Um Algoritmo para Comparação Sintática de Genomas baseado na Complexidade Condicional de Kolmogorov

Marcelo Cezar Pinto

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

Um Algoritmo para Comparação Sintática de Genomas baseado na Complexidade Condicional de Kolmogorov

Marcelo Cezar Pinto[1]

Fevereiro de 2002

Banca Examinadora:

- João Meidanis (Orientador)
- José Coelho de Pina Júnior
  IME - USP
- João Carlos Setubal
  IC - UNICAMP
- Arnaldo Vieira Moura (Suplente)
  IC - UNICAMP

[1]**Projeto financiado pela FAPESP, sob número de processo 00/04776-3.**

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Pinto, Marcelo Cezar

P658a Um algoritmo para comparação sintática de genomas baseado na complexidade condicional de Kolmogorov / Marcelo Cezar Pinto -- Campinas, [S.P. :s.n.], 2002.

Orientador: João Meidanis

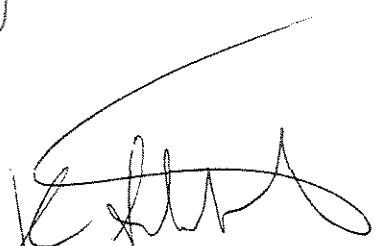
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Teoria da computação. 2. Biologia molecular. 3. Kolmogorov, Complexidade de. I. Meidanis, João. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

TERMO DE APROVAÇÃO

Tese defendida e aprovada em 12 de março de 2002, pela Banca Examinadora composta pelos Professores Doutores:

Prof. Dr. José Coelho de Pina Júnior
IME - USP



Prof. Dr. João Carlos Setubal
IC - UNICAMP

Prof. Dr. Joao Meidanis
IC - UNICAMP

Um Algoritmo para Comparação Sintática de Genomas baseado na Complexidade Condicional de Kolmogorov

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Marcelo Cezar Pinto e aprovada pela Banca Examinadora.

Campinas, 12 de março de 2002.

João Meidanis (Orientador)

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

"Bar, quando usado na dose e na medida certa,
é também uma esplêndida fonte de boas idéias."
Caninha 51

Agradecimentos

Em fevereiro de 2000 entrei em contato pela primeira vez com a área de Biologia Computacional. Até então, só tinha assistido a uma palestra em 1997 e lido alguma coisa em páginas da Internet. Não havia imaginado antes que poderia trabalhar nesta área. Entretanto, ao conversar com o professor João Meidanis, achei tão interessante o que me era apresentado que em alguns dias me convenci do quão bom seria abraçar esta nova área.

Por isso, agradeço em primeiro lugar ao meu orientador, João Meidanis, que me ajudou a iniciar meus estudos em Biologia Computacional e que confiou em minha capacidade e trabalho (espero ter correspondido!). Além disso, pelos comentários e sugestões durante nossas reuniões.

Ainda no campo profissional-acadêmico, agradeço aos colegas do LBI (Guilherme, Kita, Lin, Marília, Patrícia, Vagner, Zanoni e Zé Augusto) e do Instituto de Computação pelas ótimas discussões técnico-científicas. Sem falar no apoio da FAPESP (Fundação de Apoio à Pesquisa do Estado de São Paulo) ao me conceder uma bolsa de mestrado.

Como não se vive para estudar (estudo para viver!), vai o meu muito obrigado às festas do LBI e aos forrós da Cooperativa Brasil. Sem esquecer da galera do futsal e das inúmeras conversas nos corredores do IC quando não havia luz para trabalhar nos laboratórios.

Também sou muito grato aos companheiros de morada, Vinícius e Vanessa, que sempre me deram forças para prosseguir e descontração para não enlouquecer. Agradeço também a gauchada desgarrada que se instalou por estas bandas e compartilhou a mesma roda de chimarrão (Amanda, Celio, César, Emilene, Glaidson, Gonçalo, Gustavo, Jeferson, Luciana, Márcio, Mário, Ricardo, Tatiane, Vanessa, Vinícius Garcia e Montagner).

Agradeço ao CTT® (Conselho Tutorial do Trago) pelas cachaças e pelo *rock'n'roll* e ao PET-Informática/UFSM pela formação.

E, lá de longe, sempre presentes no coração, agradeço aos meus pais Sérgio e Elisa, a minha irmã Gisele e a minha namorada Giovana. Sem o apoio deles, nada teria conseguido. Amo vocês.

Resumo

Desde 1953, quando Watson e Crick desvendaram a estrutura do DNA (ácido desoxirribonucleico), a área de Biologia Molecular tem avançado rapidamente. Técnicas que permitem a manipulação de biomoléculas foram criadas e aperfeiçoadas desde então, gerando enormes quantidades de dados. A necessidade de processar estas informações criou um novo campo chamado de Biologia Molecular Computacional, o qual consiste em desenvolver e usar técnicas matemáticas e de computação para ajudar a resolver problemas de Biologia Molecular. Existem problemas desta área relacionados a Comparação de Genomas, que consiste, a grosso modo, em analisar e comparar seqüências de ácidos nucleicos ou aminoácidos entre espécies. A comparação de genomas busca desvendar as relações existentes entre diferentes espécies. A descoberta de genes ou porções semelhantes nos genomas pode indicar proximidade evolutiva ou regiões indispensáveis à existência da vida. Por outro lado, as diferenças podem relacionar o comportamento particular de uma espécie com uma determinada região de seu genoma. Diante destas observações, iniciamos o desenvolvimento de um algoritmo que realiza a comparação sintática de genomas baseado nos trabalhos de Li e colegas, que utilizam a Complexidade de Kolmogorov para medir a distância entre dois genomas. Ao invés de uma medida de distância, o algoritmo proposto indica as regiões similares entre genomas que são consideradas relevantes pelo critério da Complexidade Condicional de Kolmogorov.

Abstract

Since 1953, when Watson and Crick discovered the DNA (deoxiribonucleic acid) structure, Molecular Biology has advanced quickly. Techniques were developed and improved to manipulate biomolecules since that, generating huge quantities of data. The need to process this information created a new field called Computational Molecular Biology, which consists in the development and usage of mathematical and computing techniques to solve Molecular Biology problems. There are problems in that area related to Genome Comparison, which consists, roughly speaking, in analysis and comparison of nucleic acid or aminoacid sequences between species. Genome Comparison tries to reveal existing relationships between species. The discovery of similar genes or pieces in genomes can point out evolutionary proximity or indispensable regions for life existence. Besides, differences can relate the unique behavior of a species with some of its genome regions. Based on these observations, we start the development of an algorithm that makes sintatic genome comparison using some ideas of Li and colleagues. They work with Kolmogorov Complexity to measure the distance between two genomes. Instead of a distance measure, the proposed algorithm shows similar regions that are considered relevant by the Conditional Kolmogorov Complexity criteria.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Desde 1953, quando Watson e Crick desvendaram a estrutura do DNA (ácido desoxirribonucléico), a área de Biologia Molecular tem avançado rapidamente. Técnicas que permitem a manipulação de biomoléculas foram criadas e aperfeiçoadas desde então, gerando enormes quantidades de dados. A necessidade de processar estas informações criou um novo campo chamado de Biologia Molecular Computacional, o qual consiste em desenvolver e usar técnicas matemáticas e de computação para ajudar a resolver problemas de Biologia Molecular [Setubal and Meidanis, 1997].

Dentre as diversas ramificações da área, a Comparação de Genomas vem ganhando destaque nos últimos anos devido a finalização do seqüenciamento de várias espécies [Lake and Moore, 1998, De Rosa and Labedan, 1998, Archer et al., 1997, Sankoff, 2000, Glaser et al., 2001]. A Comparação de Genomas busca revelar os relacionamentos entre espécies, ou através da identificação de seqüências comuns mantidas por seleção natural, ou pela descoberta de diferenças que fazem com que cada espécie tenha sua própria identidade [Elgar et al., 1996].

Genomas podem ser comparados sintaticamente (seqüência dos elementos que constituem o genoma) ou funcionalmente (porções do genoma — normalmente genes — que desempenham funções semelhantes em organismos distintos) [Sagot, 1998].

As comparações sintáticas utilizam como dado de entrada a seqüência de nucleotídeos de uma molécula de DNA ou seqüências de aminoácidos derivadas dos genes de um organismo. As semelhanças e diferenças entre blocos destas seqüências permitem estabelecer relacionamentos entre as espécies envolvidas, tais como compartilhamento de atividades metabólicas, componentes similares e diferenças fenotípicas [Elgar et al., 1996].

Do ponto de vista da Ciência da Computação, a Comparação Sintática de Genomas pode ser analisada como uma comparação quantitativa ou qualitativa entre seqüências de caracteres. Assim, os problemas referentes a área podem ser resolvidos utilizando as bases de Teoria da Informação [Gatlin, 1972, Oommen and Kashyap, 1998, Li et al., 2001,

Wan and Wootton, 2000, Varré et al., 1999, Farach et al., 1995, Gusev et al., 1993] e de Casamento de Padrões (*String Matching*) [Allison et al., 1999, Manber and Myers, 1993, Landau et al., 2001, Hall and Dowling, 1980, Gusfield, 1997, Ukkonen, 1995, Sagot, 1998, Almeida Jr. and Setubal, 2000].

1.1 Objetivos e Importância

A proposta desta dissertação é realizar a Comparação Sintática de Genomas de bactérias utilizando métodos de Casamento de Padrões (*String Matching*) guiados por critérios de similaridade provenientes da Teoria da Informação.

O diferencial proposto neste trabalho diz respeito a utilização de um critério teoricamente fundamentado (Complexidade Condicional de Kolmogorov) para que se decida se regiões de um genoma são similares a alguma porção de outro genoma.

É importante notar que este trabalho não se propõe a indicar todas as possíveis similaridades existentes entre os genomas analisados. Pretendemos oferecer uma nova forma de olhar para esse problema para que, unindo esforços, se descubram novos conhecimentos a respeito das moléculas que armazenam e transmitem as informações necessárias à vida.

1.2 Biologia Molecular

Esta seção pretende introduzir os conceitos de Biologia Molecular necessários para que se compreendam as escolhas tomadas no desenvolvimento do algoritmo proposto. Para um estudo mais aprofundado, recomendamos o livro **Genes VII** [Lewin, 2000].

Os organismos vivos, com exceção dos vírus, são formados por uma ou mais células. Cada célula de um organismo contém a informação genética necessária para a sobrevivência deste. Esta informação está armazenada em longas moléculas de DNA (ácido desoxirribonucléico), os **cromossomos**. O conjunto dos cromossomos de um organismo é chamado de **genoma**. Em bactérias, além dos cromossomos, as células podem abrigar outras moléculas de DNA, os **plasmídeos**. Plasmídeos são moléculas circulares de DNA extra-cromossômicas e capazes de se replicar autonomamente. São distintas do genoma de um organismo e não são essenciais à sobrevivência da célula sob condições não seletivas.

As moléculas de DNA são longas fitas duplas de **nucleotídeos**. Um nucleotídeo é formado por três componentes: uma **base nitrogenada**, um carboidrato (açúcar) e um grupo fosfato. São quatro as bases nitrogenadas que formam os nucleotídeos de DNA: Adenina, Citosina, Guanina e Timina, representadas por `A`, `C`, `G` e `T`, respectivamente.

A base `A` de uma das fitas é sempre pareada com uma base `T` e a base `C` com uma `G`. As bases `A` e `T` são chamadas de **complemento** uma da outra, ou um par de **bases**

complementares. Da mesma forma, `C` e `G` são bases complementares. **Pares de bases** são a unidade de comprimento usada quando se refere a moléculas de DNA (abreviado como **bp**, do inglês *base pair*).

Uma fita de DNA é o **complemento reverso** da outra, ou seja, elas são complementares e anti-paralelas (estão em direções opostas). Assim, dada uma das fitas, sabe-se como a outra é.

O conceito de direção surge por causa das ligações da molécula de açúcar do nucleotídeo que formam a fita do DNA. A molécula de açúcar (2'-desoxirribose) é constituída por cinco átomos de carbono rotulados de 1' a 5'. O carbono 3' de um açúcar se liga ao grupo fosfato que, por sua vez, se liga ao carbono 5' do carboidrato do outro nucleotídeo. Por convenção, a fita começa na extremidade 5' e termina na 3'. Esta é a direção canônica e é denotada por direção $5' \rightarrow 3'$. A estrutura molecular pode ser melhor compreendida observando-se a figura 1.1.

Outra molécula importante no processo biológico é o RNA (ácido ribonucléico). As diferenças em relação ao DNA são:

- No RNA o açúcar é a ribose.
- No RNA não existe timina (`T`); em seu lugar encontra-se uracila (`U`), a qual liga-se com adenina (`A`).
- RNA não forma uma dupla fita como o DNA, apesar de partes dela poderem se ligar a outras por complementaridade.

A informação genética contida no DNA é utilizada sob a forma de **proteínas**, que são seqüências de moléculas mais simples chamadas **aminoácidos**. Existem as **proteínas estruturais** (são os blocos componentes de tecidos) e as **enzimas** (proteínas catalisadoras de reações químicas específicas).

Proteínas são produzidas em uma estrutura celular chamada **ribossomo**. Em um ribossomo os aminoácidos componentes de uma proteína são montados um por um graças a informação contida em uma molécula importante chamada **RNA mensageiro**.

O fluxo de informação genética em uma célula, chamado de dogma central da biologia molecular, é mostrado esquematicamente na figura 1.2.

As moléculas de DNA utilizam a **replicação** para criar novas cópias delas mesmas — transmissão da informação para gerações futuras. A **transcrição** corresponde a montagem de uma molécula de RNA a partir de um ou mais trechos de uma molécula de DNA, os **genes**. Este processo começa em uma região anterior ao gene, chamado de **promotor**. A operação inversa é chamada de **transcrição reversa**, a qual produz, a partir do RNA, uma molécula de cDNA (DNA complementar). Também a partir do RNA ocorre o processo de **tradução**, onde a fita de RNA, juntamente com o ribossomo, é responsável pela

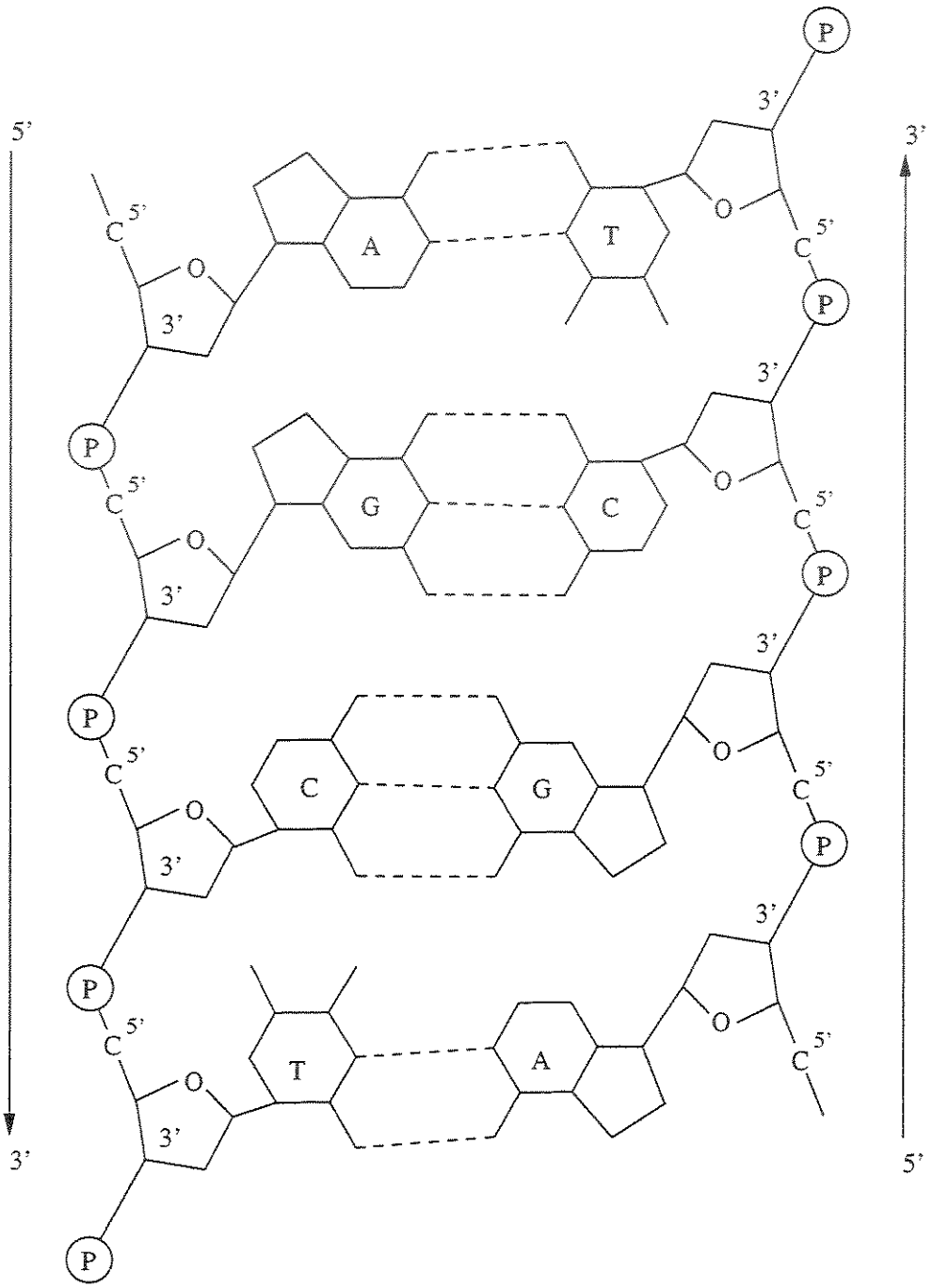


Figura 1.1: Esquema da estrutura molecular da fita dupla de DNA. A, C, G e T são as bases nitrogenadas; P é o grupo fosfato; O é oxigênio; e $C^{5'}$ é carbono.

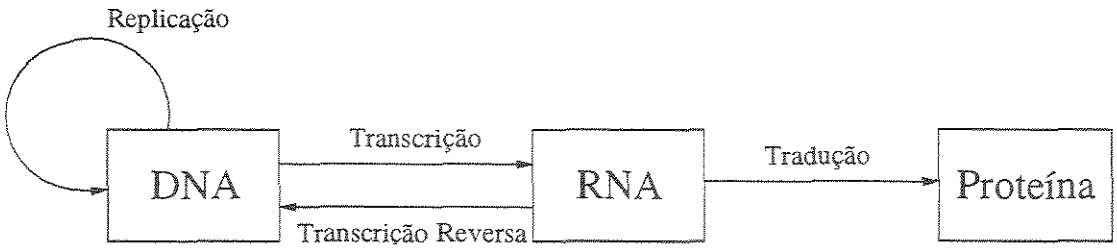


Figura 1.2: Dogma central da Biologia Molecular.

produção de proteínas. Na tradução, cada três nucleotídeos (um **códon**) geram um aminoácido. Isto ocorre porque temos quatro bases distintas para vinte aminoácidos, de forma que somente com três nucleotídeos se consegue mapear todos os tipos de aminoácidos — $4^3 = 64$. Este mapeamento é chamado **código genético**. Nem todos os organismos compartilham o mesmo código. Isto significa que um mesmo códon pode corresponder a um aminoácido diferente em organismos distintos. Porém, as diferenças são pequenas.

Os organismos vivos dividem-se em três grupos: **vírus**, **procariontes** e **eucariontes**. Os vírus são compostos por uma molécula de DNA ou RNA (os retrovírus) envolta por uma cápsula protetora. Eles não possuem maquinaria celular para se reproduzirem, necessitando parasitar outros organismos para sua replicação (alguns autores consideram os vírus um meio termo entre o vivo e o não-vivo). Os seres procariontes geralmente possuem no genoma um único cromossomo circular. Este cromossomo fica livre no interior da célula. Nos eucariontes, o genoma está confinado dentro de um núcleo. Outra diferença nos eucariontes está na organização dos genes. Muitos genes eucariontes são compostos de partes alternantes chamados **introns** e **exons**. Após a transcrição, os introns são descartados. Isto é mostrado na figura 1.3.

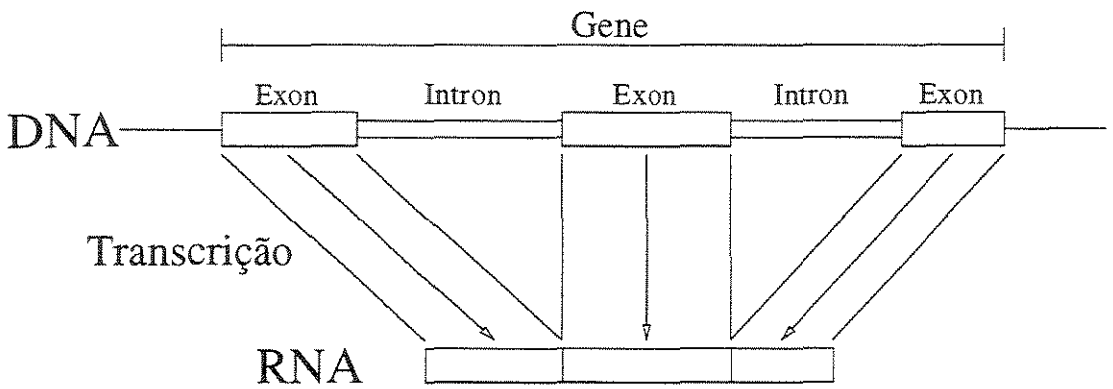


Figura 1.3: Organização dos genes de organismos eucariontes.

1.3 Estrutura da Dissertação

A dissertação está organizada da seguinte forma:

- **Capítulo 2:** Os conceitos principais de Teoria da Informação e Casamento de Padrões são abordados, visando a apresentação da base teórica de computação necessária para o algoritmo.
- **Capítulo 3:** As abordagens mais significativas da área de Comparação de Genomas são apresentadas, bem como os trabalhos mais relacionados a esta dissertação.
- **Capítulo 4:** O algoritmo proposto é apresentado, incluindo suas especificações e análise de complexidade.
- **Capítulo 5:** Discussões a respeito do algoritmo e da área são feitas e trabalhos futuros são indicados.
- **Apêndice A:** Gráficos e principais informações referentes à execução do algoritmo em cinco pares de genomas são mostrados.

Capítulo 2

Fundamentos Teóricos

O algoritmo proposto nesta dissertação utiliza conceitos de Teoria da Informação, Casamento de Padrões (*String Matching*) e Teoria dos Grafos. Entretanto, este último não é contemplado neste capítulo por serem necessários apenas conceitos simples e um algoritmo dos mais conhecidos na computação. Para uma revisão sobre grafos recomendamos o livro de West e, para o algoritmo, o livro de Cormen, Leiserson e Rivest [West, 1996, Cormen et al., 1990, páginas 527 e 550].

2.1 Teoria da Informação

A Teoria da Informação propõe-se a responder duas perguntas: *"É possível haver comunicação confiável de um ponto a outro através de um canal de comunicação com ruídos?"* e *"Como se pode medir a informação contida em uma variável aleatória?"* [MacKay, 1995].

Para isso, Shannon criou um modelo de comunicação baseado em uma fonte geradora de informação, um canal transmissor e um receptor, podendo haver codificação e decodificação desta informação na fonte e no receptor, respectivamente. Esta codificação é utilizada visando manter a integridade da informação quando esta chegar ao receptor [Shannon, 1948]. O modelo é apresentado na figura 2.1.

A partir deste modelo, Shannon provou vários teoremas, iniciando do Teorema de Codificação da Fonte, o qual motivou a escolha da entropia como medida de informação, culminando no Teorema de Codificação de Canal com Ruídos [MacKay, 1995].

Visando tornar práticos os resultados provenientes dos teoremas, códigos para compressão de dados e correção de erros foram desenvolvidos [MacWilliams and Sloane, 1977, Witten et al., 1987].

Mais recentemente, vários pesquisadores iniciaram estudos sobre como utilizar este campo para medir a complexidade de genomas. A seguir, será mostrada a relação entre Complexidade de Genomas e Teoria da Informação, bem como trabalhos que tratam deste

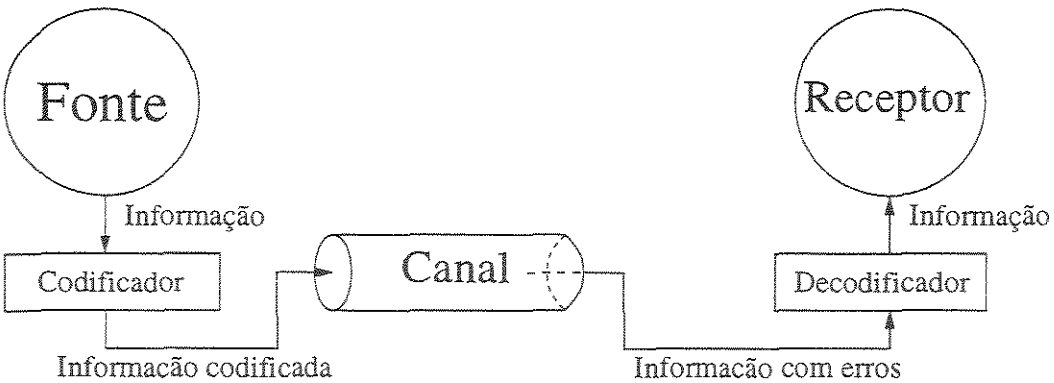


Figura 2.1: Modelo de Comunicação proposto por Shannon.

assunto.

Na subseção 2.1.2, a Complexidade de Kolmogorov será melhor apresentada e o trabalho motivador do algoritmo proposto será citado.

2.1.1 Teoria da Informação e Complexidade de Genomas

Considere que o conjunto de todos os genomas faz parte da saída de uma única fonte, de forma que, em certos intervalos de tempo, um genoma é escolhido como saída (não só os genomas existentes, como todos os possíveis).

Uma **fonte estacionária** é aquela cujas leis de probabilidade não variam com o tempo. Existe um tipo específico de fonte estacionária chamada **fonte ergódica**, onde o ponto inicial de observação não importa para se saber a distribuição de probabilidades desta fonte [Gatlin, 1972].

Com estas informações em mãos, a idéia inicial para analisar a complexidade de genomas seria utilizar o conceito de entropia de Shannon (se H é a entropia e p_i a probabilidade do evento i, temos $H = -\sum_i p_i \log p_i$). Gatlin utilizou este conceito para nortear todo o seu trabalho em análise de complexidade de genomas [Gatlin, 1972]. Entretanto, a fonte emissora de genomas não pode ser considerada ergódica, nem mesmo estacionária devido a observação da evolução das espécies. Por outro lado, deve haver alguma estrutura nesta fonte, dadas as características funcionais presentes em todos os organismos.

Assumindo isto como verdadeiro, os pesquisadores passaram a investir em estimativas de entropia [Salamon and Konopka, 1992, Farach et al., 1995]. Farach e colegas observaram que, quando consideramos a entropia de seqüências, torna-se importante apontar que entropia é um número associado a distribuições de probabilidade, não uma propriedade de uma seqüência em particular, pelo menos diretamente.

A partir desta observação, passou-se a utilizar métodos que estimam a entropia através da informação contida nas seqüências [Gusev et al., 1993, Loewenstern et al., 1995]. Le-

vando em conta a observação de Farach e colegas, o adequado seria considerar seqüências como sendo o resultado de processos estocásticos e, então, estimar a entropia da distribuição cujos dados observados sejam típicos.

Apesar do aviso, trabalhos que utilizam casamento de padrões e compressão de seqüências proliferaram [Allison et al., 1999, Chen et al., 2000, Gusfield, 1997, página 167]. Somente em 2000, Galatolo provou a relação de Beyer-Stein-Ulam entre entropia e complexidade (aqui complexidade refere-se a tamanho da menor descrição de um objeto), como descrevemos a seguir [Galatolo, 2000].

Se H é a entropia, $K(x|y)$ a Complexidade Condicional de Kolmogorov — ver seção seguinte —, $x_1, x_2, \cdots, x_{2^n}$ todas as seqüências binárias de tamanho n arranjadas em ordem não-crescente de probabilidade e $k(n)$ o menor inteiro tal que $\sum_{i=1}^{k(n)} p(x_i) > r$, onde $p(x_i)$ é a probabilidade da seqüência x_i e r é um número real no intervalo aberto $(\frac{1}{2}, 1)$, a relação é:

$$H = \lim_{n\to\infty} \frac{1}{nk(n)} \sum_{i=1}^{k(n)} K(x_i|n).$$

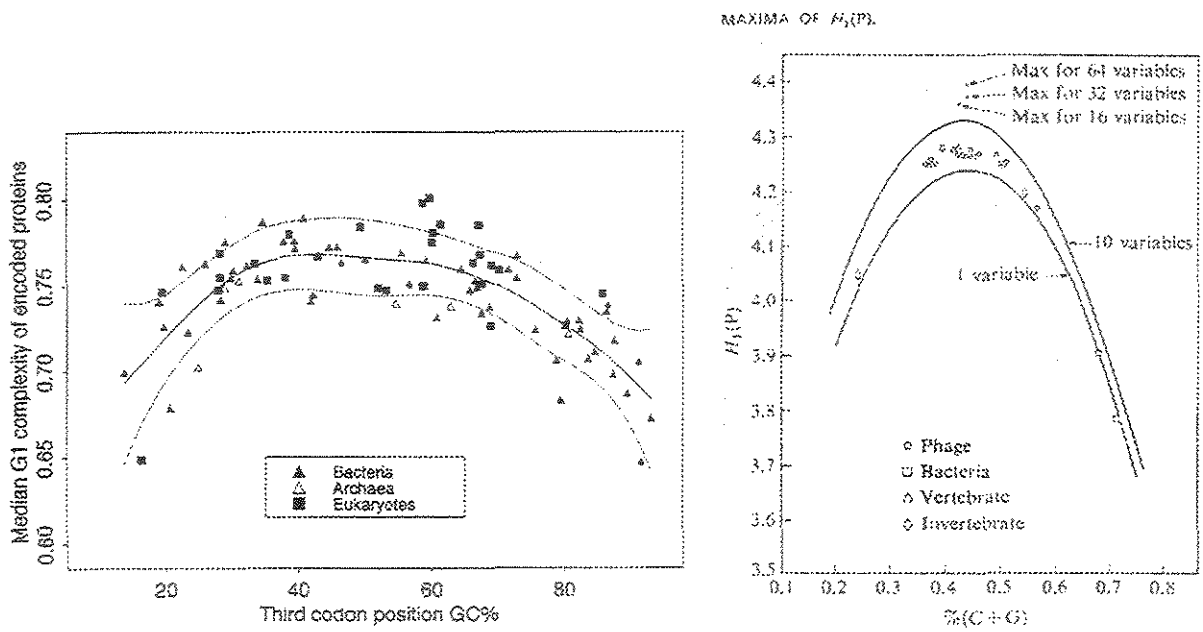
Li e Vitányi afirmam que $K(x|y)$ não é sequer computável para quaisquer seqüências x e y [Li and Vitányi, 1997, páginas 121 e 206]; entretanto, como esta medida de complexidade é intuitivamente atraente desde que captura quantitativamente estruturas repetitivas, Chen e colegas desenvolveram o algoritmo **GenCompress**, que aproxima este valor [Chen et al., 2000].

Wan e Wootton observaram que a medida de complexidade $K(x|y)$ é válida para fontes estacionárias e ergódicas, o que não corresponde a suposição feita para os genomas [Wan and Wootton, 2000]. Eles propuseram uma medida de complexidade global não estacionária, chamada G_1, baseada na distribuição de probabilidades de mononucleotídeos.

Dentre outras aplicações, as medidas de complexidade de genomas são utilizadas para indicar a "distância" entre as espécies. Wan e Wootton definiram a distância $d_w(x, y) = |G_1(x) - G_1(y)|$; já Li e colegas utilizaram o conceito de Informação Algorítmica Mútua e propuseram $d(x, y) = 1 - [K(x) - K(x|y)]/K(xy)$, que, segundo os autores, não requer que as seqüências obedeçam à condição estacionária [Li et al., 2001].

Apesar dos esforços e resultados otimistas obtidos, todas as medidas de complexidade e distâncias derivadas delas apresentam problemas. Um fato surpreendente e curioso, resultante desta observação, é o fato de Gatlin obter um gráfico muito semelhante ao de Wan e Wootton dezoito anos antes, com um método totalmente diferente [Wan and Wootton, 2000, Gatlin, 1972]. Os gráficos são mostrados na figura 2.2.

Diante disso, a melhor opção para analisar genomas é utilizar vários métodos distintos de forma a minimizar o impacto dos erros inerentes a cada um deles, não esquecendo que,



(a) Página 87 de [Wan and Wootton, 2000] (b) Página 184 de [Gatlin, 1972]

Figura 2.2: Gráficos de Análise de Complexidade de Genomas.

segundo Gusev e colegas, a filosofia das medidas de complexidade é *ir para não se sabe onde e encontrar não se sabe o quê* [Gusev et al., 1993, página 184].

2.1.2 Complexidade de Kolmogorov

Complexidade de Kolmogorov, K-complexidade, Complexidade Algorítmica, Complexidade Estocástica, Complexidade Descricional, Tamanho Mínimo de Descrição são alguns dos nomes dados à teoria que trata da quantidade de informação em objetos particulares. Cada um dos nomes acima representam uma variação da mesma idéia ou um ponto de partida diferente. Suas formulações refletem as tradições do campo onde surgiram, como teoria da probabilidade, teoria da informação, teoria da computação, estatística ou inteligência artificial.

Uma definição para a Complexidade de Kolmogorov é: dado um objeto qualquer (finito ou não), sua complexidade é o tamanho do menor algoritmo que, sem dados adicionais, computa este objeto e pára. A complexidade do objeto x é, portanto, igual ao tamanho da menor descrição de x com respeito a uma função de especificação D. Cada objeto pode ter mais de uma descrição. Ou seja, existem, em geral, vários y tais que $D(y) = x$, onde y é uma descrição para o objeto x. Esta função de especificação pode ser formalizada

como uma função parcial recursiva. Diferentes funções de especificação definem valores de complexidade para o objeto x que diferem apenas de uma constante aditiva, independente do objeto. Isto quer dizer que a Complexidade de Kolmogorov de um objeto qualquer é um atributo intrínseco do próprio objeto [Li and Vitányi, 1997].

Além disso, a Complexidade de Kolmogorov de um objeto está relacionada com o grau de compressão deste, ou seja, quanto mais aleatório for o conteúdo do objeto mais complexo ele é.

A seguir, os principais conceitos referentes a Complexidade de Kolmogorov serão mostrados.

Principais Conceitos

Dois tipos de Complexidade de Kolmogorov serão tratados aqui: a Simples e a de Prefixo. A Complexidade de Kolmogorov Simples foi a originalmente definida por seus inventores (Solomonoff, Kolmogorov e Chaitin) e satisfaz muitos dos problemas a que se propõe. Por possuir uma série de inconveniências, ela não é satisfatória para certos objetivos. Desta forma, foi proposta a Complexidade de Kolmogorov de Prefixo.

A Complexidade de Kolmogorov Simples (denotada por "C") pode ser vista como uma medida para o problema de descrever objetos pertencentes ao conjunto dos números naturais em termos de programas formados por seqüências finitas de zeros e uns.

Tanto para a Complexidade de Kolmogorov Simples, quanto para a de Prefixo, existem as versões Incondicional e Condicional.

A Complexidade Incondicional de Kolmogorov é definida como na seção anterior, ou seja, dado um objeto e sua descrição de tamanho mínimo, existe uma máquina de Turing (ou uma função parcial recursiva) que produz este objeto sem nenhuma outra informação. A máquina de Turing é considerada a descrição deste objeto.

Já a Complexidade Condicional de Kolmogorov possui, como entrada para uma máquina de Turing, um segundo objeto, do qual deve-se produzir o primeiro objeto somente com estas informações. Como a versão incondicional é um caso especial da condicional (considerando vazio o objeto dado como entrada), será mostrado somente o Teorema de Invariância para a versão condicional.

Antes disso, são necessárias as seguintes definições:

Definição 2.1 *Seja x um número natural. Definimos $\ell(x)$ como o número mínimo de bits necessários para representar x, ou seja, $\ell(x) = \lfloor \log_2 x \rfloor + 1$.*

Definição 2.2 *Seja ϕ uma função parcial recursiva. Se x, y e p são números naturais, tais que $\phi(\langle y, p \rangle) = x$, então p é uma descrição de x dado y. A complexidade C_ϕ de x condicional a y é definida por*

$$C_\phi(x|y) = min\{\ell(p) : \phi(\langle y, p\rangle) = x\},$$

e $C_\phi(x|y) = \infty$ *se não existe* p.

O parâmetro p é um programa que computa x por ϕ, dado y. Em outras palavras, a complexidade $C_\phi(x|y)$ é igual ao tamanho do menor programa que computa x dado y.

Teorema 2.1 *Existe uma função parcial recursiva universal* ϕ_0 *para a classe das funções parciais recursivas que computam* x *dado* y. *Formalmente isto diz que* $C_{\phi_0}(x|y) \leq C_\phi(x|y) + c_\phi$ *para todas funções parciais recursivas* ϕ *e todo* x *e* y, *onde* c_ϕ *é uma constante dependente de* ϕ *mas não de* x *ou* y.

Uma interpretação para a complexidade condicional $C(x|y)$ é a quantidade de informação necessária para recuperar um objeto x dado apenas y. Portanto, a complexidade é uma "informação absoluta" de um objeto (considerando y um objeto vazio). Desta forma, a Informação Algorítmica que y possui de x é definida como

$$I_C(y : x) = C(x) - C(x|y)$$

adicionada de uma constante logarítmica dependente apenas da máquina de Turing escolhida.

Ou seja, se $C(x|y)$ for muito menor do que $C(x)$, o objeto y possui muita informação a respeito de x.

A Informação Algorítmica I_C entre dois objetos é simétrica. O conceito de Simetria de Informação diz que $I_C(y : x) = I_C(x : y)$.

Ao se utilizar a Complexidade de Kolmogorov de Prefixo (denotada por "K"), obtém-se a mesma propriedade de simetria de informação para a complexidade simples, ou seja, $I_K(x : y) = I_K(y : x) = K(x) - K(x|y) = K(y) - K(y|x)$. Assim como para vários outros teoremas, esta mudança para a complexidade de prefixo não altera a essência das provas.

A Complexidade de Kolmogorov de Prefixo é induzida por máquinas de Turing com um conjunto de programas no qual nenhum programa é um prefixo próprio de outro programa. Esta restrição é motivada pela hipótese implícita de que os programas serão concatenados e devem ser unicamente decodificáveis.

Esta propriedade pode ser obtida utilizando a **versão auto-delimitada** de uma seqüência binária x, chamada $\bar{x}$. A idéia é reservar um símbolo, como "0", para ser um sinal de parada e codificar $x \in \mathbb{N}$ como $\bar{x} = 1^{\ell(x)}0x$, onde $\ell(x)$ é o tamanho, em bits, da seqüência x. Definimos $E_1(x)$ como sendo o tamanho, em bits, de $\bar{x}$. Assim, $E_1(x) = \ell(\bar{x}) = 2\ell(x) + 1$.

A versão auto-delimitada do número decimal 12 (1100, binário), por exemplo, seria:

$$\bar{x} = 111101100_2$$

e seu tamanho é $E_1(12_{10} = 1100_2) = 2(\lfloor \log_2 12 \rfloor + 1) + 1 = 9$.

O único problema, tanto da Complexidade de Kolmogorov de Prefixo, quanto da Simples, é que elas não são funções parciais recursivas e, portanto, não são computáveis [Li and Vitányi, 1997, páginas 121 e 206]. Entretanto, elas podem ser aproximadas.

Aplicações na Biologia

Li e colegas propuseram uma forma de se comparar genomas utilizando a teoria de Informação Algorítmica da Complexidade de Kolmogorov. A idéia foi considerar cada genoma um objeto e medir a distância entre dois genomas x e y da seguinte forma:

$$d(x, y) = 1 - \frac{K(x) - K(x|y)}{K(xy)}.$$

A proposta descrita acima pretende refinar o modo tradicional de se comparar genomas (alinhamentos de seqüências), além de definir uma função que pode ser considerada uma medida de distância.

Para que a função $d(x, y)$ seja uma medida de distância (ver definição na página 21), ela deve satisfazer:

1. $d(x, y) > 0$, para $x \neq y$;
2. $d(x, x) = 0$;
3. $d(x, y) = d(y, x)$ (simetria); e
4. $d(x, y) \leq d(x, z) + d(z, y)$ (desigualdade do triângulo).

Li e colegas provaram que a função $d(x, y)$ satisfaz a desigualdade do triângulo em um artigo recentemente publicado [Li et al., 2001]. As demais condições são facilmente provadas usando os teoremas da Complexidade de Kolmogorov.

Apesar de se ter uma medida de distância matematicamente rigorosa, ela não é computável. Para resolver esse problema, buscou-se uma aproximação para as complexidades de prefixo utilizadas na função por sempre existir um limitante superior computável para os valores de "K" [Li and Vitányi, 1997, página 121].

Essa aproximação segue a premissa de que a vida representa ordem, ou seja, as seqüências de DNA não são aleatórias. Em outras palavras, elas devem ser muito compressíveis. Várias evidências biológicas dão suporte a essa premissa, tais como os *repeats*,

a existência de cópias de genes essenciais e os padrões altamente conservados de segmentos de proteínas (os chamados domínios).

Desta forma, Li e colegas fizeram a aproximação da distância entre os genomas da seguinte forma:

$$d(x,y) \simeq 1 - \frac{GenCompress(x) - GenCompress(x|y)}{GenCompress(xy)},$$

onde $GenCompress(u)$ comprime o genoma u e $GenCompress(u|v)$ realiza a compressão do genoma u em relação ao genoma v. Através de resultados experimentais, GenCompress é uma das melhores ferramentas de compressão de genomas, o que garante um bom limitante superior para "K" [Chen et al., 2000, Loewenstern and Yianilos, 1999].

2.2 Casamento de Padrões

A hipótese padrão de muitos pesquisadores da Ciência da Computação que trabalham com Biologia Molecular é a de que resultados biologicamente significativos podem ser obtidos ao se considerar o DNA como uma seqüência unidimensional de caracteres, abstraindo da realidade sua estrutura tridimensional e interações no ambiente dinâmico onde esta molécula se encontra [Gusfield, 1997, página xiii].

Esta suposição permite transformar problemas biológicos em problemas de Casamento de Padrões. Um deles é: *"Dados dois genomas, quais são as semelhanças existentes entre eles?"* Este problema pode ser reformulado para: *"Dadas duas seqüências, quais são os casamentos exatos ou aproximados existentes entre elas?"*

Mas o que são seqüências? Quais são seus principais conceitos?

2.2.1 Definições

Uma **seqüência** é uma sucessão ordenada de caracteres ou símbolos extraídos de um conjunto finito chamado **alfabeto**. Seqüências podem ter caracteres repetidos, por exemplo, $S =$ `AGTTCT`. O **tamanho** de uma seqüência S, denotado por $|S|$, é o número de caracteres contidos nela. No exemplo anterior, $|S| = 6$. O caractere que ocupa a posição i na seqüência S é indicado por $S[i]$. Índices de caracteres iniciam em 1 e vão até $|S|$. Existe uma seqüência de tamanho zero, chamada **seqüência vazia**. O símbolo especial "ϵ" a representa. Existe, ainda, o **tamanho em bits** de S, definido como $\ell(S) = \lceil \log_2 |\Sigma| \rceil |S|$, onde Σ é o alfabeto.

Uma **subseqüência** de S é uma seqüência que pode ser obtida de S pela remoção de alguns caracteres. Por exemplo, `TGC` é subseqüência de `ATTGTAC`. A seqüência vazia é uma subseqüência de todas as seqüências.

Existe também o conceito de subcadeia. Uma **subcadeia** é uma seqüência formada por caracteres consecutivos de S, na mesma ordem em que eles aparecem em S. Usando o mesmo exemplo da subseqüência, `TGTA` é uma subcadeia de `ATTGTAC`. Nota-se que toda subcadeia de S é, também, uma subseqüência de S, mas nem toda subseqüência é uma subcadeia.

Muitas vezes, uma seqüência possui várias subcadeias iguais. Para distingui-las, usa-se o conceito de intervalo. Um **intervalo** de uma seqüência S é um conjunto de índices consecutivos $[i..j]$ tal que $1 \leq i \leq j+1 \leq |S|+1$. O intervalo contém todos os índices entre i e j, incluindo estes também. Para o intervalo $[i..j]$ de S, $S[i..j]$ indica a subcadeia $s[i]s[i+1]\ldots s[j]$ de S se $i \leq j$ ou indica a seqüência vazia quando $i = j+1$.

A **concatenação** de duas seqüências S_1 e S_2 é indicada por S_1S_2 e é formada pela inclusão de todos os caracteres de S_2 após S_1, na ordem em que eles aparecem em S_2. O tamanho de S_1S_2 é igual a $|S_1| + |S_2|$. A concatenação de k cópias de uma mesma seqüência é indicada por S^k. Por exemplo, com $k = 4$, temos $S^4 = SSSS$.

Um **prefixo** de S é uma subcadeia de S da forma $S[1..j]$, para $0 \leq j \leq |S|$. A inclusão de $j = 0$ indica que a seqüência vazia ($S[1..0]$) é, também, um prefixo de S. A seqüência S_1 é um prefixo de S se, e somente se, existir outra seqüência S_2, tal que $S = S_1S_2$. Para se referir ao prefixo de S com exatos k caracteres ($0 \leq k \leq |S|$), usa-se a notação $prefixo(S, k)$.

De forma semelhante, um **sufixo** de S é uma subcadeia da forma $S[i..|S|]$, para $1 \leq i \leq |S|+1$. A inclusão de $i = |S|+1$ indica que a seqüência vazia ($S[|S|+1..|S|]$) é, também, um sufixo de S. A seqüência S_2 é um sufixo de S se, e somente se, existir outra seqüência S_1, tal que $S = S_1S_2$. Para se referir ao único sufixo de S com k caracteres ($0 \leq k \leq |S|$), usa-se a notação $sufixo(S, k)$. Pode-se dizer, também, que $S[i..|S|]$ é o sufixo de índice i.

Dados os conceitos acima, podemos dizer que Casamento de Padrões consiste em encontrar todas as ocorrências da seqüência S_1 em S [Setubal and Meidanis, 1997]. Em outras palavras, S_1 é uma subcadeia de S? Se é, quais são os intervalos de S onde encontramos S_1?

Para se comparar dois genomas, devemos comparar subcadeias de um deles com a seqüência inteira do outro. Dentre as estruturas que podem ser utilizadas para isso, duas nos importam: árvores e vetores de sufixos. Nas subseções seguintes elas serão mostradas.

Uma observação importante para este trabalho é que trataremos de Casamento Exato de Padrões. A inclusão de erros nos casamentos é feito de outra forma, apesar de existirem excelentes métodos para Casamento Aproximado de Padrões [Hall and Dowling, 1980].

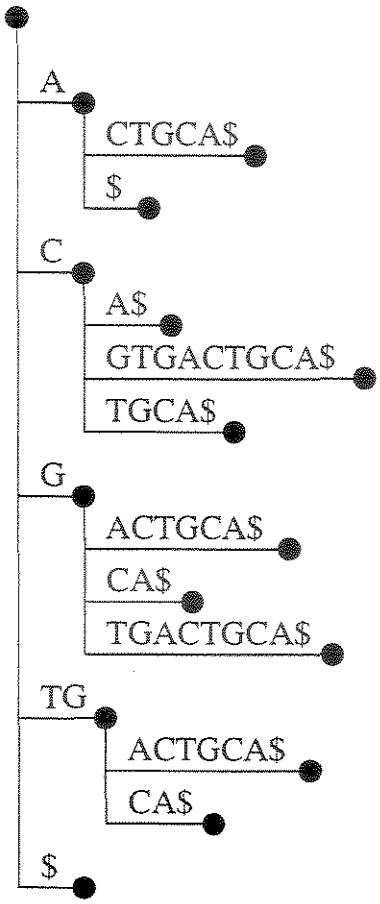


Figura 2.3: Árvore de Sufixos da Seqüência `CGTGACTGCA`.

2.2.2 Árvores de Sufixos

Uma árvore de sufixos de uma seqüência S é uma estrutura de dados que contém todos os sufixos de S tal que cada caminho da raiz a uma folha representa um sufixo. A princípio, uma árvore de sufixos de S pode ser usada para encontrar padrões que estejam contidos em S. Um exemplo de árvore de sufixos é mostrado na figura 2.3.

A seguir, serão dadas as definições básicas e uma aplicação para árvores de sufixos.

Definições

Ao descrever como construir uma árvore de sufixos para uma seqüência arbitrária, vamos nos referir a seqüência genérica S de tamanho m. Também assumiremos que o alfabeto Σ é finito e conhecido, como as bases `A`, `C`, `G` e `T` de uma molécula de DNA.

Definição 2.3 *Uma árvore de sufixos T para uma seqüência S com m caracteres é uma árvore dirigida com raiz com exatamente m folhas numeradas de 1 a m. Cada nó interno,*

exceto a raiz, tem pelo menos dois filhos e cada arco é rotulado com uma subcadeia não-vazia de S. Não existem dois arcos partindo de um mesmo nó cujos rótulos começam com o mesmo caractere. A característica chave da árvore de sufixos é que, para qualquer folha i, a concatenação dos arcos rotulados no caminho da raiz para a folha i representa exatamente o sufixo de S que começa na posição i.

Como descrito acima, a definição de uma árvore de sufixos para S não garante que uma árvore de sufixos para qualquer seqüência S exista. O problema é que se um sufixo de S for igual a um prefixo de outro sufixo de S, então nenhuma árvore de sufixos obedecendo a definição acima é possível, pois o caminho para o primeiro sufixo pode não terminar em uma folha.

Para evitar este problema, assume-se que o último caractere de S não aparece em nenhuma outra posição de S. Assim, nenhum sufixo da seqüência resultante pode ser um prefixo de qualquer outro sufixo. Na prática, adiciona-se um caractere no final de S que não faça parte do alfabeto de S. Geralmente, o caractere \$ é usado como terminador e considera-se que a seqüência S já o contém, a menos que seja explicitamente indicado o contrário [Gusfield, 1997].

Definição 2.4 *O rótulo de um caminho da raiz até um nó qualquer é a concatenação, em ordem, das subcadeias que rotulam os arcos daquele caminho. O* rótulo de um nó *é o rótulo do caminho da raiz de T até aquele nó.*

Definição 2.5 *Para qualquer nó v na árvore de sufixos, a* profundidade *de v é o número de caracteres no rótulo de v.*

Definição 2.6 *Um caminho que termina no meio de um arco (u, v) corta o rótulo de (u, v) em um ponto designado. O rótulo deste caminho é definido como o rótulo de u concatenado com os caracteres no arco (u, v) até o* ponto de corte *designado.*

Utilizando estas definições, Ukkonen propôs um algoritmo que, dada a seqüência S, constrói a árvore de sufixos T em tempo linear [Ukkonen, 1995]. As vantagens deste algoritmo em relação aos anteriores, também lineares, é a explicação mais simples e a melhor eficiência de utilização de espaço [Gusfield, 1997].

Aplicações na Biologia

Uma das aplicações de árvore de sufixos mais interessantes para este projeto é relatada por Gusfield em seu livro e chama-se **Tamanho Mínimo de Codificação de DNA** [Gusfield, 1997, páginas 167 e 168].

Recentemente, muitos pesquisadores tem usado métodos de compressão de seqüências em DNA para computar uma medida de complexidade das seqüências. A idéia básica sugere que subcadeias de grande significado biológico devem ser mais compressíveis que subcadeias essencialmente aleatórias. Dessa forma, espera-se encontrar seqüências que tenham uma função biológica definida.

A compressão também tem sido usada para estudar a distância entre duas seqüências S_1 e S_2 de DNA [Chen et al., 2000]. Uma forma de se calcular essa distância é construir uma árvore de sufixos T_1 para S_1 e, então, compactar a seqüência S_2 usando apenas T_1, ou seja, procurando subcadeias de S_2 nos sufixos de S_1 e, depois, codificá-las para aproximar o tamanho mínimo de descrição de S_2 em relação a S_1.

2.2.3 Vetores de Sufixos

Um vetor de sufixos V_S é uma estrutura de dados que contém os índices de todos os sufixos de uma seqüência S na ordem lexicográfica dos sufixos. Esta estrutura proposta por Manber e Myers é comumente construída usando um algoritmo de ordenação de seqüências [Manber and Myers, 1993]. Um exemplo de vetor de sufixos está na figura 2.4.

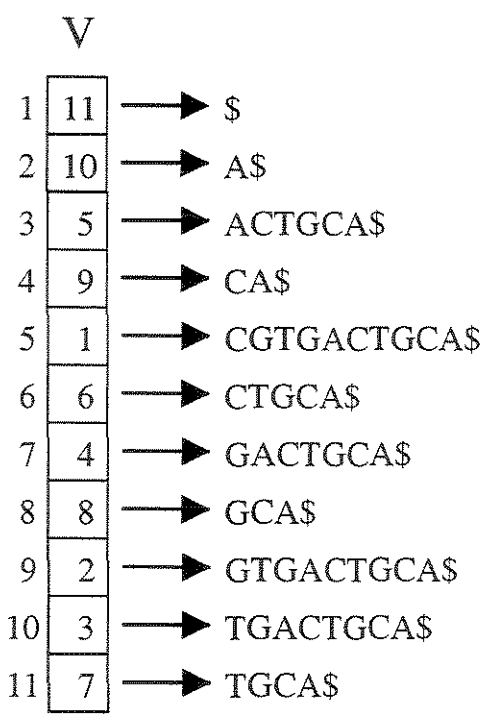


Figura 2.4: Vetor de Sufixos da Seqüência `CGTGACTGCA`.

A seguir, será mostrado como construir o vetor ordenado de sufixos, como realizar buscas neste vetor e suas vantagens sobre as árvores de sufixos.

Ordenando o Vetor de Sufixos

Para ordenar os n sufixos de S, $|S| = n$, o algoritmo executa em $\lceil \log(n+1) \rceil$ estágios. O primeiro estágio ordena os sufixos de S pelo primeiro caractere utilizando *bucketsort* [Cormen et al., 1990]. Em seguida, indutivamente, cada um dos estágios seguintes particionam os compartimentos (*buckets*) pela ordenação de duas vezes o número de símbolos do estágio anterior. Cada um dos estágios leva tempo $\mathcal{O}(n)$ e, portanto, o tempo gasto para ordenar os n sufixos de S é $\mathcal{O}(n \log n)$.

Larsson e Sadakane desenvolveram um algoritmo que ordena o vetor de sufixos mais rapidamente que o de Manber e Myers, mas o tempo de pior caso continua o mesmo [Larsson and Sadakane, 1999].

Busca de Padrões no Vetor de Sufixos

Seja $S = s_1 s_2 \cdots s_n$ uma seqüência de tamanho n, $S_i = s_i s_{i+1} \cdots s_n$ o sufixo de S que começa na posição i, V_S o vetor de sufixos de S ordenado lexicograficamente e $V_S[k]$ a posição inicial do k-ésimo menor sufixo de S.

Suponha que queiramos encontrar a seqüência $W = w_1 w_2 \cdots w_p$ de tamanho $p \leq n$ em S. Para isso, basta realizar uma busca binária em V_S e comparar o sufixo $V_S[k]$ com a seqüência W. Quando as seqüências forem lexicograficamente iguais, W foi encontrada. Caso contrário, W estará lexicograficamente entre os sufixos $V_S[k]$ e $V_S[k+1]$.

A busca em vetores de sufixos requer tempo $\mathcal{O}(p \log n)$ no pior caso. Manber e Myers propuseram um algoritmo de busca de padrões mais elaborado — que requer mais espaço durante a construção do vetor — que encontra todas as ocorrências de W em S em $\mathcal{O}(p + \log n)$ [Manber and Myers, 1993].

Vantagens sobre Árvores de Sufixos

A principal vantagem do uso de vetores de sufixos é que, na prática, estes usam de 3 a 5 vezes menos espaço do que árvores de sufixos. Além disso, Larsson e Sadakane fizeram testes com seqüências de tamanhos e distribuições de probabilidade do alfabeto diferentes e, na maioria dos casos, o tempo de execução para a construção dos vetores foi menor [Larsson and Sadakane, 1999, página 18]. O trabalho destes autores mostra que apesar de, assintoticamente, o tempo de construção de árvores de sufixos ser menor, na prática, a utilização de vetores de sufixos é melhor, até um certo valor limite para n.

Capítulo 3

Comparação de Genomas

Este capítulo apresenta uma revisão não-exaustiva referente a área de Comparação de Genomas sob a perspectiva da Ciência da Computação. Nosso trabalho é uma tentativa de transformar a idéia de Chen e colegas em uma ferramenta que produza resultados qualitativos, de maneira semelhante ao que Varré e colaboradores fizeram com o Princípio do Tamanho Mínimo de Codificação [Chen et al., 2000, Varré et al., 1999]. Para tanto, esperamos que este capítulo forneça uma visão geral da área e os relacionamentos entre suas distintas abordagens.

Comparar genomas é uma idéia útil para muitos estudos, desde problemas básicas de biologia evolucionária até questões clínicas específicas, tais como a identificação de polimorfismos genéticos, os quais podem levar ao surgimento de doenças ou a variações significativas no fenótipo.

Porque queremos comparar genomas inteiros em vez de comparar um gene por vez? À medida que os projetos genoma atingem seu final, os pesquisadores estão apenas começando a explorar, em detalhes, como a estrutura do genoma afeta as funções dele. Existem características estruturais no DNA que controlam a expressão? Existem regiões promotoras e regulatórias que ainda não foram descobertas?

A Comparação de Genomas pode ajudar a responder tais questões apontando regiões similares em trechos de DNA não caracterizados ou supostamente redundantes (repetições, cópias). A comparação genômica de organismos próximos na árvore evolutiva justifica, por sua vez, o seqüenciamento de genomas adicionais.

Pode-se comparar genomas quantitativa ou qualitativamente. A comparação quantitativa preocupa-se em estabelecer medidas de distância entre genomas, visando, principalmente, organizá-los em árvores filogenéticas. Algumas maneiras de se abordar esta questão serão tratadas na seção seguinte. Já a comparação qualitativa busca mostrar quais porções dos genomas são semelhantes, visando a descoberta ou associação de funções a seqüências genômicas específicas. Estas buscas por semelhanças podem ser baseadas ex-

clusivamente em características sintáticas (a seqüência genômica) ou incluir informações funcionais a trechos da seqüência (comparação semântica). A seção 3.2 abordará a Comparação Sintática de Genomas. Como a Comparação Semântica de Genomas não faz parte dos objetivos do presente trabalho, esta só será tratada no parágrafo seguinte.

A Comparação Semântica de Genomas, também chamada de Genômica Funcional, tenta associar funções à seqüência genômica. As funções do genoma dividem-se em poucas categorias: metabolismo, regulação, sinais e construção. Vias metabólicas convertem a energia química derivada de fontes ambientais (comida, por exemplo) em trabalho útil na célula. Vias regulatórias são mecanismos bioquímicos que controlam o que o DNA faz: quando ele é expresso e quando não é, por exemplo. A regulação gênica não envolve apenas genes expressos, mas sinais de seqüência e de estrutura no DNA, onde proteínas regulatórias podem se ligar. Vias sinalizadoras controlam, entre outras coisas, o movimento de substâncias de um compartimento da célula para outro. Desvendar as complexas redes de interações que compõem estas vias é o trabalho de bioquímicos e de biólogos moleculares, os quais podem ser auxiliados pela Computação [Gibas and Jambeck, 2001].

3.1 Medidas de Distância

A distância entre dois objetos é útil por definir quanto cada objeto é diferente do outro. Para isso, são atribuídos custos a operações elementares de edição e procuramos a composição menos custosa destas que transforma um objeto em outro. Distância é, portanto, uma medida de quanto um objeto difere do outro.

Uma **distância** em um conjunto E é uma função $d: E \times E \mapsto \mathbb{R}$ tal que:

1. $d(x, x) = 0$, para todo $x \in E$;
2. $d(x, y) > 0$, para todo $x, y \in E$, tal que $x \neq y$;
3. $d(x, y) = d(y, x)$, para todo $x, y \in E$ (d é *simétrica*);
4. $d(x, y) \leq d(x, z) + d(z, y)$, para todo x, y e $z \in E$.

As medidas de distância podem levar em conta a seqüência de caracteres de um genoma ou gene, bem como considerar o genoma um conjunto de blocos. De qualquer forma, a definição permanece válida; o que muda é a forma como os custos são atribuídos e quais são as operações elementares.

3.1.1 Distâncias entre Seqüências

No caso das seqüências, é possível definir uma distância no conjunto de todas as seqüências do alfabeto Σ baseado na quantidade de esforço necessário para transformar uma delas em outra.

Pela aplicação sucessiva de um número de operações admissíveis, qualquer seqüência pode ser transformada em outra. Se atribuímos um custo para cada operação admissível, podemos definir a distância entre duas seqüências como o custo total mínimo necessário para transformar uma em outra. As operações admissíveis são:

- Substituição de um caractere a por um caractere b;
- Inserção ou remoção de um caractere qualquer.

Para cobrar estas operações, usa-se uma medida de custo. O custo de uma série σ de operações é a soma dos custos individuais e é denotada por $custo(\sigma)$.

A distância entre duas seqüências S_1 e S_2 de acordo com a medida de custo é:

$$d(S_1, S_2) = \min_{\sigma \in \mathcal{S}(S_1, S_2)} custo(\sigma),$$

onde $\mathcal{S}(S_1, S_2)$ é o conjunto de todas as séries de operações que transformam S_1 em S_2 [Setubal and Meidanis, 1997].

Vários trabalhos definiram medidas de distância para seqüências, tais como a Distância de Edição [Levenshtein, 1966], Distância Estatística [Mironov and Alexandrov, 1988], Significância Algorítmica [Milosavljević, 1999] e a distância baseada na Simetria da Informação [Chen et al., 2000].

3.1.2 Distâncias de Rearranjo

Ao se comparar genomas inteiros entre espécies, nem sempre se quer computar distâncias que se baseiam em mutações pontuais, como substituições, remoções e inserções. O conjunto de operações básicas a serem usadas tem de envolver mutações maiores. Pedaços de cromossomos podem ser trocados de várias maneiras, afetando seções que possuem uma porcentagem significativa do cromossomo. Um pedaço pode ser movido ou copiado para outro lugar, ou pode sair de um cromossomo e parar em outro. Tais movimentos são chamados de **rearranjos de genoma**.

Considere o exemplo mostrado na figura 3.1.

Cada seta rotulada é chamada de **bloco**. Um bloco é uma seção do genoma, possivelmente contendo mais de um gene, e é transcrito, quase sempre, como uma unidade. A seta indica que os blocos possuem uma orientação. A razão pela qual os blocos são

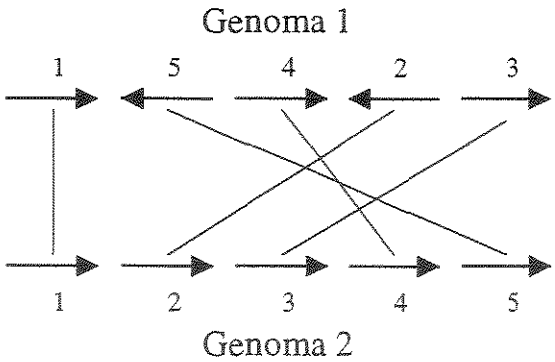


Figura 3.1: Relações entre dois genomas.

orientados tem a ver com a fita onde a transcrição ocorre, como explicado na seção 1.2. Dois blocos em diferentes genomas têm o mesmo rótulo numérico se eles são **homólogos**, isto é, se eles contém os mesmos genes.

Os blocos podem sofrer uma série de operações, tais como: reversões, inversões, translocações, duplicações, remoções, inserções, fusões e fissões [Pevzner, 2000]. Destas, a mais estudada é a reversão, a qual pode ser definida como segue.

A operação de reversão opera em segmentos contíguos de blocos. Ela inverte a ordem dos blocos afetados e suas setas também [Setubal and Meidanis, 1997]. Pevzner cita uma série de trabalhos envolvendo a distância de reversão de genomas, bem como as medidas de translocação e duplicação [Pevzner, 2000].

3.2 Comparação Sintática

Assim como a comparação clássica de seqüências via programação dinâmica, a comparação sintática de genomas busca encontrar alinhamentos entre as seqüências de dois ou mais genomas. Porém, os métodos utilizados são diferentes e também o tipo de alinhamento desejado pode diferir muito do tipo de alinhamento obtido na programação dinâmica.

De acordo com o objetivo proposto, vários autores preocuparam-se em encontrar tais alinhamentos de diversas formas. Alguns procuraram similaridades somente em *repeats*, outros buscaram por sítios de ligações ribossomais, outros visavam encontrar *motifs* e alguns tentaram achar vários alinhamentos locais entre dois genomas.

Frishman, Mironov e Gelfand desenvolveram um algoritmo que encontra sítios de ligações ribossomais (RBS, *ribosome binding sites*). Este algoritmo é útil para a análise de genomas com poucos ou nenhum gene experimentalmente mapeado, visto que não utiliza uma amostra de aprendizado [Frishman et al., 1999].

Os autores utilizam a ferramenta BLAST [Altschul et al., 1990, Gish and States, 1993, Altschul et al., 1997, Zhang and Madden, 1997] para encontrar prováveis regiões codifica-

doras. Isto é feito utilizando genes conhecidos e confiáveis (genes não associados a projetos genomas, obtidos da base de dados PIR). A partir disso, o trecho que contém os vinte caracteres anteriores ao início do suposto gene é separado. Neste trecho procura-se o tamanho e a posição da caixa de Shine-Dalgarno utilizando técnicas provenientes da Teoria da Informação. Frishman e colegas demonstraram, empiricamente, que o método por eles utilizado é um bom estimador da fração de inícios de genes corretamente identificados.

Sagot desenvolveu um algoritmo que lida, também, com padrões de seqüências que ocorrem freqüentemente em genomas. Entretanto, seu algoritmo busca por ***motifs*** (subcadeias sobre um mesmo alfabeto que ocorrem nas seqüências com, no máximo, e erros). O algoritmo extrai *motifs* comuns a um conjunto de $N \geq 2$ seqüências. Um *motif* é um padrão que ocorre com, no máximo, e erros em, no mínimo, q seqüências distintas, onde q é um número entre 1 e N. Convém salientar que as subcadeias que representam os *motifs* podem não aparecer exatamente em nenhuma seqüência. *Motifs* são, portanto, objetos externos às seqüências e são considerados **modelos válidos** se satisfizerem o quorum q [Sagot, 1998].

O método proposto por Sagot inicia construindo uma árvore de sufixos das seqüências em questão. Além disso, os nós recebem informações adicionais para que a restrição q seja respeitada. Os modelos válidos são, então, encontrados após se percorrer os nós da árvore até que se atinja o tamanho requerido para os modelos ou estes não possam mais ser estendidos sem deixarem de ser válidos. Ao final, o algoritmo retorna todas as ocorrências de cada um dos *motifs* encontrados. Isto significa que, para cada *motif*, os nós de maior profundidade que satisfizeram os critérios q e e são retornados; ou seja, os rótulos de cada um destes nós correspondem às ocorrências de um *motif*.

Normalmente, *motifs* são seqüências curtas que possuem um padrão altamente conservado. Em um genoma, podem ocorrer ***repeats***, os quais são longas subcadeias de DNA que se repetem em duas ou mais regiões de um genoma. Os *repeats* podem ser diretos ou invertidos, ou seja, se são encontrados na mesma fita ou em fitas opostas (isto é, um na fita $5' \rightarrow 3'$ e outro na $3' \rightarrow 5'$). Além disso, podem ser exatos ou aproximados e podem, ainda, estar em **tandem** (duas ou mais subcadeias idênticas uma após a outra). Rivals e colegas propuseram um algoritmo que busca por *repeats* diretos e exatos [Rivals et al., 1997]. Já Landau, Schmidt e Sokol desenvolveram um algoritmo que busca por *repeats* aproximados em tandem [Landau et al., 2001].

O algoritmo proposto por Rivals e colegas busca por *repeats* longos para que estes não possam ser confundidos com simples repetições que ocorrem ao acaso, visto que um genoma é muito maior que o tamanho do alfabeto $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ ($|\Sigma| = 4$). A condição suficiente usada por eles para este teste baseia-se na compressão do *repeat*; ou seja, o teste de compressão só é positivo se a seqüência codificada é menor do que a seqüência original. Com isso em mente, o algoritmo, após construir a árvore de sufixos do genoma,

busca os maiores prefixos que pertencem a dois ou mais sufixos da árvore. A seguir, o algoritmo tenta detectar *repeats* aproximados, os quais neste caso são compostos por *repeats* exatos intercalados por regiões de inserção/remoção, desde que eles satisfaçam duas condições: o teste de compressão continue positivo e não ocorram sobreposições entre os *repeats*. Os autores não deixaram de notar que este algoritmo pode ser estendido para múltiplos genomas desde que se considere que um *repeat* possa ser um pedaço de DNA que é encontrado em genomas distintos.

Landau e colegas, por sua vez, procuravam encontrar *repeats* aproximados em um genoma. Para isso, eles consideraram que os *repeats* podiam conter até e erros para serem considerados similares. Além disso, eles procuravam apenas por *repeats* em tandem. Um ***repeat* aproximado em tandem** é uma seqüência periódica onde uma subcadeia S ocorre consecutivamente duas ou mais vezes desde que o somatório dos erros deste *repeat* seja menor ou igual a e. Para construir os *repeats* em tandem, os autores usam uma matriz $n \times n$, onde n é o tamanho do genoma, e o método de programação dinâmica. Os problemas relativos a programação dinâmica são minimizados devido à escolha de valores pequenos para e (assim poucas posições da matriz são analisadas).

Quando não se restringe tanto a quantidade de erros possíveis em uma seqüência, o método de programação dinâmica torna-se demasiado lento para seqüências do tamanho de um genoma. Por este motivo, Varré e colegas utilizaram o algoritmo de Leung e colegas para encontrar pares de segmentos com, no máximo, e erros entre si [Leung et al., 1991]. O objetivo de Varré e colegas também é mais ambicioso: encontrar tanto mutações pontuais (inserções, remoções e substituições) entre dois genomas quanto modificações baseadas no movimento de segmentos, tais como, duplicações, inversões e inserções de regiões longas da seqüência [Varré et al., 1999].

Para isso, eles propuseram a **distância de transformação**, a qual consiste na atribuição de custos a duas operações possíveis: inserções e cópias de segmentos. Assim, dados dois genomas, a distância de transformação entre eles é o tamanho da menor descrição das operações necessárias para se montar um genoma dado o outro. Os autores utilizam o algoritmo de Leung e colegas para encontrar os segmentos comuns aos dois genomas. Em seguida, buscam pelo conjunto de segmentos que não se sobreponham e que minimizem a diferença entre o tamanho de um dos genomas e o somatório do tamanho dos segmentos utilizados. Convém notar que a distância de transformação não é, na verdade, uma medida de distância por não ser simétrica.

Capítulo 4

O Algoritmo: BioDiff

O algoritmo proposto foi desenvolvido com o intuito de indicar relações sintáticas qualitativas entre dois genomas bacterianos, mais especificamente as semelhanças entre dois genomas. Ao observarmos o aplicativo **diff** do sistema Unix, o qual é baseado no trabalho de Myers, notamos o relacionamento entre este último e os objetivos propostos para a dissertação [Myers, 1986].

Entretanto, como adaptar a idéia de **diff** para objetos com propriedades tão diferentes dos arquivos usados como entrada? Afinal, em um arquivo normal, a ordem dos caracteres é importante; em um genoma esta ordem não é tão importante, o que interessa é saber se trechos de um genoma estão presentes no outro.

Outro problema refere-se ao critério de similaridade a ser usado. O programa **diff** verifica se cada linha de um arquivo é exatamente igual a alguma linha do outro. Em um genoma, a igualdade absoluta deve ser substituída por similaridade, acrescida de algum critério de relevância. Usar o mesmo critério de similaridade de **diff** faria com que quaisquer dois genomas fossem considerados iguais, caso se editasse o genoma com um caractere em cada linha e se possibilitasse a busca de semelhanças em ordem relativa.

Para resolver estes problemas, utilizamos o conceito de Complexidade Condicional de Kolmogorov, também chamado de Tamanho de Descrição Mínima e batizamos o algoritmo como **BioDiff**.

4.1 Definições

Para que melhor se compreenda o algoritmo, três definições são importantes: casamentos exatos relevantes e relevantes em média e casamentos aproximados relevantes. É a partir destas definições que se apresenta o elo entre Complexidade Condicional de Kolmogorov e Casamento de Padrões.

Para que se determine a relevância de um casamento entre uma subcadeia S e uma

seqüência G, recorreremos à seguinte fórmula:

$$f_K(S|G) = \min\{E_1(p(S,G)) + E_1(|S|), \ell(S)\},$$

onde $f_K(u|v)$ é a aproximação para $K(u|v)$, $E_1(x)$ é o tamanho da versão auto-delimitada de x, $p(S,G)$ é a posição inicial da subcadeia S em G, $|S|$ é o tamanho, em caracteres, da subcadeia S e $\ell(S)$ é o tamanho, em bits, de S.

Isto significa que dada a seqüência G_y, queremos descrever outra seqüência G_z utilizando subcadeias da própria seqüência G_z ou alguma subcadeia comum a G_y e G_z. A menor descrição de uma subcadeia S comum às duas seqüências é, neste caso, o tamanho da posição inicial da subcadeia na seqüência G_y mais o tamanho da subcadeia, ou a própria subcadeia. Ou seja, se $f_K(S|G_y) \leq \ell(S)$, então a subcadeia S será representada pela seqüência de tamanho $f_K(S|G_y)$; caso contrário, S será representada por ela mesma.

Como exemplo, considere as seqüências $G_z =$"`ACCTGTGGAAAT`", $G_y =$"`GGAAATTACTT`" e $S =$"`GGAAAT`". Neste caso, $f_K(S|G_y) = 10$, pois $E_1(p(S,G_y)) + E_1(|S|) = E_1(1) + E_1(6) = 10 \leq 12 = \ell(S)$. Uma possível descrição de G_z dado G_y seria "`ACCTGT 101 1110110`", usando 22 bits. Os caracteres em branco são apenas para clareza, não fazem parte da descrição.

Estas considerações motivam as seguintes definições. Para elas, considere S uma subcadeia comum a G_1 e G_2, $p(S,G_2)$ a posição inicial da subcadeia S no genoma G_2, $|S|$ o tamanho, em caracteres, da subcadeia S, $\ell(S)$ o tamanho, em bits, de S e $E_1(x)$ o tamanho, em bits, da versão auto-delimitada de x.

Definição 4.1 *Se $E_1(p(S,G_2)) + E_1(|S|) \leq \ell(S)$, dizemos que S é um **casamento exato relevante**; caso contrário, não é relevante.*

Note que, no caso de bases nitrogenadas (adenina, citosina, guanina e timina), são necessários 2 bits para diferenciá-las. Se fossem aminoácidos, seriam necessários 5 bits para isso, pois $\lceil \log_2(20) \rceil = 5$.

O critério da definição 4.1 é desconfortável para implementar, pois, para saber se uma subcadeia é relevante, precisamos conhecer sua posição inicial no genoma G_2. Para melhorar isto, substituímos a posição inicial por um valor fixo de forma que estas não influenciassem os casamentos. Foi escolhido $|G_2|/2$ para representar um valor "médio" em relação às possíveis posições iniciais em G_2. A idéia é aumentar o número de casamentos analisados, mas não excessivamente. Na prática, a escolha de $|G_2|/2$ foi satisfatória.

Definição 4.2 *Se $E_1(|G_2|/2) + E_1(|S|)) \leq \ell(S)$, então S é um **casamento exato relevante em média**; caso contrário, não é relevante em média.*

Teorema 4.1 *Para todo casamento relevante em média, temos $|S| > \log_{|\Sigma|}(|G_2|)$.*

Prova:

$$E_1(|G_2|/2) + E_1(|S|)) \leq \ell(S)$$

$$2(\lfloor \log_2(|G_2|/2) \rfloor + 1) + 1 + 2(\lfloor \log_2(|S|) \rfloor + 1) + 1 \leq \lceil \log_2(|\Sigma|) \rceil |S|$$

$$2(\lfloor \log_2(|G_2|) \rfloor - 1 + 1) + 1 + 2\lfloor \log_2(|S|) \rfloor + 3 \leq \lceil \log_2(|\Sigma|) \rceil |S|$$

$$2\lfloor \log_2(|G_2|) \rfloor + 2\lfloor \log_2(|S|) \rfloor + 4 \leq \lceil \log_2(|\Sigma|) \rceil |S|$$

$$\frac{2\lfloor \log_2(|G_2|) \rfloor}{\lceil \log_2(|\Sigma|) \rceil} + \frac{2\lfloor \log_2(|S|) \rfloor + 4}{\lceil \log_2(|\Sigma|) \rceil} \leq |S|$$

Como $(2\lfloor \log_2(|S|) \rfloor + 4)/\lceil \log_2(|\Sigma|) \rceil \geq 0$,

$$|S| > \frac{2\lfloor \log_2(|G_2|) \rfloor}{\lceil \log_2(|\Sigma|) \rceil}$$

Se $|G_2| \geq 8$ e $|\Sigma| \geq 4$,

$$|S| > \frac{\log_2(|G_2|)}{\log_2(|\Sigma|)}$$

$$|S| > \log_{|\Sigma|}(|G_2|).$$

□

Pelo teorema 4.1, temos que $|S| > \log_{|\Sigma|}(|G_2|)$ para que o casamento seja relevante em média. Entretanto, usaremos logaritmo na base 2, ao invés de na base $|\Sigma|$ por dois motivos: elas só se diferenciam por uma constante ($\log_2 |\Sigma|$), o que mantém a complexidade do algoritmo e $\log_2$ é mais restritivo que $\log_{|\Sigma|}$, reduzindo, assim, o número de casamentos encontrados.

Considere C_a uma série de casamentos exatos relevantes não sobrepostos e em ordem crescente em relação aos genomas G_1 e G_2. Ou seja, os casamentos em C_a estão ordenados em relação a suas posições iniciais em G_1 e G_2.

Considere $\sigma(S_i, S_{i+1})$ a região intermediária entre os casamentos exatos relevantes S_i e S_{i+1} de C_a e $d(S_i, S_{i+1}, G_1)$ a distância, em caracteres, entre os casamentos exatos relevantes S_i e S_{i+1} de C_a em relação a G_1. De maneira análoga, $d(S_i, S_{i+1}, G_2)$ é definida.

Às regiões intermediárias aos casamentos exatos relevantes de C_a são atribuídos custos da seguinte forma:

$$custo_\sigma(S_i, S_{i+1}) = penalidade \max\{d(S_i, S_{i+1}, G_1), d(S_i, S_{i+1}, G_2)\},$$

onde $custo_\sigma(S_i, S_{i+1})$ é o tamanho, em bits, da representação de $\sigma(S_i, S_{i+1})$ e *penalidade* é o custo de uma operação de inserção, remoção ou substituição.

Para este trabalho, $penalidade = 2 + \lceil \log_2 |\Sigma| \rceil$ para qualquer uma das operações possíveis (inserção, remoção, substituição), onde Σ é o alfabeto dos genomas e o número

2 é o número de bits necessários para a codificação das operações possíveis. Como $custo_\sigma(S_i, S_{i+1})$ é definido de forma semelhante à distância entre duas seqüências da seção 3.1.1, existem $\min\{d(S_i, S_{i+1}, G_1), d(S_i, S_{i+1}, G_2)\}$ operações de substituição na região $\sigma(S_i, S_{i+1})$. As demais operações são remoções ou inserções.

Como exemplo, considere ATAC e GTAGCC as subcadeias de G_1 e G_2, respectivamente, pertencentes à região $\sigma(S_i, S_{i+1})$. Serão consideradas 4 substituições (ATAC e GTAG) e 2 remoções (CC) para representar esta região — lembramos que se quer descrever G_1 em relação a G_2. Assim, $custo_\sigma(S_i, S_{i+1}) = (2 + \lceil \log_2 |\Sigma| \rceil)6 = 24$ bits.

Isto motiva a seguinte definição:

Definição 4.3 *Seja r o número de casamentos exatos relevantes contidos em C_a. Se $E_1(p(S_1, G_2)) + \sum_{i=1}^{r} E_1(|S_i|) + \sum_{i=1}^{r-1} custo_\sigma(S_i, S_{i+1}) \leq \lceil \log_2 |\Sigma| \rceil (p(S_r, G_2) - p(S_1, G_2) + |S_r|)$, então C_a é um **casamento aproximado relevante**; caso contrário, não é relevante.*

Esta definição diz que C_a só será um casamento aproximado relevante se, apesar de se adicionar os custos para representar suas regiões intermediárias, o tamanho da representação de C_a for menor ou igual ao tamanho da seqüência com início em $p(S_1, G_2)$ e final em $p(S_r, G_2) + |S_r|$.

4.2 Algoritmo

Um resumo dos passos principais de **BioDiff** será dado e os comentários virão a seguir.

Passo 1: Construir o vetor de sufixos V_{G_2} de G_2.

O vetor de sufixos foi escolhido porque se a busca por similaridades for restrita, basta verificar o conteúdo de $V_{G_2}[j]$ (o j-ésimo sufixo de G_2 em ordem lexicográfica não-decrescente) para saber se o sufixo pertence a uma seqüência permitida ou proibida (o que não seria possível diretamente usando árvores de sufixo). Além disso, como os genomas podem ter mais de uma seqüência, a entrada para o algoritmo `SuffixSort` [Larsson and Sadakane, 1999] foi adaptada da seguinte maneira:

- É atribuído a cada seqüência, um terminador diferente;
- Os terminadores são lexicograficamente menores do que qualquer letra do alfabeto usado;
- As seqüências são concatenadas, incluindo seus terminadores, e a posição inicial e o tamanho de cada uma são armazenados para posterior identificação.

Passo 2: Localizar o intervalo $[j..k]$ de V_{G_2} de forma que o tamanho do casamento entre todos os sufixos do intervalo e a subcadeia de G_1 com início na posição i seja igual a $\lceil \log_2 |G_2| \rceil$ (ver definição 4.2). Inserir estas informações na lista de **casamentos exatos relevantes em média**.

Este passo localiza casamentos exatos de acordo com o critério de **casamento relevante em média**. Esta foi a melhor forma encontrada para não se perder **casamentos exatos relevantes**, visto que a busca no vetor de sufixos retorna apenas o intervalo onde houve o casamento do padrão pesquisado com os sufixos — o que é insuficiente para se saber se o casamento é relevante ou não.

Passo 3: Analisar cada um dos índices pertencentes ao intervalo $[j..k]$ de todos os **casamentos relevantes em média** e inserir os índices que satisfizerem o critério de relevância (ver definição 4.1) na lista de **casamentos exatos relevantes**.

Nesta etapa do algoritmo já se sabe a posição inicial do casamento na seqüência G_2 e, por isso, todos os dados necessários para decidir se o casamento é relevante estão disponíveis.

A partir do próximo passo, cada **casamento exato relevante** passa a ser um vértice de um **Grafo Acíclico Dirigido** (DAG, *Directed Acyclic Graph*). E a existência de um arco entre dois vértices indicam a possibilidade de estender os dois casamentos para um **casamento aproximado relevante**.

Passo 4: Para cada par de vértices u e v do DAG, inserir um arco dirigido de u para v se as posições iniciais do casamento v em relação a G_1 e a G_2 forem maiores do que as posições finais do casamento u em relação a G_1 e a G_2; e se a união de u e v formar um **casamento aproximado relevante** (ver definição 4.3).

Este passo é importante para que só sejam estendidos os casamentos que realmente possam gerar casamentos aproximados. Para resolver este passo, pensou-se em utilizar o algoritmo proposto por Meidanis em sua tese de doutorado, mas, infelizmente, isto não foi possível porque os arcos possuem custos associados a eles [Meidanis, 1992].

Passo 5: Atribuir pesos (ou custos) para os arcos do DAG, transferindo o peso do vértice origem para os arcos que partem dele. Para os vértices que não possuem arcos partindo deles, transferir seus pesos para os arcos que chegam neles.

Os pesos $w(\cdot)$ são calculados da seguinte maneira:

- $w(v) = \ell(v) - E_1(p(v, G_2)) - E_1(|v|)$
- $w(u, v) = w(u) - (2 + \lceil \log_2 |\Sigma| \rceil)(p(v, G_2) - p(u, G_2) - |u|)$

Onde $|t|$ é o tamanho, em caracteres, da seqüência associada ao vértice t, $\ell(t)$ é o tamanho, em bits, da seqüência em t, $p(t, G_2)$ é a posição inicial em G_2 da seqüência associada ao vértice t, Σ é o alfabeto dos genomas e $E_1(x)$ é o tamanho, em bits, da

versão auto-delimitada de x. Aos arcos que chegam aos vértices sorvedouros do DAG, faz-se $w(u,v) = w(u,v) + w(v)$. Cabe lembrar, ainda, que todos os arcos terão pesos positivos, devido ao passo 4.

Passo 6: Executar um algoritmo que resolva o problema do caminho de peso máximo entre todos os pares de vértices de um DAG cujos arcos possuem peso não-negativo. Utilizamos o algoritmo de Dijkstra para isso, entretanto poderia ser feita uma implementação mais simples com ordenação topológica.

Este passo vai produzir os caminhos (ou casamentos aproximados) mais relevantes entre todos os pares de vértices.

Passo 7: Ordenar os caminhos de peso máximo em ordem decrescente de peso e selecionar os caminhos que não se sobreponham em relação a G_1.

A ordenação é utilizada para que a seleção dos caminhos seja mais facilmente implementada. Já a questão da não-sobreposição está relacionada ao fato de que **BioDiff** é uma aproximação para $K(G_1|G_2)$ — que é o tamanho da menor descrição do genoma G_1 dado o genoma G_2 como entrada — e, portanto, a descrição de G_1 será mais econômica se cada um dos caracteres de G_1 aparecerem somente em um único casamento aproximado com G_2.

A união destas idéias gerou a versão final de **BioDiff**, o qual utiliza a implementação dos autores de `SuffixSort` para ordenar o vetor de sufixos [Larsson and Sadakane, 1999]. O algoritmo básico de **BioDiff** é mostrado na figura 4.1.

4.3 Análise

O algoritmo **BioDiff** executa em tempo $\mathcal{O}(n \log n + m^3/\log^3 n)$ e, dado que as variáveis ocupam uma palavra, utiliza espaço $\mathcal{O}(n + m^2/\log^2 n)$, onde $n = |G_2|$ e $m = |G_1|$.

Na figura 4.2, mostramos a análise de tempo e espaço para cada uma das linhas do algoritmo.

Pode-se perceber que da linha 9 em diante, a complexidade do algoritmo é proporcional ao número de casamentos exatos relevantes em média encontrados. Não se pode melhorar essa complexidade a partir da linha 11, pois não se tem garantia alguma de que, no pior caso, o número de casamentos exatos relevantes seja menor que o de casamentos exatos relevantes em média (assintoticamente falando).

Também é importante salientar que tanto a complexidade de pior caso de tempo, quanto a de espaço, são muito dependentes da escolha do menor tamanho admissível para um casamento exato relevante em média, no caso $\lceil \log_2 n \rceil$. É esta escolha que faz com que o algoritmo seja mais ou menos seletivo. Logicamente, quanto mais seletivo, mais rápido será o tempo de execução do algoritmo visto que o espaço de busca será menor.

Entrada: Genomas G_1 e G_2
Saída: Conjunto de Semelhanças $C_{aproximado}$ entre G_1 e G_2

```
BioDiff(G1, G2)
 1  m ← |G1|;
 2  n ← |G2|;
 3  V_G2 ← ordenarVetorSufixos(G2);
 4  i ← 1;
 5  while i ≤ m
 6  do intervalo ← encontrarSubcadeia(V_G2, i, G1);
 7     C_relevante em media ← inserirCasamento(intervalo.inicio, intervalo.fim, i);
 8     i ← i + ⌈log2 n⌉; \\ casamento exato relevante em media
 9  while C_relevante em media ≠ ∅
10  do C_relevante ← selecionarCasamentos(primeiro(C_relevante em media));
11  for u ← 1 to |C_relevante|
12  do
13     for v ← 1 to |C_relevante|
14     do
15        W[u][v] ← calcularPesoArcoDirigido(C_relevante, u, v);
16  Caminhos ← caminhoPesoMaximoTodosParesVertices(W);
17  ordenarArestasPesosDecrescentes(Caminhos);
18  C_aproximado ← selecionarCaminhosNaoSobrepostos(Caminhos);
19  mostrar(C_aproximado);
```

Figura 4.1: Algoritmo BioDiff.

É necessário, portanto, justificar a razão pela qual foi escolhido $\lceil \log_2 n \rceil$. Pela definição 4.2, os casamentos exatos são todos considerados como se iniciassem na posição $n/2$ do genoma G_2 (lembramos que **BioDiff** é uma aproximação para $K(G_1|G_2)$), de forma que os casamentos não sofressem, inicialmente, a influência de suas posições iniciais em G_2. A motivação foi, portanto, estabelecer um valor igual para $p(S, G_2)$ para todos os casamentos a serem encontrados.

O motivo que nos levou à escolha de $p(S, G_2) = n/2$ foi justamente a redução da complexidade do algoritmo por um fator logarítmico, sem perder, apesar disso, muitos casamentos que poderiam ser relevantes. Levamos em conta, também, a advertência de Rivals e colegas: casamentos pouco extensos podem ser obra do acaso e, portanto, não devem ser considerados [Rivals et al., 1997].

Linha(s)	Tempo	Espaço
1	$\mathcal{O}(1)$	$\mathcal{O}(m)$
2	$\mathcal{O}(1)$	$\mathcal{O}(n)$
3	$\mathcal{O}(n \log n)$	$\mathcal{O}(n)$
4	$\mathcal{O}(1)$	$\mathcal{O}(1)$
5–8	$\mathcal{O}(m \log n)$	$\mathcal{O}(m/\log n)$
9–10	$\mathcal{O}(m/\log n)$	$\mathcal{O}(m/\log n)$
11–15	$\mathcal{O}(m^2/\log^2 n)$	$\mathcal{O}(m^2/\log^2 n)$
16	$\mathcal{O}(m^3/\log^3 n)$	$\mathcal{O}(m^2/\log^2 n)$
17	$\mathcal{O}(m^2(\log m^2 - \log\log^2 n)/\log^2 n)$	$\mathcal{O}(m^2/\log^2 n)$
18	$\mathcal{O}(m^2/\log^2 n)$	$\mathcal{O}(m^2/\log^2 n)$
19	$\mathcal{O}(m^2/\log^2 n)$	$\mathcal{O}(m^2/\log^2 n)$

Figura 4.2: Análise de Tempo e Espaço de BioDiff

4.4 Opções de Entrada

O algoritmo de **BioDiff** inclui uma série de funcionalidades, as quais podem ser divididas em 3 tipos: formato do genoma, características das seqüências componentes do genoma e restrição da busca por similaridade.

Os genomas podem ser dados por suas seqüências de bases nitrogenadas, pelas seqüências de aminoácidos de seus genes ou por uma seqüência mista de bases e aminoácidos. Comparações entre formatos diferentes produzirão resultados sem nenhuma significância, pois **BioDiff** compara os genomas sintaticamente. Os dois primeiros formatos devem estar no padrão FASTA. Já o formato misto, deve ser dado também no padrão FASTA, mas com as seguintes diferenças:

- Ambas as fitas devem estar presentes no genoma;
- As bases correspondentes a um gene devem ser substituídas pelos aminoácidos derivados do gene.

Estas diferenças fazem com que as informações sobre os genes sejam mais completas, como, por exemplo, incluindo *operons*[1] da maneira correta e permitindo que genes em fitas opostas apareçam. Entretanto, uma característica dos genomas não é capturada: genes que se sobrepõe na mesma fita. Assim, decidimos deixar que este problema seja resolvido pelo usuário da forma que ele achar melhor — eliminando um ou mais genes, ou inserindo os genes lado a lado como se não houvesse sobreposição.

[1]Em bactérias, a expressão de muitos genes é regulada ao nível da transcrição. Em muitos casos, vários genes com funções coordenadas, junto com uma seqüência regulatória chamada "operador", formam um *operon* [Futuyma, 1997, página 50]

A inclusão do formato misto é interessante porque a busca por similaridades pode ser feita entre organismos que possuem diferentes códigos genéticos sem afetar os resultados. Assim, a divergência entre códons torna-se nula, caso a codificação destes em cada um dos organismos resulte no mesmo aminoácido.

Com relação às características das seqüências componentes somente uma opção pode ser feita: incluir ou não os plasmídeos. Esta decisão pode ser tomada de forma que somente os plasmídeos de um dos genomas seja incluído. Os plasmídeos devem estar presentes no mesmo arquivo do cromossomo principal.

O terceiro tipo de funcionalidade permite restringir a busca por similaridades. Ao invés de se procurar similaridades entre todas as seqüências, pode-se fazer buscas entre:

- Cromossomo principal de G_1 e cromossomo principal de G_2;
- Plasmídeos de G_1 e plasmídeos de G_2;
- Plasmídeos de G_1 e cromossomo principal de G_2;
- Cromossomo principal de G_1 e plasmídeos de G_2;
- Qualquer combinação dos itens anteriores.

Para que estas funcionalidades sejam mais facilmente manipuladas por **BioDiff** é preciso ter como entrada um arquivo de configurações que indique todas estas características. Este arquivo é dividido em 2 partes: configurações gerais e configurações específicas de cada genoma.

As configurações gerais devem estar no início do arquivo e compreendem o formato dos genomas e a restrição das buscas. Já as configurações específicas compreendem a presença ou ausência dos plasmídeos e a identificação das seqüências presentes nos genomas na mesma ordem em que aparecem no arquivo associado ao genoma. Esta identificação compreende o tipo da seqüência (cromossomo principal ou plasmídeo), a fita ($5' \rightarrow 3'$ ou $3' \rightarrow 5'$) e um nome. A figura 4.3 mostra um exemplo de arquivo de configurações. As linhas iniciadas com % indicam comentários.

Uma característica que não foi incluída é a opção de dizer se um cromossomo é circular ou linear. Esta atitude foi tomada para simplificar o algoritmo, dado que sua inclusão não iria aumentar significativamente a qualidade da solução encontrada. Isto ocorre porque os casamentos aproximados são formados pela união de dois ou mais casamentos exatos e a probabilidade de que um casamento aproximado bastante significativo seja totalmente perdido é muito baixa. Na prática, isto só aconteceria se entre dois casamentos exatos extremamente grandes estivesse presente a região que possui as bases $|G_1|$ e 1 do genoma linearizado e, além disso, esta região deveria ter muitas bases separando os dois casamentos

```
%General Settings
Format=DNA
%DNA-AA-MIX
Searches=1010
%MChromossome-MChromossome; Plasmid-Plasmid;
%Plasmid-MChromossome; MChromossome-Plasmid;
%0000 will result in no output
%Specific Settings
%Genome #1
Plasmids=YES
SequenceID=MC;5;xylella main chromossome
SequenceID=MC;3;xylella main chromossome
SequenceID=P;5;xylella pXF51 plasmid
SequenceID=P;3;xylella pXF51 plasmid
SequenceID=P;5;xylella pXF1.3 plasmid
SequenceID=P;3;xylella pXF1.3 plasmid
%Genome #2
Plasmids=NO
SequenceID=MC;5;xanthomonas main chromossome
```

Figura 4.3: Exemplo de Arquivo de Configurações.

— para que a união destes casamentos fosse considerada relevante (ver definição 4.3 na página 29).

4.5 Formato da Saída

Dada a forma como os casamentos aproximados são tratados (ver seção 4.2), o formato que achamos mais adequado para a saída é como segue.

Para cada casamento, indicamos qual seqüência casou com qual (cromossomo principal, plasmídeo) e os caracteres iniciais e finais de cada uma das seqüências. A figura 4.4 mostra um exemplo da saída de **BioDiff**.

Os casamentos mostrados são aqueles obtidos quando se faz a aproximação da Complexidade Condicional de Kolmogorov — $K(G_1|G_2)$. Outra maneira de se visualizar a saída é considerar um plano, onde cada eixo representa um genoma e cada retângulo, um casamento. Um exemplo desta representação é mostrado na figura 4.5.

Note que não há sobreposição entre os retângulos em relação ao eixo x. Isto ocorre porque não permitimos a ocorrência de casamentos sobrepostos em relação ao genoma

```
Number of approximate matches:  4
%G#1;G#2;start#1;start#2;end#1;end#2
species1 MC;species2 MC;5;5654;3246;8855
species1 MC;species2 MC;3550;236;3903;601
species1 MC;species2 MC;56300;523100;57000;523800
species1 Plasmid;species2 MC;1;56687;1400;57250
```

Figura 4.4: Exemplo da Saída de BioDiff.

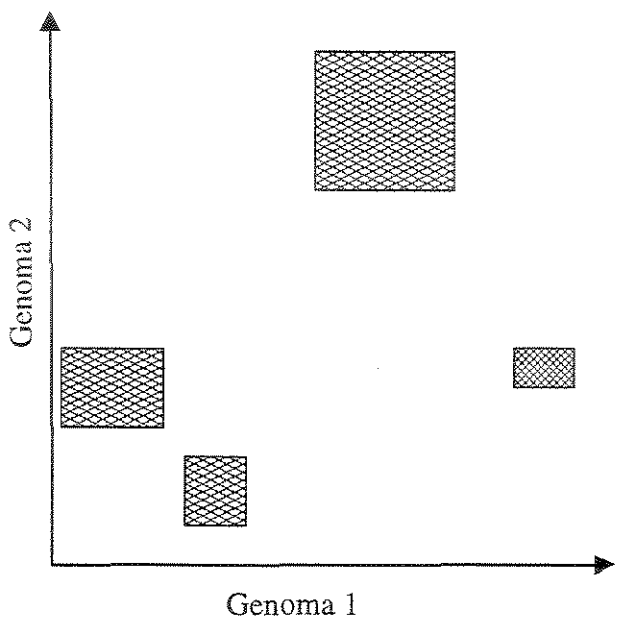


Figura 4.5: Representação no plano de casamentos.

que se quer descrever (G_1).

Capítulo 5

Discussões

O algoritmo **BioDiff** busca mostrar as regiões similares entre dois genomas utilizando um critério de similaridade teoricamente fundamentado. A utilização da *versão auto-delimitada* de um número como instrumento codificador das operações elementares foi bastante satisfatória como aproximação da Complexidade Condicional de Kolmogorov.

A dissertação traz como contribuição à área um novo algoritmo para comparar genomas sintaticamente que, diferentemente dos demais, evita a utilização de parâmetros para indicar se duas subcadeias são similares ou não por utilizar a teoria da Complexidade Condicional de Kolmogorov para isso. Desta maneira, evitam-se os necessários refinamentos destes parâmetros, os quais, quase sempre involuntariamente, são ajustados para que se produzam os resultados que esperamos encontrar.

Consideramos este trabalho a união das idéias dos trabalhos de Li e Varré por utilizar, do primeiro, o critério de similaridade baseado na Complexidade de Kolmogorov e, do segundo, a apresentação qualitativa das similaridades encontradas [Chen et al., 2000, Varré et al., 1999].

Apesar de tudo, esta é apenas uma forma de se abordar este problema de Comparação de Genomas. Muito provavelmente a realidade biológica é muito mais complexa do que qualquer um dos modelos matemáticos utilizados para a obtenção de resultados na área.

Comparar genomas é uma metodologia poderosa e emergente para explorar as forças que modelam as estruturas genômicas, tais como segmentos conservados ou blocos de genes — agrupamentos, em cada um dos genomas sendo comparados, de um conjunto de genes ortólogos que são internamente rearranjados, com ou sem a intrusão de outros genes.

Além disso, é importante para se tentar descobrir funções de regiões genômicas não-codificadoras, as quais são consideradas porções de sobra de um organismo para eventos diversos, tais como, mutações pontuais, inserções de segmentos, reversões e rearranjos diversos — todos ocorrendo ao acaso.

É no mínimo ingênuo pensar que forças seletivas não agem em estruturas genômicas que não conhecemos. É neste momento que a Computação, através de todo seu embasamento teórico, pode auxiliar as pesquisas que tentam desvendar a organização das moléculas essenciais à vida.

5.1 Trabalhos Futuros

Um dos melhoramentos mais interessantes ao algoritmo é a inclusão da busca por casamentos que não se alinhem exatamente, tal como faz o algoritmo de Leung e colegas [Leung et al., 1991]. Técnicas de Casamento Aproximado de Padrões são candidatas para isto, desde que exista uma maneira de se utilizar um limitante para a quantidade de diferenças baseado na Complexidade de Kolmogorov.

A utilização de uma nova ferramenta do conjunto BLAST, chamada `bl2seq`, também deve ser cuidadosamente analisada para se decidir se a inclusão dela é útil. `bl2seq` busca alinhamentos locais entre duas seqüências utilizando as mesmas idéias da ferramenta original [Altschul et al., 1990].

Uma alteração mais delicada que pode ser feita visando melhorar as similaridades apresentadas pelo algoritmo é utilizar uma codificação diferente da versão auto-delimitada de uma seqüência binária. Uma mudança nesta parte do algoritmo pode trazer variações na aproximação de $K(G_1|G_2)$ tanto para melhor quanto para pior.

Entretanto, deve-se ter cuidado com qualquer medida teórica para auxiliar as pesquisas em genômica, visto que, na maioria das vezes, acabamos esquecendo do problema motivador do trabalho e tentamos buscar medidas cada vez mais próximas da otimalidade. Andersson e Eriksson indicaram que simulações mostram que as distâncias menores subestimam o número de eventos reais, caso estas não levem em conta as particularidades biológicas de cada organismo [Andersson and Eriksson, 2000].

Esta advertência nos leva à última sugestão para este trabalho: incluir informações funcionais dos genomas a serem comparados. A Comparação Semântica de Genomas pode ajudar ao incluir dados sobre as funções conhecidas de porções do genoma, de modo que não se busque cegamente a melhor resposta do ponto de vista da formalização de um problema, mas a resposta mais adequada dadas as peculiaridades dos organismos.

Apêndice A

Resultados

Esse apêndice mostra resultados da execução do algoritmo entre cinco pares de genomas. Os genomas escolhidos e seus respectivos números de acesso no GenBank do NCBI (http://www.ncbi.nlm.nih.gov/) são:

- *Chlamydia trachomatis* (NC_000117);
- *Chlamydophila pneumoniae* (NC_002179);
- *Helicobacter pylori* cepa J99 (NC_000921);
- *Helicobacter pylori* cepa 26695 (NC_000915);
- *Listeria innocua* (NC_003212);
- *Listeria monocytogenes* (NC_003210);
- *Mycoplasma genitalium* (NC_000908);
- *Mycoplasma pneumoniae* (NC_000912);
- *Xylella fastidiosa* CVC (NC_002488);
- *Xylella fastidiosa* PD (Pierce's Disease);

O genoma da última espécie foi obtido a partir da página do Laboratório de Bioinformática do IC/UNICAMP (http://www.lbi.ic.unicamp.br/world/xf-grape/).

Este protótipo mostrou resultados preliminares animadores, pois detectou algumas das semelhanças conhecidas entre as espécies testadas. Revelou também que o método abordado funciona melhor entre espécies bem próximas.

Para os resultados apresentados a seguir, usamos somente o Cromossomo Principal de cada genoma. Além disso, o formato escolhido para os genomas foi sua seqüência de nucleotídeos. Nos gráficos, a primeira espécie é representada pelo **eixo y** e a segunda pelo **eixo x**.

A.1 *C. trachomatis* e *C. pneumoniae*

Tamanho, em bases, de *Chlamydia trachomatis*:	1042519
Tamanho, em bases, de *Chlamydophila pneumoniae*:	1229789
Número de casamentos exatos relevantes em média:	158
Número de casamentos exatos relevantes:	61
Número de casamentos aproximados relevantes:	56
Tamanho médio dos casamentos exatos relevantes:	34,131148
Tamanho do maior casamento exato relevante:	76
Tamanho do menor casamento exato relevante:	21

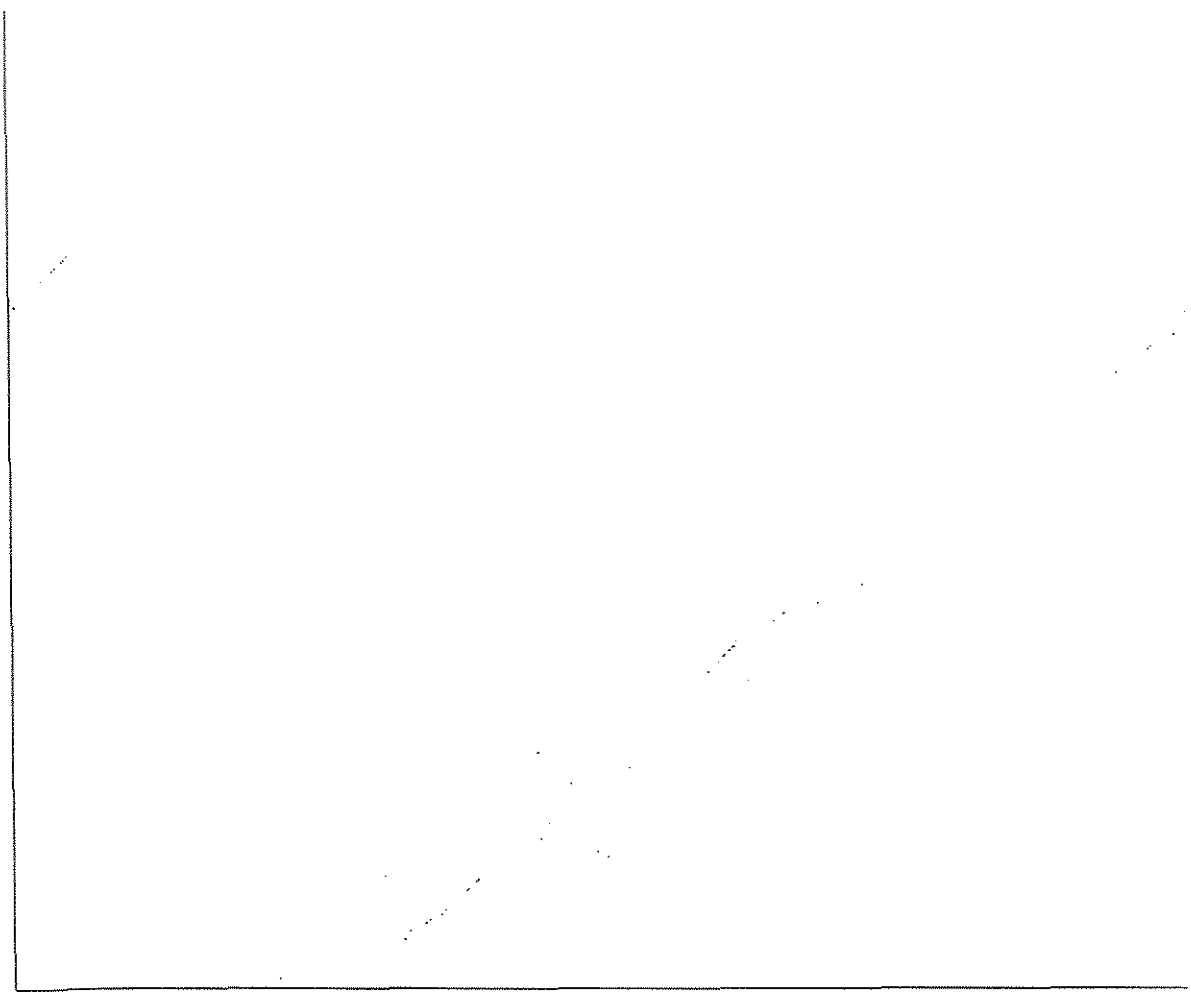


Figura A.1: *C. trachomatis* × *C. pneumoniae*

A.2 *H. pylori* cepa J99 e *H. pylori* cepa 26695

Tamanho, em bases, de *Helicobacter pylori* cepa J99:	1643831
Tamanho, em bases, de *Helicobacter pylori* cepa 26695:	1667825
Número de casamentos exatos relevantes em média:	20047
Número de casamentos exatos relevantes:	16935
Número de casamentos aproximados relevantes:	9382
Tamanho médio dos casamentos exatos relevantes:	50.696664
Tamanho do maior casamento exato relevante:	548
Tamanho do menor casamento exato relevante:	21

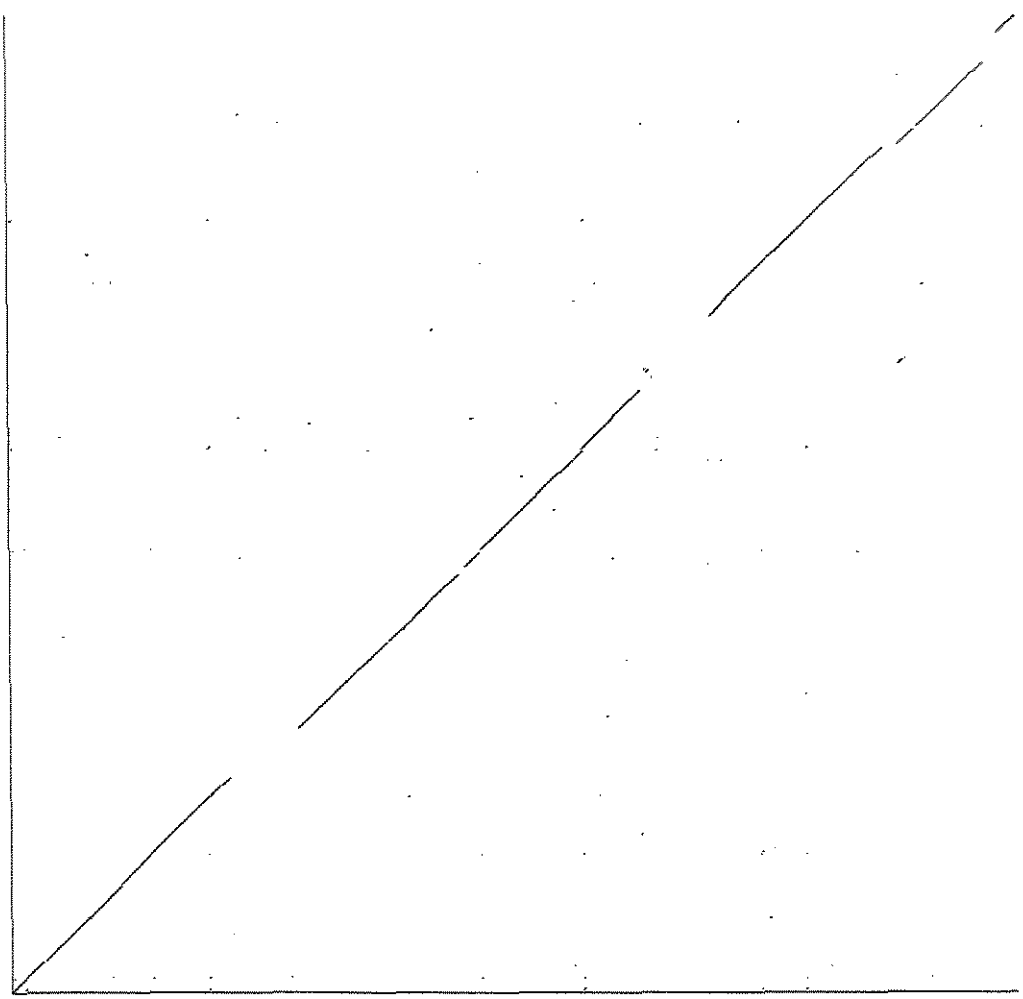


Figura A.2: *H. pylori* cepa J99 × *H. pylori* cepa 26695

A.3 *L. innocua* e *L. monocytogenes*

Tamanho, em bases, de *Listeria innocua*:	3011208
Tamanho, em bases, de *Listeria monocytogenes*:	2944528
Número de casamentos exatos relevantes em média:	20184
Número de casamentos exatos relevantes:	14622
Número de casamentos aproximados relevantes:	10075
Tamanho médio dos casamentos exatos relevantes:	55.963001
Tamanho do maior casamento exato relevante:	1256
Tamanho do menor casamento exato relevante:	22

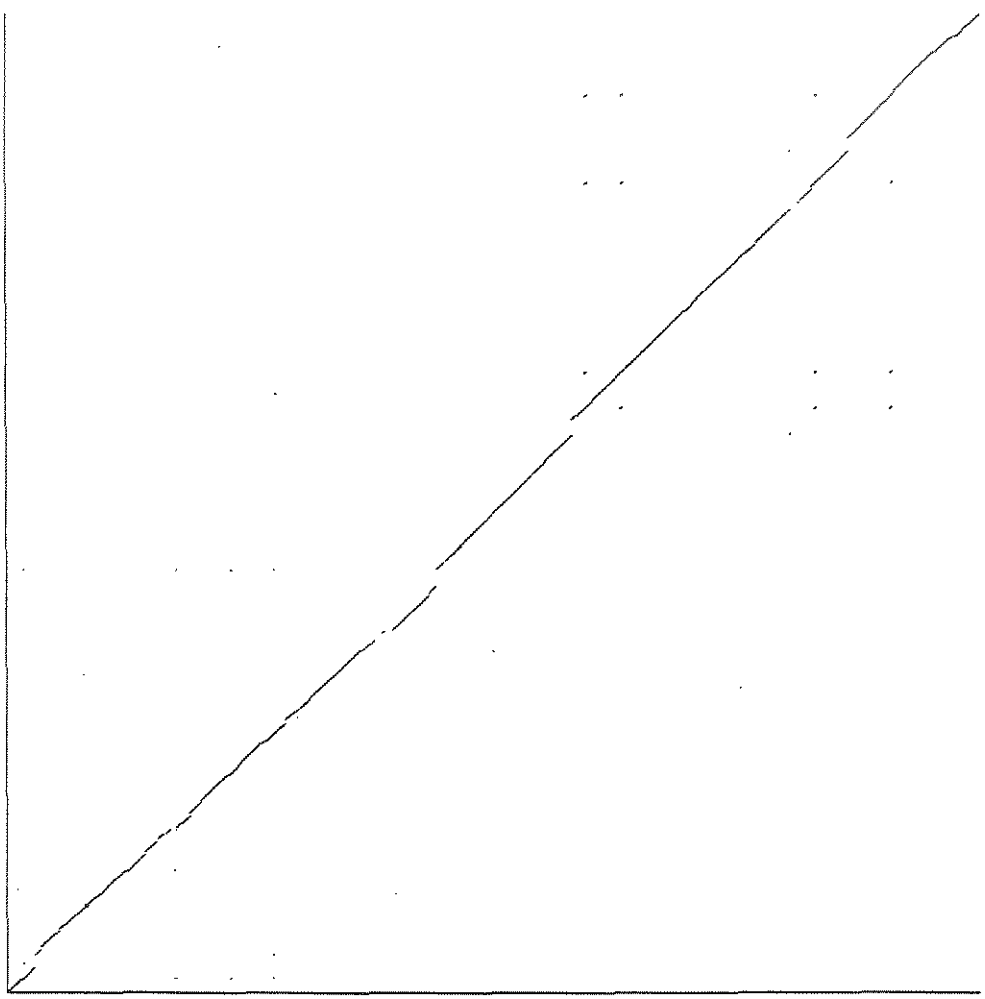


Figura A.3: *L. innocua* × *L. monocytogenes*

A.4 *M. genitalium* e *M. pneumoniae*

Tamanho, em bases, de *Mycoplasma genitalium*:	580074
Tamanho, em bases, de *Mycoplasma pneumoniae*:	816394
Número de casamentos exatos relevantes em média:	441
Número de casamentos exatos relevantes:	176
Número de casamentos aproximados relevantes:	140
Tamanho médio dos casamentos exatos relevantes:	50.500000
Tamanho do maior casamento exato relevante:	281
Tamanho do menor casamento exato relevante:	20

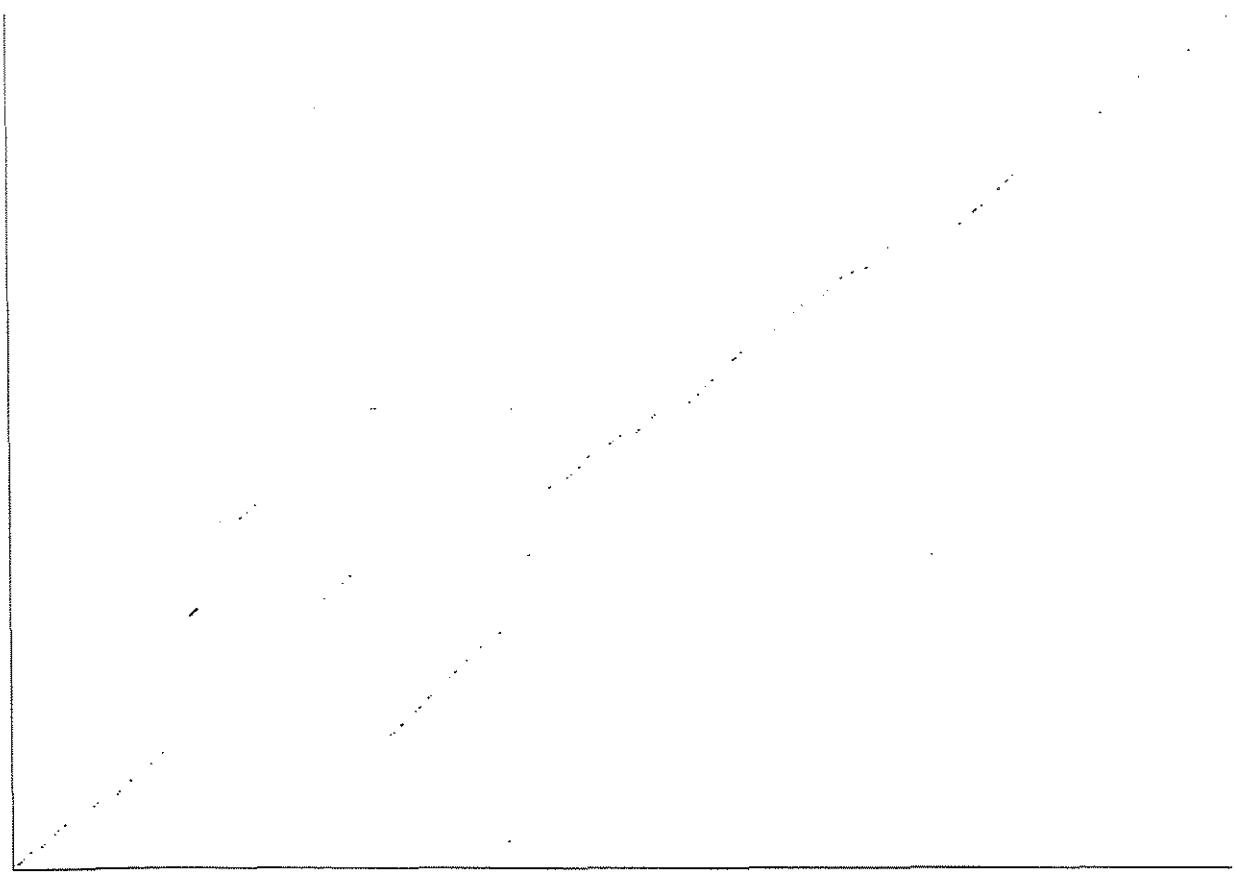


Figura A.4: *M. genitalium* × *M. pneumoniae*

A.5 *X. fastidiosa* CVC e *X. fastidiosa* PD

Tamanho, em bases, de *Xylella fastidiosa* CVC:	2679305
Tamanho, em bases, de *Xylella fastidiosa* PD:	2523614
Número de casamentos exatos relevantes em média:	21185
Número de casamentos exatos relevantes:	19021
Número de casamentos aproximados relevantes:	8521
Tamanho médio dos casamentos exatos relevantes:	65.920193
Tamanho do maior casamento exato relevante:	883
Tamanho do menor casamento exato relevante:	22

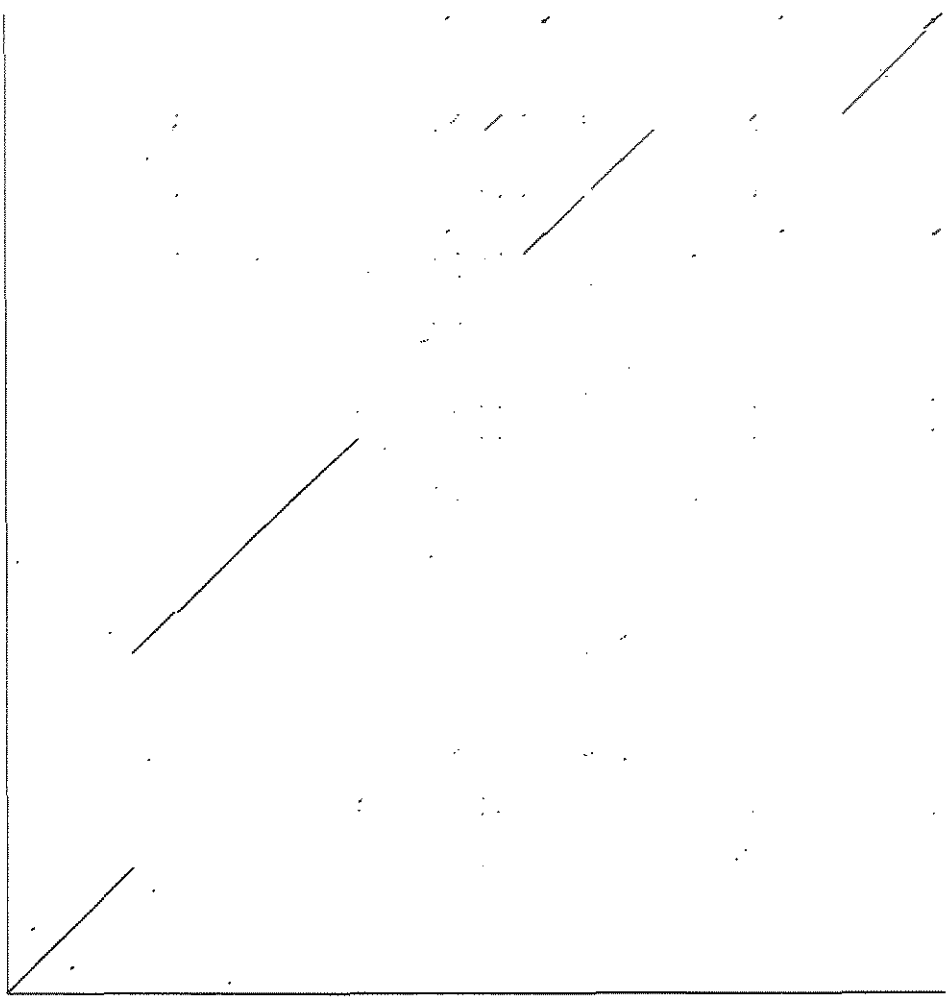


Figura A.5: *X. fastidiosa* CVC × *X. fastidiosa* PD

A.6 Tempos de Processamento

O programa **BioDiff** utilizado nos testes foi escrito na linguagem C e foi compilado com a opção de otimização máxima. A máquina dos testes é uma *Sun Enterprise* 450 com 4 processadores de 450 MHz, com memória RAM de 4 GB e sistema operacional *Unix*. Apesar de ter 4 processadores, o algoritmo só utiliza 1 deles. A tabela a seguir mostra os tempos de processamento real e de CPU e a relação entre o tempo de processamento real e o número de casamentos exatos relevantes (TR/CER) para cada par de espécies.

	Tempo real	Tempo de CPU	TR/CER
C. trachomatis e *C. pneumoniae*	11s18	11s00	0s183
H. pylori cepa J99 e *H. pylori* cepa 26695	8min46s22	8min32s72	0s031
L. innocua e *L. monocytogenes*	6min49s78	6min43s41	0s028
M. genitalium e *M. pneumoniae*	6s17	6s04	0s035
X. fastidiosa CVC e *X. fastidiosa* PD	11min28s18	10min52s45	0s036

Bibliografia

[Allison et al., 1999] Allison, L., Powell, D., and Dix, T. I. (1999). Compression and approximate matching. *The Computer Journal*, 42(1):1–10.

[Almeida Jr. and Setubal, 2000] Almeida Jr., N. F. and Setubal, J. C. (2000). A set of tools for detailed syntactic pairwise comparison of whole bacterial genomes. Manuscript.

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

[Altschul et al., 1997] Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J. H., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402.

[Andersson and Eriksson, 2000] Andersson, S. G. E. and Eriksson, K. (2000). Dynamics of gene order structures and genomic architectures. In Sankoff, D. and Nadeau, J. H., editors, *Comparative Genomics: empirical and analytical approaches to gene order dynamics, map alignment and the evolution of gene families*, volume 1 of *Computational Biology Series*, pages 267–280. Kluwer Academic Publishers.

[Archer et al., 1997] Archer, E. S., Yee, L. W., and Leung, F. C. (1997). Visual Genome Explorer (TM): a comparative visual interface to genome data. *BioTechniques*, 22(6):1164–1165.

[Chen et al., 2000] Chen, X., Kwong, S., and Li, M. (2000). A compression algorithm for DNA sequences and its applications in genome comparison. In *The Fourth Annual International Conference on Computational Molecular Biology - RECOMB 2000*, pages 107–107. ACM Press. Full version appeared in The Tenth Workshop on Genome Informatics GIW '99.

[Cormen et al., 1990] Cormen, T. H., Leiserson, C. E., and Rivest, R. L. (1990). *Introduction to Algorithms*. The MIT Electrical Engineering and Computer Science Series. MIT Press/McGraw Hill.

[De Rosa and Labedan, 1998] De Rosa, R. and Labedan, B. (1998). The evolutionary relationships between the two bacteria *Escherichia coli* and *Haemophilus influenzae* and their putative last common ancestor. *Molecular Biology and Evolution*, 15(1):17–27.

[Elgar et al., 1996] Elgar, G., Sandford, R., Aparicio, S., Macrae, A., Venkatesh, B., and Brenner, S. (1996). Small is beautiful: comparative genomics with the pufferfish (*Fugu rubripes*). *Trends in Genetics*, 12(4):145–150.

[Farach et al., 1995] Farach, M., Noordewier, M., Savari, S., Shepp, L., Wyner, A., and Ziv, J. (1995). On the entropy of DNA: algorithms and measurements based on memory and rapid convergence. In *Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms*, pages 48–57. ACM Inc.

[Frishman et al., 1999] Frishman, D., Mironov, A., and Gelfand, M. (1999). Starts of bacterial genes: estimating the reliability of computer predictions. *Gene*, 234:257–265.

[Futuyma, 1997] Futuyma, D. J. (1997). *Biologia Evolutiva*. Sociedade Brasileira de Genética/CNPq, Ribeirão Preto, SP, 2nd edition.

[Galatolo, 2000] Galatolo, S. (2000). A proof of the Beyer-Stein-Ulam relation between complexity and entropy. *Discrete Mathematics*, 223:367–372.

[Gatlin, 1972] Gatlin, L. L. (1972). *Information Theory and the Living System*. Columbia University Press.

[Gibas and Jambeck, 2001] Gibas, C. and Jambeck, P. (2001). *Developing Bioinformatics Computer Skills*. O'Reilly & Associates, UK.

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

[Glaser et al., 2001] Glaser, P., Frangeul, L., Buchrieser, C., Rusniok, C., Amend, A., Baquero, F., Berche, P., Bloecker, H., Brandt, P., Chakraborty, T., Charbit, A., Chetouani, F., Couvé, E., de Daruvar, A., Dehoux, P., Domann, E., Domínguez-Bernal, G., Duchaud, E., Durant, L., Dussurget, O., Entian, K. D., Fsihi, H., Garcia-Del Portillo, F., Garrido, P., Gautier, L., Goebel, W., Gómez-López, N., Hain, T., Hauf, J., Jackson, D., Jones, L. M., Kaerst, U., Kreft, J., Kuhn, M., Kunst, F., Kurapkat, G., Madueño,

E., Maitournam, A., Mata Vicente, J., Ng, E., Nedjari, H., Nordsiek, G., Novella, S., de Pablos, B., Pérez-Diaz, J. C., Purcell, R., Remmel, B., Rose, M., Schlueter, T., Simoes, N., Tierrez, A., Vásquez-Boland, J. A., Voss, H., Wehland, J., and Cossart, P. (2001). Comparative genomics of *Listeria* species. *Science*, 294:849–852.

[Gusev et al., 1993] Gusev, V. D., Kulichkov, V. A., and Chupakhina, O. M. (1993). The Lempel-Ziv complexity and local structure analysis of genomes. *BioSystems*, 30:183–200.

[Gusfield, 1997] Gusfield, D. (1997). *Algorithms on Strings, Trees, and Sequences: computer science and computational biology.* Cambridge University Press.

[Hall and Dowling, 1980] Hall, P. A. V. and Dowling, G. R. (1980). Approximate string matching. *ACM Computing Surveys*, 12(4):381–402.

[Lake and Moore, 1998] Lake, J. and Moore, J. (1998). Phylogenetic analysis and comparative genomics. *Trends Guide to Bioinformatics*, pages 22–23.

[Landau et al., 2001] Landau, G. M., Schmidt, J. P., and Sokol, D. (2001). An algorithm for approximate tandem repeats. *Journal of Computational Biology*, 8(1):1–18.

[Larsson and Sadakane, 1999] Larsson, N. J. and Sadakane, K. (1999). Faster suffix sorting. Technical Report LU-CS-TR:99-214, LUNDFD6/(NFCS-3140)/1–20/(1999), Department of Computer Science, Lund University, Sweden.

[Leung et al., 1991] Leung, M.-Y., Blaisdell, B. E., Burge, C., and Karlin, S. (1991). An efficient algorithm for identifying matches with errors in multiple long molecular sequences. *Journal of Molecular Biology*, 221(3):1367–1378.

[Levenshtein, 1966] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 6:707–710.

[Lewin, 2000] Lewin, B. (2000). *Genes VII.* Oxford University Press, New York.

[Li et al., 2001] Li, M., Badger, J. H., Chen, X., Kwong, S., Kearney, P., and Zhang, H. (2001). An information-based sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics*, 17(2):149–154.

[Li and Vitányi, 1997] Li, M. and Vitányi, P. (1997). *An Introduction to Kolmogorov Complexity and Its Applications.* Springer-Verlag, 2nd edition.

[Loewenstern et al., 1995] Loewenstern, D., Hirsh, H., Yianilos, P., and Noordewier, M. (1995). DNA sequence classification using compression-based induction. Technical Report DIMACS Technical Report 95-04, DIMACS, USA.

[Loewenstern and Yianilos, 1999] Loewenstern, D. and Yianilos, P. N. (1999). Significantly lower entropy estimates for natural DNA sequences. *Journal of Computational Biology*, 6(1):125–142.

[MacKay, 1995] MacKay, D. (1995). A short course in Information Theory. http://www.inference.phy.cam.ac.uk/mackay/info-theory/course.html. Livro "*Information Theory, Inference & Learning Algorithms*" publicado em janeiro de 2002.

[MacWilliams and Sloane, 1977] MacWilliams, F. J. and Sloane, N. J. A. (1977). *The theory of error-correcting codes.* North-Holland, Amsterdam.

[Manber and Myers, 1993] Manber, U. and Myers, G. (1993). Suffix arrays: a new method for on-line string searches. *SIAM Journal on Computing*, 22(5):935–948.

[Meidanis, 1992] Meidanis, J. (1992). *Algorithms for Problems in Computational Genetics.* PhD thesis, University of Wisconsin – Madison.

[Milosavljević, 1999] Milosavljević, A. (1999). *Pattern Discovery in Biomolecular Data: tools, techniques and applications*, chapter Discovering Patterns in DNA Sequences by the Algorithmic Significance Method. Oxford University Press.

[Mironov and Alexandrov, 1988] Mironov, A. A. and Alexandrov, N. N. (1988). Statistical method for rapid homology search. *Nucleic Acids Research*, 16:5169–5174.

[Myers, 1986] Myers, E. W. (1986). An $\mathcal{O}(ND)$ difference algorithm and its variations. *Algorithmica*, 1:251–266.

[Oommen and Kashyap, 1998] Oommen, B. J. and Kashyap, R. L. (1998). A formal theory for optimal and information theoretic syntatic pattern recognition. *Pattern Recognition*, 31(9):1159–1177.

[Pevzner, 2000] Pevzner, P. A. (2000). *Computational Molecular Biology: an algorithmic approach.* MIT Press Series on Computational Molecular Biology. The MIT Press.

[Rivals et al., 1997] Rivals, E., Dauchet, M., Delahaye, J.-P., and Delgrange, O. (1997). Fast discerning repeats in DNA sequences with a compression algorithm. In *Proceedings of 8th Workshop on Genome Informatics (GIW 97)*, pages 215–226. Universal Academy Press Inc.

[Sagot, 1998] Sagot, M. F. (1998). Spelling approximate repeated or common motifs using a suffix tree. In *Latin American Theoretical Informatics - LATIN 1998*, number 1380 in Lecture Notes in Computer Science, pages 374–390.

[Salamon and Konopka, 1992] Salamon, P. and Konopka, A. (1992). A maximum entropy principle for the distribution of local complexity in naturally occurring nucleotide sequences. *Computers and Chemistry*, 16(2):117–124.

[Sankoff, 2000] Sankoff, D. (2000). Genome rearrangements and comparative mapping. http://ismb00.sdsc.edu/tutorials/sanakoff.html. ISMB Tutorial.

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

[Shannon, 1948] Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423; 623–656.

[Ukkonen, 1995] Ukkonen, E. (1995). On-line construction of suffix trees. *Algorithmica*, 14:249–260.

[Varré et al., 1999] Varré, J.-S., Delahaye, J.-P., and Rivals, E. (1999). Transformation distances: a family of dissimilarity measures based on movements of segments. *Bioinformatics*, 15(3):194–202.

[Wan and Wootton, 2000] Wan, H. and Wootton, J. C. (2000). A global compositional complexity measure for biological sequences: AT-rich and GC-rich genomes encode less complex proteins. *Computers and Chemistry*, 24:71–94.

[West, 1996] West, D. B. (1996). *Introduction to Graph Theory*. Prentice-Hall.

[Witten et al., 1987] Witten, I. H., Neal, R. M., and Cleary, J. G. (1987). Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540.

[Zhang and Madden, 1997] Zhang, J. H. and Madden, T. L. (1997). PowerBLAST: A new network BLAST application for interactive or automated sequence analysis and annotation. *Genome Research*, 7(6):649–656.