias de segmentos possíveis de B e C, alinhar os segmentos correspondentes das cadeias de mesmo tamanho e, ao final, escolher aquele par de cadeias que possui maior soma de similaridades. A idéia é simples, porém computacionalmente inviável quando lidamos com uma quantidade razoável de segmentos. Para ilustrar quão dispendiosa é essa estratégia, imagine dois conjuntos B e C tais que $|B| = |C| = u$ e cujos respectivos segmentos não se sobrepõem. Observe que o número de cadeias de tamanho p que podem ser geradas a partir dos segmentos de B e C é igual a $\binom{u}{p} = \frac{u!}{p! \times (u-p)!}$. Com isso, temos um total de $\sum_{p=0}^{u} (\frac{u!}{p! \times (u-p)!})^2$ respostas possíveis para o problema. Tomando-se $u = 15$ e supondo que levaríamos um segundo para calcular a soma das similaridades para cada resposta gastaríamos um total de 155117520 segundos para resolver essa instância do problema. Esse tempo é claramente inviável levando à necessidade de uma solução mais eficiente para o problema.

Observando o problema com mais cuidado, podemos notar que ele possui a propriedade da sobreposição de problemas. Para determinarmos, por exemplo, a solução para o problema que envolve os segmentos b_k de B e c_l de C precisamos determinar a maior soma de similaridades que envolve segmentos anteriores a b_k e c_l em s e t, respectivamente. Esse mesmo valor precisará ser novamente calculado na busca pela solução do problema que envolve os segmentos posteriores a b_k e c_l nas suas respectivas sequências. Além

disso, podemos notar que o problema possui também a propriedade da subestrutura ótima. Sejam $B = \{b_1, \ldots, b_u\}$ e $C = \{c_1, \ldots, c_v\}$ os conjuntos de segmentos dados como entrada e Γ_B e Γ_C as cadeias de segmentos que possuem a maior soma de similaridades, tais que os últimos segmentos de Γ_B e Γ_C sejam, respectivamente, b_u e c_v. Excluindo-se b_u e c_v de B e C, respectivamente, temos que $\Gamma_B - \{b_u\}$ e $\Gamma_C - \{c_v\}$ serão as cadeias com maior soma de similaridades para essa instância modificada do problema, o que indica que soluções ótimas para o problema inclui soluções ótimas para os subproblemas.

Dadas as observações acima, uma solução eficiente para esse problema pode ser construída com base na programação dinâmica. Foi uma solução desse tipo que desenvolvemos neste trabalho e para uma melhor compreensão dela é importante ter em mente as seguintes definições. Dada uma sequência s e um conjunto de segmentos $B = \{b_1, b_2, \ldots, b_u\}$ derivado de s, chamamos de $first(i)$ e $last(i)$ a primeira e última posição, respectivamente, do segmento $b_i \in B$ em s. Seja $\Gamma_B = \{b_x, \ldots, b_y, \ldots, b_z\}$ uma cadeia de segmentos de B tal que algum segmento b_y de Γ_B contém a posição i de s. Denotamos por $\Gamma_B(y_{[i]}) = \{b_x, \ldots, b_y[i]\}$ o subconjunto dos segmentos em Γ_B anteriores a b_y mais os caracteres desse segmento que vão do seu início até a posição i (da sequência s). Dados esses conceitos, definimos uma matriz quadridimensional $M_{(n+1)\times(m+1)\times(u+1)\times(v+1)}$, onde cada posição $M[i, j, k, l]$ armazena o valor da maior soma de similaridades dos segmentos que compõem todos os possíveis pares de subconjuntos do tipo $(\Gamma_B(k_{[i]}), \Gamma_C(l_{[j]}))$ com a mesma quantidade de segmentos e que incluem os segmentos b_k e c_l, respectivamente.

Definida a matriz M, nossa proposta consiste basicamente do seu preenchimento e uma busca em todas as posições $M[last(k), last(l), k, l]$, para $1 \leq k \leq u$ e $1 \leq l \leq v$, de modo a encontrar a que possui o maior valor. Esse valor corresponde ao somatório das similaridades dos segmentos que compõem a resposta procurada. O preenchimento da matriz M se faz, basicamente, utilizando o algoritmo de Needleman-Wunsch quando estamos no interior dos segmentos e verificando-se qual a melhor solução obtida até o momento que utiliza segmentos anteriores ao segmento b_k e c_l quando estamos no início de um deles ou de ambos. O preenchimento de M é feito utilizando a Recorrência 4.1.

Detalhando a Recorrência 4.1, ao iniciarmos o cálculo da similaridade para o par de segmentos (b_k, c_l) ($i = first(k)$ e $j = first(l)$) devemos olhar para as similaridades já calculadas envolvendo pares de segmentos anteriores a b_k e c_l ($(b_{k'}, c_{l'})$, com $k' < k$ e $l' < l$). Encontrado o maior desses valores, podemos escolher alinhar $s[i]$ com $t[j]$, $s[i]$ com *space* ou $t[j]$ com *space* no intuito de preencher a posição $M[i, j, k, l]$.

Para os cálculos das posições da matriz que correspondem somente ao início de um segmento b_k ($i = first(k)$ e $j \neq first(l)$), temos a possibilidade de alinhar $t[j]$ com *space* em s, e nesse caso utilizamos o valor armazenado em $M[i, j-1, k, l]$. A segunda possibilidade corresponde a alinhar $s[i]$ com $t[j]$. Nesse caso, inserimos $j - first(l)$ *spaces* no início do segmento b_k. Finalmente a terceira possibilidade corresponde a alinhar $s[i]$ com um *space* em t. Nesse caso, inserimos $j - first(l) + 1$ *spaces* no início do segmento b_k. O cálculo das posições da matriz que correspondem somente ao início de um segmento c_l ($i \neq first(k)$ e $j = first(l)$) é análogo ao caso que acabamos de ver. O único detalhe é que agora inserimos *spaces* na outra sequência.

Uma vez preenchidas as posições correspondentes a $first(k)$ e/ou $first(l)$ durante o alinhamento dos segmentos (b_k, c_l), o restante das posições ($i \neq first(k)$ e $j \neq first(l)$) são preenchidas de acordo com o algoritmo de Needleman-Wunsch, descrito na Seção 2.2.2.

$$
M[i,j,k,l] = \max \left\{ \begin{array}{l}
\left\{ \begin{array}{ll}
M[i-1,j-1,k,l] + \omega(s[i],t[j]), & \text{se} \quad i \neq \mathit{first}(k) \\
M[i,j-1,k,l] + \omega(-,t[j]), & \text{e} \\
M[i-1,j,k,l] + \omega(s[i],-), & j \neq \mathit{first}(l)
\end{array} \right. \\
\left\{ \begin{array}{ll}
\max_{k'<k,l'<l} M[\mathit{last}(k'),\mathit{last}(l'),k',l'] + \omega(s[i],t[j]), & \text{se} \quad i = \mathit{first}(k) \\
\max_{k'<k,l'<l} M[\mathit{last}(k'),\mathit{last}(l'),k',l'] + \mathit{indel} + \omega(-,t[j]), & \text{e} \\
\max_{k'<k,l'<l} M[\mathit{last}(k'),\mathit{last}(l'),k',l'] + \mathit{indel} + \omega(s[i],-), & j = \mathit{first}(l)
\end{array} \right. \\
\left\{ \begin{array}{ll}
M[i,j-1,k,l] + \omega(-,t[j]), & \\
\max_{k'<k,l'<l} M[\mathit{last}(k'),\mathit{last}(l'),k',l'] + & \text{se} \quad i = \mathit{first}(k) \\
\qquad (j - \mathit{first}(l)) \times \mathit{indel} + \omega(s[i],t[j]), & \text{e} \\
\max_{k'<k,l'<l} M[\mathit{last}(k'),\mathit{last}(l'),k',l'] + & j \neq \mathit{first}(l) \\
\qquad (j - \mathit{first}(l) + 1) \times \mathit{indel} + \omega(s[i],-), &
\end{array} \right. \\
\left\{ \begin{array}{ll}
M[i-1,j,k,l] + \omega(s[i],-), & \\
\max_{k'<k,l'<l} M[\mathit{last}(k'),\mathit{last}(l'),k',l'] + & \text{se} \quad i \neq \mathit{first}(k) \\
\qquad (i - \mathit{first}(k)) \times \mathit{indel} + \omega(s[i],t[j]), & \text{e} \\
\max_{k'<k,l'<l} M[\mathit{last}(k'),\mathit{last}(l'),k',l'] + & j = \mathit{first}(l) \\
\qquad (i - \mathit{first}(k) + 1) \times \mathit{indel} + \omega(-,t[j]), &
\end{array} \right.
\end{array} \right. \tag{4.1}
$$

Observe que para o correto preenchimento de M, necessitamos apenas inicializar a posição $M[0,0,0,0]$ dessa matriz com o valor zero. Essa posição guarda o valor de uma solução vazia para o problema e é utilizada como ponto de partida para o preenchimento das posições de $M[i,j,k,l]$ quando i e/ou j correspondem ao início de um segmento. Além disso, note que, de acordo com o conteúdo da matriz M, as posições $M[i,j,k,l]$ só estão definidas para i e j pertencentes a segmentos das sequências s e t, respectivamente. Com isso, não faz sentido inicializar as posições $M[i,j,k,l]$ com $i = 0$ ou $j = 0$. Também de acordo com o conteúdo da matriz M, as posições $M[i,j,k,l]$ com $k = 0$ ou $l = 0$ (com $i \neq 0$ e $j \neq 0$) não precisam ser inicializadas já que corresponderiam a soluções sem segmento algum ($k = 0$ e $l = 0$) ou a soluções com números diferentes de segmentos ($k = 0$ e $l \neq 0$ ou $k \neq 0$ e $l = 0$).

Preenchida a matriz M e encontrada a posição $M[\mathit{last}(k),\mathit{last}(l),k,l]$ que possui o maior valor da soma das similaridades procurado, temos que os últimos segmentos de Γ_B e Γ_C são os segmentos b_k e c_l, respectivamente. A partir desse ponto podemos descobrir os penúltimos segmentos que fazem parte das cadeias Γ_B e Γ_C, visitando o par de segmentos cuja a similaridade foi utilizada no cálculo da posição $M[\mathit{last}(k),\mathit{last}(l),k,l]$. Isso é feito até qua a posição $M[0,0,0,0]$ seja alcançada.

Uma implementação direta da solução aqui proposta para o PAPS possui complexidade de espaço $O(n \times m \times u \times v)$, determinada pelo número de elementos da matriz M. Já a complexidade de tempo dessa implementação é dominada pelo preenchimento da matriz M, que é $O(n \times m \times u \times v)$, e pelo tempo gasto na procura por k' e l', que é $O(u^2 \times v^2 \times (\mathit{bmax} + \mathit{cmax}))$, uma vez que procuramos por esses valores para cada uma das posições da matriz que corresponde ao início de um segmento. Aqui, bmax e cmax correspondem ao tamanho do maior segmento em B e C, respectivamente. Dessa forma, um algoritmo baseado no preenchimento da matriz M fazendo uso da Recorrência 4.1 tal como ela se encontra descrita leva tempo $O(n \times m \times u \times v + u^2 \times v^2 \times (\mathit{bmax} + \mathit{cmax}))$. Um algoritmo com essa complexidade é proibitivo na prática levando à necessidade

de uma implementação mais eficiente da Recorrência 4.1. Na próxima seção falamos um pouco da implementação da Recorrência 4.1 detalhando a estratégia utilizada de forma a diminuir a complexidade de tempo e de espaço necessária para a solução do problema.

4.2 Detalhes de implementação

A Recorrência 4.1 foi implementada em uma ferramenta denominada GenePredictor, fazendo uso da linguagem de programação ANSI C e o ambiente integrado de desenvolvimento ECLIPSE [18]. A GenePredictor foi desenvolvido para a plataforma Linux e está sob a licença GPL. O projeto encontra-se hospedado no Google Code e pode ser acessado no endereço *http://genepredictor.googlecode.com/* (último acesso em 10 de dezembro de 2010).

Uma implementação ingênua da Recorrência 4.1 consiste, basicamente, da alocação e do preenchimento da matriz $M_{(n+1)\times(m+1)\times(u+1)\times(v+1)}$, onde calculamos a soma das similaridades dos pares de segmentos das cadeias construídas. Essa implementação consumiria uma quantidade de tempo e espaço proibitivo já que, para a aplicação proposta, as sequências de entrada podem ser muito grandes assim como os respectivos conjuntos de segmentos.

Uma implementação mais eficiente pode ser feita observando-se que $M[i, j, k, l]$ só está definida para as posições i e j de s e t, respectivamente, internas aos segmentos dessas sequências. Com isso, nem todas as posições de M precisam ser preenchidas e, consequentemente, alocadas. Pensando nisso, alteramos as dimensões da matriz de forma a alocar apenas o espaço que realmente é utilizado no cálculo das similaridades. A nova matriz M é agora indexada por u, v e o tamanho do maior segmento em B e em C, que chamamos de $bmax$ e $cmax$, respectivamente. Essa alteração mostrou-se muito eficiente na prática, uma vez que para nossos testes $bmax \ll n$ e $cmax \ll m$. A Figura 4.2 ilustra a idéia de alocação utilizada para a matriz M, utilizando uma combinação de matrizes bidimensionais.

Podemos observar na Figura 4.2 várias pequenas matrizes, as quais denominamos submatrizes de alinhamento, organizadas dentro de uma única matriz com dimensões $u \times v$, a qual denominamos matriz externa. A matriz externa é indexada pelos segmentos pertencentes aos conjuntos B e C, tendo assim a primeira linha da matriz associada ao primeiro elemento do conjunto B, a segunda linha ao segundo elemento de B, e assim sucessivamente até a u-ésima linha associada ao u-ésimo elemento de B. Uma associação semelhante é feita com as colunas da matriz e os elementos do conjunto C. Em cada posição $M[k, l]$ (considerando apenas a matriz externa) temos uma submatriz de alinhamento com dimensões $|b_k| + 1 \times |c_l| + 1$, onde é realizado o alinhamento global dos segmentos b_k e c_l, de B e C, respectivamente.

Com esse ganho prático na complexidade de espaço, diminuímos também a complexidade de tempo da nossa implementação. Além dessa otimização, estudando a Recorrência 4.1 conseguimos realizar algumas simplificações que reduziram a quantidade de tempo necessário para o preenchimento da matriz M. Podemos observar na Recorrência 4.1 que quando estamos em uma posição correspondente a $first(k)$ ou $first(l)$ sempre analisamos todas as posições $M[k', l', last(k'), last(l')]$ (adotando a nova matriz

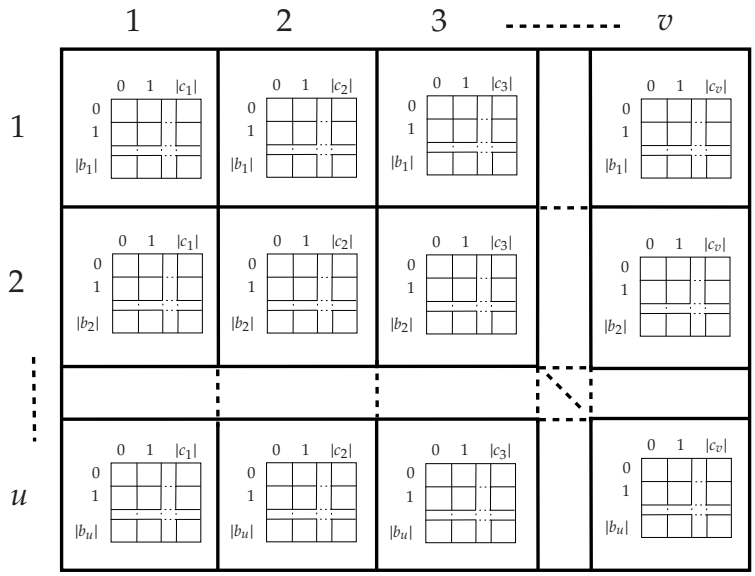


Figura 4.2: Representação da matriz quadridimensional utilizando uma combinação de matrizes bidimensionais. Nas submatrizes, considerando um segmento b_k de B, temos que as linhas 1 e $|b_k|$ equivalem, respectivamente, às posições *first*(k) e *last*(k), na sequência s, do segmento b_k. O mesmo é válido para as colunas, porém com relação à sequência t e um segmento c_l de C.

M), para $k' < k$ e $l' < l$, a fim de encontrar a solução que inclui os segmentos b_k e c_l. Observe porém que para um determinado par de segmentos (b_k, c_l) teremos sempre os mesmos valores para k' e l'. Com isso em mente podemos realizar a busca por k' e l' apenas uma vez e guardar o valor contido na posição $M[k', l', last(k'), last(l')]$ de maior soma de similaridades para que ele seja utilizado outras vezes. Explicando melhor essa otimização, no início do alinhamento do par de segmentos (b_k, c_l), ou seja, no início do preenchimento da submatriz de alinhamento $M[k, l]$, é feita uma busca por k' e l'. Com esses valores determinados, copiamos o valor de $M[k', l', last(k'), last(l')]$ para a posição $M[k, l, 0, 0]$ e então prosseguimos o preenchimento da submatriz de alinhamento como feito no algoritmo de Needleman-Wunsch. A primeira linha (linha 0) é preenchida com $M[k, l, i, j-1] + \omega(-, T[j])$, para $1 \leq j \leq |c_l|$, e a primeira coluna (coluna 0) com $M[k, l, i-1, j] + \omega(S[i], -)$, para $1 \leq i \leq |b_k|$.

Dessa forma, um algoritmo baseado no preenchimento da matriz M fazendo uso das melhorias descritas aqui possui complexidade de espaço $O(u \times v \times bmax \times cmax)$, determinada pelo número de elementos da matriz M. Já a complexidade de tempo dessa implementação é dominada pelo preenchimento da matriz M, que nesse caso é $O(u \times v \times bmax \times cmax)$, e pelo tempo gasto na procura por k' e l', que passa a ser $O(u^2 \times v^2)$, já que procuramos por esses valores apenas uma vez para cada par de segmentos. Assim, um algoritmo implementado com todas as otimizações aqui descritas tem complexidade de tempo $O(u \times v \times bmax \times cmax + u^2 \times v^2)$.

Capítulo 5

Testes

No intuito de avaliarmos quão adequado é o modelo proposto por nós para o problema da identificação de genes, assim como a precisão da ferramenta desenvolvida, a executamos sobre casos de testes reais e avaliamos os resultados obtidos. Essa avaliação foi feita comparando-se os resultados devolvidos pela ferramenta com as respostas corretas assim como com os resultados de outras quatro ferramentas de predição de genes disponíveis na literatura (AGenDA, Progen, Sgp2 e Twinscan). Este capítulo descreve os detalhes dessa avaliação experimental da nossa ferramenta e está dividido em três seções. Na primeira delas descrevemos as medidas de avaliação utilizadas. Na segunda seção descrevemos como os testes foram obtidos. Detalhes dos resultados obtidos encontram-se na última seção.

5.1 Medidas de avaliação

Para comparar os resultados das ferramentas e sabermos quão precisas elas são, nos utilizamos de um conjunto de medidas específicas para esse fim. Esse conjunto de medidas, composto basicamente pela especificidade e sensibilidade, foi proposto por Burset e Guigó em [14] e é comumente utilizado na tarefa de avaliação de ferramentas de predição de genes. Elas podem ser calculadas em dois níveis distintos: o de nucleotídeos e o de éxons.

Antes de detalharmos como é feito o cálculo das medidas de especificidade e sensibilidade, devemos levar em consideração algumas definições preliminares. Um éxon **anotado** corresponde a um éxon que existe de fato na sequência, enquanto que um éxon **predito** corresponde a um éxon identificado pela ferramenta de predição em consideração. Assim, um **nucleotídeo codificante** é aquele que se encontra dentro de um éxon anotado ou predito do gene. Por outro lado, um **nucleotídeo não-codificante** é aquele que está fora de um éxon anotado ou predito.

No nível de nucleotídeos, comparamos o *status* que um nucleotídeo recebe na predição (codificante ou não-codificante) com o *status* que ele possui na sequência anotada. Realizando essa comparação, podemos classificar os nucleotídeos em quatro classes distintas. Os **falsos-positivos** são aqueles nucleotídeos que foram preditos como codificantes mas que na realidade são não-codificantes; os **falsos-negativos** são os nucleotídeos preditos como não-codificantes mas na realidade são codificantes; os **verdadeiros-**

positivos são aqueles preditos como codificantes e que são realmente codificantes; os **verdadeiros-negativos** são aqueles preditos como não-codificantes e que não são realmente codificantes. A Figura 5.1 ilustra esses conceitos.

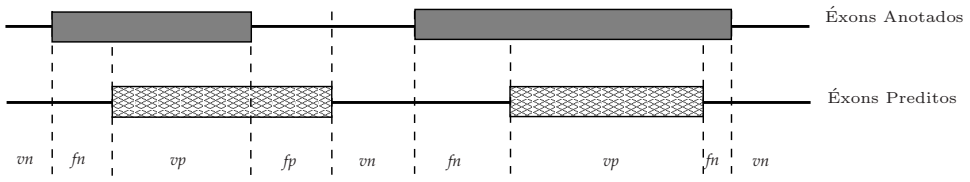


Figura 5.1: Ilustração do conceito de falsos-positivos (*fp*), falsos-negativos (*fn*), verdadeiros-positivos (*vp*) e verdadeiros-negativos (*vn*).

Com essas definições em mente, nos interessa saber a quantidade de nucleotídeos da sequência que se enquadra em cada uma das classes, ou seja, a quantidade de falsos-positivos (FP), falsos-negativos (FN), verdadeiros-positivos (VP) e verdadeiros-negativos (VN). As Fórmulas 5.1 e 5.2 resumem essas quantidades e correspondem à especificidade e à sensibilidade no nível de nucleotídeos (Sp_n e Sn_n), respectivamente.

$$Sp_n = \frac{VN}{VN + FP} \tag{5.1}$$

$$Sn_n = \frac{VP}{VP + FN} \tag{5.2}$$

A **especificidade** nos mostra a proporção de nucleotídeos não-codificantes que foram corretamente preditos pela ferramenta. A **sensibilidade**, por sua vez, é a proporção de nucleotídeos codificantes preditos de forma correta. Uma vez que o número de nucleotídeos não-codificantes em uma sequência genômica tende a ser bem maior que o de nucleotídeos codificantes, VN tende a ser maior que FP. Com isso, a especificidade (Sp_n) nem sempre informará a qualidade da predição, uma vez que o valor de Sp_n será muito alto. Devido a isso, utiliza-se uma fórmula alternativa para o cálculo da especificidade, que pode ser vista em 5.3. Esta é a fórmula que usamos para nossa avaliação.

$$Sp_n = \frac{VP}{VP + FP} \tag{5.3}$$

Tanto a especificidade quanto a sensibilidade, no nível de nucleotídeos, podem ser resumidas em apenas uma medida denominada coeficiente de correlação (CC). O CC é calculado a partir da Fórmula 5.4 e, apesar de muito utilizado, o CC possui a indesejável propriedade de não estar definido para aquelas sequências que, por exemplo, não incluam nenhum gene. Uma outra medida, que se assemelha ao CC, é a correlação aproximada (CA). A CA é calculada utilizando a Fórmula 5.5 e pode ser utilizada em qualquer circunstância.

$$CC = \frac{(VP \times VN) - (FN \times FP)}{\sqrt{(VP + FN) \times (VN + FP) \times (VP + FP) \times (VN + FN)}} \tag{5.4}$$

$$CA = \frac{1}{2}\left(\frac{VP}{VP + FN} + \frac{VP}{VP + FP} + \frac{VN}{VN + FP} + \frac{VN}{VN + FN}\right) - 1 \tag{5.5}$$

No nível de éxons, as medidas são calculadas comparando os éxons preditos com os éxons anotados da sequência. Para esse cálculo, dizemos que um éxon foi corretamente identificado quando seus limites (posição de seu primeiro e último nucleotídeo na sequência) são idênticos aos limites de algum éxon anotado da sequência (Figura 5.2).

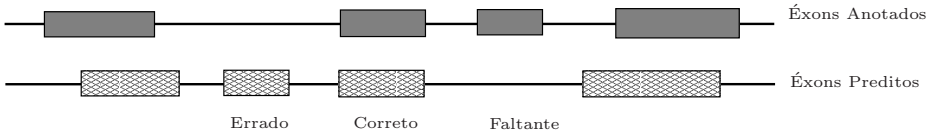


Figura 5.2: Ilustração do conceito de éxons errados, corretos e faltantes.

Seja NEP a quantidade de éxons preditos; NEC a quantidade de éxons corretamente identificados pela ferramenta e NEA a quantidade de éxons anotados; podemos calcular a especificidade e a sensibilidade no nível de éxons (Sp_e e Sn_e) utilizando as Fórmulas 5.6 e 5.7 respectivamente.

$$Sp_e = \frac{NEC}{NEP} \tag{5.6}$$

$$Sn_e = \frac{NEC}{NEA} \tag{5.7}$$

No nível de éxons, Sp_e é a proporção de éxons corretamente identificados pela ferramenta, com relação à quantidade de éxons preditos. Já Sn_e corresponde à proporção de éxons corretamente identificados com relação à quantidade de éxons reais na sequência. Essas duas medidas são comumente resumidas pela sua média aritmética Av_e, indicada na Fórmula 5.8.

$$Av_e = \frac{Sp_e + Sn_e}{2} \tag{5.8}$$

No nível de nucleotídeos estimamos quão bem as regiões codificantes da sequência foram encontradas enquanto no nível de éxon medimos quão bem os sinais (sítios de doação e aceitação e códons de início e parada) foram identificados. Observe que um alto valor das medidas no nível de nucleotídeos não necessariamente implicará num alto valor no nível de éxons.

Com base nas métricas propostas por Burset e Guigó, estendemos o conceito de especificidade e sensibilidade para um terceiro nível, o de bordas. No nível de bordas

estamos interessados em estimar o acerto de pelo menos um dos limites de um éxon predito (como borda consideramos as posições de seu primeiro e último nucleotídeo na sequência). Seja NEP a quantidade de éxons preditos, NEA a quantidade de éxons anotados e NBC a quantidade de bordas preditas corretamente (esse valor é tal que $NBC \leq 2 \times NEP$); podemos calcular a especificidade e a sensibilidade no nível de bordas (Sp_b e Sn_b) utilizando as fórmulas 5.9 e 5.10 respectivamente.

$$Sp_b = \frac{NBC}{2 \times NEP} \tag{5.9}$$

$$Sn_b = \frac{NBC}{2 \times NEA} \tag{5.10}$$

No nível de bordas, a especificidade e a sensibilidade também podem ser resumidas pela sua média aritmética Av_b, como indicado na Fórmula 5.11.

$$Avb = \frac{Sn_b + Sp_b}{2} \tag{5.11}$$

5.2 Sequências, segmentos e função de pontuação utilizados

Para o teste das ferramentas, criamos um conjunto de pares de sequências com base nas regiões do genoma humano utilizadas no projeto piloto ENCODE (do inglês, *ENCyclopedia Of DNA Elements*) [15]. Este projeto analisou uma amostra de 30 milhões de pares de bases (cerca de 1% do genoma humano) no intuito de investigar todos os elementos funcionais do genoma e como as células fazem uso das instruções codificadas no DNA. Os testes utilizados correspondem a pares de sequências genômicas que incluem, obrigatoriamente, uma sequência do *Homo sapiens* e uma segunda sequência homóloga a ela. As sequências do *Homo sapiens* correspondem a todos os genes pertencentes às regiões utilizadas no ENCODE que, obrigatoriamente, possuem um gene homólogo em outra espécie, são transcritos em uma e somente uma proteína (excluindo assim genes com *splicing* alternativo), possuem no máximo 250 mil bases e obedecem à estrutura descrita na Seção 2.1.4. No total 185 genes obedecem a todas essas restrições. Cada gene selecionado foi então extraído do genoma com um acréscimo de mil bases antes da região 5'-UTR e mil bases depois da região 3'-UTR, dando origem, assim, à sequência do *Homo sapiens* em cada par do conjunto de testes.

As sequências que complementam cada par são oriundas de genes homólogos ao gene da sequência do *Homo sapiens* e, obrigatoriamente, possuem a mesma quantidade de éxons anotados que seu par. Esses genes homólogos foram escolhidos utilizando a base de dados Homologene [51] do NCBI, dando preferência para genes do *Mus musculus*. Quando o gene do *Mus musculus* não estava de acordo com as características desejadas, ou não existia um homólogo pertencente a esse organismo, tomou-se genes de outras espécies, tais como *Rattus norvegicus*, *Bos taurus*, *Canis lupus familiaris*, *Danio rerio* ou *Pan troglodytes*. Cada gene homólogo também foi extraído com o acréscimo de mil bases

no início e no fim, dando origem à sequência homóloga do par no conjunto de testes. A preferência pela escolha do *Mus musculus* é dada pela grande quantidade de genes homólogos entre essa espécie e o *Homo sapiens*. Informações mais detalhadas sobre as sequências utilizadas podem ser encontradas no Apêncice A. Os pares de sequências utilizados podem ser encontrados no endereço *http://genepredictor.googlecode.com/* (último acesso em 10 de dezembro de 2010).

Com os pares de sequências genômicas em mãos, o próximo passo foi gerar os conjuntos de segmentos B e C a partir de cada sequência. Uma vez que esses conjuntos representam os possíveis éxons em cada uma das sequências, poderíamos pensar em obtê-los identificando os possíveis sítios de aceitação/doação e códons de início/parada nas sequências e tomando todos os segmentos delimitados por esses sinais. Porém, essa abordagem poderia dar origem a um grande número de segmentos, dificultando a execução do programa. Uma alternativa para a criação dos conjuntos B e C seria a utilização de uma ferramenta que faz um pré-processamento das sequências e nos aponta os possíveis éxons em cada uma. Utilizamos para esse fim a ferramenta GENSCAN. Como dito na Seção 3.3.4, além do conjunto de éxons ótimos que é devolvido como resposta, a GENSCAN possui a opção de devolver também um conjunto de éxons subótimos. Sendo assim, os conjuntos B e C são os conjuntos de éxons subótimos de s e t, respectivamente, devolvidos pela GENSCAN. Adotando essa estratégia obtivemos uma média de 36,8 segmentos por sequência genômica, tornando possível a execução da nossa ferramenta. A quantidade exata de segmentos derivado de cada sequência pode ser encontrada no Apêndice A.

Por fim, a função de pontuação adotada para a execução dos testes atribui um valor $p \in P = \{-2, -1, 1\}$ para cada um dos três tipos de colunas que podem ser observados em um alinhamento, conforme explicado na seção 2.2.2. Os valores atribuídos para *match*, *mismatch* e *space* são, respectivamente, 1, -1 e -2.

5.3 Resultados obtidos

Em uma primeira rodada de testes, executamos todas as ferramentas utilizando os 185 pares de sequências do nosso *benchmark*. Após a execução de todas as ferramentas, calculamos a especificidade e a sensibilidade dos resultados obtidos para cada par, nos três níveis descritos no início deste capítulo. As médias aritméticas de cada um desses valores podem ser observadas na Tabela 5.1.

Ferramenta	Sp_n	Sn_n	CA	Sp_e	Sn_e	Ave	Sp_b	Sn_b	Avb
GENEPREDICTOR	0,929	0,939	0,927	0,615	0,759	0,687	0,677	0,838	0,758
AGENDA	0,846	0,672	0,667	0,448	0,426	0,437	0,575	0,545	0,560
PROGEN	0,814	0,962	0,870	0,467	0,602	0,534	0,548	0,751	0,650
SGP2	0,858	0,800	0,783	0,550	0,508	0,529	0,673	0,631	0,652
TWINSCAN	0,908	0,773	0,794	0,651	0,520	0,585	0,760	0,613	0,686

Tabela 5.1: Qualidade das predições obtidas pelas ferramentas no conjunto de 185 pares de sequências.

Observando os valores de *CA*, *Ave* e *Avb* na Tabela 5.1, podemos concluir que a GenePredictor foi a que obteve os melhores resultados. Perdemos em qualidade apenas na sensibilidade no nível de nucleotídeos (Sn_n) para a Progen e na especificidade no nível de éxon (Sp_e) e, consequentemente, na especificidade no nível de bordas (Sp_b), para a Twinscan. De qualquer forma, vale observar que os resultados obtidos no nível de éxons pela nossa ferramenta são apenas regulares, com a identificação correta de 1874 éxons em um total de 2322 éxons anotados, mas com a identificação errada de um total de 794 éxons. Ou seja, nossa ferramenta apesar de sensível, não se mostrou tão específica para os nossos casos de teste.

Em uma análise das instâncias onde a nossa ferramenta apresentou os piores resultados, notamos que uma das causas para as falhas de predição foi a não existência de alguns éxons reais dentre aqueles devolvidos pela Genscan (e utilizados como entrada da nossa ferramenta). Outra falha relacionada ao conjunto de segmentos gerado pela Genscan deve-se à existência, nesse conjunto, de segmentos iguais a menos de umas poucas bases no início ou final deles. Nesses casos, nossa ferramenta tende a escolher o menor segmento como parte da resposta, mesmo não sendo ele um éxon anotado.

A GenePredictor também tem sua predição afetada quando da existência de outros genes muito próximos ao gene sendo buscado. Durante a criação do *benchmark*, ao acrescentarmos mil bases antes e mil bases após as sequências dos genes utilizados nos testes, vários foram os casos onde trechos (conservados) de um gene próximos ao gene utilizado faziam parte desses trechos adicionais. Assim, como analisamos apenas se a inserção de um par de segmentos vai melhorar a soma de similaridades que possuímos até o momento, houve casos onde, além dos éxons anotados serem corretamente preditos, eram colocados nas extremidades das cadeias Γ_B e Γ_C alguns segmentos adicionais (falsos éxons). Esses segmentos melhoram a soma das similaridades, mas atrapalham a predição correta do gene.

Um outro fator que atrapalha a qualidade da GenePredictor é quando as sequências possuem apenas um éxon anotado. Esse problema também está diretamente ligado ao fato de que queremos maximizar a soma de similaridade dos segmentos sem nos preocuparmos com a relevância biológica da inserção ou não de um segmento na solução.

Durante a execução das ferramentas tivemos alguns problemas com a Twinscan e a AGenDA. Para a Twinscan, não conseguimos obter respostas para um total de 11 sequências. Nesses casos, era devolvido um erro alertando que a ferramenta RepeatMasker, utilizada pelo Twinscan, não havia encontrado sequências repetitivas na sequência sendo analisada, fazendo assim que a execução do Twinscan fosse encerrada, não devolvendo uma resposta. Já com a AGenDA o problema foi um pouco maior, abrangendo 46 pares de sequências. Não sabemos as causas do erro ocorrido nesses casos, uma vez que a ferramenta foi executada via web e não devolvia muitas informações. Para o cálculo dos valores da Tabela 5.1, consideramos que as ferramentas Twinscan e AGenDA não encontraram gene algum para as sequências que não conseguiram processar.

Na intenção de deixar os resultados mais justos, excluímos do conjunto de testes os 46 pares de sequências que não foram processados pela AGenDA. Da mesma forma, foram retirados os 11 pares de sequências em que uma das sequências não foi processada pela Twinscan. Desconsiderando esses pares de sequências, passamos a ter um conjunto de testes com 131 pares (três pares não conseguiram ser processados por ambas as ferra-

mentas). Na Tabela 5.2 podemos ver as médias aritméticas das medidas de avaliação calculadas considerando esse novo conjunto de testes.

Ferramenta	Sp_n	Sn_n	CA	Sp_e	Sn_e	Ave	Sp_b	Sn_b	Avb
GenePredictor	0,938	0,958	0,941	0,599	0,752	0,675	0,671	0,847	0,759
AGenDA	0,840	0,661	0,655	0,441	0,422	0,432	0,567	0,537	0,552
Progen	0,856	0,958	0,886	0,483	0,571	0,527	0,575	0,731	0,653
Sgp2	0,888	0,850	0,828	0,522	0,505	0,513	0,667	0,652	0,660
Twinscan	0,919	0,797	0,812	0,631	0,520	0,575	0,749	0,627	0,687

Tabela 5.2: Qualidade das predições obtidas pelas ferramentas considerando o conjunto com 131 pares de sequências.

Podemos perceber nessa segunda rodada de testes que, mesmo com algumas mudanças nos valores da Tabela 5.2 com relação à Tabela 5.1, a GenePredictor continua levando vantagem sobre as outras, perdendo novamente apenas na especificidade, nos níveis de éxons e borda, para a Twinscan, mas empatando na sensibilidade no nível de nucleotídeos com a Progen.

Pelos resultados mostrados até o momento, podemos notar que a GenePredictor levou uma certa vantagem sobre as outras ferramentas. Essa vantagem poderia ser atribuída às características do conjunto de testes, mais especificamente à restrição de igualdade no número de éxons das sequências de um mesmo par. Essa característica pode facilitar a predição por parte da GenePredictor já que ela foi implementada com base na solução proposta para o PAPS. As outras ferramentas porém não possuem nenhuma restrição do tipo.

No intuito de determinarmos o quanto a restição de igualdade no número de éxons influencia o resultado da nossa ferramenta, criamos um novo conjunto de testes cujos pares possuem as mesmas características apresentadas na Seção 5.2, exceto pelo fato de que as duas sequências de cada par possuem, obrigatoriamente, uma quantidade de éxons diferentes. Esse novo conjunto de testes possui 37 pares de sequências. As médias das avaliações feitas sobre os resultados obtidos com esse novo conjunto de sequências podem ser vistas na Tabela 5.3.

Ferramenta	Sp_n	Sn_n	CA	Sp_e	Sn_e	Ave	Sp_b	Sn_b	Avb
GenePredictor	0,906	0,840	0,861	0,508	0,534	0,521	0,601	0,637	0,619
AGenDA	0,378	0,198	0,151	0,063	0,047	0,055	0,154	0,115	0,135
Progen	0,809	0,855	0,809	0,311	0,441	0,376	0,412	0,589	0,501
Sgp2	0,848	0,675	0,718	0,484	0,365	0,424	0,622	0,490	0,556
Twinscan	0,861	0,720	0,765	0,464	0,338	0,401	0,654	0,461	0,558

Tabela 5.3: Qualidade das predições obtidas pelas ferramentas considerando o conjunto com 37 pares de sequências.

Observando a Tabela 5.3 podemos notar que, de um modo geral, a GenePredictor ainda continua melhor que as outras ferramentas. Perdemos apenas na sensibilidade no nível de nucleotídeos para a Progen e na especificidade no nível de bordas para a

Twinscan. Nessa rodada de testes as ferramentas AgenDA e Twinscan também apresentaram problemas no processamento de alguns pares de sequências. Como ocorrido anteriormente, a exclusão desses pares não afetou o resultado final da avaliação. Esse novo conjunto de testes e os resultados obtidos mostram que, mesmo com a restrição imposta pela formulação desenvolvida de igualdade no número de éxons, a GenePredictor também apresenta bons resultados quando genes com números diferentes de éxons precisam ser identificados.

Além da melhora das predições em si, a GenePredictor também se mostrou mais rápida para devolver uma resposta, seguida pela Sgp2, Twinscan e Progen, respectivamente. Não foi possível computar o tempo gasto pela AGenDA, uma vez que ela foi executada via web e não nos devolvia essa informação. Considerando o conjunto de testes com os 185 pares de sequências, a GenePredictor devolveu uma resposta para todos os pares de sequências em aproximadamente 11 minutos de execução, o que vem a ser cerca de 10 vezes mais rápido que a Sgp2, 93 vezes mais rápido que a Twinscan e 132 vezes mais rápido que a Progen para executar a mesma tarefa. Os testes realizados via web foram feitos utilizando o navegador Mozilla Firefox 3.6.10. Todos os outros testes foram executados em uma máquina com 4 Gb de memória RAM e processador Core 2 Duo de 2.4 GHz.

Capítulo 6

Conclusão

O Problema da Identificação de Genes ainda não possui uma solução computacional totalmente confiável. Com isso em mente, a idéia principal deste trabalho foi estudar esse problema na intenção de obter resultados melhores que os obtidos até o momento por ferramentas já existentes. Dentre os vários métodos existentes na tentativa de solucionar o problema da identificação de genes, optamos por aquele baseado na comparação entre duas sequências relacionadas evolutivamente, procurando por éxons em ambas as sequências.

Modelamos o problema através de um problema matemático definido por nós e denominado de Problema do Alinhamento de Pares de Segmentos (PAPS). Este é um problema da Otimização Combinatória para o qual propusemos uma solução algorítmica. Para alcançar nosso objetivo, utilizamos como entrada para o algoritmo que resolve o PAPS dados provenientes de duas sequências genômicas homólogas. Primeiramente essas sequências foram pré-processadas pela ferramenta GENSCAN, a qual nos forneceu um conjunto de segmentos, retirados de cada uma das sequências, com alta chance de serem éxons nessas sequências. Fazendo uso do princípio da conservação das bases, os candidatos a éxons são comparados e então agrupados de tal forma que a soma da similaridade entre eles seja a maior possível. Tudo isso é feito em uma matriz quadridimensional que ao ser preenchida nos devolve um subconjunto de éxons como sendo os mais prováveis de serem reais.

No Capítulo 5 apresentamos os resultados obtidos com os testes comparativos entre a GENEPREDICTOR e outras ferramentas disponíveis na literatura. Analisando os resultados pudemos notar que a GENEPREDICTOR teve alguns problemas com algumas instâncias, uma vez que estávamos interessados em encontrar a maior soma de similaridade entre os segmentos das cadeias. Isso acarretou erros como, por exemplo, escolher segmentos a mais do que a quantidade de éxons anotados da sequência. Porém, mesmo assim, a GENEPREDICTOR se sobressaiu em relação às outras no que diz respeito à média das medidas de especificidade e sensibilidade em todos os níveis analisados. Isso ocorreu até mesmo nos casos em que o número de éxons eram diferentes nos genes sendo procurados. Além de uma melhora nos resultados, também tivemos um grande ganho no tempo de execução da ferramenta.

A estratégia utilizada por nós pode motivar alguns trabalhos no futuro onde, além de simplesmente procurar maximizar o valor da soma da similaridade entre os segmentos

das cadeias, também verifique se a inclusão daquele segmento na solução vai ser, do ponto de vista biológico, uma boa escolha. Dessa forma estaríamos utilizando uma estrutura mais consistente na busca pelos genes, e menos propensa a erros. Uma outra abordagem que pode vir a ser estudada futuramente seria buscar por cadeias cuja similaridade da concatenação dos seus segmentos seja máxima. Dessa forma, os genes buscados nas sequências poderiam ter quantidade de éxons diferentes. Além disso, o desenvolvimento de uma ferramenta que substitua a GENSCAN na tarefa de gerar os éxons candidatos também constitui uma possibilidade para a melhora dos resultados aqui obtidos.

A partir das tarefas empreendidas e dos resultados obtidos podemos concluir que o objetivo proposto inicialmente foi atingido, com o desenvolvimento de uma formulação adequada para o Problema da Identificação de Genes e de uma ferramenta de predição precisa e eficiente.

Apêndice A

Conjunto de Sequências

Na Tabela A.1 apresentamos informações sobre os 185 pares de sequências inicialmente utilizados. A coluna Gene corresponde ao nome do gene na espécie *Homo sapiens*. |HSap| e |Hom| equivalem, respectivamente, ao tamanho da sequência do *Homo sapiens* e da sequência homóloga, já contando os acréscimos de 1000 bases antes e depois da sequência do gene. EsHom indica de qual espécie origina a sequência homóloga utilizada. Cada espécie é representada pela sigla das suas iniciais, dessa forma temos que *mm* equivale à *Mus musculus*, *rn* à *Rattus norvegicus*, *bt* à *Bos taurus*, *clf* à *Canis lupus familiaris*, *dr* à *Danio rerio* e *pt* à *Pan troglodytes*. ER é a quantidade de éxons anotados em cada uma das sequências (lembrando que as duas sequências contêm o mesmo número de éxons). |B| e |C| são as quantidades de possíveis éxons gerados pela GENSCAN para a sequência do *Homo sapiens* e para a homóloga, respectivamente. EP é a quantidade de éxons preditos pela GENEPREDICTOR (mesma quantidade em ambas sequências). Por fim, ECB e ECC são as quantidades de éxons reais que foram corretamente preditos pela GENEPREDICTOR na sequência do *Homo sapiens* e na sequência homóloga, respectivamente.

Os pares de sequências não processados (apenas) pela ferramenta AGENDA são identificados pelo símbolo * logo após seu nome na coluna Gene da Tabela A.1. Os pares de sequências não processados (apenas) pela ferramenta TWINSCAN são identificados pelo símbolo **. Já os pares não executados em ambas ferramentas, são indicados por ***.

Tabela A.1: Especificação dos 185 pares de genes que fazem parte do conjunto de testes.

Gene	|HSap|	|Hom|	EsHom	ER	|B|	|C|	EP	ECB	ECC
aff4	90284	73008	*mm*	20	99	74	25	17	17
ankrd10*	38530	26107	*mm*	6	45	43	7	2	2
ankrd43	5457	5618	*mm*	1	2	2	2	1	1
arhgdig	4398	4168	*mm*	6	21	20	7	6	6
ascc2*	51655	47593	*mm*	19	80	102	20	16	17
asz1*	66302	60084	*mm*	13	84	86	13	10	10
bet1	14691	12024	*mm*	4	26	22	4	2	2
bgn	16594	14288	*mm*	7	80	20	8	6	6

Continua na próxima página...

Tabela A.1 – Continuação

Gene	|HSap|	|Hom|	EsHom	ER	|B|	|C|	EP	ECB	ECC
c17orf50	6183	5521	*mm*	3	16	19	4	2	2
c21orf45	12847	10060	*mm*	5	44	15	4	3	3
c21orf59	12572	10981	*mm*	7	46	33	7	7	7
c21orf63*	104946	104020	*bt*	8	172	153	12	5	5
c7orf68	4589	4775	*mm*	1	13	16	2	0	0
capza2*	58751	31437	*mm*	10	41	45	11	9	9
cav1*	38392	36955	*mm*	3	42	61	3	2	2
ccdc88b	19312	15589	*mm*	27	68	65	27	24	24
ccdc93*	100551	70367	*rn*	24	176	138	24	20	20
ccl5	10883	6719	*mm*	3	9	17	4	2	2
cdx2	9040	8350	*mm*	3	37	26	3	2	2
chmp2a	5554	4726	*mm*	5	12	20	5	3	3
cldn12	14473	11765	*mm*	1	29	24	3	1	1
csf2	4375	4630	*mm*	4	9	7	4	0	4
ctgf	5203	5174	*mm*	5	11	9	6	4	4
ctsd	13238	13955	*mm*	9	46	22	10	9	9
ddx43*	25005	29813	*mm*	16	43	58	16	13	13
decr2	12630	10954	*mm*	8	44	23	9	7	7
dnajc4	6001	6362	*mm*	6	28	20	7	3	3
dppa5	3167	3099	*mm*	3	19	9	3	2	1
drg1*	36634	18465	*mm*	9	32	18	10	9	9
dusp18	7834	8016	*mm*	1	16	25	2	1	1
eef1a1	7283	4916	*bt*	7	15	12	8	6	6
esrra	13167	12832	*mm*	6	39	36	9	4	5
fam71f1	18355	18883	*mm*	7	20	44	2	1	1
fkbp2	5195	4692	*mm*	5	18	18	6	4	4
flt3***	99319	71749	*mm*	24	114	153	25	21	20
frmd6*	243591	78721	*mm*	13	209	82	16	12	13
frs3	11717	11080	*mm*	5	29	18	7	3	4
gabrq	17189	15109	*mm*	9	26	22	9	8	8
gal3st1	12253	17603	*mm*	2	27	27	6	1	1
gas2l2	10361	9554	*mm*	6	24	24	6	5	5
gdf9	5600	6523	*mm*	2	5	9	3	2	2
gng11	6811	6459	*mm*	2	5	4	2	2	2
gng2*	111469	106691	*mm*	2	91	117	5	0	0
gngt1	6666	5425	*mm*	2	5	14	3	1	1
gpr137	5633	5410	*mm*	7	40	40	9	6	6
gsx1	3310	4252	*mm*	2	10	14	3	2	2
hba1	2842	2820	*mm*	3	10	5	4	2	2
hba2	2864	2820	*mm*	3	14	5	4	2	2
hbb	3606	3398	*mm*	3	6	5	4	3	3

Continua na próxima página...

Tabela A.1 – Continuação

Gene	|HSap|	|Hom|	EsHom	ER	|B|	|C|	EP	ECB	ECC
hbe1	3794	3454	*mm*	3	6	6	4	3	3
hbg1	3586	3525	*mm*	3	4	5	4	3	3
hbg2	3591	3629	*rn*	3	4	5	4	3	3
hbq1	2844	2805	*mm*	3	6	6	3	3	3
hbz	3651	3512	*mm*	3	11	7	4	2	2
hormad2*	98610	96700	*mm*	10	85	138	8	5	5
hoxa2	4422	4321	*mm*	2	4	10	3	1	1
hoxa4	4274	4017	*mm*	2	6	7	3	2	2
hoxa5	4292	4834	*mm*	2	8	6	3	1	1
hoxa6	4253	4260	*mm*	2	6	3	2	2	2
hoxa7	4962	4951	*mm*	2	15	15	5	2	2
hunk*	132750	115157	*mm*	11	127	170	16	7	7
il13	4937	5379	*mm*	4	15	10	4	2	4
il3	4550	4548	*pt*	5	13	10	9	4	4
il5	4079	6310	*mm*	4	8	14	5	3	3
ins**	3431	3048	*mm*	2	4	3	3	2	2
insig2	23548	15284	*bt*	5	33	38	4	4	4
irf1	11165	9246	*mm*	9	24	31	10	9	8
itfg3	33319	10288	*bt*	11	94	53	11	11	11
kcnk4	10711	10826	*mm*	6	36	33	8	6	6
leap2	3225	2956	*mm*	3	8	15	3	3	3
lep	18352	15656	*mm*	2	15	30	3	1	1
lif	8355	4775	*rn*	3	17	15	3	3	3
lrrc4	5879	5622	*mm*	1	2	4	2	1	1
lyzl6	6986	9802	*mm*	4	17	18	4	1	1
magea12	5880	3589	*pt*	1	14	4	3	1	1
map3k1***	83080	64551	*mm*	20	104	119	25	17	17
march11	114424	9542	*dr*	4	149	25	4	3	3
mettl2b	28196	15915	*mm*	9	23	24	10	9	8
mmp26	6236	6066	*clf*	6	48	17	5	4	3
morc2*	43588	22731	*bt*	23	57	84	23	20	19
moxd1*	107471	81267	*mm*	12	153	179	18	12	12
mrpl23	11338	9626	*mm*	5	39	30	6	5	5
mzf1	13659	13516	*mm*	5	29	26	6	4	4
nfxl1*	69377	48342	*mm*	22	82	87	20	16	16
nipal1*	22294	25283	*mm*	6	81	32	8	6	6
nme4	5563	5726	*mm*	5	16	13	5	5	5
nr2e1*	24799	23618	*mm*	9	41	56	9	7	7
nsdhl*	40397	42008	*mm*	7	41	55	7	6	6
of51f1	2939	2951	*mm*	1	8	7	2	0	1
ooep	3238	3184	*mm*	3	17	7	2	0	0

Continua na próxima página...

Tabela A.1 – Continuação

Gene	|HSap|	|Hom|	EsHom	ER	|B|	|C|	EP	ECB	ECC
or51a2	2942	2954	*mm*	1	9	10	2	1	0
or51a4	2942	2954	*mm*	1	10	10	2	1	0
or51a7	2939	2939	*mm*	1	8	7	2	1	1
or51b2	3055	3662	*mm*	1	5	3	1	1	1
or51b4	2933	2936	*mm*	1	2	5	2	1	1
or51b6	2939	6080	*mm*	1	3	12	2	1	1
or51f2	3029	2942	*mm*	1	5	7	2	0	0
or51g1	2966	2942	*mm*	1	12	5	2	1	1
or51g2	2945	2939	*mm*	1	4	10	2	1	0
or51i1	2945	2945	*mm*	1	5	6	2	1	1
or51i2	2939	2939	*mm*	1	4	3	2	1	1
or51l1	2948	2948	*rn*	1	2	6	2	1	1
or51m1	2981	16850	*mm*	1	7	46	2	1	1
or51q1	2954	2966	*mm*	1	3	5	2	1	0
or51s1	2972	2969	*mm*	1	7	11	2	1	1
or51t1	3065	3041	*mm*	1	4	9	2	0	0
or52a1**	2939	2948	*mm*	1	13	6	2	1	1
or52a5	2951	2951	*mm*	1	4	5	1	0	0
or52d1**	2957	2939	*mm*	1	4	4	2	1	1
or52e2**	2978	2954	*mm*	1	5	8	2	1	1
or52h1	2963	2951	*mm*	1	11	9	2	1	0
or52j3	2936	2939	*mm*	1	3	10	2	1	1
or52r1**	3185	2945	*mm*	1	3	8	2	0	1
osbp2*	215019	162173	*mm*	14	216	182	20	9	10
osm	6023	6607	*mm*	3	33	17	3	2	2
ostm1*	35329	25539	*mm*	6	40	38	6	5	5
pdk4	15117	14928	*mm*	11	20	28	12	11	11
pdx1	8284	7722	*mm*	2	22	9	3	2	2
pes1	17283	18030	*mm*	15	63	34	16	15	15
pex12	5843	6333	*mm*	3	15	15	4	2	2
pgc	12670	9637	*mm*	9	25	21	10	9	9
pla2g3	7677	13052	*mm*	7	23	42	8	6	5
plcb3*	17907	18040	*mm*	31	67	80	33	29	27
pnma3	6062	5405	*mm*	1	29	14	3	1	1
pnma5	5394	5121	*mm*	1	24	5	1	1	0
polr3k	8626	8471	*mm*	3	13	17	3	2	2
pon1*	28216	27720	*mm*	9	50	53	10	8	8
pon3*	38504	37382	*mm*	9	42	48	9	7	7
ppp1r14b	4463	4277	*mm*	4	10	6	5	4	4
ppp1r3a*	44201	43453	*mm*	4	39	60	6	3	2
prhoxnb	12532	9357	*mm*	2	37	31	3	2	1

Continua na próxima página...

Tabela A.1 – Continuação

Gene	|HSap|	|Hom|	EsHom	ER	|B|	|C|	EP	ECB	ECC
prickle4	8611	6069	*rn*	6	41	36	6	5	5
rasl10b	13862	12967	*mm*	3	11	17	3	2	2
rnf152***	80001	78740	*mm*	1	97	79	5	1	1
rps5	9536	6350	*mm*	5	23	23	6	4	4
samd9	20492	7016	*rn*	1	12	6	2	0	0
samd9l*	20313	29314	*mm*	1	25	15	3	1	1
sec14l3	14799	13776	*mm*	12	29	28	13	10	10
sec14l4*	18783	18242	*mm*	12	31	61	12	11	11
selm	4789	4650	*mm*	5	21	16	5	2	2
shroom1	5991	12551	*mm*	7	22	22	8	4	5
skap2*	199655	155385	*mm*	12	216	195	19	8	8
slc22a4*	51755	46965	*mm*	10	95	87	12	9	8
slc22a5*	27906	29119	*mm*	10	57	32	11	9	9
slc25a13*	203928	177901	*mm*	18	239	213	26	11	11
slc27a5	15733	11798	*mm*	10	19	23	10	10	10
slc35e4	14070	9643	*mm*	2	36	42	4	1	1
slc4a3	16411	14784	*mm*	22	65	55	25	19	19
slfn13	15742	20641	*mm*	4	39	49	4	4	4
snrnp25	5841	5570	*mm*	5	15	23	5	4	4
snx19*	42617	41398	*mm*	11	80	74	12	8	7
spp2*	28431	21420	*mm*	7	53	46	7	3	4
steap1	12453	14990	*mm*	4	35	30	3	2	2
stip1*	20434	21237	*mm*	14	38	34	14	13	13
taf15*	39751	35633	*mm*	16	62	44	15	13	13
taf4b*	167241	119115	*mm*	15	172	139	15	11	11
tbc1d10a*	36916	30673	*mm*	9	83	44	10	8	8
tcn2	21887	16394	*mm*	9	23	26	9	8	8
tfeb	53083	8673	*mm*	8	86	32	9	6	7
tfpi2	6321	7760	*mm*	5	13	21	5	3	3
tmem8	13175	11938	*mm*	13	56	51	14	12	12
tnni2**	4006	3938	*mm*	7	18	11	8	6	6
tomm6**	4454	3331	*bt*	2	31	8	2	1	1
trex2	3768	3626	*mm*	1	22	6	2	0	0
trim28	8247	8869	*mm*	17	34	43	18	17	16
trpm8*	104124	86662	*rn*	24	130	142	24	17	17
ube2m	5265	4741	*mm*	6	21	17	5	4	4
ubqln3	4624	4650	*mm*	1	9	3	2	1	1
ubqlnl**	4334	4288	*mm*	1	5	4	1	0	0
uqcrq	4218	4893	*mm*	2	16	9	2	1	1
urb1*	83983	62684	*rn*	39	102	119	38	37	37
vegfb	5994	7132	*mm*	6	37	41	9	5	5

Continua na próxima página...

Tabela A.1 – Continuação

Gene	|HSap|	|Hom|	EsHom	ER	|B|	|C|	EP	ECB	ECC
wnt2*	48659	43417	*mm*	5	83	73	8	4	4
zbtb45	8025	6135	*mm*	2	26	22	4	2	2
zfp92	5306	17885	*mm*	4	6	26	3	3	2
zmat5*	38025	34989	*mm*	5	32	66	5	5	5
znf135	12165	14570	*bt*	4	9	29	5	4	4
znf256	8877	15654	*bt*	3	18	47	3	2	2
znf329	26454	17083	*mm*	1	25	10	3	0	0
znf446	6803	8462	*mm*	6	38	21	5	5	4
znf606*	28275	25653	*rn*	6	59	22	4	3	3
znf622	16267	16117	*mm*	6	21	25	7	6	6
znf8	18937	14216	*mm*	4	36	31	5	2	2
zscan22	17328	12425	*mm*	2	27	21	3	2	2
zscan4	12218	5164	*rn*	3	6	5	3	2	2

Referências Bibliográficas

[1] Adi, S. S., and Ferreira, C. E. Uma avaliação de ferramentas para a predição de genes. In *Anais do XXII Congresso da Sociedade Brasileira de Computação* (2002), vol. 3, pp. 133–143.

[2] Adi, S. S., and Ferreira, C. E. Gene prediction by multiple syntenic alignment. *Journal of Integrative Bioinformatics 2*, 1 (2005).

[3] Adi, S. S., and Ferreira, C. E. Gene prediction by syntenic alignment. In *Advances in Bioinformatics and Computational Biology*, J. C. Setubal and S. Verjovski-Almeida, Eds., vol. 3594 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2005, pp. 246–250.

[4] Akhtar, M., Epps, J., and Ambikairajah, E. Signal Processing in Sequence Analysis: Advances in Eukaryotic Gene Prediction. *IEEE Journal of Selected Topics in Signal Processing, vol. 2, issue 3, pp. 310-321 2* (June 2008), 310–321.

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

[6] Arquès, D., Lacan, J., and Michel, C. Identification of protein coding genes in genomes with statistical functions based on the circular code. *BioSystems 66*, 1-2 (2002), 73–92.

[7] Batzoglou, S., Pachter, L., Mesirov, J. P., Berger, B., and Lander, E. S. Human and Mouse Gene Structure: Comparative Analysis and Application to Exon Prediction. *Genome Research 10*, 7 (2000), 950–958.

[8] Baum, L. E., Petrie, T., Soules, G., and Weiss, N. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics 41*, 1 (1970), 164–171.

[9] Baxevanis, A. D., and Ouellette, B. F. F. *Bioinformatics: A Pratical Guide to the Analysis of Genes and Proteins*, 2nd ed. John Wiley & Sons, Inc., 2001.

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

[11] Brudno, M., and Morgenstern, B. Fast and sensitive alignment of large genomic sequences. In *CSB '02: Proceedings of the IEEE Computer Society Conference on Bioinformatics* (2002), IEEE Computer Society, pp. 138 – 147.

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

[13] Burge, C., and Karlin, S. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology 268* (1997), 78–94.

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

[15] Consortium, T. E. P. Identification and analysis of functional elements in 1% of the human genome by the encode pilot project. *Nature 447*, 7146 (Jun 2007), 799–816. 10.1038/nature05874.

[16] Cooper, G. M., and Hausman, R. E. *A Célula - Uma Abordagem Molecular*, 3a. ed. Artmed, 2007.

[17] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms*, 2nd ed. MIT Press and McGraw-Hill, Cambridge - MA, USA, 2001.

[18] The Eclipse Foundation. *Eclipse CDT (C/C++ Development Tooling)*. http://www.eclipse.org/cdt/ (último acesso em 5 de março de 2010).

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

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

[21] Gelfand, M. S., Mironov, A. A., and Pevzner, P. A. Gene recognition via spliced sequence alignment. *Proceedings of the National Academy of Sciences of the United States of America 93*, 17 (1996), 9061–9066.

[22] Gish, W. Wu-blast package. http://blast.wustl.edu/ (último acesso em 17 de agosto de 2010).

[23] Gish, W., and States, D. Identification of protein coding regions by database similarity search. *Nature Genetics 3* (1993), 266–272.

[24] Gremme, G., Brendel, V., Sparks, M., and Kurtz, S. Engineering a software tool for gene structure prediction in higher organisms. *Information and Software Technology 47*, 15 (2005), 965–978.

[25] Guigo, R., Flicek, P., Abril, J., Reymond, A., Lagarde, J., Denoeud, F., Antonarakis, S., Ashburner, M., Bajic, V., Birney, E., Castelo, R., Eyras, E., Ucla, C., Gingeras, T., Harrow, J., Hubbard, T., Lewis, S., and Reese, M. EGASP: the human ENCODE Genome Annotation Assessment Project. *Genome Biology 7*, Suppl 1 (2006), S2.

[26] Guigo, R., Knudsen, S., Drake, N., and Smith, T. Prediction of gene structure. *Journal of Molecular Biology 226*, 1 (1992), 141–157.

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

[28] He, X., and Goldwasser, M. Identifying conserved gene clusters in the presence of orthologous groups. In *RECOMB* (2004), P. E. Bourne and D. Gusfield, Eds., ACM, pp. 272–280.

[29] Jones, N. C., and Pevzner, P. A. *An Introduction to Bioinformatics Algorithms*, 1st ed. The MIT Press, 2004.

[30] Junqueira, L. C. U., and Carneiro, J. *Biologia Celular e Molecular*, 6 ed. Guanabara Koogan, Rio de Janeiro, 1997.

[31] Korf, I. Gene finding in novel genomes. *BMC Bioinformatics 5*, 1 (2004), 59.

[32] Korf, I., Flicek, P., Duan, D., and Brent, M. R. Integrating genomic homology into gene structure prediction. *Bioinformatics 17* (2001), 140–148.

[33] Kulp, D., Haussler, D., Reese, M. G., and Eeckman, F. H. A generalized hidden Markov model for the recognition of human genes in DNA. In *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology* (1996), AAAI Press, pp. 134–142.

[34] Lewin, B. *Genes VII.* Artmed, 2001.

[35] Lewis, J., Walter, P., Johnson, A., Hopkin, K., Bray, D., Raff, M., Roberts, K., and Alberts, B. *Fundamentos da Biologia Celular*, 2 ed. Artmed, 2006.

[36] Liu, Q., Mackey, A. J., Roos, D. S., and Pereira, F. C. N. Evigan: a hidden variable model for integrating gene evidence for eukaryotic gene prediction. *Bioinformatics 24*, 5 (2008), 597–605.

[37] Lodish, H., Berk, A., Matsudaira, P., Kaiser, C. A., Krieger, M., Scott, M. P., Zipursky, L., and Darnell, J. *Molecular Cell Biology*, 5 ed. W.H. Freeman Custom Publishing, 2004.

[38] Majoros, W. H., Pertea, M., and Salzberg, S. L. TigrScan and GlimmerHMM: two open source ab initio eukaryotic gene-finders. *Bioinformatics 20*, 16 (2004), 2878–2879.

[39] Miller, F., Vandome, A., and McBrewster, J. *Hidden Markov Model: Markov Process, Markov Chain, Time, Speech Recognition, Handwriting Recognition, Gesture Recognition, Part-of-speech Tagging, Sheet Music, Partial Discharge, Bioinformatics, Bayesian Inference*. Alphascript Publishing, 2010.

[40] Morgenstern, B. DIALIGN: multiple DNA and protein sequence alignment at BiBiServ. *Nucleic Acids Research 32* (2004), W33–36.

[41] Morgenstern, B., Rinner, O., Abdeddaim, S., Haase, D., Mayer, K. F. X., Dress, A. W. M., and Mewes, H.-W. Exon discovery by genomic sequence alignment. *Bioinformatics 18*, 6 (2002), 777–787.

[42] Mount, D. W. *Bioinformatics: Sequence and Genome Analysis*, 1st ed. Cold Spring Harbor Laboratory Press, 2001.

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

[44] Novichkov, P. S., Gelfand, M. S., and Mironov, A. A. Gene recognition in eukaryotic DNA by comparison of genomic sequences. *Bioinformatics 17*, 11 (2001), 1011–1018.

[45] Orengo, C. A., Jones, D. T., and Thornton, J. M. *Bioinformatics: genes, proteins and computers*. BIOS Scientific Publishers Limited, 2003.

[46] Parra, G., Agarwal, P., Abril, J. F., Wiehe, T., Fickett, J. W., and Guigó, R. Comparative Gene Prediction in Human and Mouse. *Genome Research 13*, 1 (2003), 108–117.

[47] Powell, B. C., and III, C. A. H. Similarity-based gene detection: using cogs to find evolutionarily-conserved orfs. *BMC Bioinformatics 7* (2006), 31.

[48] Rabiner, L. R. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE* (1989), pp. 257–286.

[49] Robertis, E. M. F. D., and Hib, J. *Bases da Biologia Celular e Molecular*, 4 ed. Guanabara Koogan, 2006.

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

[51] Sayers, E. W., Barrett, T., Benson, D. A., Bolton, E., Bryant, S. H., Canese, K., Chetvernin, V., Church, D. M., Cuccio, M. D., Federhen, S., Feolo, M., Geer, L. Y., Helmberg, W., Kapustin, Y., Landsman, D., Lipman, D. J., Lu, Z., Madden, T. L., Madej, T., Maglott, D. R., , Marchler-Bauer, A., Miller, V., Mizrachi, I., Ostell, J., Panchenko, A., Pruitt, K. D., Schuler, G. D., Sequeira, E., Sherry, S. T.,

Shumway, M., Sirotkin, K., Slotta, D., Souvorov, A., Starchenko, G., Tatusova, T. A., Wagner, L., Wang, Y., Wilbur, W. J., Yaschenko, E., and Ye, J. Database resources of the national center for biotechnology information. *Nucleic Acids Research 38* (2010), D5–D16.

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

[53] Smit, A., Hubley, R., and Green, P. Repeatmasker. http://repeatmasker.org (último acesso em 22 de setembro de 2010).

[54] Staden, R. Computer methods to locate signals in nucleic acid sequences. *Nucleic Acids Research 12*, 1Part2 (1984), 505–519.

[55] Staden, R., and McLachian, A. Codon preference and its use in identifying protein coding regions in long DNA sequences. *Nucleic Acids Research 10*, 1 (1982), 141–156.

[56] Stanke, M., Schoffmann, O., Morgenstern, B., and Waack, S. Gene prediction in eukaryotes with a generalized hidden markov model that uses hints from external sources. *BMC Bioinformatics 7*, 1 (2006), 62.

[57] Taher, L., Rinner, O., Garg, S., Sczyrba, A., Brudno, M., Batzoglou, S., and Morgenstern, B. AGenDA: homology-based gene prediction. *Bioinformatics 19*, 12 (2003), 1575–1577.

[58] Tucker, A. B. *Computer Science Handbook*, 2nd ed. Chapman & Hall/CRC, 2004.

[59] Usuka, J., Zhu, W., and Brendel, V. Optimal spliced alignment of homologous cDNA to a genomic DNA template. *Bioinformatics 16*, 3 (2000), 203–211.

[60] Viterbi, A. G. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions Information Theory*, 13 (1967), 260–269.

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

[62] Wu, J., and Haussler, D. Coding exon detection using comparative sequences. *Journal of Computational Biology 13*, 6 (2006), 1148–1164.