Abordagens para o Problema da Seleção das Subcadeias Específicas utilizando a Distância de Hamming

Lucas Barbosa Rocha

Abordagens para o Problema da Seleção das Subcadeias Específicas utilizando a Distância de Hamming

Lucas Barbosa Rocha

Orientador: *Prof. Dr. Francisco Eloi Soares de Araujo*
Coorientador: *Prof. Dr. Said Sadique Adi*

Dissertação entregue à Faculdade de Computação da Universidade Federal de Mato Grosso do Sul - FACOM-UFMS como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

UFMS - Campo Grande
Março/2020

Agradecimentos

Agradeço aos meus orientadores Eloi Araujo e Said Sadique, pela paciência, motivação, compreensão e por me ajudarem grandemente no desenvolvimento deste trabalho.

Agradeço à minha família, aos meus pais José Aluisio Carinhanha Rocha e Zilma Gonçalves Barbosa Rocha e a minha irmã Luana Barbosa Rocha por me apoiarem em todos os momentos e compreenderem todas as vezes que precisei ficar ausente e focar nos estudos.

Agradeço aos amigos que ganhei: Angelo Maggioni, Carlos Córdoba, Denis Novaes, Edison Borghezan, Franklin Messas, Gabriel Escobar, Hernanes Almeida, Hudson Fujikawa, Jean Carlo e Valter Ferlete.

Agradeço à minha amiga Suzane Eberhart por me ajudar nos momentos que precisei (principalmente com o inglês) e agradeço às minhas amigas Jaqueline da Silva e Neiliane Capelari por estarem sempre presente.

Agradeço às professoras Edna Ayako e Graziela Araujo e aos professores Marco Aurélio, Renan Marks e Renato Ishii pelos conselhos, ajuda e motivação.

Agradeço a Deus pela oportunidade de estar vivo e ter essas pessoas maravilhosas ao meu lado.

Reservo essa linha para agradecer às minhas cachorras Lola e Mel.

Enfim, agradeço a CAPES por me disponibilizar uma bolsa de estudos durante o mestrado, agradeço a Universidade Federal de Mato Grosso do Sul – UFMS e Universidade Federal do Rio Grande do Norte – UFRN por me disponibilizarem os ambientes necessários para realizar meus estudos. Agradeço aos funcionários e professores da FACOM que contribuíram de forma direta ou indireta durante o meu trabalho de mestrado.

Abstract

An important problem in Computational Biology is to determine genetic markers that are substrings of one sequence set that do not occur in sequences of other sequence sets. The problem of determining markers is important for Biology because it allows to obtain relevant information about the sequences, for example, we can find specific regions for the designing primer in PCR context, or we can analyze metagenomas from different regions for the purpose to find markers to determine which species are present in a given specific metagenome or a given region. More specific with the metagenomes, we can analyze, for example, two metagenomes of two tissues, one healthy and the other with cancer, with the purpose of finding which DNA sequences are present in the cancer sample and are not present in the healthy sample, and vice versa. Genetic markers can be addressed by the Specific Substring Problem – SSP which consists of finding all minimal substrings in a given sequence set with at least k differences with each substring in another sequence set. In this work, we propose to do a detailed study for SSP. We implement and evaluate exact algorithms that solve SSP. We also propose heuristices for SSP and we perform comparative tests between the exact algorithms and our heuristcs. Finally, we propose, implement and evaluate a first version of the parallelism approach using MPI for our heuristics.

Keywords. Hamming Distance, Specific Substring Problem and Markers

Resumo

Um problema importante na Biologia Computacional consiste em determinar marcadores, que são subcadeias de um conjunto de sequências que não ocorrem em sequências de outros conjuntos. O problema de determinar marcadores é importante para a Biologia porque permite obter informações relevantes sobre as sequências, por exemplo, podemos encontrar regiões específicas para o design do *primer* no uso de PCR, ou podemos analisar metagenomas de diferentes locais com o propósito de encontrar marcadores para determinar quais são os indivíduos que estão presentes unicamente em um determinado metagenoma ou uma determinada região/local. Sendo mais específico com a aplicação em metagenomas, podemos analisar, por exemplo, o metagenoma de dois tecidos, um saudável e outro com câncer, com o propósito de encontrar quais sequências de DNA estão presentes na amostra com câncer e não estão presentes na amostra saudável, e vice-versa. Marcadores genéticos podem ser abordados pelo Problema da Seleção das Subcadeias Específicas – SSP que consiste em, dados dois conjuntos de sequências, encontrar todas as subcadeias minimais em um conjunto com pelo menos k diferenças em relação a todas as subcadeias do outro conjunto. Dito isso, propomos neste trabalho o estudo detalhado do SSP. Vamos implementar e avaliar algoritmos exatos que resolvam o SSP. Propomos também heurísticas para o SSP e vamos realizar testes comparativos entre os algoritmos exatos e as heurísticas. E por fim, nós propomos, implementamos e avaliamos uma primeira versão da abordagem de paralelismo usando MPI para as nossas heurísticas.

Palavras-chave. Distância de Hamming, Problema da Seleção das Subcadeias Específicas e Marcadores

Conteúdo

Sumário x
Lista de Figuras xiv
Lista de Tabelas xix
Lista de Abreviaturas xxi
Lista de Algoritmos xxiii

1 Introdução **1**
1.1 Contribuições 4
1.2 Organização da dissertação 4

2 Conceitos Preliminares **5**
2.1 Sequências 5
2.2 Distância de Hamming 6
2.3 Paralelismo 6
2.4 Problema da Seleção das Subcadeias Específicas 8
2.4.1 Definição 8
2.4.2 Algoritmos para o SSP 9
2.4.3 Extensão do problema simplificado 12
2.5 Probabilidade 12
2.6 Equação geral da reta 14
2.7 Biologia 15
2.7.1 Ácidos nucleicos 15
2.7.2 Marcadores genéticos 17
2.7.3 Metagenômica 17

3 Heurísticas **19**
3.1 Ideia Geral 19
3.2 Heurística 1 – H1 21
3.3 Heurística 2 – H2 23
3.4 Heurísticas e o SSP 25

3.5 Abordagem Paralela 26
3.6 Implementações 27

4 Análises, Experimentos e Resultados **29**
4.1 Confiabilidade do *QE* e *QV* 30
4.2 Algoritmos Exatos 33
4.2.1 Sequências Artificiais 33
4.2.2 Sequências Reais 38
4.3 Heurísticas 41
4.3.1 Acurácia dos experimentos 41
4.3.2 Tara 42
4.3.3 Bactérias 43
4.3.4 Bactérias da mesma família 45
4.3.5 Paralelismo 48

5 Discussão, Perspectivas e Conclusões **51**

Referências **57**

Lista de Figuras

2.1 Ambiente de computação paralela. O modelo CGM é utilizado neste ambiente de computação paralela que consiste de um nó mestre e de p nós interligados por uma rede dedicada. Cada nó corresponde a um computador com seu processador e memória local. O nó mestre será considerado o processador P_0, e os demais nós, processadores entre $P_1,\ldots,P_p$. No super-passo de comunicação, o nó mestre irá comunicar-se com os demais nós enviando os dados necessários. No super-passo de computação, os nós $P_1,\ldots,P_p$ vão fazer a computação. E esses super-passos serão repetidos até o fim da execução. 7

2.2 No começo de cada iteração (i,j) do laço das linhas 4–17 do **KHD**, para cada $h=1,\ldots,m$, $s[h,h+r[h]-1]$ (parte rosa na figura) é a subcadeia minimal de s com k diferenças de cada subcadeia de $t[h',n]$ (parte azul), onde $h'=h-i+j$. Embora na figura a representação seja para $i\leq h$, esta propriedade se mantém mesmo se $h<i$. Se $h<i$, então $s[h,h+r[h]-1]$ é a subcadeia minimal de s com k diferenças de cada subcadeia em t. 12

2.3 Estrutura biológica de um DNA, figura retirada do livro da Maria Ribeiro [1]. Os ácidos nucleicos são formados por nucleotídeos, que por sua vez são formados por três elementos: um radical "fosfato", uma desoxirribose (um açúcar do grupo das pentoses) e a base nitrogenada dividida em dois grupos: Purinas (Adenina e Guanina) e as Pirimidinas (Citosina, Timina e Uracila (essa última é para o RNA)). Uma cadeia de DNA é uma sequência de nucleotídeos dessa molécula unidos quimicamente pelos carbonos 3' e 5' da pentose, através do fosfato. A estrutura do DNA consiste, basicamente, de duas cadeias de nucleotídeos dispostas de forma antiparalelas, dispostas uma sobre a outra. A união das duas cadeias é feita por pontes de hidrogênio, que ligam pares de bases específicas do DNA: AT, CG, TA e GC. No RNA a base Timina (T) é substituída pela base Uracila (U). Dentre as informações nesta imagem, a mais relevante para o nosso trabalho é a forma como o DNA é representado: uma cadeia de nucleotídeos, ou seja, computacionalmente é uma concatenação de caracteres. 16

2.4 Comparação entre genoma e metagenoma. Enquanto que um genoma contém a informação genética de um organismo, o metagenoma contém a informação genética de uma comunidade de organismos. 17

3.1 Exemplo de retas criadas para o vetor $r = [4,?,?,3,?,?,6,?,?,4,?,?,2,?]$. Os pontos laranjas representam os pares $(i, r[i])$ do vetor r, onde cada posição exata i é a abscissa x e o seu respectivo valor $r[i]$ é a ordenada y. Os pontos azuis representam os pares $(i, r[i])$ onde i é uma posição deduzida e $r[i]$ o valor inferido. 23

3.2 Execução para cada A_i de tamanho $\frac{|A|}{p}$, com $i \in \{1 \dots p\}$, e B. 26

4.1 Gráfico para a Tabela 4.2, os pontos laranjas representam os valores do intervalo: $\lfloor QE - QV \rfloor : \lceil QE + QV \rceil = [32 : 40]$. A abscissa **comprimento** representa o comprimento do prefixo calculado com o algoritmo exato e a ordenada **quantidade** representa a quantidade de comprimentos calculados. Para esse teste, temos 971.897 comprimentos entre $[32 : 40]$. 31

4.2 Gráfico para os valores de $k = 30$ da Tabela 4.3. 35

4.3 Gráfico para os valores de $k = 300$ da Tabela 4.3. 35

4.4 Gráfico para os valores de $k = 600$ da Tabela 4.3. 35

4.5 Gráfico para os valores de $k = 30$ da Tabela 4.4. 35

4.6 Gráfico para os valores de $k = 300$ da Tabela 4.4. 35

4.7 Gráfico para os valores de $k = 600$ da Tabela 4.4. 36

4.8 Gráfico para os valores da Tabela 4.5. Um único gráfico é capaz de representar a Tabela 4.5 porque o tempo de execução dos algoritmos são os mesmos independente do valor de k. 37
4.9 Gráfico para os valores da sequência *hsarhgdig* para $k = 30$ da Tabela 4.6. 39
4.10 Gráfico para os valores da sequência *hsarhgdig* para $k = 300$ da Tabela 4.6. 39
4.11 Gráfico para os valores da sequência *hsarhgdig* para $k = 600$ da Tabela 4.6. 40
4.12 Gráfico para os valores da sequência *hsankr10* para $k = 30$ da Tabela 4.6. 40
4.13 Gráfico para os valores da sequência *hsankr10* para $k = 300$ da Tabela 4.6. 40
4.14 Gráfico para os valores da sequência *hsankr10* para $k = 600$ da Tabela 4.6. 40
4.15 Gráfico para os valores da sequência *hsaff4* para $k = 30$ da Tabela 4.6. 40
4.16 Gráfico para os valores da sequência *hsaff4* para $k = 300$ da Tabela 4.6. 40
4.17 Gráfico para os valores da sequência *hsaff4* para $k = 600$ da Tabela 4.6. 41
4.18 Gráfico para os valores da Tabela 4.8 para a quantidade de 100 sequências considerando o teste com o KHD. 44
4.19 Gráfico para a Tabela 4.9 para o teste com 100 sequências, representando a quantidade para cada comprimento calculado/inferido pelos programas KHD, H1 e H2. 44
4.20 Gráfico para os valores da Tabela 4.10 para a quantidade de 100 sequências para o primeiro experimento considerando o teste com o KHD. 47
4.21 Gráfico para a Tabela 4.11 para o teste com 100 sequências, representando as quantidades para cada comprimento calculado/inferido pelos programas KHD, H1 e H2 para o primeiro experimento. . . 47
4.22 Gráfico para os valores da Tabela 4.10 para a quantidade de 100 sequências para o segundo experimento considerando o teste com o KHD. 47
4.23 Gráfico para a Tabela 4.11 para o teste com 100 sequências, representando as quantidades para cada comprimento calculado/inferido pelos programas KHD, H1 e H2 para o segundo experimento. . . 47

4.24 Gráfico para os valores da Tabela 4.10 para a quantidade de 100 sequências para o terceiro experimento considerando o teste com o KHD. 47
4.25 Gráfico para a Tabela 4.11 para o teste com 100 sequências, representando as quantidades para cada comprimento calculado/inferido pelos programas KHD, H1 e H2 para o terceiro experimento. . . . 47
4.26 Gráfico para os valores da Tabela 4.13 para o primeiro caso, para 10 e 100 sequências. 50
4.27 Gráfico para os valores da Tabela 4.13 para o segundo caso, para 10 e 100 sequências. 50
4.28 Gráfico para os valores da Tabela 4.13 para o terceiro caso, para 10 e 100 sequências. 50
4.29 Gráfico para os valores da Tabela 4.13 para o quarto caso, para 10 e 100 sequências. 50

5.1 Pipeline com *scripts* do CD-HIT-EST para reduzir o número de sequências nos conjuntos. Os passos são os seguintes: **CD-HIT-EST-2D** – clusterizar (agrupar) as sequências similares (utilizamos 95% de similaridade) gerando diversos arquivos com diversos *clusters* (se existirem). **Merge** – reunir todos os *clusters* em um único arquivo. **Clstr_renumber** – reorganizar a numeração dos *clusters*. **MultiSeq** – gerar arquivos Fasta com as sequências contidas em cada *cluster*. **Merge** – reunir todas as sequências Fasta em um único arquivo. **CD-hit-dup** – retirar sequências duplicadas. Para realizar o *merge* podemos utilizar o comando **cat** do linux, sendo esse, o único passo sem *script* do CD-HIT-EST. . 54

Lista de Tabelas

3.1 Exemplo do vetor P com a probabilidade dos comprimentos possuírem $k = 2$ diferenças para uma sequência t de comprimento $n = 14$. Para o comprimento 1 é impossível de acontecer por causa do valor de k, por isso a probabilidade é 0. 22

3.2 Exemplo do vetor C com os pares de pontos cartesiano (x,y) para vetor o $r = [4,?,?,3,?,?,6,?,?,4,?,?,2,?]$, onde as abscissas são as posições exatas i do vetor r e as ordenadas são os respectivos $r[i]$. O índice de cada par (x,y) é representado por c. 23

3.3 Resumo das complexidades. Nesta tabela, para as versões exatas e heurísticas apresentamos a complexidade considerando duas sequências s e t, para $|s| = |t| = n$. Sendo P o custo máximo para construir a tabela de probabilidade e $\mathcal{T}$ o custo máximo para um algoritmo exato calcular o comprimento para uma determinada posição exata. Para a versão paralela, considere A e B dois conjuntos de sequências de comprimento n. A versão paralela possui em cada processador a complexidade de espaço $O(n \cdot \frac{|A|}{p} + n \cdot |B|)$. . 28

4.1 Testes variando o comprimento das sequências onde cada teste contém 1 milhão de pares de sequências. Para cada milhar de testes, a segunda coluna contém o valor de k, a terceira e quarta coluna possuem os valores QE e QV, respectivamente, que são valores teóricos para o valor esperado e a sua variância referente ao comprimento da sequência analisada. As colunas cinco e seis, possuem os valores QEE e QVE que são valores esperados e variância que baseado nos comprimentos obtidos nos testes. A colunas sete e oito possuem o menor e maior comprimento dos prefixos calculado com o algoritmo exato. A última coluna % representa a quantidade de comprimentos dos prefixos que estão entre $\left[\lfloor QE - QV \rfloor : \lceil QE + QV \rceil\right]$. Por exemplo, para o comprimento 100 e $k = 20$ na primeira linha da tabela, o menor comprimento do prefixo calculado com o algoritmo exato em 1-milhão de pares de sequências foi de 29 e o maior foi de 57, de todas as sequências, 97,19% das sequências possuem prefixo com o comprimento dentro do intervalo $[32 : 40]$. 32

4.2 Quantidade total de comprimentos dos prefixos calculados com o algoritmo exato, para 1 milhão de pares de sequências s e t, onde $|s| = |t| = 100$. Cada prefixo x de s tem pelo menos 20 diferenças com todas as subcadeias y em t, onde $|x| = |y|$. Na primeira coluna contém o comprimento de um prefixo e a segunda coluna contém a quantidade desse comprimento. Por exemplo, o algoritmo exato calculou um prefixo de comprimento 32 em 59.281 sequências (linha 2 na tabela). Essa tabela detalha a primeira linha da Tabela 4.1. 32

4.3 Testes com s e t geradas aleatoriamente, para $k = 30,\ 300$ e 600. A primeira coluna contém o comprimento das sequências s e t e as seis colunas a seguir contém o tempo de execução do FB e do KHD no formato hora:minuto:segundo. 34

4.4 Testes com s e t completamente diferentes, para $k = 30,\ 300$ e 600. A primeira coluna contém o comprimento das sequências s e t e as seis colunas a seguir contém o tempo de execução do FB e do KHD no formato hora:minuto:segundo. 36

4.5 Testes com s e t iguais, para $k = 30,\ 300$ e 600. A primeira coluna contém o comprimento das sequências s e t e as seis colunas a seguir contém o tempo de execução do FB e do KHD no formato hora:minuto:segundo. A tabela contém tempos idênticos para qualquer valor de k e isso se aplica pelo fato das sequências serem totalmente iguais, ou seja, tanto o FB quanto o KHD vão sempre fazer a mesma verificação independente do k escolhido. 37

4.6 Testes com s e t reais (*Homo Sapiens*), para $k = 30,\ 300$ e 600. A primeira e a segunda coluna contém o nome e o comprimento da sequência s. A terceira e a quarta coluna contém o nome e o comprimento da sequência t. As seis colunas a seguir contém o tempo de execução do FB e o KHD no formato hora:minuto:segundo. . . 39

4.7 Exemplo de avaliação da acurácia dos experimentos considerando duas sequências s no conjunto A, com $|s| = 14$. Os valores nesta tabela são dados ilustrativos. A primeira coluna contém a identificação da sequência. A segunda e terceira colunas temos rótulos para as próximas colunas. Entre as colunas quatro e dezessete temos quatorze posições, para a linha **KHD** são os comprimentos exatos do vetor r, para a linha **H2** são os comprimentos deduzidos do vetor r', para a linha **diferença**, são as diferenças (valor absoluto) entre os comprimentos do KHD e H2. Para a coluna μ, temos a média das diferenças. Para a linha Φ, temos a média das médias das diferenças. Para H2 considerando essas duas sequências, sua acurácia é de $0,32$. 42

4.8 Teste para sequências dos metagenomas da expedição Tara. A primeira coluna contém a quantidade de sequências no conjunto A (O conjunto B contém 1-milhão de sequências). A segunda coluna contém o tempo de execução do KHD. A terceira e quarta coluna contém o tempo de execução das heurísticas utilizando o algoritmo FB para as posições exatas, e a quinta e sexta coluna, o tempo de execução das heurísticas utilizando o algoritmo KHD. O formato do tempo é hora:minuto:segundo. Os testes consideram $k = 20$. 44

4.9 Acurácia para os resultados da Tabela 4.8. A primeira coluna contém a quantidade de sequências do conjunto A. As últimas dez colunas contém o valor Φ (média das médias das diferenças) entre as respostas do KHD e das heurísticas H1 e H2 (fonte em negrito) e a porcentagem das respostas das heurísticas que diferem em algumas unidade do exato (fonte sem negrito). Por exemplo, para 100 sequências, H2 está errando na média em $1,6$, então muitos comprimentos estão próximos do algoritmo exato, podemos ver isso na colunas $1-4$ onde $60,6\%$ dos comprimentos diferem em 1, 2, 3 ou 4 unidades. 44

4.10 Teste para sequências de bactérias obtidas do NCBI. A primeira coluna contém a quantidade de sequências no conjunto A (O conjunto B contém 1-milhão de sequências). A segunda coluna contém o tempo de execução do KHD. A terceira e quarta coluna contém o tempo de execução das heurísticas utilizando o algoritmo FB para as posições exatas, e a quinta e sexta coluna, o tempo de execução das heurísticas utilizando o algoritmo KHD. Os testes são divididos em três casos. O primeiro caso contém sequências de comprimento entre 100 e 200, a segundo, sequências com comprimento entre 800 e 1000. O terceiro caso, mantém 1, 10 e 100 sequências em A, mas o conjunto B tem 10.000 sequências, ambos os conjuntos contém sequências com o comprimento entre 8000 e 10.000. O formato do tempo é hora:minuto:segundo. Os testes consideram $k = 20$. 46

4.11 Acurácia para os resultados da Tabela 4.10. A primeira coluna contém a quantidade de sequências do conjunto A. As últimas dez colunas contém o valor Φ (média das médias das diferenças) entre as respostas do KHD e das heurísticas H1 e H2 (fonte em negrito) e a porcentagem das respostas das heurísticas que diferem em algumas unidade do exato (fonte sem negrito). 46

4.12 Acurácia para os testes com bactérias da mesma família. A primeira coluna contém a quantidade de sequências do conjunto A. As últimas dez colunas contém o valor Φ (média das médias das diferenças) entre as respostas do KHD e das heurísticas H1 e H2 (fonte em negrito) e a porcentagem das respostas das heurísticas que diferem em algumas unidade do exato (fonte sem negrito). . . 48

4.13 Resumo do tempo de execução da heurística H2-P para as sequências das seções 4.3.2 e 4.3.3 nas quatro primeiras tabelas. A primeira coluna contém a quantidade de sequências do conjunto A (o conjunto B contém 1-milhão de sequências). A segunda coluna o tempo de execução da Heurística 2. As próximas quatro colunas contém o tempo de execução da Heurística 2 na sua versão paralela. O formato do tempo é dia-hora:minuto e o $K = 20$. A última tabela contém o *Speedup*. A primeira coluna contém a quantidade de sequências do conjunto A (o conjunto B contém 1-milhão de sequências). A segunda coluna não contém nenhum valor. As quatro próximas colunas contém o *speedup* médio para cada coluna do processador. 49

Lista de Abreviaturas

FB Força Bruta

GCM Coarse Granularity Multicomputer

H1 Heurística 1

H2 Heurística 2

H2-P Heurística 2 Paralela

KHD k–Hamming Distance

MPI Message Passing Interface

QE Quasi-Valor Esperado

QV Quasi-Variância

SSP$_s$ Simplified Specific Substring Problem

SSP Specific Substring Problem

Lista de Algoritmos

1 FORÇA BRUTA – FB 10
2 *k-Hamming Distance* – KHD 11
1 H1 22
2 H2 24

CAPÍTULO

1

Introdução

Um algoritmo eficiente é aquele cuja complexidade de tempo satisfaz algum critério[1]. Um problema é tratável se admite um algoritmo eficiente e caso contrário ele é intratável. Em geral, a Teoria da Computação associa o conceito de tratabilidade a algoritmos polinomiais pois na prática, muitos algoritmos que são processados em tempo polinomial podem ser computados em um tempo razoável. Entretanto, se a quantidade de dados na entrada é muito grande como é o caso em muitas aplicações biológicas atuais, usar soluções que gastam tempo polinomial eventualmente não é o suficiente; em alguns casos a base de dados é tão grande que quando o grau do referido polinômio é maior ou igual a dois, a computação pode ser impraticável. Nas últimas quatro décadas, por exemplo, com o avanço da tecnologia de sequenciamento de DNA, o número de genomas sequenciados cresceu notavelmente [2, 3]. Essas bases de dados podem ter tamanhos muito grandes e as aplicações que envolvem buscas nessas bases podem usar terabytes de informações o que implica que algoritmos quadráticos poderiam levar milhares de anos para serem processados nos mais velozes computadores atuais. Na ausência de algoritmos mais rápidos que forneçam respostas exatas, lançamos mão de métodos que fornecem respostas boas mas não exatas tais como heurísticas que, embora nem mesmo garantam uma relação entre boas resposta e respostas exatas, são muito rápidas e na prática fornecem respostas práticas muito boas na maioria das vezes.

A Biologia Computacional é a área que aplica a computação para a solução de problemas da Biologia e mais especificamente com muita frequência da

[1]Na verdade esse conceito é também usando para a complexidade de espaço gasto, mas não nos preocupamos muito com esse conceito nesse texto.

Biologia Molecular que é a área da Biologia que estuda moléculas de ácido nucleico que por sua vez definem o genoma de um organismo. No contexto de genômica, que é o termo atribuído a todo e qualquer estudo do genoma, um problema importante em Biologia Computacional é determinar marcadores genéticos. Entende-se que **marcadores genéticos** [4, 5, 6] são características ímpares que podem ser utilizadas para definir um indivíduo. Neste trabalho entendemos por marcador genético como sendo uma sequência de símbolos que representa uma região de DNA em particular, ou seja, são subcadeias (regiões) diferentes de qualquer outra subcadeia de outras sequências de DNA; em outras palavras são subcadeias específicas. Desta forma, dados dois ou mais conjuntos de sequências, um marcador genético pode ser descrito como uma subcadeia específica, que são subcadeias de um conjunto de sequências que não ocorrem em sequências dos outros conjuntos, ou seja, subcadeias encontradas apenas em um único conjunto.

A determinação de marcadores pode ajudar na identificação de quais espécimes estão contidos dentro de um determinado conjunto de sequências. Por exemplo, a fundação Tara *Ocean*, realizou uma expedição, conhecida como *Tara expedition*, que foi uma expedição de pesquisa oceânica. Eles realizaram um trabalho de sequenciamento de metagenoma de mais de 35.000 amostras diferentes de 210 regiões diferentes do mundo, resultando em um total de 7,2TB de dados genômicos [7, 8]. No nosso contexto, analisar estas amostras poderia nos revelar quais são os indivíduos que estão presentes em determinadas regiões e não estão em outras. Uma outra aplicação consiste de através da análise do metagenoma de dois tecidos, sendo um saudável e o outro com câncer, determinar quais sequências de DNA estão presentes na amostra com câncer e não estão presentes na amostra saudável, para então identificar padrões específicos em metagenomas que poderiam estar presentes em um determinado tecido doente para o diagnóstico precoce de câncer [9, 10]. Por fim, biólogos interessados em detectar doenças podem utilizar a técnica de PCR (do inglês, *Polymerase Chain Reaction*) [11] para amplificar determinadas regiões do DNA do agente causador da doença, possibilitando sua identificação. Para essa técnica, as regiões do DNA precisam ser específicas, então identificar marcadores (subcadeias específicas) é relevante para a conclusão do PCR [12].

Neste trabalho, o problema de encontrar subcadeias específicas é modelado como segue. Dadas duas sequências de mesmo comprimento, elas possuem k diferenças se a distância de Hamming [13] entre elas é k. Considerando um inteiro $k > 0$ e dois conjuntos de sequências A e B, o objetivo é encontrar todas as subcadeias específicas x em A com pelo menos k diferenças com relação a qualquer subcadeia y em B. Nós chamamos esse problema de o PROBLEMA

DA SELEÇÃO DAS SUBCADEIAS ESPECÍFICAS – SSP. Maaß [14] descreveu uma solução utilizando a distância de Hamming para um problema similar denominado *Inexact Characteristic String Problem*. Dados dois conjuntos A e $B \subsetneq A$, e um inteiro $k > 0$, o objetivo é encontrar a menor subcadeia x em B com pelo menos k diferenças em relação a qualquer subcadeia y em $A \backslash B$. Gusfield [15] descreveu uma solução utilizando a distância de edição que resolve um outro problema similar denominado de *k-Difference Primer Problem*. Para esse último problema, sejam duas sequências s e t, e um inteiro $k > 0$, o objetivo é encontrar para cada índice i de s, a menor subcadeia y de s que se inicia em i com distância de edição de pelo menos k com relação a qualquer subcadeia em t.

Identificar subcadeias específicas requer analisar sequências, muitas das vezes envolvendo a análise de inúmeras sequências. Devido à grande quantidade de sequências que podem estar envolvidas e/ou aos extensos comprimentos que uma sequência pode ter, a identificação das subcadeias específicas pode levar muito tempo para ser feita, tornando a tarefa complicada de ser realizada. Quando temos conjuntos com milhares de símbolos, algoritmos polinomiais permanecem ineficientes devido ao número de comparações. Por exemplo, para o algoritmo quadrático do Maaß [14], tome dois conjuntos de sequências com 1TB de símbolos, o algoritmo realiza 10^{12}x10^{12} comparações. Considerando que uma máquina é capaz de realizar menos de 10^8 comparações por segundo, o algoritmo levaria pelo menos 10^{16} segundos para finalizar, ou seja, alguns milênios. Portanto, o tempo gasto pelos algoritmos mais rápidos propostos até o momento para o SSP é impraticável nesses casos.

Dado o que foi exposto anteriormente, o propósito principal deste trabalho é o estudo detalhado do Problema da Seleção das Subcadeias Específicas – SSP, implementar algoritmos exatos que resolvem o SSP, propor e implementar heurísticas, propor e implementar um algoritmo paralelo para as heurísticas. Além das implementações, realizar testes comparativos entre os algoritmos exatos com dados artificiais e reais, testes comparativos entre os algoritmos exatos e as heurísticas com dados reais, e por fim, realizar testes comparativos entre o tempo das heurísticas e o algoritmo paralelo. Faz parte também dos objetivos deste trabalho um estudo teórico sobre a probabilidade, dadas duas sequências x e t com $|x| = l$, de x ter pelo menos k diferenças em relação a qualquer subcadeia de comprimento l em t. Para este estudo, é necessário discutir se x tem pelo menos k diferenças em relação a qualquer subcadeia de comprimento l em t quando as sequências x e t são geradas aleatoriamente.

1.1 Contribuições

Dentre as principais contribuições deste trabalho, listamos as principais como sendo:

- a implementação de dois algoritmos exatos para o Problema da Seleção das Subcadeias Específicas – SSP, implementação de duas heurísticas propostas para o SSP e a implementação de uma versão paralela para as heurísticas. Os códigos estão disponíveis no GitHub: https://github.com/LucasBarbosaRocha/SSP;
- apresentação oral do trabalho intitulado de Soluções Paralelas para o Problema das Subcadeias Específicas, na ação de extensão **V Workshop de Computação Paralela e Distribuída 2018**, realizada pela Faculdade de Computação da Universidade Federal de Mato Grosso do Sul;
- a submissão e o aceite de um artigo para a publicação na décima primeira edição do evento internacional *Brazilian Symposium on Bioinformatics* – BSB [16], intitulado *Specific Substring Problem: an application in bioinformatics*, 2018;
- a submissão e o aceite de um artigo para a publicação na décima nona edição do evento internacional *International Conference on Bioinformatics and Bioengineering* – BIBE [17], intitulado *Heuristics for the Specific Substring Problem with Hamming Distance*, 2019.

1.2 Organização da dissertação

Esta dissertação está dividida em quatro outros capítulos, além desta introdução, organizados da seguinte forma: o Capítulo 2 apresenta conceitos computacionais, definições, problemas e algoritmos que estão relacionados com o problema estudado neste trabalho. No Capítulo 3 apresentamos as heurísticas propostas neste trabalho. No Capítulo 4 descrevemos os testes realizados, incluindo diversas tabelas e gráficos detalhando os resultados obtidos. No Capítulo 5 apresentamos as considerações finais, os trabalhos em andamento e o que ainda pode ser explorado em trabalhos futuros.

CAPÍTULO

2

Conceitos Preliminares

Para um melhor entendimento do problema tratado neste trabalho, é necessário conhecer alguns conceitos preliminares descritos neste capítulo. Na Seção 2.1 descrevemos conceitos sobre sequências, na Seção 2.2 apresentamos o conceito da distância de Hamming e na Seção 2.3 abordamos sobre paralelismo. Na Seção 2.4 abordamos o problema das subcadeias específicas e algoritmos exatos para solucionar o problema. Na Seção 2.5 descrevemos conceitos de probabilidade e discutimos como esse conceito se encaixa ao nosso contexto. Na Seção 2.6 descrevemos sobre a equação da reta. E finalmente, na Selão 2.7 descrevemos conceitos biológicos relevantes para a compreensão dos tipos de dados que estamos utilizando neste trabalho.

2.1 Sequências

Um **alfabeto** Σ é um conjunto finito de elementos chamados **símbolos** ou **caracteres**. Uma **sequência** ou **cadeia** s sobre Σ é uma lista de símbolos $s_1 s_2 \ldots s_n$ onde cada $s_i \in \Sigma$. O **comprimento** de s, denotado por $|s|$, é n e corresponde à quantidade de símbolos que s possui. A **sequência vazia** ou **cadeia vazia** é denotada por ε e corresponde à sequência com nenhum símbolo, ou seja, $|\varepsilon| = 0$.

Dada uma sequência s, a sequência $s_i s_{i+1} \ldots s_j$, com $i \geq 1$ e $j \leq |s|$ é uma **subcadeia** de s, sendo denotada por $s[i, j]$. Portanto, uma subcadeia de uma sequência s é um segmento ou trecho de s com zero ou mais caracteres consecutivos. Se $i > j$, a subcadeia corresponde à cadeia vazia ε. Se $1 \leq i \leq |s|$, dizemos que i é a **posição** do símbolo s_i. Para qualquer i, a subcadeia $x = s[1, i]$ ($s[i, |s|]$) é chamada de **prefixo** (**sufixo**) de s, e x é **prefixo próprio** (**sufixo pró-**

prio) de s se $x \neq \varepsilon$ e $x \neq s$. Para exemplificar as definições acima, considere $s =$ AAGGT uma sequência sobre $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. Logo, $|s| = 5$, $s_1 = \mathtt{A}$, $s_5 = \mathtt{T}$, $s[2,4] = \mathtt{AGG}$ é uma subcadeia de s, $s[1,3] = \mathtt{AAG}$ é um prefixo próprio de s, $s[4,5] = \mathtt{GT}$ é um sufixo próprio de s e $s[4,3] = \varepsilon$ é a subcadeia vazia.

2.2 Distância de Hamming

A **distância de Hamming**, estabelecida por Richard Hamming [13], entre duas sequências s e t de mesmo comprimento, corresponde ao menor número de substituições necessárias para transformar s em t, ou seja, a distância de Hamming entre duas sequências é o número de posições que elas diferem entre si. Formalmente, a distância de Hamming d ou simplesmente distância entre s e t, ambas de comprimento n, é definida como:

$$d(s,t) = \sum_{i=1}^{n} [s_i \neq t_i], \text{ onde}$$

$$[s_i \neq t_i] = \begin{cases} 1 & \text{se } s_i \neq t_i, \\ 0 & \text{caso contrário.} \end{cases}$$

Dizemos que s e t possuem $d(s,t)$ **diferenças** e essa medida pode ser obtida percorrendo ao mesmo tempo as sequências s e t e contando o número de posições i tais que $s_i \neq t_i$, o que podemos fazer em tempo $\Theta(n)$.

Para exemplificar, considere $s =$ AAGGT e $t =$ AATAT. Temos $d(s,t) = 2$ pois $s_3 \neq t_3$ e $s_4 \neq t_4$ e $s_i = t_i$ para todo $i \neq 3$ e $i \neq 4$. Então para transformar s em t precisamos realizar duas substituições: s_3 em $t_3 = \mathtt{T}$ e s_4 em $t_4 =$ A.

2.3 Paralelismo

Nesta seção, apresentamos os conceitos de paralelismo que serão necessários para descrever a nossa abordagem paralela para as heurísticas.

Cluster

No contexto de computação, o termo *cluster* faz referência à arquitetura de sistema que une dois ou mais computadores como se fossem apenas um. Os computadores ficam interligados em um único sistema e atuam conjuntamente no processamento de dados e execução de tarefas mais complexas que exigem muitos processadores. Em um *cluster*, cada **computador interligado** é denominado de **nó** e não existe uma quantidade limite para o números de nós de um *cluster*. O tipo de ambiente de computação paralela utilizado neste

trabalho consiste em um cluster de CPUs interconectadas por meio de uma rede dedicada de alta velocidade (ver Figura 2.1).

Modelo

Para o desenvolvimento de programas paralelos, existe uma variedade de modelos [18, 19]. A seguir descrevemos aquele que utilizamos nesse trabalho. No modelo CGM (*Coarse Granularity Multicomputer*) [18], um algoritmo realiza uma sequência de super-passos, onde cada super-passo é dividido em uma fase de comunicação global e uma fase de computação local. Neste modelo temos diversos nós conectados em rede. Para o funcionamento do nosso modelo, algum nó será o nó mestre. O nó mestre irá fazer a comunicação entre os nós e distribuir as tarefas para realizar a computação na próxima fase do super-passo. O nó mestre pode estar incluído na etapa de computação, mas o nosso modelo não o incluirá (ver Figura 2.1).

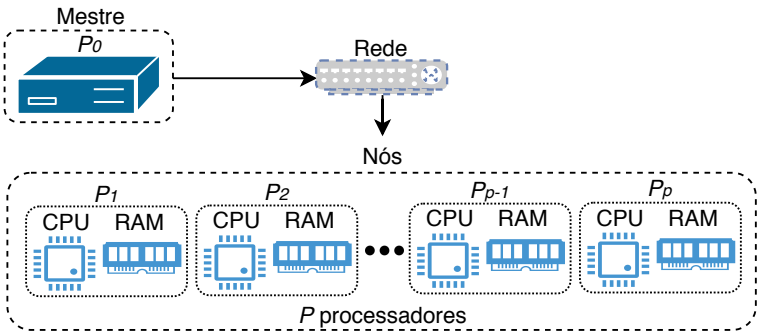

Figura 2.1: Ambiente de computação paralela. O modelo CGM é utilizado neste ambiente de computação paralela que consiste de um nó mestre e de p nós interligados por uma rede dedicada. Cada nó corresponde a um computador com seu processador e memória local. O nó mestre será considerado o processador P_0, e os demais nós, processadores entre $P_1, \ldots, P_p$. No super-passo de comunicação, o nó mestre irá comunicar-se com os demais nós enviando os dados necessários. No super-passo de computação, os nós $P_1, \ldots, P_p$ vão fazer a computação. E esses super-passos serão repetidos até o fim da execução.

MPI

MPI (*Message Passing Interface*) é uma biblioteca para troca de mensagens. O paradigma de troca de mensagens consiste em um conjunto de funções que torna possível a gerência, comunicação e sincronização entre processadores. O MPI está disponível para as linguagens C, C++ e Fortran [18, 20] e mais recentemente para Python [21]. O objetivo do MPI é ser usado em programas que exploram a existência de múltiplos processadores através da troca de mensagens.

Na especificação do MPI, os processadores são identificados por números naturais, ou seja, se temos $p+1$ processadores (ou nós), então cada um será identificado como um valor entre $0,\ldots,p$. No nosso modelo utilizado (ver Figura 2.1), temos $p+1$ processadores, considerando o nó mestre como processador P_0, e os demais nós, processadores entre $P_1,\ldots,P_p$. No nosso modelo, P_0 divide a entrada entre os $P_1,\ldots,P_p$ processadores. [18, 19, 20]

2.4 Problema da Seleção das Subcadeias Específicas

O objetivo desta seção é introduzir o Problema da Seleção das Subcadeias Específicas – SSP. Apresentamos então os conceitos necessários para a descrição de dois algoritmos conhecidos para uma versão estreitamente relacionada ao SSP; essa versão simplificada é denominada aqui como Problema Simplificado da Seleção das Subcadeias Específicas – SSP_s. Por ser uma versão simplificada do SSP, diversos conceitos como os algoritmos exatos descritos nesta mesma seção, a esperança e a variância estudadas na Seção 2.5 e as heurísticas propostas no Capítulo 3 são mais simples. Entretanto, a extensão desses conceitos para o SSP é imediata. A descrição do SSP, que é a versão mais geral, é justificada pelas aplicações estudadas, cujos testes são apresentados no Capítulo 4. A organização desta seção é feita conforme a seguir. Na Seção 2.4.1 definimos o SSP_s e na Seção 2.4.2 descrevemos e analisamos dois algoritmos exatos para o SSP_s. Analisamos a complexidade dos algoritmos apresentados e, ao final, na Seção 2.4.3 estendemos o problema simplificado SSP_s para SSP.

2.4.1 Definição

Dados as sequências s, t, tais que $|s| = m$ e $|t| = n$, e $k \in \mathbb{N}$, definimos a função $p : \Sigma^* \times \Sigma^* \times \mathbb{N} \to \mathbb{N}$; onde $p(s,t,k)$ é o menor inteiro l tal que $d(s[1,l],t[1,l]) = k$; se tal l não existe, então $p(s,t,k) = \min\{m,n\}+1$. O valor $p(s,t,k)$ pode ser encontrado percorrendo s e t ao mesmo tempo da esquerda para direita até encontramos k diferenças ou até chegarmos ao fim de uma das sequências s ou t. Esse procedimento pode ser feito em tempo $O(\min\{m,n\})$. Definimos também um vetor r indexado por $1,\ldots,m$ com $r[i] = \max_j\{p(s[i,m],t[j,n],k)\}$. Se $r[i] \leq \min\{m-i+1,n\}$, então dizemos que $s[i,r[i]+i-1]$ é o **prefixo mínimo de** $s[i,m]$ **em relação a** t e é a **subcadeia minimal de** s **começando em** i **em relação a** t; caso contrário dizemos que não existe o prefixo mínimo de $s[i,m]$ nem a subcadeia minimal de s começando em i. Se $s[i,j]$ é uma subcadeia minimal de s, qualquer cadeia $s[i,h]$, com $j \leq h \leq m$ é dita ser uma **(sub)cadeia específica** em relação à sequência t. Note que se x é uma cadeia específica, então $d(x,y) \geq k$ para toda subcadeia y em t.

Dadas as definições anteriores, o problema da seleção das subcadeias específicas simplificado pode ser definido como segue:

Definição 2.1 (O Problema da Seleção das Subcadeias Específicas Simplificado – SSP$_s$ (*Simplified Specific Substring Problem*)). *Dadas duas sequências s e t, $|s| = m$ e $|t| = n$, e um inteiro k, encontrar todas as subcadeias minimais em s com pelo menos k diferenças em relação a toda subcadeia em t.*

De acordo com a definição do problema, diversas subcadeias minimais em s podem existir. De fato, em geral, para cada posição i de s, existe uma subcadeia minimal de s começando em i, ou seja, um prefixo mínimo do sufixo $s[i,m]$. Por outro lado, se $m-i+1 \leq n$ e $x = s[i,m]$ não é subcadeia específica, então existe uma subcadeia y em t tal que $d(x,y) < k$.

Dizemos que o vetor r definido acima é a **resposta** do SSP$_s$; se $r[i] \leq \min\{m-i+1, n\}$, $r[i]$ é a **resposta do SSP$_s$ para a posição i de s**; caso contrário dizemos que não existe resposta do SSP para a posição i de s e temos que $r[i] = m-i+2$. Note que a resposta do SSP$_s$ não são as subcadeias minimais em s, mas essas últimas podem ser imediatamente obtidas a partir do vetor r. Em outras palavras, uma cadeia x é uma subcadeia minimal de s se e somente se existe i tal que $r[i] \leq \min\{m-i+1, n\}$ e $s[i, i+r[i]-1] = x$.

Para exemplificar uma resposta para o SSP$_s$ e sua relação com cadeias minimais, tome duas sequências $s = \texttt{TCGTGT}$ e $t = \texttt{GCTGCG}$, e $k = 3$. O vetor r tem $m = |s| = 6$ posições. A subcadeia minimal começando em 1 é TCGT o que implica que $r[1] = |\texttt{TCGT}| = 4$; além disso, se compararmos $x = \texttt{TCGT}$ com todas as subcadeias y de comprimento 4 em t, temos que $d(x,y) \geq 3$. Não existem subcadeias minimais começando em 4, 5 ou 6 porque para cada $i = 4, 5,$ e 6, temos que existe pelo menos uma subcadeia y de t de comprimento $m-i+1$ tal que $d(s[i,m], y) < 3$. Logo, pela definição, essas posições não possuem respostas, $r[4] = 6-4+2 = 4$, $r[5] = 6-5+2 = 3$ e $r[6] = 6-6+2 = 2$.

2.4.2 Algoritmos para o SSP

Nesta seção abordamos dois algoritmos exatos para o SSP: um algoritmo de Força Bruta – FB e outro algoritmo baseado no algoritmo do Maaß [14] que denominamos de KHD – k-Hamming Distance.

Força Bruta – FB

Dadas duas sequências s e t, com $|s| = m$ e $|t| = n$, e um inteiro $k > 0$, o Algoritmo 1 é um algoritmo iterativo ingênuo que descreve em detalhes como encontrar uma resposta para o SSP$_s$. Em cada iteração $i = 1, 2, \ldots, m$ correspondente ao laço na linha 2, o algoritmo calcula o valor de $r[i]$. Em cada iteração

i, o algoritmo determina para cada j, o valor $p(s[i,m],t[j,n],k)$, o que é feito pelo laço da linha 3 e armazena em $r[i]$ o maior valor encontrado conforme linha 9.

Algoritmo 1: FORÇA BRUTA – FB

Entrada: duas sequências s e t, com $|s| = m$ e $|t| = n$, e um inteiro $k > 0$.
Saída: um vetor de inteiros r onde $r[i] = \max_j\{p(s[i,m],t[j,n],k)\}$.

```
1  r[i] ← 0 para i = 1,...,m
2  para i ← 1 até m faça
3      para j ← 1 até n faça
4          l ← err ← 0
5          enquanto err < k e i+l < m e j+l < n faça
6              se s[i+l] ≠ t[j+l] então err ← err+1
7              l ← l+1
8          se err = k então r[i] ← MAX(r[i],l)
9          senão r[i] ← MAX(r[i],l+1)
10 devolve r
```

Complexidade

Seja $|s| = m$ e $|t| = n$. Para computar o valor de cada $l = r[i]$, a linha 5 é executada $O(\min\{m-i,n-j\}) = O(\min\{m,n\})$ vezes. Como são executadas $O(n)$ vezes o laço começando na linha 3 e em cada iteração são executadas $O(\min\{m,n\})$ vezes a linha 5, temos que em cada iteração da linha 3 o tempo gasto é $O(n) \cdot O(\min\{m,n\}) = O(\min\{mn,n^2\})$. Então, sendo m vezes executada a linha 2, no total o algoritmo gasta tempo $O(m) \cdot O(\min\{mn,n^2\} = O(\min\{m^2n,mn^2\})$. Portanto, se $n = m$, então a complexidade total do algoritmo é $O(n^3)$.

Maaß – KHD

Dadas duas sequências s e t, com $|s| = m$ e $|t| = n$, e um inteiro $k > 0$, o Algoritmo 2 é um algoritmo iterativo baseado no algoritmo do Maaß que também descreve em detalhes como encontrar uma resposta para o SSP_s. Em cada iteração da linha 4, o algoritmo armazena, em $r[h]$ o valor do comprimento da subcadeia minimal começando em h em relação à $t[h',n]$, onde $h' = h - i + j$ para cada $h = i,\ldots,m$.

Corretude

Invariante 2.1. *No começo de cada iteração das linhas 4–17 do* ***KHD****,* $r[h]$ *vale* $\max_{h'}\{p(s[h,m],t[h',n],k)\}$ *para cada* $h = 1,\ldots,m$*, com* $h' \geq j - i + h$*.*

A Figura 2.2 representa a propriedade do Invariante 2.1.

Demonstração. Indução no número de iterações $I = n - j + i + 1$.

Algoritmo 2: *k-Hamming Distance* – KHD

Entrada: duas sequências s e t, com $|s| = m$ e $|t| = n$, e um inteiro $k > 0$.
Saída: um vetor de inteiros r onde $r[i] = \max_j\{p(s[i,m],t[j,n],k)\}$.

```
1  r[i] ← 1 para i = 1,...,m
2  i ← 1
3  j ← n+1
4  enquanto i ≤ m faça
5      se j ≠ 1 então j ← j−1
6      senão i ← i+1
7      max ← min{m, i+n−j}
8      l ← p(s[i,m], t[j,n], k−1)
9      L ← 0
10     enquanto i+L ≤ max faça
11         enquanto l+i+L < max e s_{l+i+L} = t_{l+j+L} faça  l ← l+1
12         se max ≥ l+i+L então  l ← l+1
13         faça
14             se r[i+L] < l então  r[i+L] ← l
15             L ← L+1
16             l ← l−1
17         enquanto i+L ≤ max e s_{i−1+L} = t_{j−1+L} ;
18 devolve r
```

No início da primeira iteração, quando $j = n+1$, $i = 1$, isto é, quando $I = n-j+i+1 = 1$, como $h > 0$, vale que $h' \geq j-i+h = n+1-1+h > n$, o que implica que $t[h',n] = \varepsilon$ para todo h'. Isso implica que $p(s[h,m],t[h',n],k) = 1$ para todo h independentemente de h'. Segue da inicialização de r na linha 1 que vale o invariante quando $I = 1$.

Suponha que $I > 1$. Pela hipótese de indução, temos que no início da iteração $I-1$, $r[h]$ vale $\max_{h'}\{p(s[h,m],t[h'+1,n],k)\}$ para cada $h = 1,\ldots,m$, com $h' \geq j-i+h$. No laço da linhas 10 armazenamos em l o valor de $p(s[h,m],t[h',n],k)$ ($h = i+L$ e $h' = j+L$). Portanto, no final da iteração, temos armazenado em $r[h]$ o valor de $\max_{h'}\{p(s[h,m],t[h'+1,n],k)\}$, para cada $h = 1,\ldots,m$ e $h' \geq j-i+h$.

□

Segue como consequência do Invariante 2.1 o seguinte resultado.

Teorema 2.2. *Depois de rodar o KHD, para cada* $h = 1,\ldots,m$, $r[h]$ *vale*

$$\max_{h'}\{p(s[h,m],t[h',n],k)\}$$

para todo $h = 1,\ldots,m$.

Complexidade

Seja $|s| = m$ e $|t| = n$, o número de (i,j) executados na linha 4 é $m+n-1 = O(m+n)$. Portanto, o número de iterações é $O(m+n)$. Na linha 8, no pior caso,

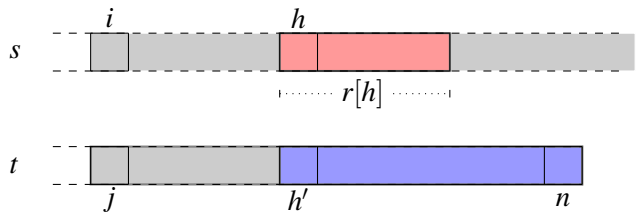

Figura 2.2: No começo de cada iteração (i, j) do laço das linhas 4–17 do **KHD**, para cada $h = 1, \ldots, m$, $s[h, h+r[h]-1]$ (parte rosa na figura) é a subcadeia minimal de s com k diferenças de cada subcadeia de $t[h', n]$ (parte azul), onde $h' = h - i + j$. Embora na figura a representação seja para $i \leq h$, esta propriedade se mantém mesmo se $h < i$. Se $h < i$, então $s[h, h+r[h]-1]$ é a subcadeia minimal de s com k diferenças de cada subcadeia em t.

percorre-se toda a sequência, com isso sendo feito em $O(\min\{m,n\})$. Durante cada iteração do laço da linha 4, a estrutura de repetição na linha 10 executa $O(min\{m,n\})$. A complexidade total é $O(m+n) \cdot (O(\min\{m,n\}) + O(min\{m,n\}))$. Se $|s| = |t| = n$, então $\min\{m,n\} = n$. Logo, a complexidade total é $O(n+n) \cdot (O(n) + O(n)) = O(4n^2) = O(n^2)$.

2.4.3 Extensão do problema simplificado

O problema definido anteriormente foi o SSP_s, uma versão simplificada do problema das subcadeias específicas que considera como entrada duas sequências e um inteiro $k > 0$. O problema é facilmente estendido para uma versão que considera como entrada dois conjuntos A e B de sequências e um inteiro $k > 0$. Considerando os dois conjuntos, o problema pode ser definido como segue:

Definição 2.2 (O Problema da Seleção das Subcadeias Específicas – SSP (*Specific Substring Problem*)). *Dados dois conjuntos A e B de sequências e um inteiro k, encontrar todas as subcadeias minimais em A com pelo menos k diferenças em relação a toda subcadeia em B.*

Neste novo contexto, ambos os algoritmos descritos anteriormente devolvem para cada sequência s em A um vetor r.

2.5 Probabilidade

Nessa seção consideramos um alfabeto Σ com 4 símbolos, $n, m \in \mathbb{N}$ e s, t sequências escolhidas uniformemente num conjunto C de sequências, onde C contém todos os pares de sequências com elementos em Σ com comprimentos m, n, respectivamente.

Sejam $k,l \in \mathbb{N}$, tais que $k \leq l \leq \min\{m,n\}$, x prefixo de s e y subcadeia de t, ambas cadeias de comprimento l. A probabilidade de x e y terem pelo menos k diferenças é

$$p(l) = \sum_{i=k}^{l} \frac{\binom{l}{i} \cdot 3^i}{4^l},$$

ou seja $p(l)$ é a probabilidade de $d(x,y) \geq k$.

Seja $X_j(l)$ uma variável aleatória:

$$X_j(l) = \begin{cases} 1 & \text{se } d(x,t[j,j+l-1]) \geq k; \\ 0 & \text{caso contrário.} \end{cases}$$

e denote por $\mathbb{P}$ a função de probabilidade da variável aleatória $X_j(l)$. Note que, de acordo com o parágrafo anterior, a probabilidade de x ter pelo menos k diferenças com cada subcadeia $t[j,j+l-1]$ é $p(l)$, ou seja, $\mathbb{P}[X_j(l)=1] = p(l)$.

Considere agora a variável aleatória

$$X(l) = \begin{cases} 1 & \text{se } x \text{ e cada subcadeia} \\ & \text{de } t \text{ de comprimento } l \\ & \text{tem pelo menos } k \text{ dife-} \\ & \text{renças;} \\ 0 & \text{caso contrário.} \end{cases}$$

Para existir pelo menos k diferenças entre x e cada subcadeia de t de comprimento l, é necessário que $X_j(l) = 1$ para cada j, $1 \leq j \leq n-l+1$, em outras palavras,

$$X(l) = \prod_{j=1}^{n-l+1} X_j(l)$$

e

$$\mathbb{P}[X(l)=1] = \mathbb{P}\left[\prod_{j=1}^{n-l+1} X_j(l) = 1\right].$$

Como $t[i+1,i+l-1]$ é uma subcadeia comum de $t[i,i+l-1]$ e $t[i+1,i+l]$, os eventos $X_i(l)=1$ e $X_{i+1}(l)=1$ não são independentes. Entretanto, podemos ver na Seção 4.1 que $X_i(l)$ e $X_{i+1}(l)$ se comportam como variáveis independentes, pelo menos considerando os comprimentos e grandezas que utilizamos na prática. Nesse sentido,

$$\mathbb{P}\left[\prod_{i=1}^{n-l+1} X_i(l) = 1\right] \approx \prod_{i=1}^{n-m+1} \mathbb{P}[X_i(l)=1] = p(l)^{n-l+1}.$$

Definimos também uma variável aleatória Y que é o comprimento l do menor prefixo x de s tal que x e qualquer subcadeia de t de comprimento r tenham

k diferenças. Então $Y = l$ se e somente se, $X(l) = 1$ e $X(l-1) = 0$ e

$$\mathbb{P}[Y = l] \quad = \quad \mathbb{P}[X(l) = 1] - \mathbb{P}[X(l-1) = 1]. \tag{2.1}$$

Portanto, o valor esperado de Y é

$$\begin{aligned}
\mathtt{E}(Y) &= \sum_{l=k}^{n} (\mathbb{P}[Y = l] \cdot l) \\
&= \sum_{l=k}^{n} \Big(\mathbb{P}[(X(l) = 1] \cdot l - \mathbb{P}[X(l-1) = 1] \cdot l \Big) \\
&= n \cdot \mathbb{P}[X(n) = 1] - \sum_{l=k}^{n-1} \mathbb{P}[X(l) = 1] - k \cdot \mathbb{P}[X(k-1) = 1] \\
&= n \cdot \mathbb{P}[X(n) = 1] - \sum_{l=k}^{n-1} \mathbb{P}[X(l) = 1],
\end{aligned}$$

na qual a última igualdade é porque não existe uma subcadeia com comprimento $(k-1)$ com k diferenças com outra sequência de comprimento $(k-1)$ o que implica que $\mathbb{P}[X(k-1) = 1] = 0$.

Definimos QE o **quasi-valor esperado** de l, que é um valor aproximado para o valor esperado considerando que $\mathbb{P}[X(l) = 1] \approx p(l)^{n-l+1}$. Mais precisamente,

$$\begin{aligned}
QE &= n \cdot p(n) - \sum_{l=k}^{n-1} p(l)^{n-r+1} \\
&\approx n \cdot \mathbb{P}[X(n) = 1] - \sum_{l=k}^{n-1} \mathbb{P}[X(l) = 1] = \mathtt{E}(Y).
\end{aligned}$$

A *variância* de Y é

$$\mathtt{V} = \mathtt{E}(Y^2) - \mathtt{E}(Y)^2.$$

Similarmente, definimos como **quasi-variância** – QV uma aproximação para a variância de l como segue:

$$QV = \Big(\sum_{l=k}^{n} l^2 \cdot \mathbb{P}[Y = l] \Big) - QE^2.$$

2.6 Equação geral da reta

Um ponto no plano cartesiano pode ser representado pelo par $p = (x_1, y_1)$ de números, onde x_1 é a abscissa e y_1 é a ordenada de p. O conjunto de pontos (x, y) da reta r que passa por p e possui coeficiente angular α é dada pela **equação geral da reta** $y - y_1 = \alpha(x - x_1)$. A equação da reta pode também ser dada na forma $y = ax + b$, onde $a = \alpha$ e $b = y_1 - \alpha \cdot x_1$. Se α não é dado, mas conhecemos dois pontos $p_1 = (x_1, y_1)$ e $p_2 = (x_2, y_2)$ da reta, com $x_1 \neq x_2$, então podemos determinar α como

$$\alpha = \frac{y_2 - y_1}{x_2 - x_1}, \tag{2.2}$$

ou seja, é a reta que passa pelos pontos p_1 e p_2. O conjunto de pontos (x,y) da reta que passa por p_1 e p_2 tais que $x_1 \leq x \leq x_2$ é o segmento de reta que liga os pontos p_1 e p_2. O inteiro mais próximo de $x \in \mathbb{R}$ é $\lfloor x \rfloor$ se $x - \lfloor x \rfloor \leq 0.5$; caso contrário é $\lceil x \rceil$. Um ponto inteiro é o ponto cujas coordenadas são inteiras. Dada uma reta w no plano, o ponto inteiro mais próximo de w que tem abscissa $x \in \mathbb{N}$, é (x,y'), tal que y' é o inteiro mais próximo de y com (x,y) ponto de w.

2.7 Biologia

Nesta seção vamos descrever alguns conceitos biológicos relevantes para compreendermos o tipo de dados que temos nos testes e a aplicação biológica do Problema da Seleção das Subcadeias Específicas ao determinar marcadores.

2.7.1 Ácidos nucleicos

Os **nucleotídeos**, em Biologia Molecular, são blocos construtores dos ácidos nucleicos. O **ácidos nucleicos**, então, são moléculas com extensas cadeias carbônicas, formadas por sequência desses blocos de nucleotídeos. Os dois tipos de ácidos nucleicos são: DNA e RNA. As bases nitrogenadas que compõem os ácidos nucleicos são: (**A**) adenina, (**G**) guanina, (**C**) citosina, (**T**) timina e (**U**) uracila, sendo a timina a base exclusiva do DNA e a uracila do RNA. As outras três bases ocorrem em ambos os ácidos nucleicos [1, 11]. O DNA é um material hereditário que contém informações necessárias para definir e executar todas as funções biológicas de um indivíduo e pode ser representado pela sequência de nucleotídeos que aparece na molécula. Dentre as funções do DNA temos, armazenar instruções genéticas, controlar a atividade celular e produzir RNA. O DNA de duas espécies diferentes é diferente, e indivíduos que pertencem a mesma espécie possuem com pequenas variações o mesmo DNA. Neste trabalho detectamos diferentes DNAs pelas suas diferenças entre si.

Na Figura 2.3 podemos ver a estrutura do DNA [1]. Das informações contidas na imagem, a presença dos nucleotídeos e como eles estão representado no DNA são os pontos importantes para o nosso trabalho. Com base nesta imagem, entendemos que o DNA pode ser representado com uma concatenação de nucleotídeos, ou seja, pode ser representado computacionalmente por uma cadeia de caracteres.

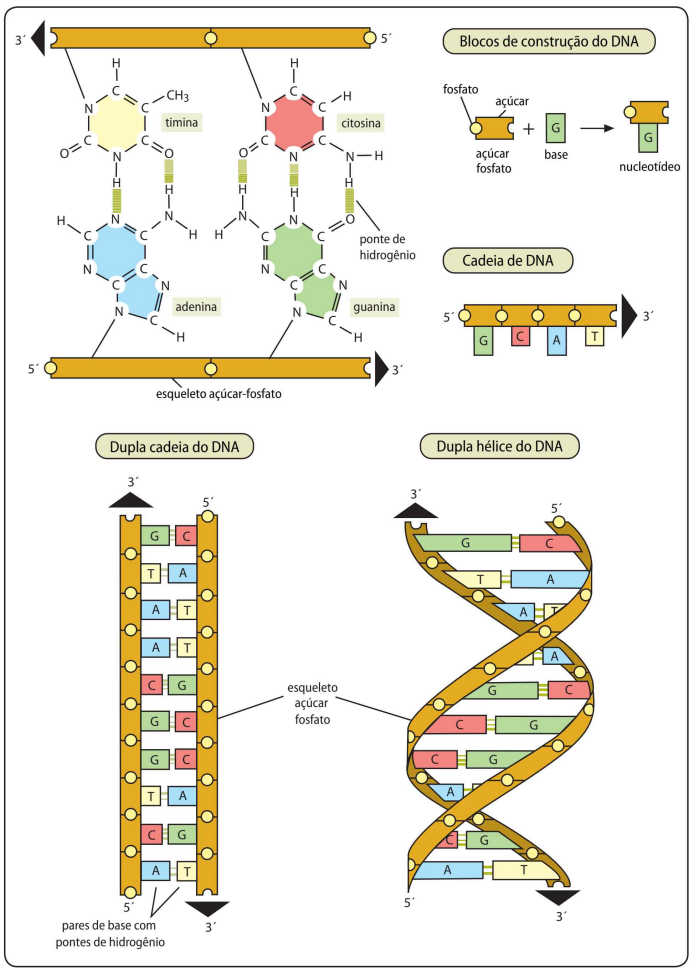

Figura 2.3: Estrutura biológica de um DNA, figura retirada do livro da Maria Ribeiro [1]. Os ácidos nucleicos são formados por nucleotídeos, que por sua vez são formados por três elementos: um radical "fosfato", uma desoxirribose (um açúcar do grupo das pentoses) e a base nitrogenada dividida em dois grupos: Purinas (Adenina e Guanina) e as Pirimidinas (Citosina, Timina e Uracila (essa última é para o RNA)). Uma cadeia de DNA é uma sequência de nucleotídeos dessa molécula unidos quimicamente pelos carbonos 3' e 5' da pentose, através do fosfato. A estrutura do DNA consiste, basicamente, de duas cadeias de nucleotídeos dispostas de forma antiparalelas, dispostas uma sobre a outra. A união das duas cadeias é feita por pontes de hidrogênio, que ligam pares de bases específicas do DNA: AT, CG, TA e GC. No RNA a base Timina (T) é substituída pela base Uracila (U). Dentre as informações nesta imagem, a mais relevante para o nosso trabalho é a forma como o DNA é representado: uma cadeia de nucleotídeos, ou seja, computacionalmente é uma concatenação de caracteres.

2.7.2 Marcadores genéticos

Bered et al. [5] definem que **marcadores genéticos** são caracteres com mecanismo de herança simples que podem ser utilizados para avaliar diferenças genéticas entre dois ou mais indivíduos. Turchetto et al. [4] definem marcador genético como qualquer carácter visível ou fenótipo que de alguma forma seja analisável. Nodari et al. [6] definem que marcadores genéticos é uma característica que é capaz de detectar diferenças genéticas entre dois ou mais indivíduos ou organismos. Definimos nesse texto marcadores genéticos, ou simplesmente marcadores, como trechos (pedaços) de DNA presente em uma única amostra ou espécie. Na computação, podemos entender tais trechos como subcadeias de sequências.

2.7.3 Metagenômica

Genoma é a sequência completa de DNA (ou, em alguns vírus, RNA) de um organismo, ou seja, o conjunto de todos os genes de um ser vivo. Um genoma coletivo da microbiota total encontrada em um determinado *habitat*, ou biodiversidade do ambiente, é conhecida como **metagenoma** [22]. Para obter o metagenoma de um determinado ambiente, é realizado o processamento de sequenciamento de uma amostra obtida desse ambiente, que é a obtenção de cadeias de DNA contidas no metagenoma. Na Figura 2.4 apresentamos uma ideia da diferença entre genoma e metagenoma.

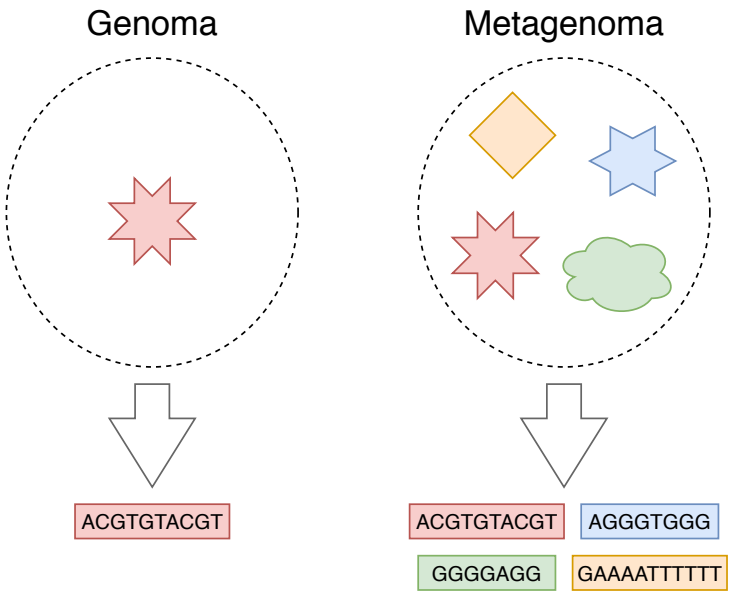

Figura 2.4: Comparação entre genoma e metagenoma. Enquanto que um genoma contém a informação genética de um organismo, o metagenoma contém a informação genética de uma comunidade de organismos.

Com o avanço da Microbiologia, Genética e Biologia Molecular, novas áreas foram surgindo, como por exemplo a **Metagenômica** – o estudo de metageno-

mas. O surgimento da metagenômica resultou em um avanço na compreensão da biodiversidade nos estudos da ecologia microbiana – estudo das interações entre os micro-organismos entre si e com o ecossistema. A metagenômica foi desenvolvida com o intuito de auxiliar os métodos da microbiologia clássica a entender melhor a diversidade de micro-organismos nos ambientes. Esse conceito oferece um caminho para acessar de forma mais completa a diversidade de micro-organismos presentes no ambiente em análise [22, 23].

Contextualização

Neste trabalho estamos interessados em determinar marcadores genéticos. Para dois conjuntos A e B de sequências de DNA, estamos interessados em estudar algoritmos exatos e propor heurística capazes de determinar marcadores que estejam presentes apenas no conjunto A, ou seja, não acontecem no conjunto B. Dito isso, podemos computar metagenomas de biodiversidades diferentes com o propósito de encontrar marcadores que estejam presentes em um determinado metagenoma.

CAPÍTULO

3

Heurísticas

Neste capítulo abordamos duas heurísticas que utilizam um algoritmo exato que resolva o SSP_s, como por exemplo os algoritmos exatos descritos no Capítulo 2. As heurísticas são descritas, inicialmente, para o SSP_s e ao final falamos como aplicá-las no SSP. Como vimos no capítulo anterior, os algoritmos exatos devolvem um vetor r que representa o conjunto de sequências minimais. Denotamos neste capítulo o vetor r indexado por $\{1,\ldots,m\}$ como $r = [x_1, x_2, \ldots, x_m]$ onde $r[i] = x_i$. Quando o SSP_s não tem resposta para algum i, para evidenciar esse fato nós apresentamos em nossos exemplos nesse capítulo como $r[i] = -1$ em vez de um valor positivo conforme a definição.

Na Seção 3.1 descrevemos a ideia geral das heurísticas, e nas Seções 3.2 e 3.3 descrevemos o funcionamento de cada uma das duas heurísticas. Na Seção 3.5 descrevemos uma abordagem paralela utilizando o MPI. Finalmente, na Seção 3.6 descrevemos um resumo de todas as implementações que fizemos neste trabalho e respectivas complexidades.

3.1 Ideia Geral

Dadas duas sequências s e t, com $|s| = m$ e $|t| = n$, e um inteiro $k > 0$, um algoritmo exato de SSP_s devolve o vetor r, tal que $r[i]$ é o comprimento do prefixo mínimo de $s[i,m]$ para $i = 1,\ldots,m$ quando $r[i] \leq m - i + 1$ e -1 caso contrário. Por exemplo, se $s =$ `TGAGAATCTCGGAA`, $t =$ `TGCTCTGGCCGTGT` e $k = 2$, após a execução de um algoritmo exato, temos o seguinte vetor r:

$$r = [4,4,3,3,2,5,6,6,5,4,4,3,2,-1].$$

Para exemplificar, $r[1] = 4$, indica que 4 é o comprimento do prefixo mínimo começando na posição 1 de s, ou seja, a subcadeia $s[1,4] = \texttt{TGAG}$ é minimal, pois $r[i] = 4 < 14 = 14 - 1 + 1 = m - i + 1$.

Como vimos anteriormente, o vetor r possui m posições, onde cada $r[i]$ é resposta do SSP_s para a posição i de s. Dito isso, as heurísticas consistem em, inicialmente, para alguns índices i do vetor r, calcular o comprimento $r[i]$ utilizando um algoritmo exato e denominamos esses índices de **posições exatas**. Os demais índices denominamos de **posições deduzidas** onde armazenamos em $r[i]$ um valor aproximado para o comprimento do prefixo mínimo de $s[i,m]$. As posições exatas são aquelas posições i tais que $i \bmod p = 1$ para algum inteiro p. O valor escolhido nesse trabalho é $p = \lfloor m/\log m \rfloor$ (onde $\log x$ significa $\log_2 x$). Isso significa então que a quantidade de posições exatas é $\lceil m/p \rceil = O(\log m)$. Ou seja, particionamos a sequência de índices em r em aproximadamente $\log m$ blocos onde cada bloco começa com um elemento cujo índice é uma posição exata e os demais são aqueles cujos índices são posições deduzidas. A ideia das heurísticas para aumentar a eficiência é gastar pouco tempo para, em cada posição deduzida i, calcular um valor aproximado para $r[i]$.

Por exemplo, para as sequências $s = \texttt{TGAGAATCTCGGAA}$ e $t = \texttt{TGCTCTGGCCGTGT}$, e $k = 2$, temos então $p = \lfloor 14/\log(14) \rfloor = 3$ e $\lceil 14/3 \rceil = 5$ posições exatas, sendo elas $i = 1$ $(1 \bmod 3 = 1)$, 4 $(4 \bmod 3 = 1)$, 7 $(7 \bmod 3 = 1)$, 10 $(10 \bmod 3 = 1)$ e 13 $(13 \bmod 3 = 1)$. Então calculamos usando um algoritmo exato, o comprimento do prefixo mínimo de $s[i,m]$, para $i = 1,\ 4,\ 7,\ 10$ e 13. Para facilitar, nesse exemplo, para cada posição exata i, s_i foi pintado na cor laranja representando o começo de cada $s[i,m]$ na sequência $s = \texttt{TGAGAATATCGGAA}$. A ideia é executar o algoritmo exato para encontrar $r[i]$ para essas posições exatas i. Após encontrarmos esses valores temos o seguinte vetor r:

$$r \ = \ [4,?,?,3,?,?,6,?,?,4,?,?,2,?].$$

No vetor r acima, as posições exatas i estão com respostas $r[i]$ na cor laranja. As posições deduzidas estão com um ponto de interrogação, representando que para cada posição i no vetor r com o ponto de interrogação, o valor $r[i]$ ainda não foi calculado. Então, denotamos por ? as entradas $r[i]$ quando i é uma posição deduzida ainda não calculada. Os pontos de interrogação são apenas *flags* que indicam quais são as posições que terão seus comprimentos inferidos. Note que o SSP pode não ter resposta para uma posição exata i, então, nesses casos vamos calcular os comprimentos das subcadeias maximais até a posição i, e para as posições a partir de i até o final da sequência, vamos considerar que o SSP não tem resposta. A seguir mostramos as duas estratégias para inferir os valores das posições deduzidas.

3.2 Heurística 1 – H1

Como vimos na Seção 3.1, o algoritmo exato calcula o valor de $r[i]$ para as posições exatas $i = i_1, i_2, \ldots, i_{\lceil m/p \rceil}$, onde $i_1 = 1$ e $i_j = i_{j-1} + p$ para $j = 2, \ldots, \lceil m/p \rceil$. Seja P um vetor de probabilidade onde, imprecisamente falando, cada posição em P representa a probabilidade de um comprimento de um prefixo mínimo ser a resposta para o vetor r. Na Seção 2.5, apresentamos como calcular a probabilidade de l ser o comprimento de um prefixo mínimo de uma subcadeia em relação à uma sequência de comprimento n. A Heurística 1 atribui um valor para $r[h]$ para cada posição induzida h, levando em consideração a tabela P e os resultados previamente calculados para as posições exatas e as posições deduzidas anteriores.

A ideia desta heurística detalhada em Heurística 1 segue os seguintes passos: tome duas sequências s e t, com $|s| = m$ e $|t| = n$, e um $k > 0$. Primeiro, no laço da linha 4, vamos executar o algoritmo exato para calcular $r[i]$ para cada posição exata i em s, como explicado na Seção 3.1. Segundo, na linha 7, vamos criar um vetor de probabilidade P com n posições, onde cada $P[j]$, para $j = 1, \ldots, n$, contém a probabilidade do comprimento do prefixo mínimo ser j em relação à uma sequência de comprimento n (que é o comprimento da sequência t). Para calcular cada entrada de P, vamos utilizar a Fórmula (2.1). Note que, para calcular P, não precisamos conhecer s ou t; basta apenas conhecer os valores de k e de n. Para descrever o terceiro passo, considere para algum inteiro l, a função $P'[l] = P[l]/(P[l-1] + P[l] + P[l+1])$ e $g(P,l)$ que devolve um valor entre $l-1, l, l+1$. Mais precisamente, a função $g(P,l)$ é igual a $l-1$ com probabilidade $P'[l-1]$, igual a l com probabilidade $P'[l]$ e igual a $l+1$ com probabilidade $P'[l+1]$. Então no laço da linha 8, vamos percorrer o vetor r e atribuir a $r[i]$ para cada posição deduzida i (posições marcadas com ?) o valor obtido por $g(P, r[i-1])$.

Por exemplo, dadas $s = \texttt{TGAGAATCTCGGAA}$ e $t = \texttt{TGCTCTGGCCGTGT}$, as duas sequências apresentadas na Seção 3.1 e $k = 2$.

1. Executar o algoritmo exato para as posições 1, 4, 7, 10 e 13, resultando no vetor $r = [4, ?, ?, 3, ?, ?, 6, ?, ?, 4, ?, ?, 2, ?]$;

2. Determinar o vetor de probabilidades P (ver Tabela 3.1);

3. Percorrer o vetor r e inferir um valor para cada posição deduzida i. Para inferir o valor de $r[2]$, por exemplo, vamos computar o comprimento $l = r[1] = 4$, pois 1 é uma posição exata e com a função $g(P,4)$ usando o vetor P(ver Tabela 3.1) a probabilidade dos comprimentos $l-1$, l e $l+1$, ou seja, 3, 4 e 5 ser o comprimento do prefixo mínimo em relação à uma sequência de comprimento n. A função pode escolher entre os três valores

Heurística 1: H1

Entrada: duas sequências s e t, com $|s| = m$ e $|t| = n$, e um inteiro $k > 0$.
Saída: um vetor r, onde cada comprimento $r[i]$ é um valor aproximado da resposta do SSP_s.

```
1  r[i] ←? para i = 1,...,m
2  p ← ⌊m/log m⌋
3  i ← 1
4  enquanto i ≤ m faça
5  |  Calcular r[i] com um dos algoritmos exatos
6  |  i ← i + p
7  Seja P o vetor definido conforme a Fórmula (2.1)
8  para i ← 2 até m faça
9  |  se r[i] = ? então r[i] ← g(P, r[i − 1])
10 devolve r
```

Comprimento	1	2	3	4	5	6	7
Probabilidade	0	0,0001	0,001	0,153	0,439	0,274	0,094
Comprimento	8	9	10	11	12	13	14
Probabilidade	0,027	0,007	0,002	0,001	0.0001	0.00001	0.000001

Tabela 3.1: Exemplo do vetor P com a probabilidade dos comprimentos possuírem $k = 2$ diferenças para uma sequência t de comprimento $n = 14$. Para o comprimento 1 é impossível de acontecer por causa do valor de k, por isso a probabilidade é 0.

anteriores, caso escolha 5, então $r[2] = 5$. Ao final da execução, um dos possíveis vetores é $r = [4,5,5,3,4,5,6,5,5,4,5,5,2,3]$. Para o $r[14]$ não existe resposta do SSP porque $r[14] = 3$ e não existe subcadeias de comprimento 3 para a posição 14 em s.

Complexidade

Considere duas sequências s e t, com $|s| = m$ e $|t| = n$. Considere $\mathcal{T}$ o custo máximo de um algoritmo exato calcular o comprimento $r[i]$ para uma determinada posição exata i. O laço da linha 4, então, tem o custo máximo $\mathcal{T}$ para calcular o comprimento $r[i]$ de cada uma das $O(\log m)$ posições exatas i. O tempo total gasto é então $O(\mathcal{T} \cdot \log m)$. Considere o vetor de probabilidades P com n posições que é definido ma linha 7 com o custo $P(n,k)$. O laço da linha 8 tem o custo de percorrer todo o vetor r, que é feito em $O(m)$. A complexidade da Heurística 1 é portanto $O(\log m \cdot \mathcal{T}) + P(n,k) + O(m) = O(\mathcal{T} \cdot \log m + P(n,k) + m)$.

Para processarmos duas sequências, vamos utilizar de memória $m+n$ para as sequências s e t, m para o vetor r e n para o vetor de probabilidade P. O custo total da complexidade de espaço , portanto, $O(m+n+n+m) = O(2m+2n) = O(m+n)$.

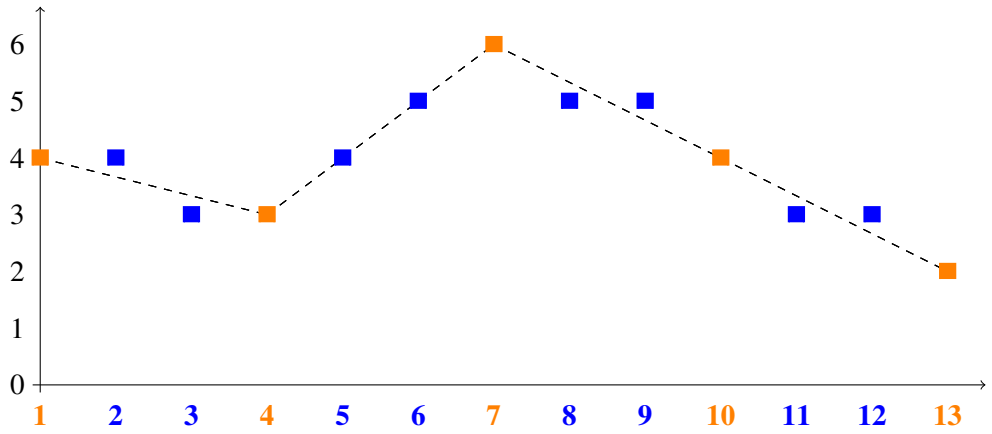

Figura 3.1: Exemplo de retas criadas para o vetor $r = [4,?,?,3,?,?,6,?,?,4,?,?,2,?]$. Os pontos laranjas representam os pares $(i, r[i])$ do vetor r, onde cada posição exata i é a abscissa x e o seu respectivo valor $r[i]$ é a ordenada y. Os pontos azuis representam os pares $(i, r[i])$ onde i é uma posição deduzida e $r[i]$ o valor inferido.

c	1	2	3	4	5
x_c	1	4	7	10	13
y_c	4	3	6	5	2

Tabela 3.2: Exemplo do vetor C com os pares de pontos cartesiano (x,y) para vetor o $r = [4,?,?,3,?,?,6,?,?,4,?,?,2,?]$, onde as abscissas são as posições exatas i do vetor r e as ordenadas são os respectivos $r[i]$. O índice de cada par (x,y) é representado por c.

3.3 Heurística 2 – H2

Como vimos na Seção 3.1, o algoritmo exato calcula o valor de $r[i]$ para as posições exatas $i = i_1, i_2, \ldots, i_{\lceil m/p \rceil}$, onde $i_1 = 1$ e $i_j = i_{j-1} + p$. Para cada $j = 2, \ldots, \lceil m/p \rceil$, considere o segmento de reta w que liga os pontos $(i_{j-1}, r[i_{j-1}])$ e $(i_j, r[i_j])$ no plano cartesiano. A Figura 3.1 mostra a concatenação dos segmentos de reta descrita a partir de $r = [4,?,?,3,?,?,6,?,?,4,?,?,2,?]$. A Heurística 2 define o valor de $r[h]$ com $i_j \leq h \leq i_{j+1}$ tal que $(h, r[h])$ é o ponto mais próximo de w com abscissa h.

Seja C o vetor dos pontos cartesianos com $\lceil m/p \rceil$ pares (x,y) onde x é uma posição exata. Denotamos por $C[i].x$ e $C[i].y$ os valores (x,y) de $C[i]$. O vetor C é indexado por $\{1,2,\ldots,\ldots,\lceil m/p \rceil\}$ e $C[i].x < C[i+1].x$. Por exemplo, o plano cartesiano representando o vetor r apresentado na Figura 3.1 possui 5 posições exatas i na cor laranja, então, o vetor C possui 5 posições representando os 5 pares $(i, r[i])$. Por exemplo, para $x = i = 1$ e $y = r[1] = 4$, temos $C[1].x = 1$ e $C[1].y = 4$, os demais pares estão representados na Tabela 3.2.

A ideia desta heurística detalhada em Heurística 2 segue os seguintes passos: tome duas sequências s e t, com $|s| = m$ e $|t| = n$, e um $k > 0$. Primeiro,

no laço da linha 5, vamos executar o algoritmo exato para calcular $r[i]$ para cada posição exata i em s, como explicado na Seção 3.1. Segundo, na linha 7, vamos criar um vetor C conforme definimos anteriormente. Terceiro, no laço da linha 9, vamos percorrer o vetor C criando equações da reta para dois pares (x,y) consecutivos conforme definimos anteriormente. Na linha 13, para cada posição deduzida i (posições marcadas com ?) entre dois pares (x,y) e (x',y') consecutivos, vamos percorrer o vetor r e atribuir a $r[i]$ o resultado da equação da reta $\lfloor (y+\alpha\cdot(i-x))+0,5\rfloor$, onde α é o coeficiente angular e $i = x+1,\ldots,x'-1$. Após o último par (x,y), no laço da linha 15, vamos percorrer o vetor r e definir que não existe resposta do SSP para cada $r[i]$, com $i = x,\ldots,m$.

Heurística 2: H2

Entrada: duas sequências s e t, com $|s| = m$ e $|t| = n$, e um inteiro $k > 0$.
Saída: um vetor r, onde cada comprimento $r[i]$ é um valor aproximado da resposta do SSP_s.

```
1  r[i] ← ? para i = 1,...,m
2  p ← ⌊m/log m⌋
3  i ← 1
4  enquanto i ≤ m faça
5  |  Calcular r[i] com um dos algoritmos exatos
6  |  i ← i + p
7  Seja C um vetor conforme definimos anteriormente
8  c ← 1
9  enquanto c < ⌈m/p⌉ − 1 faça
10 |  x1 ← C[c].x; y1 ← C[c].y
11 |  x2 ← C[c+1].x; y2 ← C[c+1].x
12 |  α ← (y2−y1)/(x2−x1); c ← c+1
13 |  para i ← x1 + 1 até x2 − 1 faça
14 |  |  r[i] ← ⌊(y1 + α·(i−x1)) + 0,5⌋
15 para i ← C[⌈m/p⌉].x até m faça
16 |  r[i] ← não existe resposta do SSP para a posição i
17 devolve r
```

Por exemplo, dadas as duas sequências $s = \texttt{TGAGAATCTCGGAA}$, $t = \texttt{TGCTCTGGCCGTGT}$ apresentadas na Seção 3.1 e $k = 2$.

1. Executar o algoritmo exato para as posições 1, 4, 7, 10 e 13, resultando no vetor $r = [4,?,?,3,?,?,6,?,?,4,?,?,2,?]$;

2. O vetor C contém todos os pares x e y, então vamos criar a primeira reta com os dois primeiros pares de pontos $r[1] = 4$ e $r[4] = 3$ e com o coeficiente angular $\alpha = \frac{3-4}{4-1} = \frac{-1}{3} \approx -0,33$. Utilizando o primeiro ponto $r[1] = 4$, $x = 1$ e

$y = 4$, temos a seguinte reta:

$$\begin{aligned} y - 4 &= -0,33(x-1) \\ y - 4 &= -0,33x + 0,33 \\ y &= -0,33x + 4,33 \end{aligned}$$

3. Para os pontos de interrogações nas posições $i = 2,\ 3$ e 4 no vetor r, os valores de $y's$ que representarão os comprimentos para $r[i]$ são:

$$\begin{aligned} &\text{para } x = i = 2,\ y = \lfloor(-0,33 \cdot 2 + 4,33) + 0,5\rfloor = 4,0 \\ &\text{para } x = i = 3,\ y = \lfloor(-0,33 \cdot 3 + 4,33) + 0,5\rfloor = 3,0 \end{aligned}$$

   Então, $r[2] = 4$ e $r[3] = 3$. Ao final da execução temos o seguinte vetor $r = [4,4,3,3,4,5,6,5,5,4,3,3,2,-1]$. Nesta heurística, definimos que não existe resposta do SSP para as posições a partir da última posição exata. Observe a Figura 3.1 para verificar o quão próximos estão as comprimentos inferidos da reta utilizada.

Complexidade

Considere duas sequências s e t, com $|s| = m$ e $|t| = n$. Considere $\mathcal{T}$ o custo máximo de um algoritmo exato calcular o comprimento $r[i]$ para uma determinada posição exata i. O laço da linha 4, então, tem o custo máximo $\mathcal{T}$ para calcular $O(\log m)$ posições exatas i. O tempo total gasto é então $O(\mathcal{T} \cdot \log m)$. Considere o vetor de pontos cartesianos C que é definido na linha 7 com $\log m + 1$ pares (x,y). Para determinar o vetor C basta percorrer o vetor r em $O(m)$. O laço da linhas 9 tem o custo de percorrer o vetor r até a posição $C[m/p].x \geq m$ e o laço da linha 15 tem o custo de percorrer o vetor r do último $C[m/p].x$ até m. Os dois laços juntos custa $O(m)$. A complexidade de tempo da Heurística 2 é $O(\mathcal{T} \cdot \log m) + O(m) + O(m) = O(\mathcal{T} \cdot \log m + 2m)$.

Para processarmos duas sequências, vamos utilizar de memória $m+n$ para s e t, m para o vetor r e $\lceil m/p \rceil$ para o vetor C. O custo total da complexidade de espaço é $O(m+n+\log m+m) = O(2m+\log m+n) = O(m+\log m+n)$.

3.4 Heurísticas e o SSP

As heurísticas foram descritas para o SSP_s, a versão simplificada do SSP. Sabemos que a diferença entre os dois problemas consiste na entrada, enquanto o SSP_s considera duas sequências s e t, e $k > 0$, o SSP considera dois conjuntos A e B, e $k > 0$. Para estendermos as heurísticas para que também funcionem com o SSP, devemos executar para cada sequência de A e todas sequências de B o algoritmo exato, ou seja, um vetor r para cada sequência

em A preenchido conforme descremos na ideia geral (ver Seção 3.1). Na Heurística 1 o laço na linha 4 (na Heurística 2 é a linha 4 também) é o responsável por executar o algoritmo exato.

3.5 Abordagem Paralela

As implementações paralelas das heurísticas H1 e H2 são semelhantes, pois o passo de maior consumo é o mesmo nas duas heurísticas, ou seja, no passo para executar o algoritmo exato na linha 4 (para ambas as heurísticas). Sejam dois conjuntos A e B com sequências. Inicialmente as sequências são distribuídas igualmente entre os p processadores pelo nó mestre, de modo que cada processador execute o algoritmo algoritmo exato com $\frac{|A|}{p}$ sequências do conjunto A e todas as sequências do conjunto B (ver Figura 3.2). Após receber os dados de entrada, cada processador calcula independentemente para cada sequência do conjunto A, em relação a cada sequência do conjunto B, o comprimento de seus prefixos mínimos gerando um vetor r para todas $\frac{|A|}{p}$ sequências. Todos os nós devolvem para o nó mestre os vetores r para que a Heurística 1 ou 2 continuem a execução.

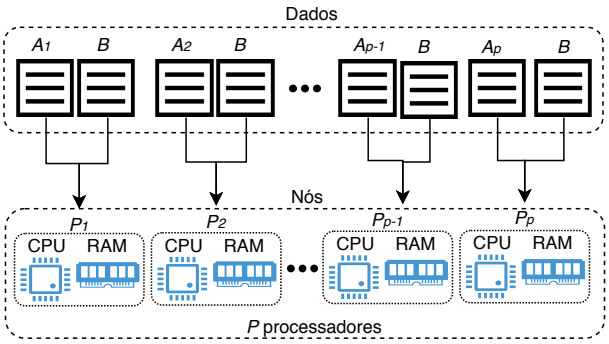

Figura 3.2: Execução para cada A_i de tamanho $\frac{|A|}{p}$, com $i \in \{1 \dots p\}$, e B.

Complexidade

Considere que todas as sequências de A e de B tenham comprimento n. Conforme visto anteriormente, para computar o vetor r referente a uma sequência de A e uma sequência de B, FB gasta tempo $O(n^3)$. Desde que temos que computar um vetor r para cada par de sequências em A e B e temos $\frac{|A|}{p}|B|$ pares de sequências, o tempo total gasto por FB é $\frac{|A|}{p}|B| \cdot O(n^3) = O\left(\frac{|A|}{p}|B|n^3\right)$. Similarmente, o KHD gasta tempo total $O\left(\frac{|A|}{p}|B|n^2\right)$. O nó mestre distribui as sequências entre p processadores, sendo assim, desconsiderando o nó mestre, essa abordagem requer quantidade de memória de $O(2n\frac{|A|}{p} + n|B|)$ em cada processador, $2n\frac{|A|}{p}$ para as $\frac{|A|}{p}$ sequências e seus respectivos vetores r, e $|B|$ para

todas as sequências em B. O nó mestre tem o custo de criar os vetores auxiliares de cada heurística, para a Heurística 1, o vetor P, para a Heurística 2, o vetor C.

3.6 Implementações

Implementamos os algoritmos exatos descritos no Capítulo 2, as heurísticas e a versão paralela das heurísticas descritas anteriormente, ao todo foram implementados cinco programas diferentes. Uma implementação de um programa exato de **F**orça **B**ruta – FB, cuja complexidade de tempo é cúbica. Um programa exato baseado no algoritmo do Maaß que denominamos de k-**H**amming **D**istance – KHD, cuja complexidade de tempo é quadrática. Duas heurísticas que denominamos de **H**eurística **1** – H1 e **H**eurística **2** – H2. Finalmente uma versão paralela da Heurística 2 denominada de H2-P. Todos os programas foram implementados na linguagem orientada a objetos C++. A verão H2-P foi implementada utilizando o MPI com a linguagem C++. Um resumo da análise de complexidade das implementações é apresentado na Tabela 3.3 e as implementações estão disponíveis no github: https://github.com/LucasBarbosaRocha/SSP. Na Tabela 3.3 apresentamos a complexidade de tempo considerando duas sequências s e t, onde $|s| = |t| = n$. Se consideramos dois conjuntos A e B de sequências de comprimento n, o FB tem complexidade de tempo $|A| \cdot |B| \cdot O(n^3)$ e o KHD $|A| \cdot |B| \cdot O(n^2)$. Para H1 as complexidades de tempo são $(|A| \cdot |B| \cdot O((n^2) \cdot \log n) + O(n)) + O(P(n,k))$ usando o FB e $(|A| \cdot |B| \cdot O(n \cdot \log n) + O(n)) + O(P(n,k))$ usando o KHD. Para H2 as complexidades de tempo são $|A| \cdot |B| \cdot (O(n^2 \cdot \log n) + O(n)$ usando o FB e $|A| \cdot |B| \cdot (O(n \cdot \log n) + O(n)$ usando KHD. Note que a complexidade do FB é $O(n^3)$ para encontrar todos os prefixos mínimos, mas o custo de encontrar apenas um prefixo mínimo é $O(n^2)$. Para o KHD, o custo de encontrar um prefixo mínimo é $O(2n) = O(n)$.

			Complexidade	
Sigla	Algoritmo	Versão	Tempo	Espaço
FB	Força Bruta	Exata	$O(n^3)$	$O(n)$
KHD	k-hamming Distance	Exata	$O(n^2)$	$O(n)$
H1	FB ou KHD	Heurística	$O(\mathcal{T} \cdot \log n) + O(P(n,k)) + O(n)$	$O(n)$
H2	FB ou KHD	Heurística	$O(\mathcal{T} \cdot \log n) + O(n)$	$O(n + \log n)$
H2-P	k-hamming Distance	Paralelo	$O((\frac{\|A\|}{p}\|B\|n^2))$	$O(n \cdot \frac{\|A\|}{p} + n \cdot \|B\|)$

Tabela 3.3: Resumo das complexidades. Nesta tabela, para as versões exatas e heurísticas apresentamos a complexidade considerando duas sequências s e t, para $|s| = |t| = n$. Sendo P o custo máximo para construir a tabela de probabilidade e $\mathcal{T}$ o custo máximo para um algoritmo exato calcular o comprimento para uma determinada posição exata. Para a versão paralela, considere A e B dois conjuntos de sequências de comprimento n. A versão paralela possui em cada processador a complexidade de espaço $O(n \cdot \frac{|A|}{p} + n \cdot |B|)$.

CAPÍTULO

4

Análises, Experimentos e Resultados

Neste capítulo abordamos algumas análises e testes relevantes para o trabalho. Na Seção 4.1 descrevemos um estudo para avaliar a confiabilidade dos valores aproximados QE e QV apresentados na Seção 2.5, ou seja, vamos verificar se os valores de QE e QV estão próximos aos valores esperado e a sua variância do comprimento do prefixo mínimo em simulações. Na Seção 4.2 apresentamos os testes com os algoritmos exatos com o propósito de avaliarmos o tempo de execução do algoritmo de força bruta – FB e o algoritmo do Maaß – KHD. Para esses testes consideramos o SSP_s e sequências artificiais e reais e apresentamos tabelas e gráficos com o tempo de execução dos algoritmos.

Na Seção 4.3 apresentamos os testes realizados com as heurísticas, essa seção é dividida da seguinte forma: realizamos testes com o metagenoma da expedição tara, onde as sequências possuem aproximadamente o mesmo comprimento. Apresentamos também testes com sequências de bactérias obtidas da plataforma NCBI com sequências de comprimentos variados e testes com os conjuntos de sequências compartilhando entre si bactérias da mesma família. Para esses testes consideramos o SSP e apenas sequências reais e apresentamos tabelas e gráficos com o tempo de execução das heurísticas e do algoritmo exato e apresentamos tabelas e gráficos de acurácia para avaliarmos a qualidade das heurísticas em relação ao algoritmo exato. Finalmente, apresentamos testes com a versão paralela da Heurística 2 e tabela e gráficos com tempo de execução entre a Heurística 2 e a sua versão paralela.

Os testes experimentais da seções 4.1 e 4.2 foram executados no *cluster*[1] da Universidade Federal do Mato Grosso do Sul com 40 nós, cada nó possui um

[1] http://cluster.ctei.ufms.br/ganglia/

processador Xeon(R) X3440@2.53GHz de 4 cores e 4 GB de RAM. Os testes experimentais da Seção 4.3 foram executados no *cluster*[2] da Universidade Federal do Rio Grande do Norte com 64 nós, cada nó possui um processador Xeon Sixteen-Core E5-2698v3 e 128 GB de RAM. Apresentamos nas seções abaixo uma série de tabelas geradas a partir dos testes realizados, juntamente a algumas observações baseadas nos resultados obtidos.

4.1 Confiabilidade do QE e QV

Na Seção 2.5, nós realizamos um estudo sobre a probabilidade de um prefixo x de uma sequência s ter pelo menos k diferenças com cada subcadeia y de t, onde $|x| = |y|$. Naquela seção definimos os valores QE – quasi-valor esperado e QV – quasi-variância, onde QE é um valor aproximado para o valor esperado Y do comprimento do prefixo, e QV um valor aproximado para a variância de Y do comprimento. Dito isso, nesta seção, estamos interessados em verificar na prática se esses valores podem ser usados para representar a realidade.

Nossos experimentos funcionam da seguinte maneira: geramos uniformemente e randomicamente s e t que são duas sequências de comprimento n. E considere um dado inteiro k. Vamos utilizar um algoritmo exato para calcular o comprimento do prefixo x de s com pelo menos k diferenças com todas as subcadeias y em t. Cada experimento contém um conjunto de testes que corresponde a 1-milhão de pares de sequências de comprimento n. Os experimentos são feitos variando os valores de n e de k conforme a Tabela 4.1. Para cada conjunto de teste, temos $v_1, v_2, \ldots, v_{10^6}$ valores de comprimentos calculados com o algoritmo exato. Calculamos então para cada experimento os valores esperado e a variância, ou seja, para esses valores, nós definimos o QEE – **quasi-valor esperado experimental** e o QVE – **quasi-variância experimental** como segue

$$QEE = \sum_{i=1}^{10^6} \frac{v_i}{10^6} \quad \text{and} \quad QVE = \sum_{i=1}^{10^6} \frac{(v_i - QEE)^2}{10^6}.$$

Estamos interessados em verificar se QEE está próximo de QE e QVE de QV. Além disso, vamos computar a quantidade de sequências que possuem o comprimento do prefixo dentro do intervalo $\left[\lfloor QE - QV \rfloor : \lceil QE + QV \rceil\right]$ e calcular a porcentagem dessas sequências.

Os resultados são mostrados na Tabela 4.1. O valor da esperança experimental QEE do comprimento do prefixo mínimo de s é o valor real da esperança quando se considera o espaço amostral das sequência obtidas randomicamente. Por outro o QE é um valor teórico aproximado para o valor dessa esperança. Verificamos naquela tabela que esses valores estão muito

[2] http://npad.ufrn.br/hardware.php

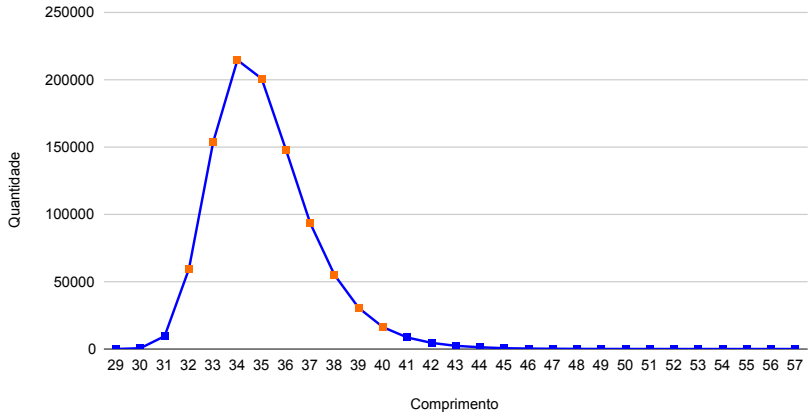

Figura 4.1: Gráfico para a Tabela 4.2, os pontos laranjas representam os valores do intervalo: $\left[\lfloor QE - QV \rfloor : \lceil QE + QV \rceil\right] = [32 : 40]$. A abscissa **comprimento** representa o comprimento do prefixo calculado com o algoritmo exato e a ordenada **quantidade** representa a quantidade de comprimentos calculados. Para esse teste, temos 971.897 comprimentos entre $[32 : 40]$.

próximos, ou seja, em nossos experimentos os valores de QEE diferem de QE na média em 0,42%, no menor valor em 2,37% e no maior valor em 0,43% de QE. Isso significa que para nossas aplicações o valor de QE pode de fato ser usado como uma boa aproximação para a esperança do comprimento do prefixo mínimo. Analogamente observamos que QVE difere de QV na média em 26,17%, no menor valor em 23,61% e no maior valor em 20,79% indicando que esse também pode ser uma boa aproximação para o valor da variância embora em todos os nossos experimentos os valores de $QE - QV$ são na média muito maiores em relação à $QEE - QVE$. Isso significa que o intervalo teórico $\left[\lfloor QE - QV \rfloor : \lceil QE + QV \rceil\right]$ é menor do que o intervalo $\left[\lfloor QEE - QVE \rfloor : \lceil QEE + QVE \rceil\right]$. Entretanto, verificamos que mais de 90% das sequências possuem o comprimento do prefixo mínimo dentro do primeiro intervalo indicando mais uma vez que esse intervalo pode ser considerado aquele que contém os comprimentos de prefixo mínimo mais comuns.

Na Tabela 4.2, para um conjunto de testes, detalhamos a distribuição da quantidade de comprimentos dos prefixos mínimos calculados com o algoritmo exato para sequências de comprimento 100 e $k = 20$, para essa tabela apresentamos um gráfico na Figura 4.2. Verificamos que a quantidade de sequências dentro do intervalo teórico $\left[\lfloor QE - QV \rfloor : \lceil QE + QV \rceil\right]$ é de 971.897 de 1 milhão. Dito isso, podemos concluir que o estudo teórico representa bem a realidade nos testes práticos.

						Comprimento		
Comprimento	k	QE	QV	QEE	QVE	Menor	Maior	%
100	20	35,92	3,74	35,09	4,37	29	57	97,19%
	40	64,32	6,29	63,77	7,87	55	89	98,56%
	60	90,00	12,00	89,12	15,15	80	100	99,92%
200	20	37,33	3,46	36,60	4.27	31	58	95,04%
	40	66,95	5,63	66,71	7,72	59	92	97,76%
	60	95,56	7,77	95,44	10,73	87	131	97,86%
300	20	38,03	3,32	37,36	4.20	32	57	92,35%
	40	68,00	5,32	67,92	7,52	61	97	95,95%
	60	97,04	7,25	97,18	10,48	89	130	97,14%
400	20	38,49	3,23	37,87	4,17	33	61	96,04%
	40	68,65	5,14	68,69	7,43	61	97	96,32%
	60	97,89	6,95	98,19	10,23	89	130	97,22%
500	20	38,84	3,17	38,26	4.15	33	59	92,11%
	40	69,13	5,01	69,26	6,71	63	97	95,50%
	60	98,49	6,74	98,91	10,16	90	128	96,25%

Tabela 4.1: Testes variando o comprimento das sequências onde cada teste contém 1 milhão de pares de sequências. Para cada milhar de testes, a segunda coluna contém o valor de k, a terceira e quarta coluna possuem os valores QE e QV, respectivamente, que são valores teóricos para o valor esperado e a sua variância referente ao comprimento da sequência analisada. As colunas cinco e seis, possuem os valores QEE e QVE que são valores esperados e variância que baseado nos comprimentos obtidos nos testes. A colunas sete e oito possuem o menor e maior comprimento dos prefixos calculado com o algoritmo exato. A última coluna % representa a quantidade de comprimentos dos prefixos que estão entre $\left[\lfloor QE - QV \rfloor : \lceil QE + QV \rceil\right]$. Por exemplo, para o comprimento 100 e $k = 20$ na primeira linha da tabela, o menor comprimento do prefixo calculado com o algoritmo exato em 1-milhão de pares de sequências foi de 29 e o maior foi de 57, de todas as sequências, 97,19% das sequências possuem prefixo com o comprimento dentro do intervalo $[32 : 40]$.

Comprimento do Prefixo	Quantidade
31 <	10.124
32	59.281
33	153.645
34	214.662
35	200.888
36	148.077
37	93.528
38	55.093
39	30.503
40	16.220
> 41	17.979

Tabela 4.2: Quantidade total de comprimentos dos prefixos calculados com o algoritmo exato, para 1 milhão de pares de sequências s e t, onde $|s| = |t| = 100$. Cada prefixo x de s tem pelo menos 20 diferenças com todas as subcadeias y em t, onde $|x| = |y|$. Na primeira coluna contém o comprimento de um prefixo e a segunda coluna contém a quantidade desse comprimento. Por exemplo, o algoritmo exato calculou um prefixo de comprimento 32 em 59.281 sequências (linha 2 na tabela). Essa tabela detalha a primeira linha da Tabela 4.1.

4.2 Algoritmos Exatos

Para os testes desta Seção, vamos considerar o algoritmo de Força Bruta – FB e o algoritmo do Maaß – KHD apresentados no Capítulo 2 e o SSP_s. Estamos interessados em comparar o tempo de execução dos algoritmos exatos estudados, e também verificar na prática quão melhor é o KHD em relação ao FB. Na Seção 4.2.1 apresentamos os resultados para os experimentos considerando sequências artificiais. Finalmente, na Seção 4.2.2 apresentamos os resultados para os experimentos considerando sequências reais.

4.2.1 Sequências Artificiais

Nesta seção, estamos interessados em verificar o tempo de execução do FB e do KHD. Para realizar este experimento, vamos utilizar dados artificiais. Vamos chamar de sequências artificiais aquelas que são geradas aleatoriamente. Para gerar as sequências, desenvolvemos um *script* na linguagem C, que gera randomicamente duas sequências (ou cadeia) de caracteres sobre o alfabeto $\sum = \{A, C, G, T\}$.

Nossos experimentos são divididos da seguinte maneira: primeiro, geramos aleatoriamente s e t que são duas sequências de comprimento 1000 a 10.000, em intervalos de 1000 caracteres, e de 10.000 a 100.000 caracteres, em intervalos de 10.000 caracteres. Segundo, para os mesmos intervalos anteriores, geramos sequências s e t tais que, cada uma dessas sequências é formada por apenas um símbolo, mas os símbolos de s são diferentes dos símbolos de t , e para cada par de sequência, $s \neq t$. Terceiro, geramos sequências s e t tais que, cada uma dessas sequências é formada por apenas um símbolo e os símbolos de s são iguais aos símbolos de t, e para cada par de sequência, $s = t$. Cada experimento contém um conjunto de 19 pares de sequências s e t, os experimentos são feitos variando o comprimento das sequências, mas mantendo $|s| = |t|$ em cada par e variando o valor de k em 30, 300 e 600 conforme as tabelas 4.3, 4.4 e 4.5. Assim, queremos simular o caso aleatorizado, melhor e pior caso, respectivamente.

Para o primeiro experimento, os resultados são apresentados na Tabela 4.3, onde podemos ver como o KHD é mais rápido que o FB. O KHD executa exatamente n^2 comparações enquanto o FB realiza $O(n^3)$ comparações, por essa razão o KHD mantém tempo de execução independente do valor de k. Para o segundo experimento (melhor caso), os resultados são apresentados na Tabela 4.4, KHD se mantém com o tempo de execução próximo aos tempos apresentados no primeiro experimento, enquanto para o FB o tempo diminuiu em relação ao experimento anterior. Para o terceiro experimento (pior caso), os resultados são apresentados na Tabela 4.5, KHD mantém seu tempo de exe-

cução e foi extremamente mais rápido em relação ao FB. Observando o tempo de execução entre o FB e KHD, podemos notar, para duas sequências em todos os experimentos, que o KHD é mais rápido comprovando em algum grau o que foi discutido na complexidade de tempo daqueles algoritmos. Note que o melhor e pior caso tem um grande efeito no FB, mas não tem efeitos no KHD. Isso ocorre porque o FB no pior caso é $O(n^3)$ enquanto que no melhor caso gasta tempo $O(k \cdot n^2)$. Já o KDH sempre gasta tempo $O(n^2)$ independentemente do caso. Entretanto, não identificamos as razões do KHD demorar alguns segundos no caso aleatorizado (Tabela 4.3).

Para os resultados apresentados na Tabela 4.3, apresentamos na Figura 4.2 o gráfico de tempo de execução dos programas FB e KHD para $k = 30$ e nas figuras 4.3 e 4.4 para $k = 300$ e 600, respectivamente. Para os resultados apresentados na Tabela 4.4, apresentamos na Figura 4.5 o gráfico de tempo de execução dos programas FB e KHD para $k = 30$ e nas figuras 4.6 e 4.7 para $k = 300$ e 600, respectivamente. Finalmente, para os resultados apresentados na Tabela 4.5, apresentamos na Figura 4.8 o gráfico de tempo de execução dos programas FB e KHD para $k = 30$, 300 e 600. Note que para esta última tabela temos apenas um gráfico representando os diferentes valores de k. Isso acontece porque o tempo de execução é o mesmo independente de k. Apresentamos os gráficos com o tempo em segundos e em escala logarítmica.

	30		300		600	
Comprimento	FB	KHD	FB	KHD	FB	KHD
1000	00:00:01	00:00:00	00:00:01	00:00:00	00:00:04	00:00:00
2000	00:00:02	00:00:00	00:00:15	00:00:00	00:00:24	00:00:00
3000	00:00:04	00:00:00	00:00:38	00:00:00	00:01:06	00:00:00
4000	00:00:06	00:00:00	00:01:10	00:00:00	00:02:07	00:00:00
5000	00:00:09	00:00:00	00:01:52	00:00:00	00:03:27	00:00:00
6000	00:00:13	00:00:01	00:02:43	00:00:01	00:05:07	00:00:01
7000	00:00:19	00:00:01	00:03:44	00:00:01	00:07:06	00:00:01
8000	00:00:25	00:00:01	00:04:56	00:00:01	00:08:56	00:00:01
9000	00:00:30	00:00:01	00:05:38	00:00:01	00:09:24	00:00:01
10.000	00:00:40	00:00:01	00:06:18	00:00:01	00:10:03	00:00:01
20.000	00:02:58	00:00:05	00:20:52	00:00:05	00:43:13	00:00:05
30.000	00:06:26	00:00:10	01:12:07	00:00:11	01:48:47	00:00:11
40.000	00:11:51	00:00:19	01:49:07	00:00:19	03:10:51	00:00:20
50.000	00:16:19	00:00:29	03:01:56	00:00:30	05:06:19	00:00:30
60.000	00:25:57	00:00:43	04:19:40	00:00:43	07:21:46	00:00:43
70.000	00:35:40	00:01:00	05:34:08	00:01:00	10:01:19	00:01:00
80.000	00:43:07	00:01:15	06:42:54	00:01:18	12:03:54	00:01:19
90.000	00:52:50	00:01:39	08:23:17	00:01:39	16:35:00	00:01:40
100.000	01:04:37	00:02:00	10:19:33	00:02:01	20:34:44	00:02:01

Tabela 4.3: Testes com s e t geradas aleatoriamente, para $k = 30,\ 300$ e 600. A primeira coluna contém o comprimento das sequências s e t e as seis colunas a seguir contém o tempo de execução do FB e do KHD no formato hora:minuto:segundo.

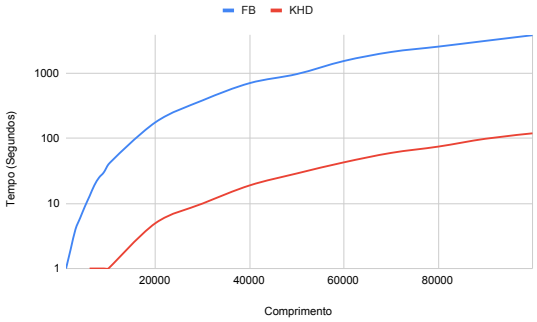

Figura 4.2: Gráfico para os valores de $k = 30$ da Tabela 4.3.

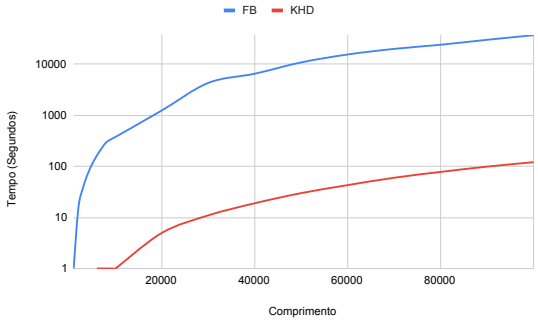

Figura 4.3: Gráfico para os valores de $k = 300$ da Tabela 4.3.

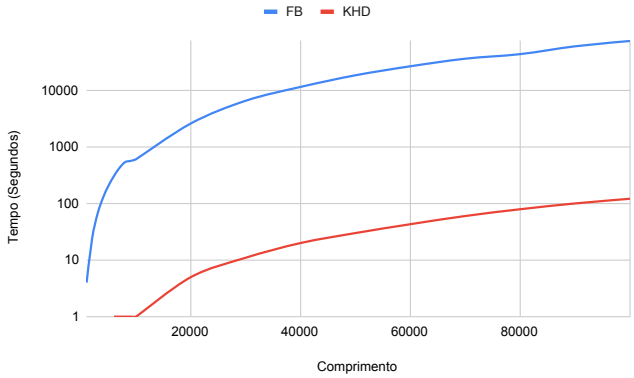

Figura 4.4: Gráfico para os valores de $k = 600$ da Tabela 4.3.

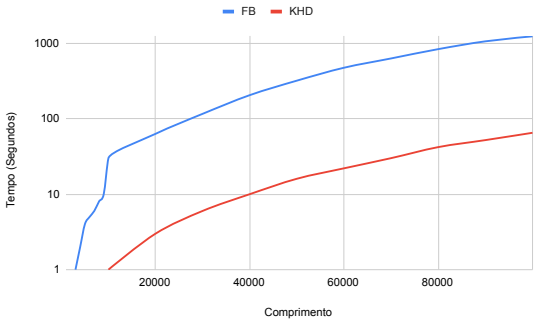

Figura 4.5: Gráfico para os valores de $k = 30$ da Tabela 4.4.

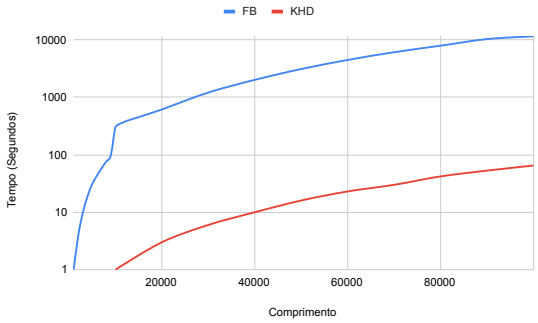

Figura 4.6: Gráfico para os valores de $k = 300$ da Tabela 4.4.

	30		300		600	
Comprimento	FB	KHD	FB	KHD	FB	KHD
1000	00:00:00	00:00:00	00:00:01	00:00:00	00:00:01	00:00:00
2000	00:00:00	00:00:00	00:00:04	00:00:00	00:00:07	00:00:00
3000	00:00:01	00:00:00	00:00:10	00:00:00	00:00:18	00:00:00
4000	00:00:02	00:00:00	00:00:19	00:00:00	00:00:34	00:00:00
5000	00:00:04	00:00:00	00:00:31	00:00:00	00:00:55	00:00:00
6000	00:00:05	00:00:00	00:00:43	00:00:00	00:01:21	00:00:00
7000	00:00:06	00:00:00	00:00:58	00:00:00	00:01:50	00:00:00
8000	00:00:08	00:00:00	00:01:16	00:00:00	00:02:25	00:00:00
9000	00:00:10	00:00:00	00:01:38	00:00:00	00:03:09	00:00:00
10.000	00:00:30	00:00:01	00:05:02	00:00:01	00:10:05	00:00:01
20.000	00:01:03	00:00:03	00:10:14	00:00:03	00:21:01	00:00:03
30.000	00:01:56	00:00:06	00:20:12	00:00:06	00:41:02	00:00:06
40.000	00:03:25	00:00:10	00:33:35	00:00:10	01:05:14	00:00:10
50.000	00:05:19	00:00:16	00:51:49	00:00:16	01:40:59	00:00:17
60.000	00:07:53	00:00:22	01:14:29	00:00:23	02:23:03	00:00:23
70.000	00:10:28	00:00:30	01:41:41	00:00:30	03:13:02	00:00:31
80.000	00:13:54	00:00:42	02:12:03	00:00:42	04:11:03	00:00:42
90.000	00:17:38	00:00:52	02:52:03	00:00:53	05:18:09	00:00:53
100.000	00:20:32	00:01:05	03:12:05	00:01:05	06:33:34	00:01:05

Tabela 4.4: Testes com s e t completamente diferentes, para $k = 30,\ 300$ e 600. A primeira coluna contém o comprimento das sequências s e t e as seis colunas a seguir contém o tempo de execução do FB e do KHD no formato hora:minuto:segundo.

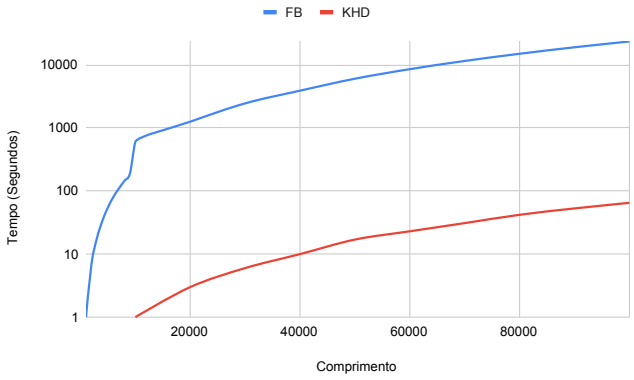

Figura 4.7: Gráfico para os valores de $k = 600$ da Tabela 4.4.

	30		300		600	
Comprimento	FB	KHD	FB	KHD	FB	KHD
1000	00:00:02	00:00:00	00:00:02	00:00:00	00:00:02	00:00:00
2000	00:00:15	00:00:00	00:00:15	00:00:00	00:00:15	00:00:00
3000	00:00:49	00:00:00	00:00:49	00:00:00	00:00:49	00:00:00
4000	00:02:01	00:00:00	00:02:01	00:00:00	00:02:01	00:00:00
5000	00:03:00	00:00:00	00:03:00	00:00:00	00:03:00	00:00:00
6000	00:04:02	00:00:00	00:04:02	00:00:00	00:04:02	00:00:00
7000	00:07:10	00:00:00	00:07:10	00:00:00	00:07:10	00:00:00
8000	00:10:39	00:00:00	00:10:39	00:00:00	00:10:39	00:00:00
9000	00:15:10	00:00:00	00:15:10	00:00:00	00:15:10	00:00:00
10.000	00:24:23	00:00:01	00:24:23	00:00:01	00:24:23	00:00:01
20.000	03:19:12	00:00:03	03:19:12	00:00:03	03:19:12	00:00:03
30.000	11:40:36	00:00:05	11:40:36	00:00:05	11:40:36	00:00:05
40.000	25:01:44	00:00:11	25:01:44	00:00:11	25:01:44	00:00:11
50.000	50:02:20	00:00:17	50:02:20	00:00:17	50:02:20	00:00:17
60.000	83:20:05	00:00:21	83:20:05	00:00:21	83:20:05	00:00:21
70.000	124:23:00	00:00:30	124:23:00	00:00:30	124:23:00	00:00:30
80.000	174:04:15	00:00:38	174:04:15	00:00:38	174:04:15	00:00:38
90.000	233:23:04	00:00:49	233:23:04	00:00:49	233:23:04	00:00:49
100.000	299:03:00	00:01:06	299:03:00	00:01:06	299:03:00	00:01:06

Tabela 4.5: Testes com s e t iguais, para $k = 30,\ 300$ e 600. A primeira coluna contém o comprimento das sequências s e t e as seis colunas a seguir contém o tempo de execução do FB e do KHD no formato hora:minuto:segundo. A tabela contém tempos idênticos para qualquer valor de k e isso se aplica pelo fato das sequências serem totalmente iguais, ou seja, tanto o FB quanto o KHD vão sempre fazer a mesma verificação independente do k escolhido.

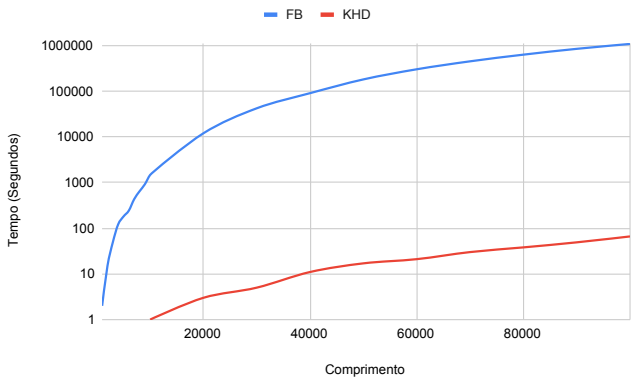

Figura 4.8: Gráfico para os valores da Tabela 4.5. Um único gráfico é capaz de representar a Tabela 4.5 porque o tempo de execução dos algoritmos são os mesmos independente do valor de k.

4.2.2 Sequências Reais

Nesta seção, estamos interessados em verificar o tempo de execução do FB e do KHD em sequências reais. Para realizar este experimento, vamos utilizar sequências obtidas do projeto do Dobre [24]. As sequências utilizadas são dez sequências da espécie *Homo sapiens*, com diversos comprimentos, sendo elas: *hsarhgdig* (4398 caracteres), *hsascl2* (4455 caracteres), *hsankrd43* (5457 caracteres), *hsa1bg* (8694 caracteres), *hsalg10b* (14.972 caracteres), *hsbad* (16.877 caracteres), *hsankr10* (38.530 caracteres), *hsascc2* (51.655 caracteres), *hsasz1* (66.302 caracteres) e *hsaff4* (90.284 caracteres).

Os nossos experimentos foram divididos em três diferentes experimentos. Primeiro, a sequência s é a sequência *hsarhgdig* e as sequências t são todas as outras sequências. Segundo, a sequência s é a sequência *hsankr10* e as sequências t são todas as outras sequências. Terceiro, a sequência s é a sequência *hsaff4* e as sequências t são todas as outras sequências. As sequências *hsarhgdig*, *hsankr10* e *hsaff4* foram escolhidas como sequência s para representarem a sequência de menor comprimento, comprimento médio e maior comprimento. Cada experimento contém uma sequência s e cada uma das 9 sequências restantes t. Os experimentos são feitos variando o valor de k em 30, 300 e 600 conforme a Tabela 4.6.

Para os experimentos, os resultados são apresentados na Tabela 4.6. Assim como nos resultados apresentados para as sequências artificiais, aqui nesses resultados o comportamento relativo dos algoritmos são similares, ou seja, KHD é mais rápido. Apesar dos excelentes resultados para o KHD, precisamos lembrar que estes experimentos representam o SSP_s, então os testes foram para duas sequências. Para dois conjuntos, mesmo o KHD sendo quadrático e mais rápido que o FB, não resolve o problema em um tempo viável. Por exemplo, tome dois conjuntos A e B de sequências com 1TB, o KHD realiza 10^{12}x10^{12} comparações. Considerando que uma máquina é capaz de realizar 10^8 comparações por segundo, o algoritmo levaria 10^{16} segundos para finalizar, ou seja, séculos. Portanto, o tempo gasto pelo KHD é impraticável.

Para os resultados apresentados na Tabela 4.6, apresentamos nas figuras 4.9, 4.10 e 4.11 o gráfico de tempo de execução dos programas FB e KHD para a sequência *hsarhgdig* para $k = 30$, 300 e 600, respectivamente. Apresentamos nas figuras 4.12, 4.13 e 4.14 o gráfico de tempo de execução dos programas FB e KHD para a sequência *hsank10* para $k = 30$, 300 e 600, respectivamente. Finalmente, para a sequência *hsaff4*, apresentamos nas figuras 4.15, 4.16 e 4.17 o gráfico de tempo de execução dos programas FB e KHD para $k = 30$, 300 e 600, respectivamente. Apresentamos os gráficos com o tempo em segundos e em escala logarítmica.

s		t		30		300		600	
Nome	Com.	Nome	Com.	FB	KHD	FB	KHD	FB	KHD
hsarhgdig	4398	*hsascl2*	4455	00:00:09	00:00:01	00:01:16	00:00:01	00:02:18	00:00:01
hsarhgdig	4398	*hsankrd43*	5457	00:00:09	00:00:01	00:01:24	00:00:01	00:02:34	00:00:01
hsarhgdig	4398	*hsa1bg*	8694	00:00:16	00:00:00	00:02:19	00:00:00	00:04:17	00:00:01
hsarhgdig	4398	*hsalg10b*	14.972	00:00:24	00:00:00	00:03:37	00:00:00	00:06:47	00:00:01
hsarhgdig	4398	*hsbad*	16.877	00:00:30	00:00:01	00:04:30	00:00:01	00:08:25	00:00:01
hsarhgdig	4398	*hsankr10*	38.530	00:01:00	00:00:02	00:09:36	00:00:02	00:18:11	00:00:02
hsarhgdig	4398	*hsascc2*	51.655	00:01:28	00:00:03	00:13:33	00:00:03	00:25:29	00:00:03
hsarhgdig	4398	*hsasz1*	66.302	00:01:45	00:00:03	00:16:07	00:00:04	00:30:29	00:00:04
hsarhgdig	4398	*hsaff4*	90.284	00:02:27	00:00:04	00:22:44	00:00:04	00:42:52	00:00:05
hsankr10	38.530	*hsarhgdig*	4398	00:01:03	00:00:02	00:09:36	00:00:02	00:18:54	00:00:02
hsankr10	38.530	*hsascl2*	4455	00:01:04	00:00:02	00:10:07	00:00:02	00:19:10	00:00:02
hsankr10	38.530	*hsankrd43*	5457	00:01:22	00:00:03	00:11:32	00:00:03	00:22:51	00:00:03
hsankr10	38.530	*hsa1bg*	8694	00:02:11	00:00:04	00:09:36	00:00:05	00:18:53	00:00:05
hsankr10	38.530	*hsalg10b*	14972	00:03:52	00:00:08	00:37:17	00:00:08	01:13:22	00:00:08
hsankr10	38.530	*hsbad*	16.877	00:04:09	00:00:08	00:39:50	00:00:08	01:18:35	00:00:08
hsankr10	38.530	*hsascc2*	51.655	00:12:56	00:00:24	02:03:45	00:00:24	04:09:41	00:00:24
hsankr10	38.530	*hsasz1*	66.302	00:17:12	00:00:32	02:44:59	00:00:32	05:32:19	00:00:32
hsankr10	38.530	*hsaff4*	90.284	00:23:06	00:00:42	03:42:55	00:00:43	07:24:02	00:00:43
hsaff4	90.284	*hsarhgdig*	4398	00:02:32	00:00:05	00:23:05	00:00:05	00:55:52	00:00:05
hsaff4	90.284	*hsascl2*	4455	00:02:29	00:00:05	00:23:23	00:00:05	01:08:18	00:00:05
hsaff4	90.284	*hsankrd43*	5457	00:03:18	00:00:06	00:31:13	00:00:06	01:11:29	00:00:06
hsaff4	90.284	*hsa1bg*	8694	00:05:10	00:00:10	00:48:56	00:00:10	01:37:58	00:00:10
hsaff4	90.284	*hsalg10b*	14.972	00:09:28	00:00:17	01:29:42	00:00:17	03:20:31	00:00:17
hsaff4	90.284	*hsbad*	16.877	00:10:05	00:00:19	01:37:48	00:00:19	03:20:26	00:00:19
hsaff4	90.284	*hsankr10*	38.530	00:23:48	00:00:43	03:52:10	00:00:43	07:43:29	00:00:44
hsaff4	90.284	*hsascc2*	51.655	00:30:49	00:00:56	05:03:01	00:00:57	10:02:01	00:00:57
hsaff4	90.284	*hsasz1*	66.302	00:41:18	00:01:14	06:39:51	00:01:14	13:22:54	00:01:14

Tabela 4.6: Testes com s e t reais (*Homo Sapiens*), para $k = 30,\ 300$ e 600. A primeira e a segunda coluna contém o nome e o comprimento da sequência s. A terceira e a quarta coluna contém o nome e o comprimento da sequência t. As seis colunas a seguir contém o tempo de execução do FB e o KHD no formato hora:minuto:segundo.

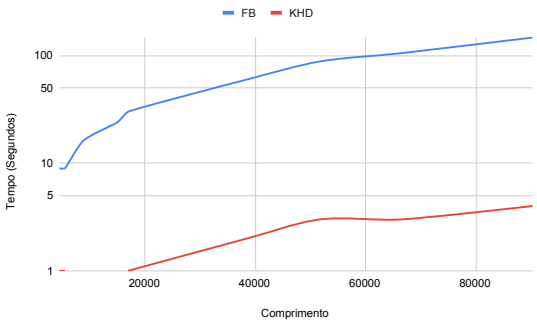

Figura 4.9: Gráfico para os valores da sequência *hsarhgdig* para $k = 30$ da Tabela 4.6.

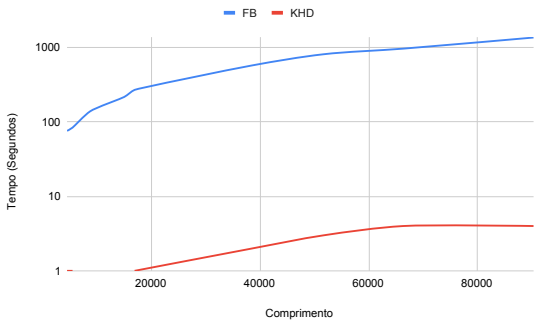

Figura 4.10: Gráfico para os valores da sequência *hsarhgdig* para $k = 300$ da Tabela 4.6.

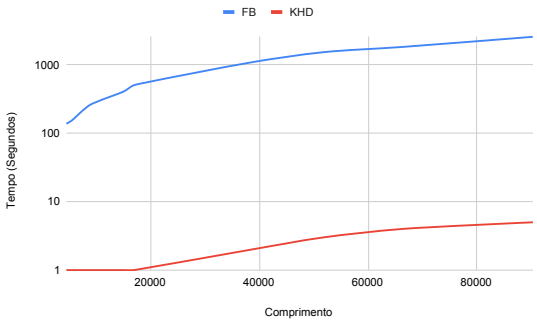

Figura 4.11: Gráfico para os valores da sequência *hsarhgdig* para $k = 600$ da Tabela 4.6.

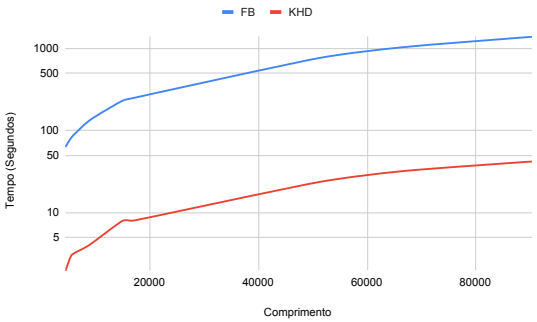

Figura 4.12: Gráfico para os valores da sequência *hsankr10* para $k = 30$ da Tabela 4.6.

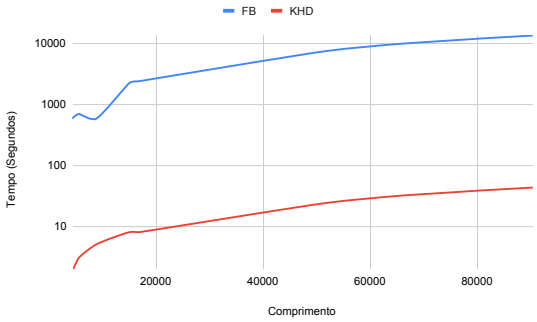

Figura 4.13: Gráfico para os valores da sequência *hsankr10* para $k = 300$ da Tabela 4.6.

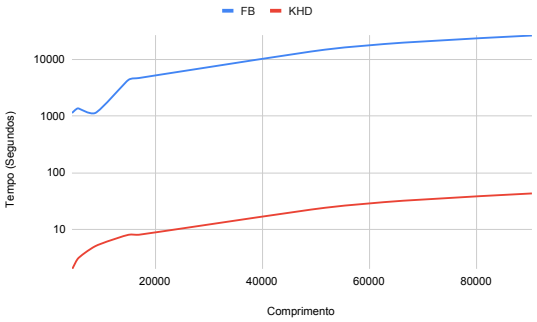

Figura 4.14: Gráfico para os valores da sequência *hsankr10* para $k = 600$ da Tabela 4.6.

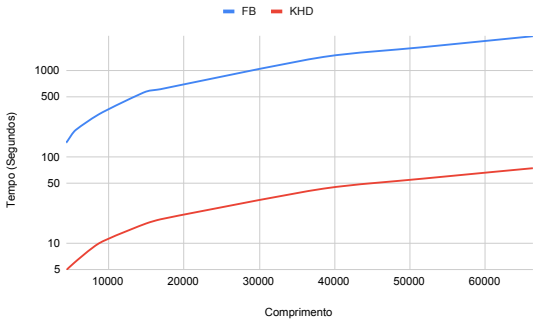

Figura 4.15: Gráfico para os valores da sequência *hsaff4* para $k = 30$ da Tabela 4.6.

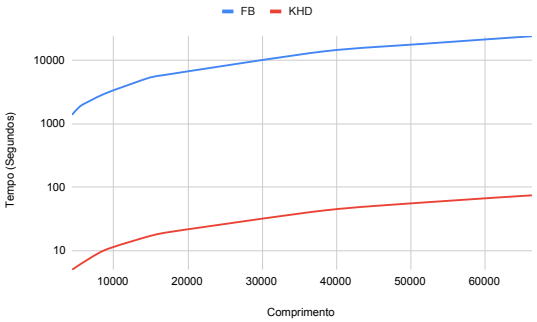

Figura 4.16: Gráfico para os valores da sequência *hsaff4* para $k = 300$ da Tabela 4.6.

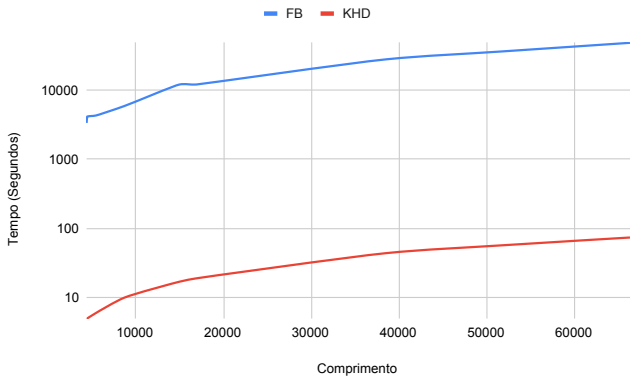

Figura 4.17: Gráfico para os valores da sequência *hsaff4* para $k = 600$ da Tabela 4.6.

4.3 Heurísticas

Para os testes desta Seção, vamos considerar o algoritmo do Maaß – KHD apresentado no Capítulo 2, as heurísticas apresentadas no Capítulo 3, e o SSP. Na Seção anterior avaliamos os algoritmos exatos, comprovamos como o KHD é mais rápido que o FB para o SSP_s. Para esta seção, estamos interessados em comparar o tempo de execução do KHD e das heurísticas utilizando dois conjuntos de sequências. Também estamos interessados em comparar os comprimentos inferidos pelas heurísticas com os comprimentos exatos e determinar a qualidade dos comprimentos inferidos. Na Seção 4.3.1 apresentamos como avaliamos a qualidade das respostas das heurísticas. Na Seção 4.3.2 apresentamos os resultados para as sequências da expedição Tara. Nas seções 4.3.3 e 4.3.4 apresentamos os resultados para as sequências de bactérias obtidas do NCBI. Finalmente, Na Seção 4.3.5, apresentamos os tempos de execução para a Heurística 2 e sua versão paralela.

4.3.1 Acurácia dos experimentos

Para determinar a qualidade dos comprimentos calculados pelas heurísticas, utilizamos o termo **acurácia dos experimentos**. Para isso, precisamos lembrar que o KHD devolve um vetor, digamos r, com os comprimentos exatos das subcadeias mínimas e as heurísticas também devolvem um vetor, digamos r', com os comprimentos deduzidos. Para avaliarmos a acurácia, vamos calcular a diferença $r[i] - r'[i]$ para cada posição i, com $i = 1, \ldots, n$. Sejam A e B os dois conjuntos de entrada, ou seja, queremos encontrar todas as subcadeias de A que possuem pelo menos k diferenças em relação a qualquer subcadeia em B. Para cada sequência $s \in A$, temos um vetor r e um vetor r', e vamos então

calcular a **média das diferenças** (com o valor absoluto) como segue:

$$\mu(s) = \frac{\sum_{i=1}^{n} |r[i] - r'[i]|}{n}.$$

Para avaliarmos a acurácia do experimento, vamos calcular a **média das médias das diferenças** que definimos como:

$$\Phi = \frac{\sum_{s \in A} \mu(s)}{|A|}.$$

Na Tabela 4.7 apresentamos um exemplo hipotético assumindo que executamos o KHD e o H2 para um conjunto A com duas sequências e um conjunto B. Relembrando, temos um vetor r e r' para cada sequência do conjunto A.

Sequência	Posições	i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
1	KHD	r	4	4	3	3	2	5	6	6	5	4	4	3	2	-1	
	H2	r'	4	4	3	3	4	5	6	5	5	4	3	3	2	-1	μ
	Diferença	$r[i] - r'[i]$	0	0	0	0	2	0	0	1	0	0	1	0	0	0	0,29
2	KHD	r	4	4	5	5	4	4	4	4	3	2	4	2	-1	-1	
	H2	r'	4	4	4	5	4	4	4	3	2	2	2	2	-1	-1	μ
	Diferença	$r[i] - r'[i]$	0	0	1	0	0	0	0	1	1	0	2	0	0	0	0,36
																Φ	**0,32**

Tabela 4.7: Exemplo de avaliação da acurácia dos experimentos considerando duas sequências s no conjunto A, com $|s| = 14$. Os valores nesta tabela são dados ilustrativos. A primeira coluna contém a identificação da sequência. A segunda e terceira colunas temos rótulos para as próximas colunas. Entre as colunas quatro e dezessete temos quatorze posições, para a linha **KHD** são os comprimentos exatos do vetor r, para a linha **H2** são os comprimentos deduzidos do vetor r', para a linha **diferença**, são as diferenças (valor absoluto) entre os comprimentos do KHD e H2. Para a coluna μ, temos a média das diferenças. Para a linha Φ, temos a média das médias das diferenças. Para H2 considerando essas duas sequências, sua acurácia é de $0,32$.

4.3.2 Tara

Nesta seção, estamos interessados em avaliar sequências dos metagenomas da expedição Tara. A expedição Tara pertence a fundação Tara *Oceans*, eles realizaram uma expedição oceânica com o propósito de sequenciar o metagenoma de mais de 35.000 amostras diferentes de 210 regiões diferentes do mundo, resultando em um total de 7,2TB de dados genômicos [7, 8].

Para o nosso experimento, vamos considerar dois metagenomas da expedição Tara, sendo eles: ERR599000 que denominamos de conjunto A e ERR599040 que denominamos de conjunto B. Cada metagenoma contém milhares de sequências. Como os metagenomas possuem muitas sequências, vamos considerar um amostra reduzida com apenas 1, 10 e 100 sequências

no conjunto A e 1 milhão de sequências no conjunto B e $k = 20$ conforme a Tabela 4.8 para os tempos de execução e a Tabela 4.9 para a acurácia do experimento. Todas as sequências usadas neste experimento possuem comprimento de aproximadamente 100.

Para os resultados apresentados na Tabela 4.8, vamos analisar as seguintes informações: primeiro, podemos notar que utilizar o FB ou o KHD nas posições exatas não faz diferença, o tempo de execução é muito similar. Isso ocorre pelo fato de que o KHD é um excelente algoritmo (em relação ao FB) ao ser executado quando queremos encontrar todas as subcadeias mínimas. Devemos lembrar que o KHD faz reaproveitamento de comprimentos, e se estamos calculando para um comprimento em específico, ele se comporta igualmente ao FB quando buscamos exatamente um comprimento. Segundo, entre as heurísticas e o algoritmo exato KHD, tem uma diminuição no tempo, mas ela não é tão satisfatória. Por exemplo, para H1 utilizando o KHD, a heurística foi 1,4 vezes mais rápida que o KHD. Isso ocorre porque as sequências possuem um comprimento pequeno, e a quantidade de posições exatas não foi suficiente para conseguirmos uma boa melhoria no tempo, em outras palavras, estamos executando muitas vezes o KHD nas posições exatas.

Para os resultados apresentado na Tabela 4.9, vamos analisar as seguintes informações: primeiro, os valores Φ (média das médias das diferenças), quanto mais próximo de zero Φ estiver, menos as heurísticas estão errando. Podemos notar que H2 erra menos que H1. Segundo, as outras colunas representa em porcentagem quanto os comprimentos das subcadeias inferidas pelas heurísticas diferem dos comprimentos das subcadeias minimais calculadas por algoritmos exatos. Podemos notar que para 100 sequências, 80% dos comprimentos inferidos diferem do comprimento calculado pelo algoritmo exato em até 4 unidades, ou seja, diferem pouco.

Referente aos resultados apresentados na Tabela 4.8, considerando 100 sequências, apresentamos na Figura 4.18 o gráfico do tempo de execução e apresentamos na Figura 4.19 as quantidades para cada comprimento calculados/inferidos pelos programas KHD, H1 e H2. Utilizamos este último gráfico para mostrar como a quantidade de comprimentos calculados e inferidos estão distribuídos.

4.3.3 Bactérias

Nesta seção, estamos interessados em aumentar o comprimento das sequências para avaliarmos melhor a performance das heurísticas em relação ao KHD. Para o nosso experimento, vamos considerar sequências de bactérias de vários comprimentos. As sequências foram obtidas da plataforma NCBI e os experimentos foram divididos em três diferentes experimentos de acordo com

	Exato	FB		KHD	
Quantidade	KHD	H1	H2	H1	H2
1	00:04:23	00:03:36	00:03:00	00:03:25	00:03:03
10	00:36:41	00:31:36	00:31:00	00:30:42	00:29:48
100	05:56:49	04:23:21	04:22:29	04:12:17	04:10:23

Tabela 4.8: Teste para sequências dos metagenomas da expedição Tara. A primeira coluna contém a quantidade de sequências no conjunto A (O conjunto B contém 1-milhão de sequências). A segunda coluna contém o tempo de execução do KHD. A terceira e quarta coluna contém o tempo de execução das heurísticas utilizando o algoritmo FB para as posições exatas, e a quinta e sexta coluna, o tempo de execução das heurísticas utilizando o algoritmo KHD. O formato do tempo é hora:minuto:segundo. Os testes consideram $k = 20$.

	Φ	%				Φ	%			
Quantidade	H1	0	1-4	5-8	>9	H2	0	1-4	5-8	>9
1	**3,4**	13,7	51,0	29,4	5,9	**3,3**	31,4	62,7	0	5,9
10	**2,3**	20,8	63,1	10,8	5,3	**1,3**	30,0	64,5	0,4	5,1
100	**2,8**	21,0	64,3	9,2	5,5	**1,6**	33,6	60,6	1,1	4,7

Tabela 4.9: Acurácia para os resultados da Tabela 4.8. A primeira coluna contém a quantidade de sequências do conjunto A. As últimas dez colunas contém o valor Φ (média das médias das diferenças) entre as respostas do KHD e das heurísticas H1 e H2 (fonte em negrito) e a porcentagem das respostas das heurísticas que diferem em algumas unidade do exato (fonte sem negrito). Por exemplo, para 100 sequências, H2 está errando na média em 1,6, então muitos comprimentos estão próximos do algoritmo exato, podemos ver isso na colunas $1-4$ onde 60,6% dos comprimentos diferem em 1, 2, 3 ou 4 unidades.

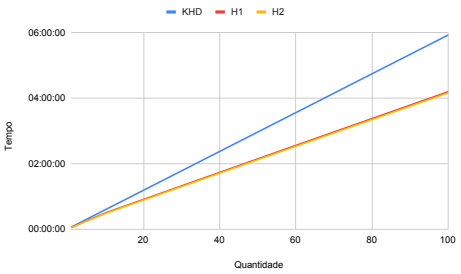

Figura 4.18: Gráfico para os valores da Tabela 4.8 para a quantidade de 100 sequências considerando o teste com o KHD.

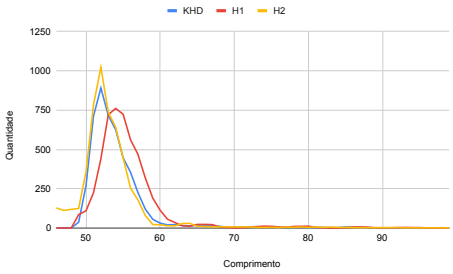

Figura 4.19: Gráfico para a Tabela 4.9 para o teste com 100 sequências, representando a quantidade para cada comprimento calculado/inferido pelos programas KHD, H1 e H2.

o comprimento das sequências. O primeiro experimento contém sequências com comprimento entre 100 e 200, o segundo experimento contém sequências com comprimentos entre 800 e 1000, e finalmente, o terceiro experimento contém sequências com comprimentos entre 8000 e 10.000. Para todos os experimentos, vamos considerar 1, 10 e 100 sequências no conjunto A e 1-milhão de sequências no conjunto B especificamente para o primeiro e o segundo experimento, para o terceiro experimento, o conjunto B contém 10.000 sequências e $k = 20$ conforme a Tabela 4.11 para os tempos de execução e Tabela 4.11 para a acurácia dos experimentos.

Para os resultados de tempo de execução apresentados na Tabela 4.11, podemos notar que as heurísticas dependem do comprimento das sequências para executarem mais rápido do que o KHD. Para o primeiro experimento, os tempos não são tão bons, mas são melhores do que tempos apresentados na seção anterior. Para 100 sequências, H1 e H2 foram 1,6 vezes mais rápido do que KHD. Conforme aumentamos o comprimento das sequências no segundo experimento, temos um grande ganho, onde H1 foi 5,85 e H2 5,9 vezes mais rápido do que o KHD. Para o terceiro experimento, H1 foi 41,8 e H2 foi 44,62 vezes mais rápido do que o KHD.

Para os resultados de acurácia apresentados na Tabela 4.11, podemos notar que conforme aumentamos o comprimento das sequências nos experimentos, maiores são os valores Φ, ou seja, as heurísticas estão errando mais. Isso ocorre porque as posições exatas estão ficando cada vez mais distantes, então temos mais posições deduzidas. H1 é a heurística que mais tem errado. Para H2, considerando 100 sequências, podemos notar para o primeiro e o segundo experimento que 75% dos comprimentos inferidos estão variando em até 4 unidades e 50% para o terceiro experimento.

Referente aos resultados apresentados na Tabela 4.10 para o primeiro experimento, apresentamos na Figura 4.20 para o teste com 100 sequências o gráfico do tempo de execução e na Figura 4.21 apresentamos as quantidades para cada comprimento encontrado pelos programas KHD, H1 e H2. Apresentamos o mesmo tipo de gráfico para segundo e o terceiro experimento da Tabela 4.10. A Figura 4.22 contém o tempo de execução e na Figura 4.23 as quantidades para os comprimentos referente ao segundo experimento. Para o terceiro experimento, as Figuras 4.24 e 4.25 representam o tempo de execução e as quantidades de comprimentos, respectivamente.

4.3.4 Bactérias da mesma família

Nesta seção estamos interessados em avaliar se as heurísticas são capazes de identificar quando uma sequência não possui respostas, ou seja, não existe resposta do SSP para nenhuma posição de uma determinada sequência. Para

	Exato	FB		KHD	
Quantidade	KHD	H1	H2	H1	H2
1	00:04:57	00:03:48	00:03:40	00:02:31	00:02:28
10	00:46:15	00:29:18	00:28:54	00:28:05	00:28:40
100	07:33:05	04:43:21	04:39:02	04:41:48	04:31:32
Quantidade	KHD	H1	H2	H1	H2
1	03:22:45	00:43:05	00:44:04	00:36:18	00:35:54
10	36:03:05	07:37:58	07:19:06	07:10:43	07:01:31
100	418:59:43	74:57:55	73:18:39	71:31:22	70:10:41
Quantidade	KHD	H1	H2	H1	H2
1	02:50:16	07:55:12	00:06:53	00:07:25	00:05:26
10	32:41:31	01:13:00	01:08:01	01:03:02	00:58:41
100	386:00:26	09:45:24	09:29:20	09:13:14	08:39:49

Tabela 4.10: Teste para sequências de bactérias obtidas do NCBI. A primeira coluna contém a quantidade de sequências no conjunto A (O conjunto B contém 1-milhão de sequências). A segunda coluna contém o tempo de execução do KHD. A terceira e quarta coluna contém o tempo de execução das heurísticas utilizando o algoritmo FB para as posições exatas, e a quinta e sexta coluna, o tempo de execução das heurísticas utilizando o algoritmo KHD. Os testes são divididos em três casos. O primeiro caso contém sequências de comprimento entre 100 e 200, a segundo, sequências com comprimento entre 800 e 1000. O terceiro caso, mantém 1, 10 e 100 sequências em A, mas o conjunto B tem 10.000 sequências, ambos os conjuntos contém sequências com o comprimento entre 8000 e 10.000. O formato do tempo é hora:minuto:segundo. Os testes consideram $k = 20$.

	Φ	%				Φ	%			
Quantidade	H1	0	1-4	5-8	>9	H2	0	1-4	5-8	>9
1	**11,2**	6,0	26,3	22,6	45,1	**3,8**	62,4	24,8	1,5	11,3
10	**8,8**	18,0	40,7	14,1	27,2	**8,2**	42,0	41,6	5,0	11,4
100	**7,9**	17,5	50,4	11,5	20,6	**5,5**	37,9	47,5	3,6	11,0
Quantidade	H1	0	1-4	5-8	≥ 9	H2	0	1-4	5-8	≥ 9
1	**8,3**	6,6	35,8	14,1	43,5	**6,3**	10,2	48,8	13,8	27,2
10	**24,2**	25,6	20,4	8,5	45,5	**13,8**	41,8	32,8	6,3	19,1
100	**24,9**	28,4	24,5	9,7	37,4	**19,3**	44,4	31,8	6,9	16,9
Quantidade	H1	0	1-4	5-8	≥ 9	H2	0	1-4	5-8	≥ 9
1	**31,9**	4,2	24,0	12,0	59,8	**34,5**	5,1	28,7	13,5	52,7
10	**266,1**	8,7	13,4	4,2	73,7	**161,3**	26,9	15,9	4,9	52,3
100	**264,7**	7,7	16,3	5,7	70,3	**155,2**	29,3	21,0	5,0	44,7

Tabela 4.11: Acurácia para os resultados da Tabela 4.10. A primeira coluna contém a quantidade de sequências do conjunto A. As últimas dez colunas contém o valor Φ (média das médias das diferenças) entre as respostas do KHD e das heurísticas H1 e H2 (fonte em negrito) e a porcentagem das respostas das heurísticas que diferem em algumas unidade do exato (fonte sem negrito).

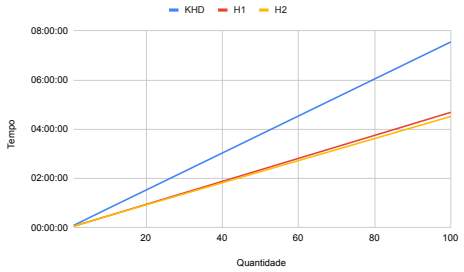

Figura 4.20: Gráfico para os valores da Tabela 4.10 para a quantidade de 100 sequências para o primeiro experimento considerando o teste com o KHD.

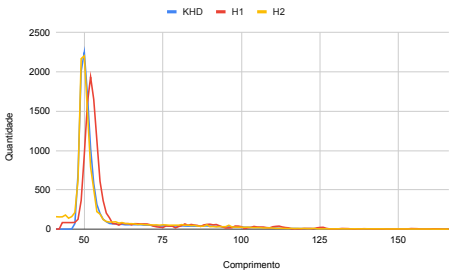

Figura 4.21: Gráfico para a Tabela 4.11 para o teste com 100 sequências, representando as quantidades para cada comprimento calculado/inferido pelos programas KHD, H1 e H2 para o primeiro experimento.

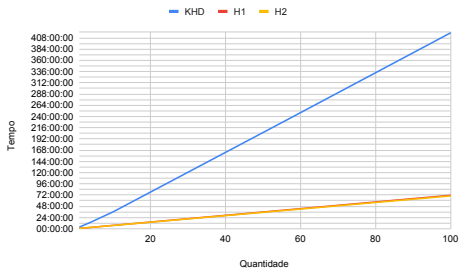

Figura 4.22: Gráfico para os valores da Tabela 4.10 para a quantidade de 100 sequências para o segundo experimento considerando o teste com o KHD.

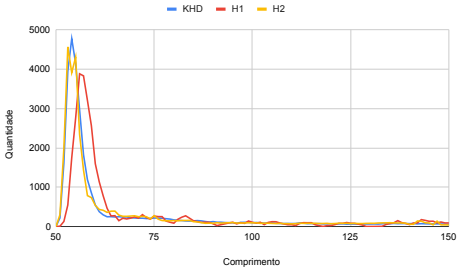

Figura 4.23: Gráfico para a Tabela 4.11 para o teste com 100 sequências, representando as quantidades para cada comprimento calculado/inferido pelos programas KHD, H1 e H2 para o segundo experimento.

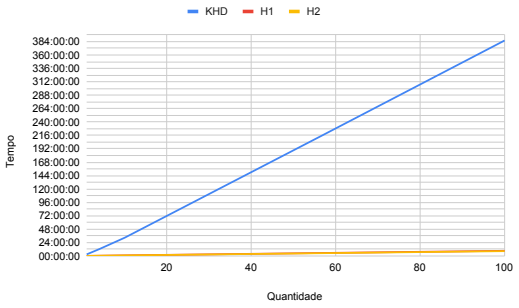

Figura 4.24: Gráfico para os valores da Tabela 4.10 para a quantidade de 100 sequências para o terceiro experimento considerando o teste com o KHD.

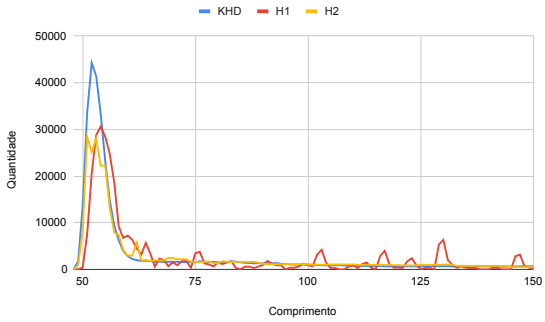

Figura 4.25: Gráfico para a Tabela 4.11 para o teste com 100 sequências, representando as quantidades para cada comprimento calculado/inferido pelos programas KHD, H1 e H2 para o terceiro experimento.

o nosso experimento, vamos considerar as sequências de comprimento entre 800 e 1000 do segundo experimento da Tabela 4.10.

Neste experimento, vamos inserir tanto no conjunto A quanto no conjunto B sequências da mesma família de bactéria. Como as sequências vão estar mais similares entre si, vamos ter várias sequências que não vão ter k diferenças, em outras palavras, quando procurarmos por subcadeias com pelo menos k diferenças, o algoritmo encontrará longos trechos idênticos entre os pares de sequências porque há sequências da mesma família nos dois conjuntos, elas vão compartilhar nucleotídeos similares. Para esse experimento nós omitimos a tabela de tempo porque estamos interessados na acurácia, nós calculamos a acurácia e apresentamos na Tabela 4.12.

Os resultados de acurácia apresentados na Tabela 4.12. Para o primeiro experimento com 1 sequência, KHD, H1 e H2 a identificaram sem respostas. Para o segundo experimento com 10 sequências, os algoritmos identificaram 6 sequências sem respostas, e finalmente, para o último experimento com 100 sequências, os algoritmos identificaram 72 sequências sem respostas. As heurísticas acertaram bastante nesses experimentos, então as médias mostradas na Tabela 4.12 são menores em comparação com as médias para o segundo experimento apresentado na Tabela 4.11.

	Φ	%				Φ	%			
Quantidade	H1	0	1-4	5-8	>9	H2	0	1-4	5-8	>9
1	**2,3**	100,0	0	0	0	**1,6**	100,0	0	0	0
10	**10,3**	69,7	12,8	3,3	14,2	**11,4**	74,8	12,8	3,3	9,1
100	**5,6**	79,4	8,2	2,3	10,1	**5,5**	82,9	9,8	2,0	5,3

Tabela 4.12: Acurácia para os testes com bactérias da mesma família. A primeira coluna contém a quantidade de sequências do conjunto A. As últimas dez colunas contém o valor Φ (média das médias das diferenças) entre as respostas do KHD e das heurísticas H1 e H2 (fonte em negrito) e a porcentagem das respostas das heurísticas que diferem em algumas unidade do exato (fonte sem negrito).

4.3.5 Paralelismo

Nesta seção estamos interessados em avaliarmos o tempo de execução das heurísticas e a versão paralela. Como ambas as heurísticas têm abordagens paralelas semelhantes (ver Seção 3.5) e tempo de execução próximo, escolhemos apenas a H2 para calcular a aceleração da abordagem paralela. Para os experimentos paralelos, vamos considerar a Heurística 2 que utiliza o KHD como algoritmo exato. Os resultados da H2 paralela, chamada **H2-P**, são apresentados na Tabela 4.13.

Para os resultados apresentados na Tabela 4.13, podemos notar que conforme aumentamos o números de processadores, diminuímos o tempo de exe-

cução da H2-P. Para 20 processadores, o *speedup* é de 16,7 para o nosso experimento com 100 sequências. Apesar de simples, esse teste no revela que as heurísticas são escaláveis e se tivermos um número razoável de processadores, teremos um grande ganho no tempo.

Para cada experimento apresentado na Tabela 4.13, mostramos os seus respectivos gráficos. Na Figura 4.26 mostramos o gráfico de tempo de execução para o número de processadores utilizados para o primeiro experimento. Para os outros três experimentos, apresentamos o mesmo tipo de gráfico nas figuras 4.27, 4.28 e 4.29, respectivamente.

Tara	H2	H2-P			
Processador(es)	1	2	5	10	20
10	00:29:48	00:15:00	00:06:00	00:03:00	00:02:00
100	04:10:23	02:24:00	01:04:00	00:30:00	00:17:00

NCBI 1	H2	H2-P			
Processador(es)	1	2	5	10	20
10	00:28:40	00:16:00	00:07:00	00:04:00	00:02:00
100	04:31:32	02:36:00	01:03:00	00:37:00	00:17:00

NCBI 2	H2	H2-P			
Processador(es)	1	2	5	10	20
10	07:01:31	03:39:00	01:45:00	00:45:00	00:24:00
100	70:10:41	36:31:00	15:41:00	07:20:00	03:41:00

NCBI 3	H2	H2-P			
Processador(es)	1	2	5	10	20
10	00:58:41	00:029:00	00:12:00	00:07:00	00:04:00
100	08:39:49	04:47:00	02:08:00	01:02:00	00:30:00

Speedup	H2	H2-P			
Processador(es)	1	2	5	10	20
10	-	1.901	4.407	8.368	15.078
100	-	1.801	4.185	8.400	16.726

Tabela 4.13: Resumo do tempo de execução da heurística H2-P para as sequências das seções 4.3.2 e 4.3.3 nas quatro primeiras tabelas. A primeira coluna contém a quantidade de sequências do conjunto A (o conjunto B contém 1-milhão de sequências). A segunda coluna o tempo de execução da Heurística 2. As próximas quatro colunas contém o tempo de execução da Heurística 2 na sua versão paralela. O formato do tempo é dia-hora:minuto e o $K = 20$. A última tabela contém o *Speedup*. A primeira coluna contém a quantidade de sequências do conjunto A (o conjunto B contém 1-milhão de sequências). A segunda coluna não contém nenhum valor. As quatro próximas colunas contém o *speedup* médio para cada coluna do processador.

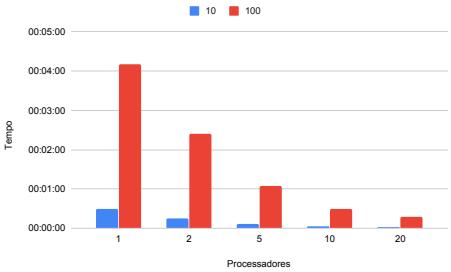

Figura 4.26: Gráfico para os valores da Tabela 4.13 para o primeiro caso, para 10 e 100 sequências.

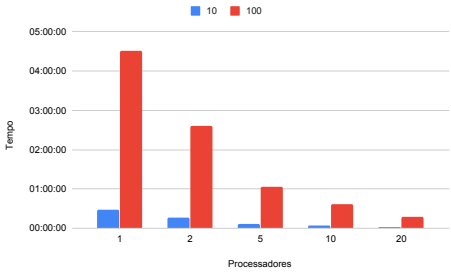

Figura 4.27: Gráfico para os valores da Tabela 4.13 para o segundo caso, para 10 e 100 sequências.

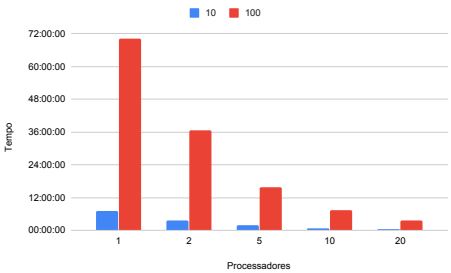

Figura 4.28: Gráfico para os valores da Tabela 4.13 para o terceiro caso, para 10 e 100 sequências.

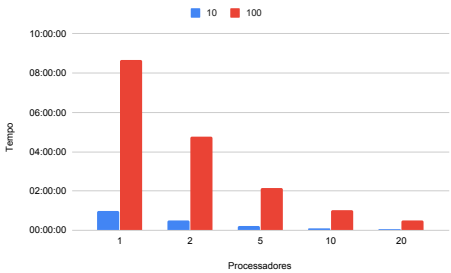

Figura 4.29: Gráfico para os valores da Tabela 4.13 para o quarto caso, para 10 e 100 sequências.

CAPÍTULO

5

Discussão, Perspectivas e Conclusões

Essa dissertação é um texto referente a todo o trabalho de pesquisa desenvolvido durante o programa de mestrado em ciência da computação da UFMS. Ele descreve o problema estudado e os principais resultados encontrados na literatura, assim como resultados por nós obtidos que foram publicados e apresentados em *Brazilian Symposuin on Bioinformatics* – BSB [16] e *International Conference on Bioinformatics and Bioenginerring* – BIBE [17], eventos qualis B3 e B1, respectivamente. Análise e testes práticos também foram feitos.

Neste trabalho abordamos o problema da Seleção das Subcadeias Específicas – SSP e sua versão simplificada – SSP_s. Após as pesquisas na literatura e a compreensão do problema, implementamos dois algoritmos exatos. O primeiro é um algoritmo de força bruta denominado neste trabalho de FB, com a complexidade de tempo cúbica. O segundo algoritmo é baseado no algoritmo do Maaß [14] denominado neste trabalho de KHD, com a complexidade de tempo quadrática.

Realizamos testes com sequências artificiais e reais para avaliar o comportamento do FB e do KHD. Conforme esperado o KHD foi mais rápido que o FB. O tempo gasto por FB no pior caso é $O(n^3)$ enquanto que no melhor caso gasta tempo $O(k \cdot n^2)$. Já o KDH sempre gasta tempo $O(n^2)$ independentemente do caso. Embora não fizemos uma análise de complexidade de tempo do caso médio, sabemos que o KHD realiza n^2 comparações no pior e no melhor caso, por esse motivo o KHD gasta tempo quadrático independente da entrada e é mais rápido que o algoritmo de força bruta.

Duas heurísticas nomeadas de Heurística 1 – H1 e Heurística 2 – H2 foram propostas, implementadas e testadas. Observamos que as heurísticas propos-

tas são relativamente mais rápidas em comparação ao KHD. No entanto, para instâncias maiores, com inúmeras sequências, as heurísticas também poderiam levar muito tempo de processamento. O comprimento das sequências influenciam no tempo de execução e na qualidade das respostas das heurísticas. De modo geral, quanto menor são os comprimentos das sequências de entrada, mais rápidas são as heurísticas em comparação aos algoritmos exatos. Além disso, quanto menor é o comprimento da sequência, mais próximos são os comprimentos aproximados das subcadeias minimais calculadas para as posições induzidas pelas heurísticas em relação aos comprimentos exatos das subcadeias minimais calculadas por algoritmos exatos sugerindo que as heurísticas podem ter bons resultados se os comprimentos das sequências não são grandes. Tratando-se da qualidade das respostas, H2 possui uma qualidade melhor (acurácia). Podemos ver nos experimentos com as heurísticas nas Tabelas 4.9 e 4.11 que H2 possui o valor Φ (média das médias das diferenças) menor do que H1. Essa média representa que os comprimentos inferidos por H2 estão mais próximos dos comprimentos exatos.

Os testes realizados com as heurísticas foram divididos com base no comprimento das sequências. Na Tabela 4.9 e no primeiro experimento da Tabela 4.11, os comprimentos não passam de 200, e neste cenário, as heurísticas possuem tempo similar ao KHD e uma boa qualidade nas respostas. Conforme aumentamos o comprimento das sequências, podemos notar algumas mudanças, por exemplo, no segundo experimento da Tabela 4.11 para sequências com comprimentos até 1000, o tempo foi bem menor. Entretanto, os comprimentos inferidos pelas heurísticas estão ficando distantes em relação aos comprimentos do KHD. Para o último experimento na Tabela 4.11 com sequências com comprimento até 10.000, notamos que o tempo de execução foi realmente menor, mas os comprimentos inferidos foram os piores de todos os experimentos. Para os primeiros experimentos com sequências menores, esperávamos uma qualidade boa e um tempo próximo ao KHD. Para as sequências mais compridas no último experimento, esperávamos uma qualidade ruim e um tempo excelente. Uma surpresa encontrada durante o trabalho, foram os testes para o segundo experimento na Tabela 4.11. Como aumentamos o comprimento das sequências nesse experimento, acreditávamos que teríamos respostas ruins nas heurísticas, mas observamos a Figura 4.23 e notamos que a grande maioria dos comprimentos das subcadeias minimais estão próximas ao do algoritmo exato, então a piora na qualidade se dá pelo fato de que alguns comprimentos estão realmente errados, mas a grande maioria estão próximos dos comprimentos exatos.

Conforme aumentamos a quantidade de símbolos nos conjuntos, tanto os algoritmos exatos quanto as heurísticas podem demorar muito tempo para

terminar, tornando-se impraticáveis. Para contornar o problema do tempo de execução, realizamos um estudo inicial com o MPI para paralelizar as heurísticas. Nos nossos experimentos, podemos ver que conforme aumentamos o número de processadores, melhoramos bastante no tempo de execução. Este ponto pode ser explorado nos trabalhos futuros, com a intenção de melhorar também na qualidade das heurísticas. Se tivermos inúmeros processadores, sabemos que o tempo de execução diminuirá, além disso, podemos aumentar o número de posições exatas, melhorando, dessa forma, a qualidade das respostas. O estudo realizado neste trabalho com o paralelismo considerou o MPI e uma paralelização simples de distribuir as sequências entre os processadores. Como trabalho futuro, podemos paralelizar utilizando programação multi-processo de memória compartilhada com OpenMP [25] ou realizar paralelismo utilizando placas gráficas (GPU) com o CUDA [26]. Também como trabalho futuro, podemos melhorar a distribuição das posições exatas, escolhendo outras funções e avaliando quais apresentam melhores resultados, neste trabalho utilizamos a função $\log$ para determinar a quantidade de posições exatas.

Também como trabalhos futuros, está no estudo deste problema numa aplicação prática e na avaliação das subcadeias específicas como marcadores genéticos. Entretanto, como já mencionado na dissertação, as bases de dados reais podem possuir milhares de sequências, e uma forma de contornar esse problema é reduzindo de forma estratégica a quantidade de sequências. O CD-HIT-EST [27] é uma ferramenta de clusterização de sequências, que podemos utilizar para agrupar sequências similares. Atualmente estamos realizando um trabalho com a UFRN que consiste em analisar o metagenoma do tecido do estômago de dez humanos, onde temos dois metagenomas para cada humano: um metagenoma do tecido saudável e outro do tecido com o câncer. Como os metagenomas possuem milhares de sequências, estamos estudando a utilização de *scripts* do CD-HIT-EST para diminuir a quantidade de sequências, e como trabalho futuro, possivelmente, com o CD-HIT-EST bem definido e os algoritmos e heurísticas deste trabalho, a criação e a disponibilização de um pipeline com as ferramentas necessárias para reduzir e analisar amostras de sequências. Apresentamos na Figura 5.1 a versão inicial do pipeline.

Os resultados obtidos neste trabalho mostram que cumprimos com o objetivo de estudar detalhadamente o problema da Seleção das Subcadeias Específicas e abordagens para este problema.

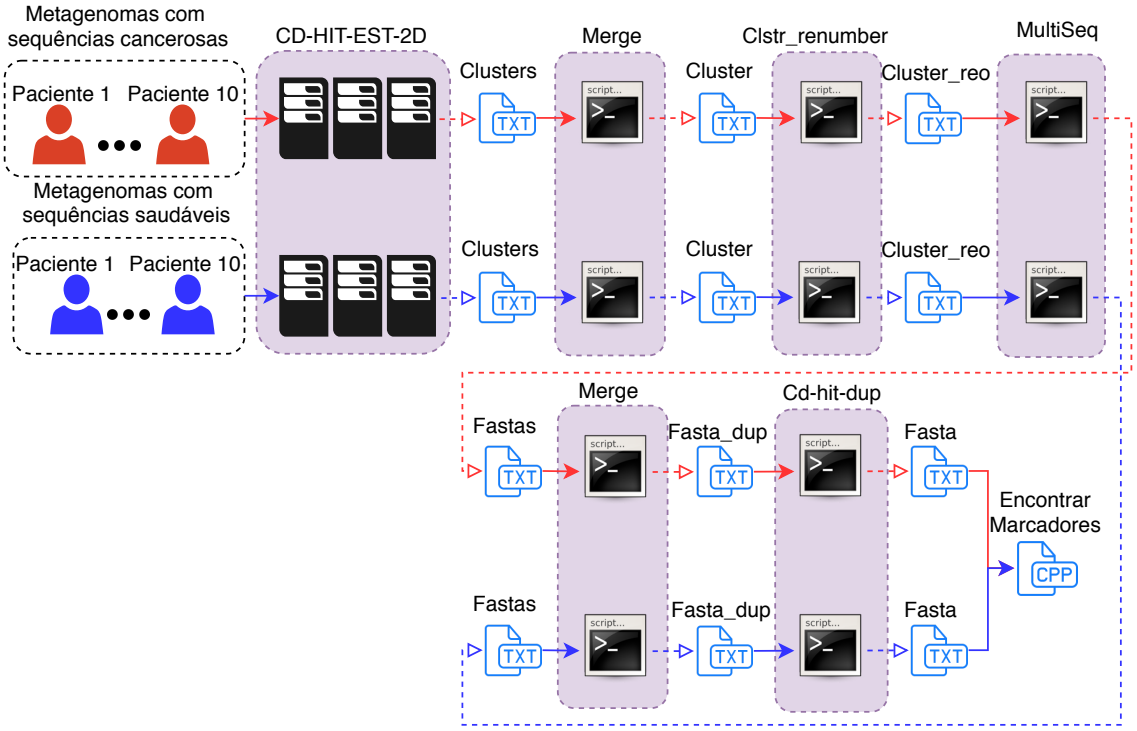

Figura 5.1: Pipeline com *scripts* do CD-HIT-EST para reduzir o número de sequências nos conjuntos. Os passos são os seguintes: **CD-HIT-EST-2D** – clusterizar (agrupar) as sequências similares (utilizamos 95% de similaridade) gerando diversos arquivos com diversos *clusters* (se existirem). **Merge** – reunir todos os *clusters* em um único arquivo. **Clstr_renumber** – reorganizar a numeração dos *clusters*. **MultiSeq** – gerar arquivos Fasta com as sequências contidas em cada *cluster*. **Merge** – reunir todas as sequências Fasta em um único arquivo. **CD-hit-dup** – retirar sequências duplicadas. Para realizar o *merge* podemos utilizar o comando **cat** do linux, sendo esse, o único passo sem *script* do CD-HIT-EST.

Bibliografia

[1] M. C. M. Ribeiro. *Genética molecular*. Universidade Federal de Santa Catarina, 2009. Citado nas páginas xii, 15, e 16.

[2] K. Onimaru and L. Marcon. Systems biology approach to the origin of the tetrapod limb. *arXiv preprint arXiv:1907.02730*, 2019. Citado na página 1.

[3] B. M. Peixoto. Classificação de sequências e análise de diversidade em metagenômica. Universidade Estadual de Campinas, 2013. Master's thesis. Citado na página 1.

[4] A. Turchetto-Zolet, C. Turchetto, and G. Zanella, C. Passaia. *Marcadores Moleculares na Era Genômica: Metodologias e Aplicações*. Sociedade Brasileira de Genética, 2017. Citado nas páginas 2 e 17.

[5] F. Bered, J. F. B. Neto, and F. I. F. de Carvalho. Marcadores moleculares e sua aplicação no melhoramento genético de plantas. *Ciencia Rural. Santa Maria, p. 513-520*, 1997. Citado nas páginas 2 e 17.

[6] R. O. Nodari, M. P. Guerra, A. C. de M. Dantas, V. M. Stefenon, and G. H. F. Klabunde. Fit 5806-biotecnologia. 2016. Citado nas páginas 2 e 17.

[7] P. Bork, C. Bowler, C. de Vargas, G. Gorsky, E. Karsenti, and P. Wincker. Tara oceans studies plankton at planetary scale. *Science*, 348(6237):873–873, 2015. Citado nas páginas 2 e 42.

[8] S. Pesant, F. Not, M. Picheral, S. Kandels-Lewis, N. Le Bescot, G. Gorsky, D. Iudicone, E. Karsenti, S. Speich, R. Troublé, et al. Open science resources for the discovery and analysis of tara oceans data. *Scientific data*, 2:150023, 2015. Citado nas páginas 2 e 42.

[9] L. Zitvogel, Y. Ma, D. Raoult, G. Kroemer, and T. F. Gajewski. The microbiome in cancer immunotherapy: Diagnostic tools and therapeutic strategies. *Science*, 359(6382):1366–1370, 2018. Citado na página 2.

[10] R. Shigefuku, T. Watanabe, Y. Kanno, H. Ikeda, H. Nakano, N. Hattori, K. Matsunaga, N. Matsumoto, S. Kanno, K. Nosho, et al. Fusobacterium nucleatum detected simultaneously in a pyogenic liver abscess and advanced sigmoid colon cancer. *Anaerobe*, 48:144–146, 2017. Citado na página 2.

[11] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter, and S. Morgan. *Biologia molecular da célula*. Artmed Editora, 6 edition, 2017. Citado nas páginas 2 e 15.

[12] L. Montera and M. C. Nicoletti. The PCR primer design as a metaheuristic search process. In *International Conference on Artificial Intelligence and Soft Computing*, pages 963–973. Springer, 2008. Citado na página 2.

[13] R. W. Hamming. Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160, 1950. Citado nas páginas 2 e 6.

[14] M. G. Maaß. A fast algorithm for the inexact characteristic string problem. Technical Report TUM-I0312, Fakultät für Informatik, TU München, aug 2003. Citado nas páginas 3, 9, e 51.

[15] D. Gusfield. Algorithms on strings, trees and sequences: Computer science and computational biology, 1st edition. *New York, United States*, page 534, 1997. Citado na página 3.

[16] L. B. Rocha, S. S. Adi, and E. Araujo. Specific substring problem: an application in bioinformatics. In *BSB*, 2018. Citado nas páginas 4 e 51.

[17] L. B. Rocha, S. S. Adi, M. A. Stefanes, and E. Araujo. Heuristics for the specific substring problem with hamming distance. In *2019 IEEE 19th International Conference on Bioinformatics and Bioengineering (BIBE)*, pages 250–257. IEEE, 2019. Citado nas páginas 4 e 51.

[18] E. N. Cáceres, H. Mongelli, and S. W. Song. Algoritmos paralelos usando cgm/pvm/mpi: uma introduçao. *AS TECNOLOGIAS da Informação e a Questão Social. Porto Alegre: Sociedade Brasileira de Computação*, pages 1–11, 2001. Citado nas páginas 7 e 8.

[19] M. A. Mariano. Comparação de algoritmos paralelos para a extração de regras de associação no modelo de memória distribuída. Master's thesis, 2011. Citado nas páginas 7 e 8.

[20] MPI. https://computing.llnl.gov/tutorials/mpi/. Citado nas páginas 7 e 8.

[21] L. Dalcin. mpi4py: Mpi for python, 2013. Citado na página 7.

[22] M. A. Filho. Metagenômica e sua aplicação no estudo de diversidade e função de microrganismos de solos do cerrado. *Embrapa Cerrados-Documentos (INFOTECA-E)*, 2010. Citado nas páginas 17 e 18.

[23] A. K. Meneghine. Análise metagenômica e potencial biotecnológico de microrganismos de solo e água de uma área agrícola com adubação orgânica. 2016. Citado na página 18.

[24] J. A. Dobre. O problema da seleção de segmentos específicos: algoritmos e aplicações. Universidade Federal de Mato Grosso do Sul, 2017. Master's thesis. Citado na página 38.

[25] OpenMP architecture review board: OpenMP application programming interface version 4.5. https://www.openmp.org/wp-content/uploads/openmp-4.5.pdf. Citado na página 53.

[26] NVIDIA corporation: CUDA parallel computing platform. https://developer.nvidia.com/cuda-zone. Citado na página 53.

[27] W. Li and A. Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, 2006. Citado na página 53.