Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Avaliação de estratégias Alignment-free para determinar o fator de pruning em comparação paralela de sequências

Rafael Neiva da Cunha - 130016594

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof.a Dr.a Alba Cristina M. A. de Melo - 149870

Brasília
2019

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Avaliação de estratégias Alignment-free para determinar o fator de pruning em comparação paralela de sequências

Rafael Neiva da Cunha - 130016594

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof.a Dr.a Alba Cristina M. A. de Melo - 149870 (Orientador)
CIC/UnB

Prof. Dr. Eduardo Adilo Pelinson Alchieri
CIC/UnB

Prof.a Dr.a Genaina Nunes Rodrigues
CIC/UnB

Prof. Dr. José Edil Guimarães de Medeiros
Coordenador do Curso de Engenharia da Computação

Brasília, 09 de julho de 2019

Dedicatória

Dedico esse trabalho de graduação para todos que me apoiaram durante os anos de graduação e especialmente ao meu falecido avô Teté!

Agradecimentos

Agradeço a todos que me apoiaram nessa jornada e em especial a Professora Alba que me orientou na realização desse trabalho com muita paciência e empenho.

Resumo

A Bioinformática tem como uma de suas principais operações a comparação entre sequências biológicas. Através de métodos exatos baseados no alinhamento, são obtidos resultados ótimos, porém o tempo para obtenção desses alinhamentos é muito grande. Com a adição da técnica de *block pruning*, o tempo pode ser bastante reduzido. Porém, no *block pruning* a área de poda é obtida ao longo da execução do alinhamento. Para determinar o tempo de execução de uma comparação com pruning antes do alinhamento existe uma regressão multi-linear. No entanto, essa fórmula requer que o fator de *block pruning* seja conhecido antes da execução da comparação, o que não acontece. O presente trabalho de graduação tem como objetivo analisar algoritmos que fazem a comparação sem se basear no alinhamento, *alignment-free*, e que são executados rapidamente, com o objetivo de utilizá-los para gerar os valores de *block pruning* antes do alinhamento das sequências. Os resultados da análise de diferentes algoritmos mostraram que a ferramenta *ALF-N2 (ALignment-Free framework)*, através de seu algoritmo *N2*, apresenta resultados satisfatórios para a geração de uma correlação entre o fator de *block pruning* e a similaridade entre as sequências por meio do *alignment-free*. Usando a estratégia proposta no presente trabalho de graduação, conseguimos obter taxas de *block pruning* com muito boa acurácia, apresentando erros de até 7,0% para sequências pequenas e 3,7% para sequências maiores.

Palavras-chave: Bioinformática, MASA, Comparação de Sequências Biológicas, Algoritmos Alignment-free, Block Pruning

Abstract

Sequence comparison is one of the most basic operations in Bioinformatics. With alignment-based exact methods it is possible to obtain very good results, but these methods require huge execution times, when the sequences are long. Block pruning is an optimization that may reduce considerably the execution times if the sequences involved have high similarity. In order to predict execution times of comparisons with pruning, a multiple regression formula was proposed. However, this formula requires that the block pruning rate is known before the comparison, which is not the case, since the pruned area is computed during the alignment process. This graduation project evaluates different alignment-free algorithms that execute much faster than the exact methods, with the goal to generate block pruning values before the sequences alignment itself. The results of the evaluation of various algorithms showed that the ALF-N2 tool provided very good results using its algorithm N2. With the ALF-N2 tool, we were able to generate a correlation between block pruning percentages and sequences similarity. Using the proposed strategy, we were able to obtain block pruning rates that are very close to the real ones, with maximum error rates of 7.0% for small sequences and 3.7% for longer ones.

Keywords: Bioinformatics, MASA, Biological Sequence Comparison, Alignment-Free Algorithms, Block Pruning

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A Bioinformática é uma área de pesquisa que vem se desenvolvendo muito nos últimos anos e atraindo grande atenção da área científica. Ela engloba aspectos da biologia, computação, estatística e matemática. Uma de suas operações mais importantes é a comparação entre sequências biológicas, que determina um escore relacionado com o grau de similaridade entre as sequências e um alinhamento entre as mesmas [2]. A comparação de sequências é um importante fator no estudo e análise dos dados biológicos, como sequências de DNA, RNA e proteínas, para auxiliar em um maior entendimento do mundo e sua evolução.

A comparação e alinhamento entre sequências longas de DNA utiliza geralmente programação dinâmica e é uma tarefa que demanda um alto custo computacional, demorando longas horas e até dias para ser finalizada [7]. De modo a reduzir drasticamente o tempo de execução e fornecer uma aproximação do resultado ótimo, vários algoritmos procuram obter o grau de similaridade entre duas sequências através de técnicas que não utilizam a programação dinâmica para produzir o alinhamento e geralmente têm complexidade linear. Tais técnicas são fundamentadas nos campos da álgebra linear, da teoria da informação e da estatística, sendo chamadas de técnicas *alignment-free* [2]. Nos últimos tempos, essas técnicas estão ganhando muita força e aparecendo cada vez mais no meio acadêmico em pesquisas.

Algoritmos *alignment-free* usam basicamente dois métodos: baseado na frequência de palavras e o baseado na teoria da informação [2]. Para cada método existem diversos algoritmos disponíveis com diferentes ideias e metodologias. O método baseado em frequência de palavras utiliza a noção de que sequências similares serão formadas por subsequências (ou *k-mers*) iguais. As seguintes ferramentas utilizam esse método: ALF-N2 (*ALignment-Free framework*) [8] e Mash (*Matching-Hashes*)[9]. O método baseado em teoria da informação reconhece e computa a quantidade de informações compartilhadas entre as sequências analisadas [2]. O método de representação de sequências biológicas [10]

e o baseado na entropia [10] são exemplos de técnicas baseadas na teoria da informação.

Um dos problemas das técnicas de *alignment-free* é que não existe um *benchmark* específico sendo usado nos diferentes trabalhos sobre o tema. Portanto, fica difícil escolher o melhor algoritmo existente para cada cenário.

O *block pruning* é uma técnica aplicada aos métodos exatos com programação dinâmica, que elimina do cálculo do alinhamento blocos da matriz que, matematicamente, não podem fazer parte do alinhamento ótimo. Em [11] foi determinada uma fórmula de previsão de tempo de execução para a ferramenta MASA (*Multi-platform Architecture for Sequence Aligners*) [4], que usa a porcentagem de *block pruning* (%BP) como um dos parâmetros. O problema é que, até o momento, a %BP é obtida somente ao final da comparação entre as sequências, ou seja, é necessário executar a ferramenta até o final para obtê-la, o que praticamente inviabiliza a fórmula de previsão.

O presente trabalho de graduação tem como objetivo definir uma abordagem para a obtenção da porcentagem de *block pruning* com algoritmos de *alignment-free*. Um dos desafios desse trabalho reside em correlacionar a similaridade entre as sequências (expressa em porcentagens de 0 a 100%) com a taxa de *block pruning* (expressa em porcentagens de 0 a 60%) de modo a obter regressões estatísticas. De posse dessas regressões, iremos determinar se a similaridade obtida com os algoritmos *alignment-free* oferece uma boa previsão da porcentagem de *block pruning* (%BP). Com isso, pode-se determinar se é viável a utilização desses métodos para se estimar a quantidade de *block pruning* que será utilizada na fórmula de previsão do tempo de execução proposta em [12]. Portanto, visamos descobrir a *%BP* entre duas sequências através de algoritmos de *alignment-free*, que executam rapidamente mesmo que a comparação seja entre muito longas sequências de DNA. Com isso, poderemos utilizar os valores encontrados/estimados para a %BP e utilizar a fórmula de previsão de tempo de execução no framework MASA [4] de maneira efetiva.

O restante deste documento está organizado como se segue. O Capítulo 2 apresenta inicialmente o problema da comparação entre sequências biológicas. A seguir, apresenta algoritmos exatos que são usados para resolvê-lo. Por fim, são apresentados algoritmos que não realizam o alinhamento entre as sequências (*alignment-free*) para se chegar ao resultado da comparação. O capítulo 3 apresenta o framework MASA, explorando toda sua história e a otimização *block pruning* (BP) que é um dos motivadores deste trabalho. Ao final do capítulo 4, é descrito o MASA - OpenCL [12], que propõe a fórmula de previsão de tempo de execução, trabalho do aluno de doutorado Marco A. C. Figueirêdo Jr.. O Capítulo 4 apresenta a abordagem proposta no presente trabalho de graduação para a obtenção da taxa de *block pruning*. Primeiramente, avaliamos diversos algoritmos *alignment-free* e escolhemos o ALF-N2. Com a similaridade obtida com o ALF-N2 e

as porcentagens reais de BP, geramos uma regressão polinomial. Finalmente, usamos a regressão polinomial para obter a %BP para comparações não usadas na obtenção da fórmula. Além disso, com os valores gerados de %BP, utilizamos a fórmula de previsão de tempo de execução para comparar os valores de tempo de execução do *block pruning* gerado com os tempos de execução obtidos com o *block pruning* real. Finalmente, o Capítulo 5 apresenta a conclusão e sugere trabalhos futuros.

1.1 Objetivos

O objetivo geral do presente trabalho de graduação é descobrir a porcentagem de *block pruning* da comparação entre dois pares de sequências biológicas antes da execução do alinhamento entre as sequências. Objetivos mais específicos são entender se algoritmos de *alignment-free* são capazes de executar a tarefa de comparação não exata entre as sequências de forma relativamente precisa, gerar uma correlação entre a similaridade das sequências e a %BP e ao final conseguir calcular a fórmula de previsão do tempo de execução da comparação na ferramenta MASA antes de de fato executar a comparação.

Capítulo 2

Comparação de sequências biológicas

A comparação entre duas sequências biológicas permite, entre outros, analisar o parentesco evolutivo entre diferentes espécies, descobrindo, portanto, semelhanças e diferenças genéticas entre espécies, e também compreender melhor um organismo específico [7]. Essa é uma das operações mais básicas da Bioinformática [13]. Além de gerar um escore do alinhamento entre as sequências, que reflete o quão próximas as duas sequências de fato são, ainda gera o alinhamento entre as sequências que ressalta onde estão as maiores similaridades e diferenças. No presente capítulo, apresentamos inicialmente os conceitos de alinhamento e escore entre sequências biológicas. A seguir, são apresentados quatro algoritmos exatos de comparação de sequências, que obtém o resultado ótimo. Ao final do capítulo, são apresentados os métodos *alignment-free* e ferramentas que os implementam.

2.1 Escore e Alinhamento

Para se comparar duas sequências biológicas, o primeiro passo é alinhá-las. Todo alinhamento possui um escore, que representa a similaridade entre as sequências. Existem dois tipos básicos de alinhamento, o global e o local, como mostra a Figura 2.1. No alinhamento global as sequências são analisadas na sua totalidade. Já no alinhamento local, o objetivo é descobrir áreas que contém uma forte similaridade e o alinhamento se resume a elas. Em ambos, são levados em consideração *matches* (pareamento entre caracteres iguais), *mismatches* (pareamento entre caracteres diferentes) e *gaps* (inserção de espaços).

Para se obter um bom alinhamento é importante entendermos o modelo de escore, que consiste em uma soma de termos para cada par alinhado mais termos para cada *gap*. Como eventos conservativos (*matches*) são mais esperados em um bom alinhamento, eles contribuem com valores positivos, enquanto eventos não-conservativos (*mismatches* e *gaps*) contribuem com valores negativos [7]. Para o valor de um conjunto de *gaps* é

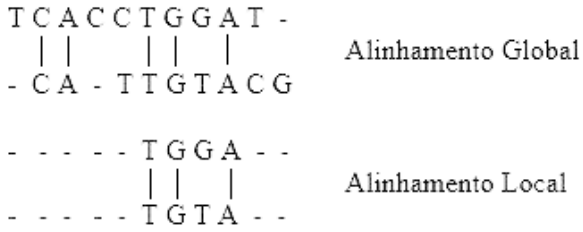


Figura 2.1: Alinhamento Global x Local entre as sequências S0=TCACCTGGAT e S1=CATTGTACG.

interessante penalizar o alinhamento com um valor maior quando um *gap* é introduzido e, caso ocorra uma sequência de *gaps*, esse valor é menor para os *gaps* subsequentes [7].

2.2 Algoritmo Needleman-Wunsch (NW)

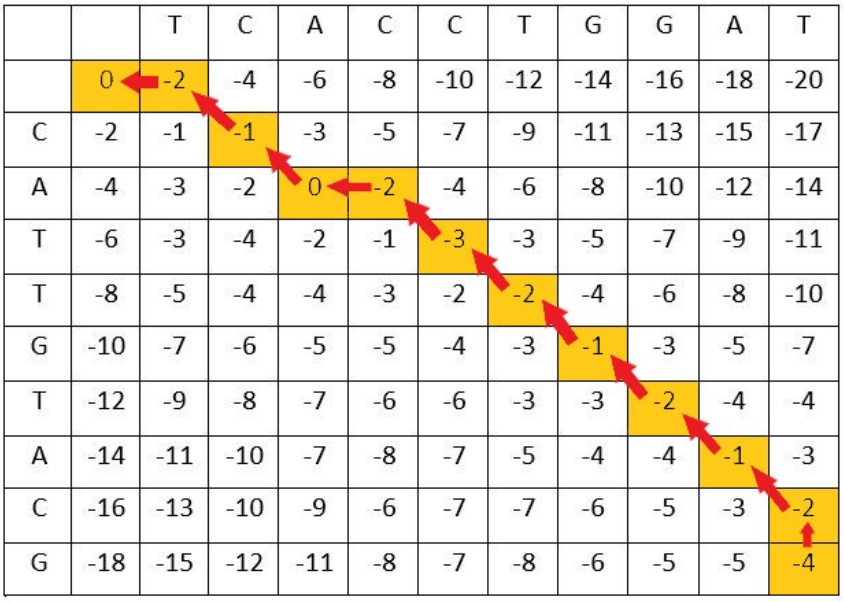
Para a obtenção do alinhamento global ótimo entre duas sequências x e y, de tamanhos m e n com a presença de *gaps* é utilizado o algoritmo de programação dinâmica Needleman-Wunsch [14]. A ideia do algoritmo é que o alinhamento ótimo seja construído através do alinhamento ótimo de subsequências, portanto, a matriz de programação dinâmica F com índices i e j onde $F(i,j)$ é o escore do alinhamento ótimo entre os prefixos $x[0..i]$ e $y[0..j]$. $F(i,j)$ é calculada conforme a Equação 2.1.

$$F(i,j) = max \begin{cases} F(i-1,j-1) + s(x_i, y_j), \\ F(i-1,j) - d, \\ F(i,j-1) - d \end{cases} \tag{2.1}$$

Para que a matriz seja calculada com a Equação 2.1 é necessário que sua primeira linha e primeira coluna sejam inicializadas com as penalidades de *gap*. A partir daí já é possível utilizar a Equação 2.1, onde $s(x_i, y_j)$ é igual à pontuação de *match*, se $x_i = y_i$ ou à pontuação de *mismatch*, caso contrário, e d é a penalidade de *gap*. Após isso, o valor máximo retornado é colocado em $F(i,j)$, além de armazenar uma indicação sobre qual das três células $F(i-1,j-1)$, $F(i-1,j)$ ou $F(i,j-1)$ gerou esse valor. Caso valores máximos iguais sejam obtidos, a indicação geralmente segue a ordem diagonal, esquerda e acima.

Quando a matriz F estiver com todas as posições preenchidas, o escore do alinhamento ótimo encontra-se na célula $F(m,n)$. Passa-se então à fase de *traceback*. Partindo do canto inferior direito da matriz ($F(m,n)$), segue-se o caminho das células armazenadas em direção à origem da matriz ($F(0,0)$). A Figura 2.2 mostra a matriz preenchida e o

caminho do *traceback* para obter o alinhamento global ótimo. Quando a célula guardada é acima ou à esquerda é necessário incluir um *gap* no alinhamento e caso seja na diagonal será incluído um *match* ou *mismatch*.



		T	C	A	C	C	T	G	G	A	T
	0	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20
C	-2	-1	-1	-3	-5	-7	-9	-11	-13	-15	-17
A	-4	-3	-2	0	-2	-4	-6	-8	-10	-12	-14
T	-6	-3	-4	-2	-1	-3	-3	-5	-7	-9	-11
T	-8	-5	-4	-4	-3	-2	-2	-4	-6	-8	-10
G	-10	-7	-6	-5	-5	-4	-3	-1	-3	-5	-7
T	-12	-9	-8	-7	-6	-6	-3	-3	-2	-4	-4
A	-14	-11	-10	-7	-8	-7	-5	-4	-4	-1	-3
C	-16	-13	-10	-9	-6	-7	-7	-6	-5	-3	-2
G	-18	-15	-12	-11	-8	-7	-8	-6	-5	-5	-4

Figura 2.2: Matriz de programação dinâmica Needleman-Wunsch.

Na Figura 2.2 os parâmetros utilizados foram: *match*: +1; *mismatch*: -1; *gap*: -2 e o valor do escore do alinhamento global ótimo produzido é -4.

2.3 Algoritmo Smith-Waterman (SW)

Para se obter o alinhamento local ótimo entre duas sequências x e y de tamanho m e n com a presença de *gaps* utiliza-se o algoritmo de programação dinâmica Smith-Waterman [15]. O algoritmo encontra o alinhamento local ótimo através do melhor alinhamento entre todas as subsequências das sequências x e y. Descobrir o alinhamento local ótimo é importante em situações onde, por exemplo as sequências têm um parentesco evolutivo muito distante. Nesse caso, é possível encontrar partes das sequências que se mantiveram conservadas durante a seleção natural e ainda contém alta similaridade [15].

$$F(i,j) = max \begin{cases} 0, \\ F(i-1,j-1) + s(x_i, y_j), \\ F(i-1,j) - d, \\ F(i,j-1) - d \end{cases} \tag{2.2}$$

O algoritmo Smith-Waterman é parecido com o Needleman-Wunsch, porém é incluído um 0 na equação de recorrência (Equação 2.2) para impedir valores negativos. O valor zero significa começar um novo alinhamento, já que, se em alguma parte do alinhamento ótimo se encontra um valor negativo, é melhor começar um novo alinhamento do que estender o antigo [15]. Portanto, para se inicializar a matriz de Smith-Waterman ao invés da utilização do valor baseado no *gap* na primeira linha e primeira coluna, preenchemo-las com zeros.

		T	C	A	C	C	T	G	G	A	T
	0	0	0	0	0	0	0	0	0	0	0
C	0	0	1	0	1	1	0	0	0	0	0
A	0	0	0	2	0	0	0	0	0	1	0
T	0	1	0	0	1	0	1	0	0	0	2
T	0	1	0	0	0	0	1	0	0	0	1
G	0	0	0	0	0	0	0	2	1	0	0
T	0	1	0	0	0	0	1	0	1	0	1
A	0	0	0	1	0	0	0	0	0	2	0
C	0	0	1	0	2	1	0	0	0	0	1
G	0	0	0	0	0	1	0	1	1	0	0

Figura 2.3: Matriz de programação dinâmica Smith-Waterman.

A Figura 2.3 ilustra a matriz de programação dinâmica Smith-Waterman. No traceback, ao invés de começarmos na última célula da matriz $F(m,n)$ começamos da célula com maior valor presente na matriz (valor 2 na Figura 2.3). Caso exista mais de uma célula com valor máximo é escolhida uma dessas células. O alinhamento termina quando uma célula com valor zero é encontrada.

Os parâmetros de *match*, *mismatch* e *gap* foram iguais aos utilizados no Needleman-Wunsch, portanto, *match*: +1; *mismatch*: -1; *gap*: +2; Sendo assim, o escore do alinhamento local ótimo representado na Figura 2.3 é +2.

2.4 Gotoh

O algoritmo de Gotoh [16] é uma adaptação do algoritmo de Needleman-Wunsch (Seção 2.2) para o modelo *affine gap*, que é mais próximo da realidade biológica. O modelo *affine gap* utiliza uma penalidade maior para abertura de uma sequência de gaps e uma penalidade menor para a extensão da mesma sequência.

$$F(i,j) = max \begin{cases} F(i-1,j-1) + s(xi,yj), \\ E(i-1,j-1) + s(xi,yj), \\ G(i-1,j-1) + s(xi,yj); \end{cases} \tag{2.3}$$

$$E(i,j) = max \begin{cases} F(i-1,j) - d, \\ E(i-1,j) - e; \end{cases} \tag{2.4}$$

$$G(i,j) = max \begin{cases} F(i,j-1) - d, \\ G(i,j-1) - e. \end{cases} \tag{2.5}$$

Para isso, o algotimo de Gotoh calcula 3 matrizes de programação dinâmica, ao invés de uma, conforme as Equações (2.3), (2.4) e (2.5). Na equação 2.3, $F(i,j)$ corresponde ao escore máximo até (i,j) caso exista um *match* ou *mismatch* entre $x[i]$ e $y[j]$. Na Equação 2.4, $E(i,j)$ corresponde ao escore máximo caso x esteja alinhado com *gap* e, na equação 2.5, $G(i,j)$ corresponde ao escore máximo caso y esteja alinhado com *gap*. O valor d corresponde à penalidade de abertura de *gap* e o valor e corresponde à penalidade de extensão do *gap*.

2.5 Myers e Miller

Os algoritmos descritos nas seções 2.2, 2.3 e 2.4 possuem complexidade quadrática de memória. Sendo assim, à medida que o tamanho das sequências cresce, o custo de memória para armazernar as matrizes fica muito alto. O algoritmo proposto por Myers e Miller [1] não armazena a matriz inteira e mesmo assim obtém um alinhamento ótimo e seu respectivo escore. Ele é uma adaptação do algoritmo de Hirschberg [17] que foi

desenvolvido para resolver o problema da Maior Subsequência Comum (*LCS - Longest Commom Subsequence*).

O algoritmo de Myers e Miller utiliza a recursividade para encontrar o ponto médio (u, v) do alinhamento. No início, o algoritmo calcula as equações de recorrência do início para o meio da matriz, armazenando somente a linha que está sendo calculada e a linha anterior. A seguir, o algoritmo calcula as equações do final para o meio, sobre o reverso das sequências. Ao final dessa etapa, tem-se duas linhas do meio. Hirschberg [17] provou que o ponto i onde a soma dos valores das duas linhas na mesma posição é máximo pertence ao alinhamento ótimo (*optimal midpoint*). Esse ponto é utilizado para dividir a matriz em 4 quadrantes, conforme a Figura 2.4. O algoritmo é executado nos quadrantes superior esquerdo e inferior direito, recursivamente, até que todos os pontos do alinhamento ótimo sejam obtidos, em espaço $O(m + n)$, onde m e n são os tamanhos das sequências.

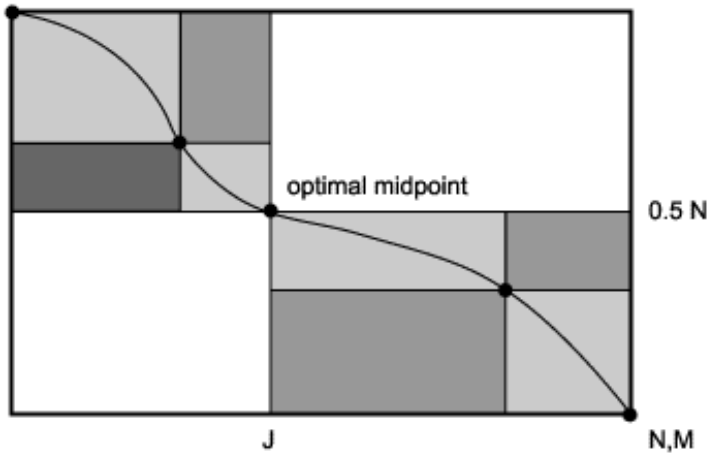


Figura 2.4: Metódo dividir e conquistar de Myers e Miller [1].

2.6 Métodos Alignment-Free

Os programas que produzem alinhamentos entre sequências normalmente assumem que sequências homólogas são formadas por trechos de sequências linearmente organizados e relativamente conservados, o que é chamado de colinearidade. Porém, essa presunção é violada muitas vezes no mundo real [2]. Além disso, as técnicas de alinhamento clássicas, como as detalhadas nas seções 2.2 a 2.5, geralmente exigem um alto tempo de execução. Mesmo utilizando técnicas de *block pruning*, o tempo de execução continua muito alto, quando comparado aos métodos *alignment-free*.

As técnicas de *alignment-free* são fundamentadas matematicamente nos campos da álgebra linear, da teoria da informação e da estatística e calculam medidas de dissimi-

laridade ou distância entre pares de sequências [2]. Elas tem como objetivo quantificar a similaridade ou dissimilaridade entre sequências sem usar ou produzir um alinhamento [18]. Essas técnicas não dependem da programação dinâmica para produzir o alinhamento e, portanto, são computacionalmente mais baratas, geralmente com complexidade linear de tempo e memória. Devido ao tempo de execução reduzido, são utilizadas para comparações entre sequências muito longas e genomas completos. Essa técnicas podem ser divididas em dois grupos [18]: (a) métodos baseados nas frequências das subsequências de um comprimento definido, também chamados de métodos baseados em palavras (*word-based*); e (b) métodos que avaliam o conteúdo informacional entre as sequências completas, também chamado de métodos baseados em teoria da informação. Além destes, existem alguns métodos que não são classificados em nenhum dos dois grupos e possuem diferentes formas de quantificar a similaridade.

2.6.1 Método baseado na frequência de palavras

Esse método é baseado na ideia de que sequências similares compartilham palavras (ou *k-mers*, subsequências de tamanho k) iguais. Com isso, operações matemáticas que se baseiam na ocorrência de palavras semelhantes entre as duas sequências mostram o quão similares ou dissimilares elas são. A ideia de uma assinatura genômica também é fortemente relacionada com esse método [2].

O método baseado na frequência de palavras pode ser dividido em três passos essenciais que podem ser vistos na Figura 2.5. No primeiro passo, as sequências a serem comparadas são quebradas em diversas palavras de um tamanho definido (tamanho de *k-mer*). Quanto maior o tamanho mais preciso será o resultado porém mais complexo será o cálculo. No exemplo da Figura 2.5, o tamanho de *k-mers* é 3. Assim, são criadas duas coleções de palavras únicas ($W_3^x = ATG, TGT, GTG$ e $W_3^y = CAT, ATG, TGT, GTG$). Como algumas palavras estão presentes em ambas coleções é feita uma coleção geral que contém todas as diferentes palavras, ou seja, a união entre as duas coleções W_3^x e W_3^y, para deixar os cálculos mais simples $W_3 = ATG, CAT, GTG, TGT$.

O segundo passo consiste em transformar as coleções W_3^x e W_3^y em um vetor de números onde as palavras que estão presentes na coleção W_3 serão contadas quando aparecerem em cada uma das coleções W_3^x e W_3^y. Assim, serão produzidos dois vetores C_3^x e C_3^y de acordo com a frequência com que cada palavra de W_3 ocorre em W_3^x e W_3^y respectivamente. Os valores no exemplo são apresentados na Figura 2.5 $C_3^x = (0, 1, 2, 2)$ e $C_3^y = (1, 1, 1, 1)$.

O último passo consiste em quantificar o nível de dissimilaridade entre as sequências através da distância entre os vetores que representam cada uma das sequências, no exemplo, C_3^x e C_3^y. A diferença é comumente computada como a distância euclidiana (Figura 2.5), porém outras métricas podem ser aplicadas. Quanto maior for o valor de

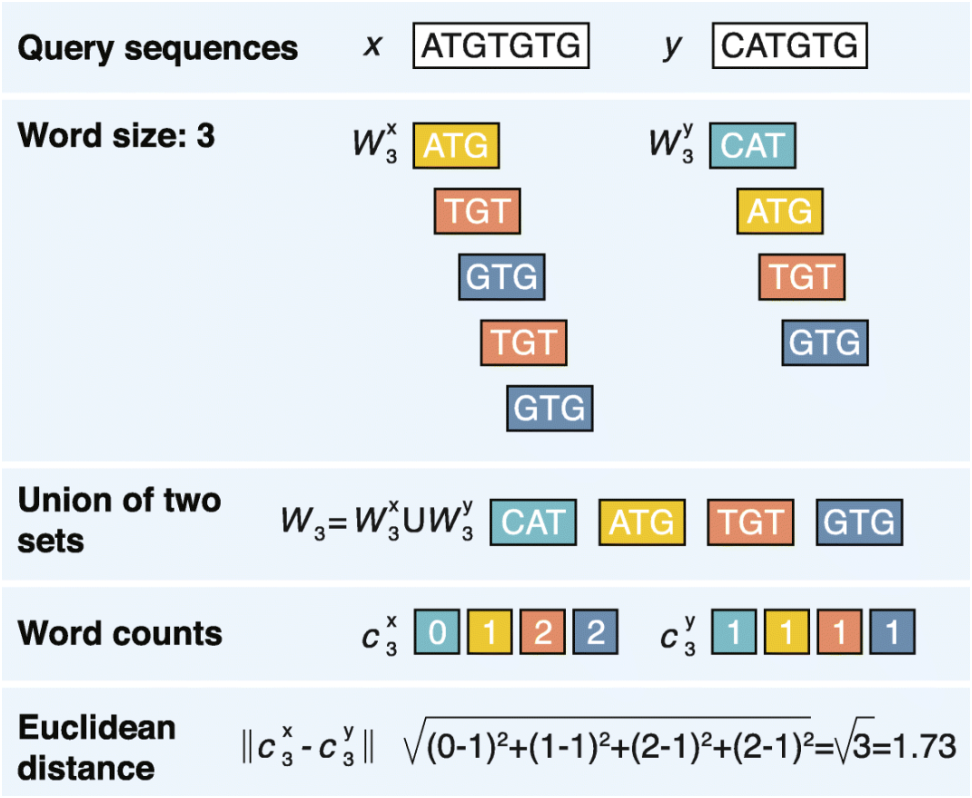


Figura 2.5: Método baseado na frequência de palavras.

dissimilaridade, mais distantes as sequências são entre si, portanto, o resultado para duas sequências idênticas será zero. No exemplo da Figura 2.5, a distância é 1,73. No melhor caso a distância é 0, ou seja, a sequências são iguais e no pior caso a distância é 2,828, ou seja, as sequências são completamente diferentes. Esses valores variam de acordo com o tamanho da sequência, mas 0 sempre refletirá sequências iguais caso a distância euclidiana seja utilizada.

2.6.2 Método baseado na teoria da informação

Métodos baseados na teoria da informação reconhecem e computam a quantidade de informações compartilhadas entre as sequências biológicas analisadas [2]. Nucleotídeos e aminoácidos são um conjunto de símbolos e sua organização é interpretada com ferramentas de teoria da informação, como a complexidade e a entropia.

Como exemplo, pode-se usar a complexidade de Kolmogorov [2] de uma sequência, que é o tamanho da menor descrição da sequência. Por exemplo, a descrição da sequência $GGGGGG$ seria $6G$, já a sequência $AGTCG$ seria descrita como $1A1G1C1G$. Portanto, quanto maior for o tamanho da descrição de uma sequência, maior será a complexidade. Kolgomorov não desenvolveu o método para encontrar a menor descrição da sequência

baseado em um conjunto de símbolos. Assim, a complexidade é mais comumente aproximada por algoritmos de compressão (por exemplo, os programas zip e gzip) onde o tamanho da sequência comprimida dá uma estimativa da complexidade [2]. Nesse caso, sequências mais complexas serão menos comprimíveis. O cálculo da distância entre duas sequências através da complexidade (compressão) é relativamente simples. O procedimento recebe como entrada duas sequências a serem comparadas e as concatena para formar uma sequência mais longa. Por exemplo, sendo $x = ATGTGTG$ e $y = CATGTG$, tem-se $xy = ATGTGTGCATGTG$. Se x e y são exatamente iguais, então a complexidade the xy será bem próxima da complexidade de x ou y. Caso x e y sejam bem diferentes, a complexidade de xy (compressão de xy) tenderá a ser a soma da complexidade de x com a complexidade de y. Existem diferentes modos de calcular a distância através da teoria da informação, um exemplo é a complexidade de Lempel-Ziv que pode ser visto na Figura 2.6. Nessa figura, obtém-se inicialmente a concatenação (xy) das sequências x e y. A seguir, é calculada a complexidade de Lempel-Ziv para x, y e xy. Finalmente, a distância de compressão normalizada é calculada (Figura 2.6). Nesse caso, a distância é 0,6, no caso de duas sequências exatamente iguais a complexidade de xy seria exatamente igual à complexidade de x e y, portanto, a distância normalizada seria 0 e no caso de sequências opostas a complexidade de xy seria exatamente a soma da complexidade de x e y, portanto a distância normalizada seria 1.

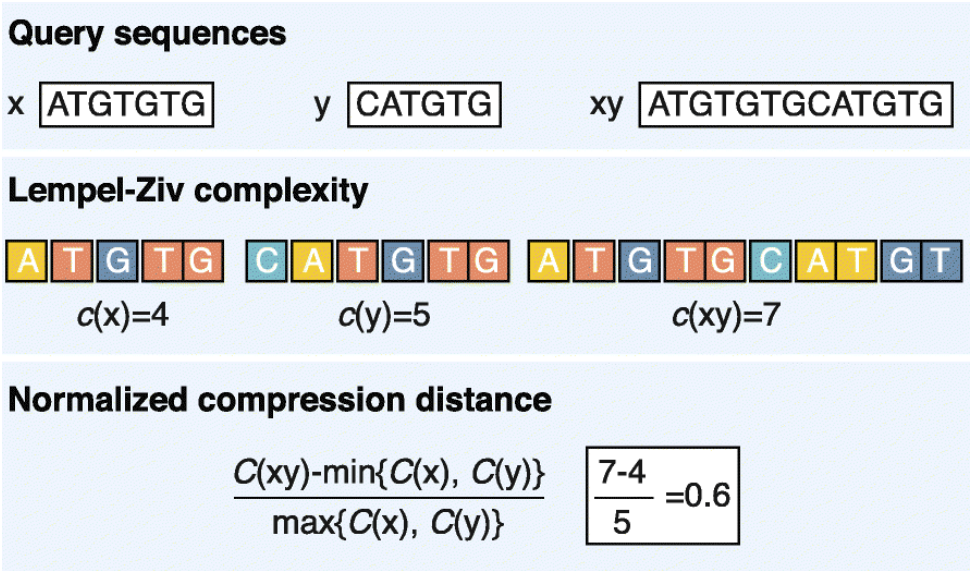


Figura 2.6: Método baseado na teoria da informação [2].

Outro exemplo de uma medição baseada na teoria da informação aplicada nas sequências biológicas é a entropia [2]. A entropia é expressa através de uma fórmula para quantificar a incerteza de se encontrar uma palavra qualquer em uma sequência de palavras.

Usando o conceito de entropia criado por Shannon, Kullback e Leibler [19], foi introduzida a medida de entropia relativa, que possibilita a comparação de duas sequências biológicas. O procedimento consiste no cálculo da frequência dos símbolos ou palavras em uma sequência e a soma de suas entropias nas sequências comparadas.

Os dois conceitos de teoria da informação, complexidade e entropia, tem uma associação muito clara, embora existam diferenças metodológicas. Por exemplo, uma sequência de baixa complexidade (ex. $AAAAAAA$) terá uma menor entropia do que uma sequência mais complexa (ex. $ACGATGT$) [2].

2.6.3 Avaliação dos Métodos *Alignment-Free*

Para a avaliação da qualidade dos resultados produzidos pelos métodos *alignment-free* não existe um *benchmark* padronizado. Toda vez que um novo algoritmo é publicado, um novo processo de avaliação ou a comparação com conjunto de dados diferentes é introduzido. Por exemplo, a maior parte dos algoritmos foi avaliada usando dados de sequências de DNA, genomas de mitocôndrias, genomas procariontes, genomas de plantas e outros genomas homólogos [2].

Dada toda essa heterogeneidade entre os procedimentos para avaliação dos algoritmos, fica muito difícil saber qual algoritmo é o melhor para os diversos cenários possíveis. Algumas tentativas de padronização foram feitas em trabalhos mais específicos [20] e [21], e o fator tempo de execução fica muito evidenciado nesses testes. A comparação de 22 milhões de pares de proteínas feita através do algoritmo de Smith-Waterman levou aproximadamente 3 dias, enquanto na média os algoritmos de *aligment-free* levaram 4 minutos para completar a mesma comparação. Logo, esses métodos se executam em um tempo muito reduzido. A principal questão é determinar se seus resultados possuem boa qualidade.

2.7 Ferramentas de *Alignment-Free*

2.7.1 andi - Anchor Distances

De acordo com a documentação, o andi (*anchor distances*) [3] foi otimizado para computar a distância evolucionária entre genomas intimamente relacionados, medindo a distância baseado em alinhamentos locais sem a presença de gaps amparados por pares de *matches* únicos com o menor tamanho. De acordo com o autor [3], o programa executa de maneira extremamente rápida e consome pouca memória RAM e os pares de *matches* dividem o genoma entre regiões homólogas e não homólogas para estimar a distância. O andi [3] foi proposto com o objetivo ter boa acurácia, ser rápido e eficiente em termos de utilização

de memória. A estratégia utilizada foi procurar por *mismatches* que ocorrem entre duas sequências de *matches* maiores que um tamanho mínimo. Essa formação, vista na Figura 2.7, foi chamada de âncora (*anchors*).

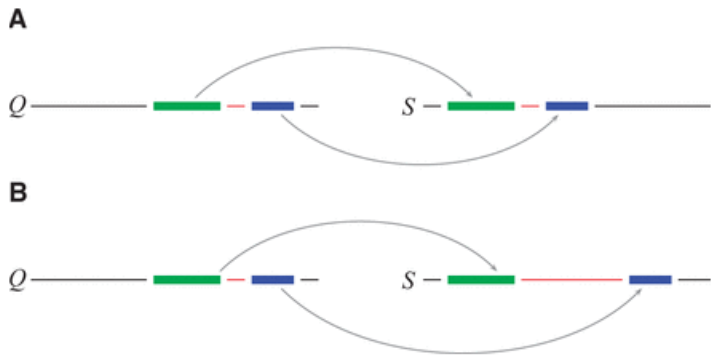


Figura 2.7: Estratégia de âncoras [3]. A: Âncoras com tamanhos iguais são contabilizadas. B: Âncoras com tamanhos diferentes são ignoradas. .

Para se chegar à estratégia de âncoras são necessárias duas etapas. Na primeira, define-se a distância entre as âncoras, olhando para os *matches* e para três critérios: tamanho mínimo, equidistância e singularidade. Considerando duas sequências de DNA Q e S, procura-se pares de âncoras que tenham a mesma distância em Q e em S, como mostra a Figura 2.7. Na segunda etapa somam-se os *mismatches* presentes dentro da âncora e, caso a distância entre Q e S seja diferente, os pares de âncoras são ignorados. O número total de *mismatches* presentes nas âncoras equidistantes dividido pelo número de nucleotídeos cobertos pelas regiões de âncora e não-âncora representam o número estimado de *mismatches* por *site*, $dm(Q, S)$. Um dos parâmetros usados na fórmula que compara as duas sequências é o tamanho mínimo de uma âncora. Esse valor é importante já que ele vai influenciar o número total de âncoras. Quanto mais âncoras forem consideradas válidas, maior será o número de *mismatches* por *site*.

Para a análise dos resultados foram utilizados em [3] diferentes conjunto de dados, alguns tirados dos artigos que definem as ferramentas que foram utilizadas para comparar com o andi. O tamanho médio das sequências de DNA utilizadas foi de 5 milhões de pares de bases. De acordo com os resultados, a ferramenta andi cumpre bem o seu objetivo de ser mais eficiente, em termos de tempo e consumo de memória. Porém, os resultados acabam influenciados pelo número de substituições por *site* (K), que segue a fórmula 2.6. Na simulação usada em [3], quando os valores de K começam a passar de 0,5 a proporção de falhas na estimativa aumenta muito, saindo de 0,7% para $K = 0,5$ para 94% para

$K = 0,65$. Isso acontece porque a busca por âncoras válidas começa a não ser viável para K maior que 0,5, e mesmo diminuindo o tamanho mínimo das âncoras a acurácia continua baixa. Portanto, após as análises a utilização da ferramenta andi só é recomendada para sequências que o K seja menor que 0,5.

$$K(Q,S) = \frac{3}{4} ln(1 - \frac{4}{3} dm(Q,S)) \tag{2.6}$$

2.7.2 Mash

Dos anos 1990 até os dias de hoje saímos de um banco de sequências com menos de 50 milhões de bases de nucleotídeos, fonte os arquivos do GenBank [22], para mais de vários trilhões de bases. Portanto, novos métodos precisavam ser criados para gerenciar e organizar essa enorme quantidade de informações. A técnica de MinHash [23] é uma técnica de *hash* sensível a localidade, que foi criada na década de 1990 para detectar páginas muito parecidas na Web. Mais recentemente, começou a ganhar força no meio da Bioinformática. A ferramenta Mash [9] utiliza essa técnica para expressar sequências muito grandes em um esboço comprimido, fazendo depois disso a comparação entre esboços. Assim, a ferramenta Mash combina a alta especificidade das abordagens baseadas em *match* e a redução da complexidade das abordagens estatísticas, calculando a distância entre os esboços de tamanho reduzido rapidamente com resultados que correlacionam com abordagens baseadas em alinhamento como o ANI (Average Nucleotide Identity) [24].

A ferramenta Mash propõe duas funções básicas para a comparação entre as sequências, o *sketch* e o *dist*. Com a função *sketch* a sequência de caracteres que forma a sequência biológica é convertida para um *sketch* de MinHash. Depois, com a função *dist*, os dois *sketchs* convertidos das sequências biológicas são comparados, uma estimativa do índice de Jaccard (Equação 2.7), um valor P e a distância Mash são retornados. A distância Mash estima a taxa de mutação de acordo com um modelo evolucionário simples [25]. O Mash compara subsequências de tamanho k, ou *k-mers*, portanto, os *inputs* podem ser genomas inteiros. Cada *input* é simplesmente tratado como um conjunto de *k-mers* tirados de algum alfabeto. O índice de Jaccard é um bom indicador de similaridade entre as sequências porque seu resultado se correlaciona bem com os de ferramentas baseadas em alinhamento [9].

$$J(A,B) = \frac{A \cup B}{A \cap B} \tag{2.7}$$

A ferramenta Mash habilita o agrupamento (*clustering*) escalável de genomas inteiros, que não é possível através de ferramentas baseadas no alinhamento. Durante os testes do Mash, foi feito o *sketch* e o *clustering* de todo o NCBI RefSeq Release 70 [26]. Isso totalizou 54,118 organismos e 618 GBP (618 bilhões de pares de bases). Nesse caso, o

sketch resultante tinha apenas 93 MB. Portanto, a ferramenta se mostrou extremamente útil para a construção de grupos de genomas relacionados [9].

$$D = 1 - ANI \tag{2.8}$$

Outro resultado encontrado foi que a distância Mash se correlaciona muito bem com o ANI [24] de acordo com a Equação 2.8, considerando-se múltiplos *sketchs* e diferentes tamanhos de subsequências. Porém, a correlação começa a se degradar para genomas mais divergentes, já que a variação da estimativa do Mash aumenta com a distância evolucionária. A escolha do k é um compromisso entre sensibilidade e especificidade: valores pequenos de k são mais sensíveis em genomas divergentes, mas perdem a especificidade para grandes genomas devido à grande probabilidade de uma colisão. Essas colisões enviesam a distância Mash [9], porém escolher um valor de *k-mer* de acordo com o tamanho do genoma diminui as colisões. No entanto, um valor de *k-mer* muito grande reduz a sensibilidade.

2.7.3 *ALignment-Free framework* (ALF-N2)

A regulação de genes dependente das sequências é normalmente feita através da ligação entre fatores de transcrição com pequenos padrões do DNA. Essas regiões de ligação dos fatores ocorrem em aglomerados regulatórios no genoma, chamados módulos *cis-regulatory* (CRMs) [27].

A ideia de descrever uma sequência através de suas palavras se encaixa muito bem com o modelo das CRMs, onde uma função similar se reflete em um conteúdo similar na região de ligação. Métodos baseados na contagem de palavras (Seção 2.6.1) foram muito usados na comparação de sequências regulatórias [28]. Porém, esses métodos se baseiam na contagem exata de palavras, enquanto as regiões de ligação com fator de transcrição são mais flexíveis. Como a orientação genômica de CRMs e da região de ligação normalmente não é conhecida, é necessário comparar as sequências de acordo com a contagem de palavras nas duas fitas simultaneamente. Como exemplo, a palavra $w = CATAAT$ pode ser delimitada pelo mesmo fator de transcrição que as palavras CTTAAT e ATTATG, a primeira tendo uma substituição e a segunda estando na fita oposta. Métodos de comparação exata entre palavras consideram essas palavras dissimilares. Portanto, foi necessário desenvolver o método que compare as sequências baseado na vizinhança das palavras. Chamado N2, que é um método de comparação *alignment-free* baseado em contagem de palavras (Seção 2.6.1) que integra todas as palavras na vizinhança w [8].

Utiliza-se a ideia do *alignment-free* para comparar duas sequências S_1 e S_2 de tamanho l_1 e l_2, baseadas no número de ocorrências de todas as palavras w de tamanho k dentro do

alfabeto possível. O vetor de contagem de palavras de uma sequência S de tamanho l é calculado com a Equação 2.9. Para superar as restrições da contagem exata de palavras, a Equação 2.9 é extendida para a contagem de palavras da vizinhança. O conjunto de palavras na vizinhança da palavra w é $n(w)$ e a vizinhança pode ser definida apropriadamente para cada aplicação. Essa abordagem faz com que a contagem de palavras fique mais suavizada (*smoothing*), por exemplo, palavras inexatas são consideradas similares [8]. Para controlar isso é utilizado um peso, $a_{w'}$ nessa associação e para computar a contagem de palavras na vizinhança com o peso é gerada a Equação 2.10 [8].

$$N_w^S = \sum_{i=1}^{l-k+1} 1(S[i...i+k-1] = w) \tag{2.9}$$

$$N_{n_{(w)}}^S = \sum_{w' \epsilon n(w)} a_{w'} N_{w'}^S \tag{2.10}$$

Adicionalmente, a variação da contagem de uma palavra específica deve ser considerada e algumas palavras são mais prováveis que ocorrer do que outras. Por exemplo, palavras ricas em GC são menos frequentes em mamíferos do que palavras ricas em AT [8]. A correção para dependência inter-variáveis, variações da contagem de palavras e probabilidade de palavras é feita padronizando a contagem de palavras vizinhas com a Equação 2.11. Já que a contagem de palavras pode ser dependente, a co-variância de todas as palavras na vizinhança deve ser contabilizada para obter $V[N_{n(w)}^S]$.

$$N_w'^S = \frac{N_{n(w)}^S - E[N_{n(w)}^S]}{V[N_{n(w)}^S]} \tag{2.11}$$

A definição da similaridade N2 entre duas sequências é o produto interno do padrão da norma da contagem de palavras vizinhas (Equação 2.12), onde o símbolo $|x|$ representa a norma Euclidiana, onde sequências iguais terão o valor máximo de similaridade entre os pares que é 1, $N2(S1, S2) = 1$.

$$N2(S1, S2) = \sum_{w \epsilon A} \left(\frac{N_w'^{S_1}}{|N'^S|} * \frac{N_w'^{S_2}}{|N'^S|}\right) \tag{2.12}$$

A implementação do N2 faz parte da biblioteca SeqAn [29] e foi incorporada à ferramenta ALF-N2 [8]. Para a execução é necessário um conjunto de arquivos no formato .fasta como entrada e a saída será uma matriz com todos os escores de similaridade entre pares de sequências. O cálculo dos escores é dividido em duas etapas: a primeira faz o pré-processamento e a segunda faz a comparação. No pré-processamento, cada sequência é testada individualmente e o modelo de Markov é estimado [8]. Na etapa da comparação, o produto interno do padrão da norma da contagem de palavras vizinhas é computado

para todos os pares das sequências, como mostrado na Equação 2.12.

A comparação da similaridade entre pares de sequências deve ser eficiente para um uso em larga escala. O algoritmo N2 teve seu desempenho testado com dados simulados, e sequências randômicas foram geradas com o conteúdo de um dinucleotídeo similar ao genoma de um rato, baseado no estudo [30]. Para a comparação dos resultados do N2 foram utilizados diversos outros algoritmos como o D2 [31], o D2z [32] e o D2* [33].

Todos os resultados foram baseados nas comparações com os algoritmos citados acima. Em termos de desempenho, o algoritmo N2 sempre foi mais rápido que os outros algoritmos. Para se computar escores de 5000 pares de sequências aleatórias com tamanho de 1000 bp e com $k = 6$ o N2 demorou 2 horas, sendo que o D2* demorou 42 horas e o D2z demorou 91 horas [8].

Capítulo 3

Multi-platform Architecture for Sequence Aligners (MASA)

O MASA (Figura 3.1) é um *framework* de software flexível que simplifica a criação de aplicações de alinhamento de sequências biológicas com métodos exatos (Seções 2.2 a 2.5) em múltiplas plataformas de hardware como: GPUs de qualquer fabricante, CPUs, Intel PHIS e plataformas de software como o OpenCL [34], CUDA [35], OpenMP e OmpSs [36]. A arquitetura MASA foi desenvolvida por Sandes e co-autores [5], reprojetando a arquitetura do CUDAlign [37] para que as otimizações utilizadas no CUDAlign pudessem ser utilizadas por várias plataformas já que o CUDAlign se executa na arquitetura CUDA da NVIDIA.

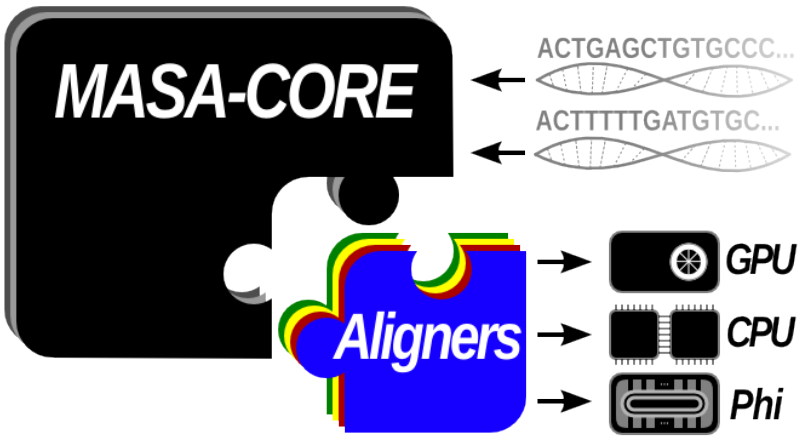


Figura 3.1: Visão geral do *framework* MASA [4].

3.1 Histórico CUDAlign

O CUDAlign é uma ferramenta para comparação de sequências biológicas longas de DNA que utiliza os algoritmos de Gotoh (Seção 2.4) e Myers e Miller (Seção 2.5) em GPU da NVIDIA com arquitetura CUDA. Um breve histórico de suas versões é abordado nas Seções 3.1.1 à 3.1.5.

3.1.1 CUDAlign 1.0

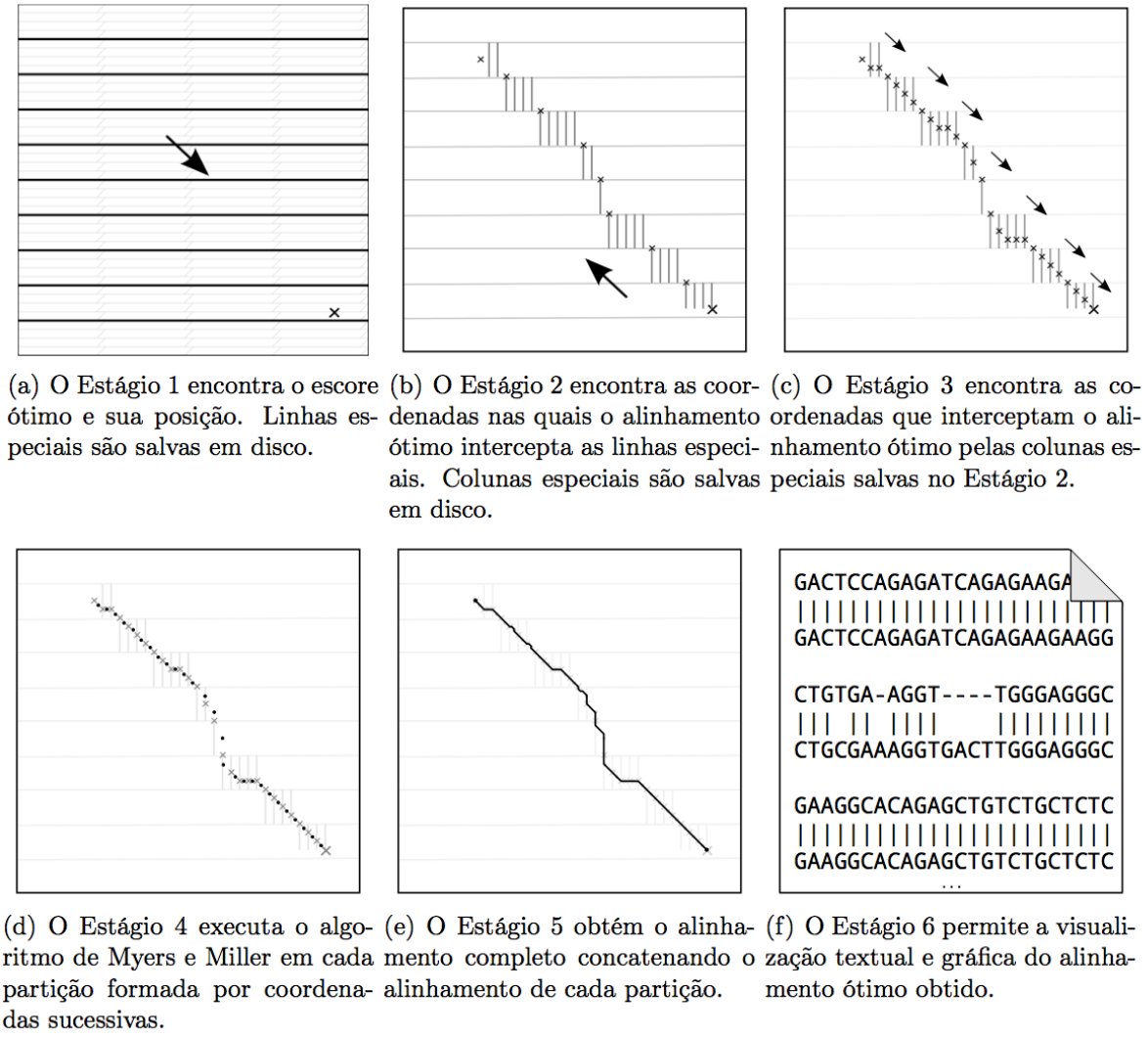
O CUDAlign 1.0 foi a primeira versão do CUDAlign [38]. Ele executa o algoritmo de Gotoh (seção 2.4) em uma GPU com uso linear de memória e retorna o escore ótimo. O paralelismo da GPU é explorado através da técnica de *wavefront* [39] em dois níveis: interno e externo. No paralelismo externo, que representa o primeiro nível, as células da matriz de programação são agrupadas em blocos, esses agrupamentos criam diagonais externas onde a técnica de *wavefront* é aplicada. Blocos de uma mesma diagonal externa são calculados em paralelo. Já no paralelismo interno, ou seja, dentro de um bloco, as *threads* de um mesmo bloco trabalham em paralelo para calcular uma mesma diagonal interna. A otimização *cells delegation* [38] foi proposta para permitir a execução do *wavefront* com paralelogramos, ao invés do tradicional retângulo.

3.1.2 CUDAlign 2.0

O CUDAlign 2.0 [40] executa o CUDAlign 1.0 para obter o escore ótimo. Para o *traceback*, executa uma variante do algoritmo de Myers e Miller (Seção 2.5) e o FastLSA [41] para obter alinhamento local ótimo em tempo reduzido e uso linear de memória para sequências longas de DNA. Sua execução foi dividida em 6 estágios conforme a Figura 3.2. O CUDAlign 1.0 é executado no estágio 1. Como saída desse estágio, temos o escore ótimo e sua posição na matriz de programação dinâmica. Além disso, algumas linhas da matriz de programação dinâmica são salvas em disco.

O *traceback* é executado nos estágios 2 a 5. No estágio 2, a linha especiai salva no estágio 1 que está mais próxima do escore ótimo é lida e uma variante do Myers e Miller (execução ortogonal) é usada para descobrir um ponto nessa linha que pertence ao alinhamento ótimo (*crosspoint*). Esse procedimento é repetido até que o início do alinhamento ótimo seja alcançado. Colunas especiais são salvas em disco. No estágio 3 o número *crosspoints* é aumentado a partir das colunas especiais salvas no estágio anterior. O estágio 4 aplica o algoritmo original de Myers e Miller em cada conjunto de *crosspoints* sucessivos aumentando iterativamente o número de *crosspoints* obtidos. Quando as partições formadas (definidas por *crosspoints* sucessivos) ficam com tamanho

menor que a constante de tamanho máximo de partição, parte-se para a próxima partição. No estágio 5 as partições são concatenadas e utiliza-se o algoritmo de Needleman-Wunsch (Seção 2.2) para se obter o alinhamento ótimo de cada partição e, assim, o alinhamento completo é obtido. O estágio 6 apenas permite a visualização do alinhamento.



(a) O Estágio 1 encontra o escore ótimo e sua posição. Linhas especiais são salvas em disco.

(b) O Estágio 2 encontra as coordenadas nas quais o alinhamento ótimo intercepta as linhas especiais. Colunas especiais são salvas em disco.

(c) O Estágio 3 encontra as coordenadas que interceptam o alinhamento ótimo pelas colunas especiais salvas no Estágio 2.

(d) O Estágio 4 executa o algoritmo de Myers e Miller em cada partição formada por coordenadas sucessivas.

(e) O Estágio 5 obtém o alinhamento completo concatenando o alinhamento de cada partição.

(f) O Estágio 6 permite a visualização textual e gráfica do alinhamento ótimo obtido.

Figura 3.2: Estágios do CUDalign 2.0 [5].

3.1.3 CUDAlign 2.1

O CUDAlign 2.1 [40] propôs a técnica de *Block Pruning*, que descarta blocos da matriz de programação dinâmica que contêm valores de escore tão baixos que é matematicamente impossível que esses blocos façam parte de um alinhamento ótimo. Esse processo pode acelerar em mais de 50% o cálculo da matriz de programação dinâmica para sequências

similares em uma GPU [40]. O *Block Pruning* é aplicado no estágio 1 do CUDAlign, onde o escore máximo não é conhecido.

A otimização por *Block Pruning* possui pouco *overhead* já que a verficação só é feita uma vez para cada bloco e não para cada célula. Porém, caso os blocos sejam muito grandes o procedimento de *pruning* tende a encontrar menos blocos capazes de serem ignorados no cálculo do alinhamento ótimo.

Existem algumas formas diferentes de se realizar o *pruning*, como é possível ver na Figura 3.3 [6]. Em (a) vemos o algoritmo específico por linha onde o processamento é feito por linha ou coluna tornando a complexidade de memória $O(1)$. Em (b) vemos o algoritmo específico por diagonal onde o processamento é feito por diagonal e a complexidade de memória também é $O(1)$. Em (c) vemos o algoritmo genérico que pode ser aplicado em qualquer forma de processamento da matriz, mas pode ter complexidade de memória linear $O(n + m)$ [6].

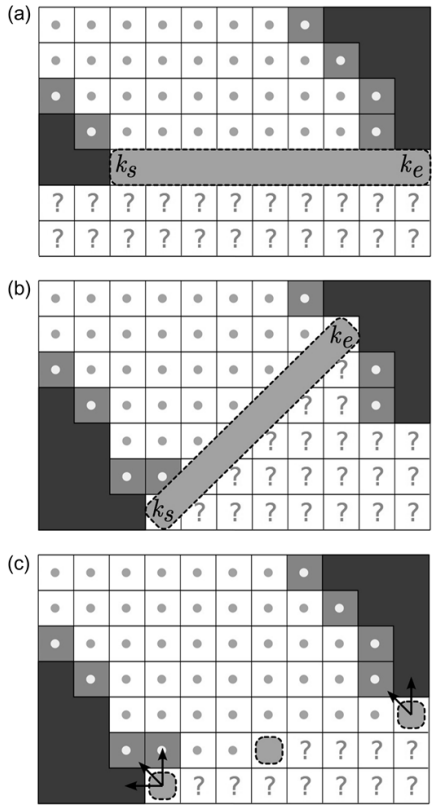


Figura 3.3: Algoritmos de Pruning [6].

3.1.4 CUDAlign 3.0

A versão 3.0 do CUDAlign [42] que introduz a arquitetura com múltiplas GPUs para que uma mesma comparação utilize o poder de várias GPUs simultaneamente. A arquitetura

de Multi-GPU distribui o cálculo de uma única comparação entre várias GPUs, acelerando o estágio 1 do CUDAlign e possibilita encontrar o escore ótimo entre sequências longas de DNA com mais de 100 milhões de pares de base em um tempo razoável [42].

O cálculo da matriz de programação dinâmica é dividido entre as diversas GPUs de maneira que cada GPU calcula um subconjunto de colunas. Para desacoplar computação e comunicação, são usadas *threads* que transmitem os valores calculados da última coluna de cada GPU entre as GPUs vizinhas. Nessa versão, somente o escore é obtido e o *block pruning* é desabilitado.

3.1.5 CUDAlign 4.0

A versão 4.0 do CUDAlign [43] faz o cálculo do *traceback* de alinhamentos ótimos longos com múltiplas GPUs, paralelizando os estágios de 2 a 5 do CUDAlign. A paralelização do estágio 2 em múltiplas GPUs foi desafiadora, já que a obtenção de *crosspoints* é feita de maneira sequencial. Com isso, foi proposta uma técnica de especulação incremental, onde cada GPU (com exceção da última) assume que o máximo valor obtido na sua última coluna pertence ao alinhamento ótimo. Caso a especulação seja correta, o tempo total de cálculo é reduzido. Caso contrário, os valores tem que ser recalculados.

Usando 384 GPUs, o CUDAlign 4.0 obteve todos os alinhamentos entre cromossomos homólogos do homem e do chimpanzé, com taxa de acerto de especulação maior que 80% em todos os casos. Na comparação dos cromossomos homólogos 5 (180 milhões de caracteres x 180 milhões de caracteres) em 384 GPUs, o CUDAlign 4.0 atingiu a taxa de 10.3 TCUPS (Trilhões de células atualizadas por segundo) que, a nosso conhecimento, é a melhor taxa da literatura.

3.2 Arquitetura MASA

Na arquitetura MASA (Multi-platform Architecture for Sequence Aligners) [4], código do CUDAlign foi analisado e separado em duas partes, uma parte contém o código específico de uma plataforma e a outra o código que independe da plataforma utilizada. Também foi proposta uma generalização da técnica de *Block Pruning* (Figura 3.3c) para suportar o processamento da matriz de uma forma assíncrona, onde o processamento de forma diagonal não é necessário.

A divisão da arquitetura aconteceu por meio de 5 módulos que podem ser vistos na Figura 3.4: Gerência de Dados, Comunicação, Estatísticas, Gerenciamento de Estágios e *Aligner*. Os 4 primeiros módulos são reutilizáveis em caso de migração entre plataformas e o módulo *Aligner* (Figura 3.4) deve ser específico para cada plataforma que realizará

o alinhamento. Ele foi dividido em 3 partes: Tipo de Processamento, *Block Pruning* e o módulo que realiza os cálculos da matriz de programação dinâmica.

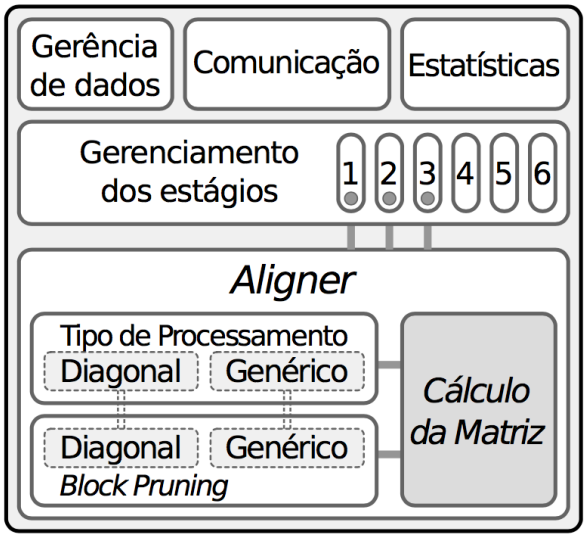


Figura 3.4: Arquitetura MASA [5][4].

O módulo *Aligner* precisa ter código específico para cada plataforma pois é ele que executa a equação de recorrência de Smith-Waterman com *affine gap* (Gotoh) ou Needleman-Wunsch. Porém, uma versão básica do algoritmo de Gotoh é fornecida no MASA de maneira portável [4] e diferentes técnicas de *Block Pruning* podem ser utilizadas dependendo do tipo de processamento que a plataforma utilizará (Figura 3.4).

O módulo de Gerência de dados é responsável pelo controle da entrada e saída do MASA, sendo responsável por tarefas de leitura das sequências de entrada, da geração dos arquivos de saída, do salvamento da informação e da restauração de *checkpoints*, caso a execução seja interrompida.

O módulo de Estatísticas é responsável pela coleta de estatísticas, como informações de quanto tempo cada estágio do alinhamento demorou para ser executado. O módulo de Comunicação é responsável pela comunicação interna, permitindo que o alinhamento seja executado por múltiplas GPUs ou múltiplos dispositivos.

O Gerenciamento de estágios é responsável por coordenar a execução dos estágios 1 a 6, sendo que os estágios 1 a 3 utilizam o módulo Aligner e os estágios 4 a 6 utilizam a CPU. Nos 3 primeiros estágios a matriz de programação dinâmica é particionada e cada partição é enviada para o módulo *Aligner*, que retorna a célula com escore máximo da partição e as linhas ou colunas especiais.Na Figura 3.5, no estágio 1, a partição coincide com o tamanho máximo da matriz e a partir dos estágios 2 e 3 as partições são divididas em várias áreas menores que são definidas pelas linhas especiais dos estágios anteriores e o

crosspoint é definido utilizando o procedimento de *matching* por objetivo [40]. Quando o *crosspoint* é encontrado, o cálculo da partição é encerrado e a próxima partição é enviada ao Aligner. Nos 3 últimos estágios o código é reutilizável e é executado na CPU.

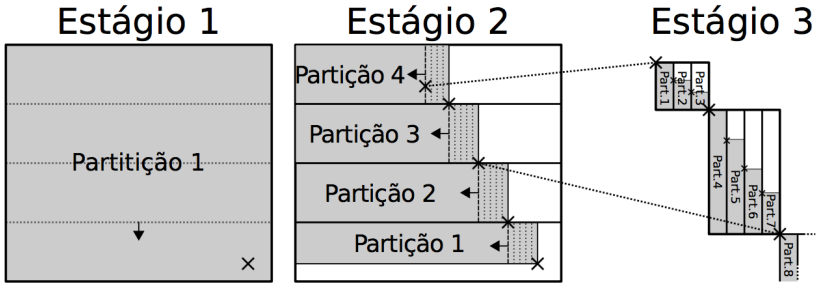


Figura 3.5: Processamento em partições [5].

O módulo *Aligner* possui 3 partes: (a) a definição do tipo de processamento, já que a arquitetura MASA fornece duas formas portáveis para o cálculo da matriz de programação, podendo ser por diagonal ou de maneira genérica; (b)o cálculo da matriz; (c) o módulo responsável pelo *Block Pruning*, que foi implementado de maneira independente da plataforma no MASA por se tratar de uma otimização muito importante. Foram implementados dois algoritmos de *Pruning* que são o Diagonal BP e o Generic BP que podem ser vistos na Figura 3.3.

3.3 Implementações MASA

As implementações de um alinhador que vai utilizar o MASA devem seguir o diagrama de classes da Figura 3.6 e implementar o métodos virtuais da classe IAligner, para que o código independente de plataforma possa processar os argumentos, fazer a leitura das sequências e coordenar a execução dos estágios. Nessa figura, é possível ver como o código foi transformado em independente de plataforma e foi chamado de MASA-API.

Com o desenvolvimento dessa API foram criadas quatro implementações do MASA para tratar plataformas de hardware específicas e que usam diferentes ferramentas de programação, são elas o MASA-OpenMP para CPU, o MASA-OpenMP para arquitetura Phi, o MASA-OmpSs para CPU e o MASA-CUDAlign e na Figura 3.7 [5] é possível ver quais componentes do MASA cada uma das implementações utiliza.

3.3.1 MASA-OpenCL

Em 2015, foi criado o MASA-OpenCL [44] pelo aluno de doutorado Marco Figueirêdo Jr. Além de implementar uma nova extensão do MASA, o MASA-OpenCL propõe uma fór-

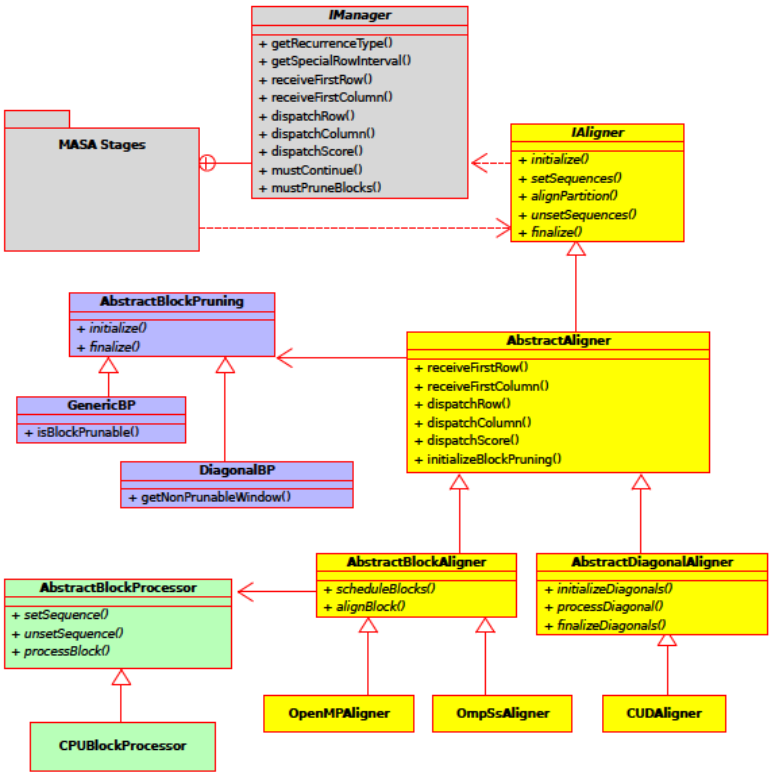


Figura 3.6: Diagrama de Classes MASA-API [5].

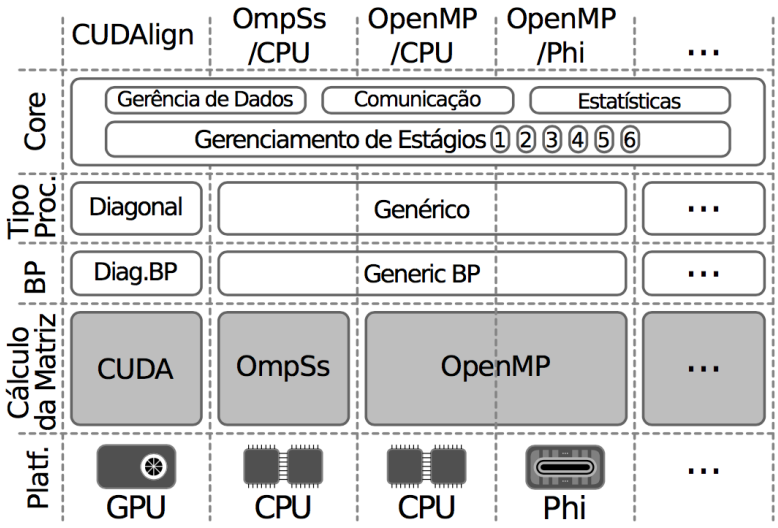


Figura 3.7: Extensões MASA [5].

mula para estimar o tempo de execução da operação de alinhamento através de regressão linear múltipla, baseando-se nas características das sequências, da GPU e da porcentagem de *block pruning* (%BP). Conforme a Equação 3.1, o tempo de execução ($log(t)$) é calculado com base nos tamanhos das sequências (m e n); o CP, que representa o poder computacional da GPU utilizada, sendo calculado através da multiplicação da quantidade de núcleos da GPU pelo *clock*; o BW, que representa a *bandwidth* da GPU, ou seja, a largura de banda do barramento da memória da GPU; o BP, ou seja, a taxa de *block pruning* (%BP) no alinhamento entre as sequências. Um dos trabalhos futuros que foram apontados após a realização do artigo [12] foi a necessidade de se determinar a porcentagem de *block pruning* (%BP) entre as sequências antes da comparação propriamente dita, ou seja, baseado em métricas de similaridade das sequências.

$$\log(t) = -3{,}036 + 0{,}979*\log(m*n) - 0{,}344*\log(CP) - 1{,}001*\log(BW) + 0{,}777*\log(1-BP) \quad (3.1)$$

Capítulo 4

Estratégia para obtenção da taxa de *pruning* com ferramentas *alignment-free*

Esse capítulo descreve a estratégia proposta para a obtenção de uma estimativa da taxa de *pruning* antes da execução do MASA (Capítulo 3). Como a estimativa deve ser obtida rapidamente, para não comprometer o desempenho, foram consideradas as ferramentas *alignment-free* (Seção 2.6). Para a utilização da Equação 3.1 proposta em [12] (Seção 3.3.1) devemos inicialmente obter a similaridade das sequências antes do processo de alinhamento, o que faz com que precisemos de uma técnica que nos dê essa informação de uma maneira rápida e, consequentemente, sem fazer o alinhamento das sequências. Por isso, investigamos a utilização de técnicas de *alignment-free* para descobrir a similaridade.

Outra necessidade encontrada para a previsão da porcentagem de *block pruning* ocorre na execução do MASA com múltiplos dispositivos. No trabalho de doutorado de Figueiredo Jr, será feita a execução com descarte de blocos em múltiplas GPUs, tornando necessário se conhecer a %BP antes do alinhamento entre as sequências, já que essa informação será muito importante para fazer o balanceamento de carga entre as GPUs.

Devido a essa necessidade, o presente trabalho de graduação tem como objetivo analisar diferentes programas que calculam a similaridade entre duas sequências através de técnicas *alignment-free* e verificar se essas técnicas já estão maduras e otimizadas o suficiente para atender essa necessidade de se obter a %BP antes do alinhamento. Consequentemente, desejamos chegar a uma conclusão se uma possível solução para a geração da fórmula que possa derivar a %BP está de fato em algoritmos de *aligment-free*.

A estratégia proposta possui 5 passos. No passo 1, escolhemos 15 pares de sequências reais de DNA com diversos tamanhos e similaridades (Seção 4.1). No passo 2, comparamos as sequências escolhidas com as ferramentas *alignment-free* andi, ALF-N2, Mash e

decaf+py e comparamos suas taxas de similaridade com as taxas de similaridade obtidas com a ferramenta MASA, que obtém o melhor escore em comparações locais e globais (Seção 4.2). Com base nos resultados comparativos, a ferramenta ALF-N2 foi escolhida. No passo 3, foram analisados diversos valores para o parâmetro k do ALF-N2 (Seção 4.3). Tendo a ferramenta e o valor de k, foram geradas no passo 4 diversas regressões relacionando os valores de similaridade gerados no MASA com a taxa de *block pruning*, também do MASA, e uma delas foi escolhida (Seção 4.4). No passo 5, foi usada a regressão escolhida no passo 4 para estimar o valor de *block pruning*, tendo como entrada a similaridade gerada pelo ALF-N2, para 5 novos pares de sequências. Finalmente, o valor de *block pruning* foi usado na fórmula (3.1) para a obtenção do tempo estimado do alinhamento na ferramenta MASA (Seção 4.5).

4.1 Sequências utilizadas nos testes

Foram utilizados 15 pares de sequências diferentes que tinham entre 10 mil pares de bases (10KBP) e 63 milhões de pares de bases (63MBP). A Tabela 4.1 apresenta o tamanho de cada sequência, seus *accession numbers*, a %BP, a similaridade local e a similaridade global ótima de todas as sequências que foram utilizadas. Nessa tabela, o valor de similaridade local foi calculado através da ferramenta MASA utilizando o algoritmo Gotoh (Seção 2.4) e o valor de similaridade global através da ferramenta MASA utilizando o algoritmo Needleman-Wunsch (Seção 2.3).

Como pode ser visto na Tabela 4.1, sequências com similaridade muito alta - $0,998$ na comparação 5M e $0,999$ na comparação 10M - possuem taxa de *block pruning* (%BP) de cerca de 53%. Devido às características do BP, sua taxa máxima fica em torno de 60% [4].

4.2 Análise das Ferramentas *Alignment-Free*

4.2.1 Análise do andi

O primeiro programa testado foi o andi (Seção 2.7.1) [3], que é utilizado para estimar a distância evolucionária entre genomas e não computa o alinhamento total, portanto, é eficiente e rápido.

Ao analisar os resultados da ferramenta andi, como é possível ver na Tabela 4.2, os valores encontrados para a similaridade foram bem diferentes das similaridades local e global encontradas com a ferramenta MASA, que produz os resultados ótimos. Até em comparações de sequências muito parecidas, onde o andi deveria encontrar valores mais

Identificação	Sequência 1	Sequência 2	Porcentagem de BP	Similaridade Local (SW)	Similaridade Global (NW)
10k	10k - AF133821.1	10k - AY352275.1	28.83%	0.297	0.507
57k	57k - NC_001715.1	57k - AF494279.1	0%	0.000	-1.129
162k	162k - NC_000898.1	172k - NC_007605.1	0%	0.000	-1.287
543k	543k - NC_003064.2	536k - NC_000914.1	0%	0.000	-1.092
1M	1M - CP000051.1	1M - AE002160.2	10.96%	0.0845	-1.138
3M	3M- BA000035.2	3M - BX927147.1	0.13%	0.001	-0.845
5M	5M - AE016879.1	5M - AE017225.1	53.70%	0.998	0.998
7M	7M - NC_005027.1	7M - NC_003997.3	0%	0.000	-1.569
10M	10M - NC_017186.1	10M - NC_014318.1	53.34%	0.999	0.999
23M	23M - NT_033779.4	25M - NT_037436.3	0.04%	0.000	-1.192
47M	47M - NC_000021.7	32M - BA000046.3	34.79%	0.829	-0.017
Chr 19	58 M	63M	28.33%	0.292	0.269
Chr 20	63M	61M	44.73%	0.648	0.646
Chr 21	48M	46M	47.08%	0.774	0.754
Chr 22	51M	49M	46.47%	0.633	0.590

Tabela 4.1: Sequências utilizadas nos testes. A identificação reflete como elas foram mostradas nas outras tabelas

altos de similaridade genética entre as sequências isso não ocorreu, portanto, a ferramenta foi descartada e não foram feitos mais testes.

4.2.2 Análise do ALF-N2

A segunda ferramenta testada foi o ALF-N2 (Seção 2.7.3). Os resultados obtidos apresentam uma melhora considerável se comparados com o andi (Seção 4.2.1), sem comprometer a velocidade. Utilizado para calcular a similaridade entre os pares de uma comparação, sua implementação é baseada no método de contagem de *k-mer*, que conta a frequência de todas as possíveis sub-séries de letras dentro de uma série de letras. Uma vantagem do algoritmo N2 é que ele extende a contagem para palavras vizinhas, cobrindo *matches* entre

Sequência	10k	57k	162k	543k	1M	3M	5M	7M	10M	23M	47M
Tempo (s)	0,117	0,126	0,174	0,409	0,793	2,614	4,360	5,835	9,24	26,62	34,39
Similaridade Andi	0,097	0,0326	NaN	0,262	0,1834	0,203	0,002	0,239	0,001	0,007	0,011
Similaridade Local (SW)	0,297	0,000	0,000	0,000	0,084	0,001	0,998	0,000	0,999	0,000	0,829
Similaridade Global(NW)	0,507	-1,129	-1,287	-1,092	-1,138	-0,84	0,998	-1,56	0,999	-1,19	-0,01

Tabela 4.2: Resultados do andi

palavras aproximadas e independentes da orientação. Portanto, o N2 tem boa performance para encontrar similaridade entre sequências baseado em palavras compartilhadas entre as sequências.

Sequência	10k	57k	162k	543k	1M	3M	5M	7M	10M	23M	47M
K = 4	0,792	0,099	0,571	0,940	0,982	0,818	0,999	0,239	1,000	0,997	0,999
K = 8	0,377	0,069	0,058	0,496	0,731	0,696	0,999	0,182	0,999	0,979	0,995
Tempo(s)	0,005	0,025	0,027	0,097	0,141	0,390	0,528	0,705	1,184	2,666	4,004
Similaridade Local (SW)	0,297	0,000	0,000	0,000	0,084	0,001	0,998	0,000	0,999	0,000	0,829
Similaridade Global (NW)	0,507	-1,129	-1,287	-1,092	-1,138	-0,84	0,998	-1,56	0,999	-1,19	-0,01

Tabela 4.3: Resultados do ALF-N2 onde o k é o tamanho máximo da série de letras

Com os resultados da Tabela 4.3 foi possível ver que alguns valores encontrados podiam ser relacionados com a similaridade local e global entre as sequências, como no caso de comparações de sequências com 99% de similaridade no ALF-N2 e que tinham 99% de similaridade local e global na ferramenta MASA (5M e 10M). O resultado da comparação de 47M ($0,995$ no ALF-N2 e $0,829$ na similaridade local) também ficou bom. Também foi possível perceber que quando o k aumentava, a similaridade do ALF-N2 ficava mais próxima da similaridade local. Portanto, seria interessante fazer testes alterando os valores de k para melhorar os resultados encontrados e ver se a correlação melhorava.

4.2.3 Análise do Mash e decaf+py

Os resultados do Mash (Seção 2.7.2) foram computados e analisados conforme a Tabela 4.4. Pode ser visto que a similaridade do Mash (*Matching-hashes*) é próxima da similaridade local SW quando as sequências são muito similares (5M e 10M). O resultado da comparação de 47M, no entanto, não é tão bom como o apresentado pelo ALF-N2.

Sequência	10k	57k	162k	543k	1M	3M	5M	7M	10M	23M	47M
Tempo (s)	0,018	0,058	0,133	0,450	0,868	2,647	4,300	5,152	8,440	20,08	30,24
Mash-distance	0,090	1	1	1	0,174	0,230	0	1	0	0,203	0,017
Matching-Hashes	0,08	0	0	0	0,001	0	0,999	0	0,999	0	0,536
Similaridade Local (SW)	0,297	0,000	0,000	0,000	0,084	0,001	0,998	0,000	0,999	0,000	0,829
Similaridade Global (NW)	0,507	-1,129	-1,28	-1,09	-1,13	-0,84	0,998	-1,56	0,999	-1,19	-0,01

Tabela 4.4: Resultados do Mash

O programa chamado decaf+py [45] e [46] também foi analisado, porém por retornar tempos de execução muito altos acabou sendo descartado, já que o objetivo do trabalho era encontrar programas que geram a similaridade entre sequências sem utilizar o alinhamento e de maneira rápida, para que esses valores possam ser utilizados quando as sequências forem de fato alinhadas. Por exemplo, a comparação das sequências de $57k$ demorou 10 minutos.

4.2.4 Discussão

A análise comparativa entre os resultados obtidos com as ferramentas andi, ALF-N2 e Mash mostra que a ferramenta andi apresenta resultados muito distantes das similaridades calculadas com métodos exatos de alinhamento local e global obtidos com a ferramenta MASA. Em particular, as comparações de 5M e 10M, que possuem similaridade maior que 97%, retornaram no andi similaridades de 2% e 1%, respectivamente.

As ferramentas ALF-N2 e Mash apresentaram muito bons resultados para sequências muito díspares e muito similares. Para sequências de similaridade média (e.g. 47M), o ALF-N2 apresentou resultados melhores. Na comparação entre os tempos todas as três ferramentas executaram em poucos segundos.

Com base nos resultados apresentados nas Tabelas 4.2 a 4.4, escolhemos a ferramenta ALF-N2 para a determinação rápida da similaridade.

4.3 Definição do parâmetro k da ferramenta ALF-N2

Para a determinação do parâmetro k foi feita uma análise específica do ALF-N2 variando os valores de k e fazendo testes no computador do laboratório LAICO e no ambiente Casa. Para essa análise o ambiente Casa tinha como hardware um processador Intel Core i7-4710HQ CPU @2.50GHz, 15,7 GB de memória RAM e sistema operacional Ubuntu 16.04 LTS com o GCC na versão 5.4.0. Já o ambiente chamado de LAICO tinha como hardware um processador Intel Xeon CPU e5645 @ 2.40 GHz, 11,7 GB de memória RAM e sistema operacional Ubuntu 14.04 com o GCC na versão 4.9.4. Nessa segunda análise os tempos de execução do ALF-N2 também foram computados e comparados, já que é fator determinante na escolha do melhor programa para a tarefa.

Sequência	10k	57k	162k	543k	1M	3M	5M	7M	10M	23M	47M
K = 4	0,792	0,099	0,571	0,940	0,982	0,818	0,999	0,239	1,000	0,997	0,999
K = 8	0,377	0,069	0,058	0,496	0,731	0,696	0,999	0,182	0,999	0,979	0,995
K = 11	0,268	0,024	0,002	0,032	0,175	0,285	0,999	0,056	0,999	0,823	0,989
K = 12	0,241	0,017	0,000	0,010	0,123	0,155	0,999	0,020	0,999	0,714	0,986
K = 13	0,219	0,013	0,000	0,003	0,102	0,090	0,998	0,006	0,999	0,602	0,982
Similaridade Local (SW)	0,297	0,000	0,000	0,000	0,084	0,001	0,998	0,000	0,999	0,000	0,829
Similaridade Global (NW)	0,507	-1,129	-1,287	-1,092	-1,138	-0,84	0,998	-1,56	0,999	-1,19	-0,01

Tabela 4.5: Resultados do ALF-N2 com a variação do k na máquina LAICO

Após a coleta dos resultados, o ALF-N2 mostrou melhores resultados tanto para o k igual a 12 tanto para o k igual a 13, como é possível ver na Tabela 4.5. Os valores foram exatamente iguais independente do ambiente testado (Casa ou LAICO). Com $k = 12$ e $k = 13$, a maior parte das comparações teve resultados muito próximos das similaridades locais. Por exemplo, para a comparação de 23 M é possível ver que os valores foram melhorando a medida que o k aumentou, mas ainda foram obtidos valores distantes para os maiores valores de k testados. O k igual a 16 foi testado nos dois ambientes, mas o programa retornou "*segmentation fault*". Analisando as Tabelas 4.5 e 4.6, o valor de k igual a 12 foi escolhido como o de melhor custo/benefício (qualidade do resultado x tempo de execução) e, portanto, será o utilizado quando o ALF-N2 for executado.

Ambiente Casa		Ambiente Laboratório	
Tamanho do K	Tempo médio	Tamanho do K	Tempo médio
K = 11	3 minutos	K = 11	1 minuto
K = 12	13 minutos	K = 12	2 minutos
K = 13	~ 1 hora	K = 13	~ 1 hora

Tabela 4.6: Tempo de execução das 15 comparações nos 2 ambientes

4.4 Regressões para obtenção da taxa de *Block Pruning*

Tendo escolhido a ferramenta ALF-N2 ($k = 12$), o próximo passo foi a obtenção de regressões que correlacionassem a similaridade com a taxa de *block pruning*. Essa correlação não é óbvia, pois o maior valor de similaridade é 100% e o maior valor de *block pruning* fica em torno de 60% [6].

Para gerar as regressões foram utilizados os valores de *block pruning* e suas respectivas similaridades geradas pelo MASA, que podem ser vistos na Tabela 4.1. Nesse caso foram consideradas tanto a similaridade de *Smith-Waterman* (SW) que gera o alinhamento local e a similaridade de *Needleman-Wunsch* (NW) que gera o alinhamento global. O cálculo da similaridade foi feito dividindo-se o escore gerado pelo MASA pelo tamanho da menor sequência de cada comparação. Como no caso do alinhamento global, *Needleman-Wunsch*, houve valores negativos para a similaridade, removemos esses valores para construir os gráficos.

A partir dos gráficos gerados foram traçadas diferentes curvas, variando a potência da função polinomial 2, 3 e 4 para as similaridades locais e globais. Cada uma das seis curvas tem sua equação apresentada em sua respectiva figura (Figuras 4.1, 4.2, 4.3, 4.4, 4.5 e 4.6).

Após a comparação entre os valores de *block pruning*, o objetivo foi identificar qual equação mais se aproxima dos pontos. Os valores que utilizavam a similaridade global foram desconsiderados após uma primeira análise pois as equações foram geradas com menos pontos, uma vez que os valores negativos foram descartados. A análise visual das regressões para alinhamento local com curvas polinomiais potência de 2 (Figura 4.4), 3 (Figura 4.5) e 4 (Figura 4.6) mostra que a regressão com curva polinomial potência de 4 mais se aproxima dos pontos. Sendo assim, a Equação 4.1, onde x é o valor da similaridade, foi escolhida.

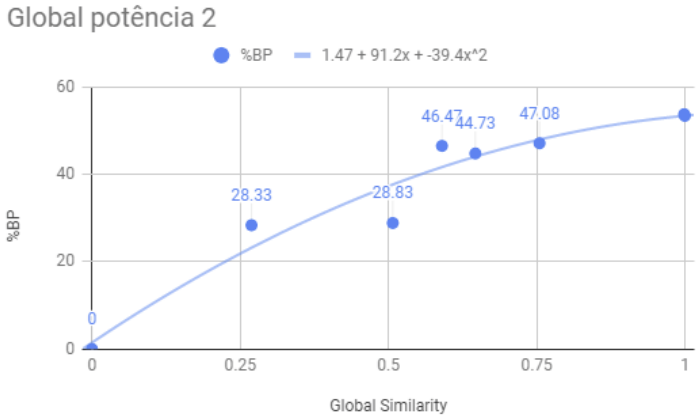


Figura 4.1: Gráfico similaridade global x *block pruning* com a curva polinomial com potência 2 .

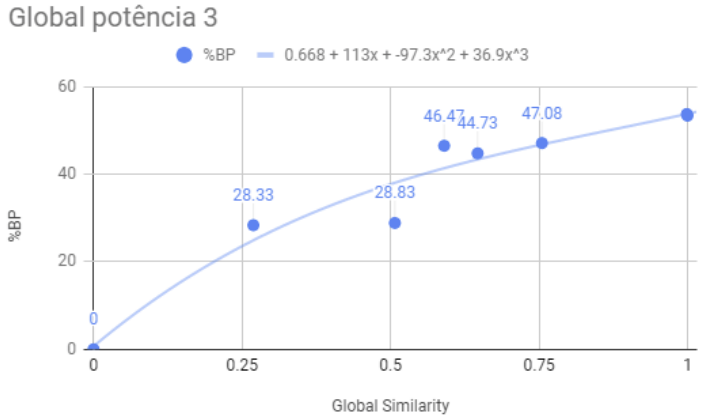


Figura 4.2: Gráfico similaridade global x *block pruning* com a curva polinomial com potência 3 .

$$BP = 0,375 + 45,4x + 417x^2 - 919x^3 + 510x^4 \tag{4.1}$$

4.5 Validação da regressão para previsão de %BP

Para validar a Equação 4.1, foram utilizados 5 novos pares de sequências de tamanhos (35k a 280k) e similaridades (0 a 83,2%) variados. A Tabela 4.7 apresenta as sequências comparadas, a similaridade do ALF-N2, a %BP real, obtida com a execução da comparação no MASA, a %BP estimada com a Equação 4.1 e o erro.

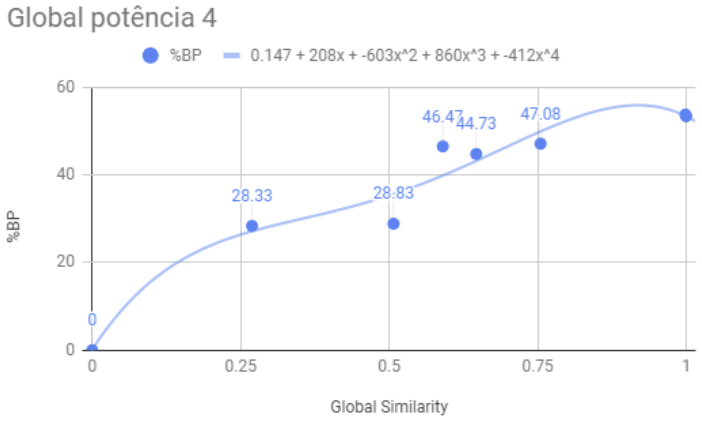


Figura 4.3: Gráfico similaridade global x *block pruning* com a curva polinomial com potência 4 .

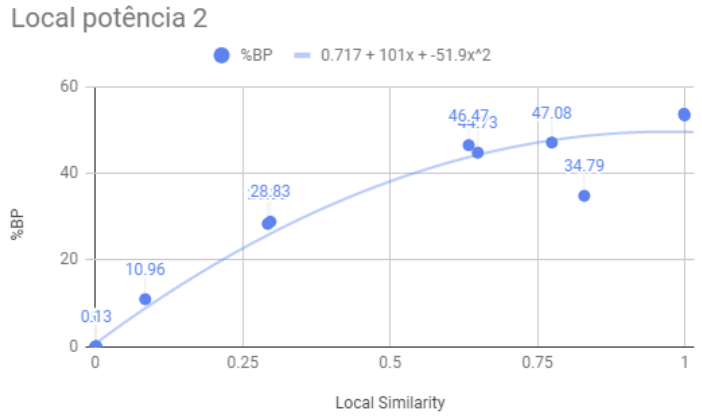


Figura 4.4: Gráficos similaridade local x *block pruning* com a curva polinomial com potência 2.

Os valores de *block pruning* obtidos com a Equação 4.1 (%BP estimado - Local pot 4 na Tabela 4.7) foram usados na Equação 3.1, já que o objetivo é utilizar as %BP geradas de forma rápida através do *alignment-free* nessa equação. A GPU utilizada para gerar os demais valores foi uma GTX 680, com o valor do CP na equação sendo 1545216 e o BW sendo 192,2. A Tabela 4.8 apresenta os tempos gerados através da Equação 3.1 com os valores de *block pruning* reais e os valores gerados através da regressão polinomial potência 4 (Equação 4.1 e Figura 4.6). É interessante perceber que os valores de tempo gerados com a equação estão bem próximos aos obtidos com os valores reais de *block pruning*, o que sugere que os valores gerados são adequados para a utilização na Equação 3.1. Sendo assim, mostramos que não é necessário rodar o MASA para se obter uma aproximação muito boa do *block pruning* real em poucos minutos.

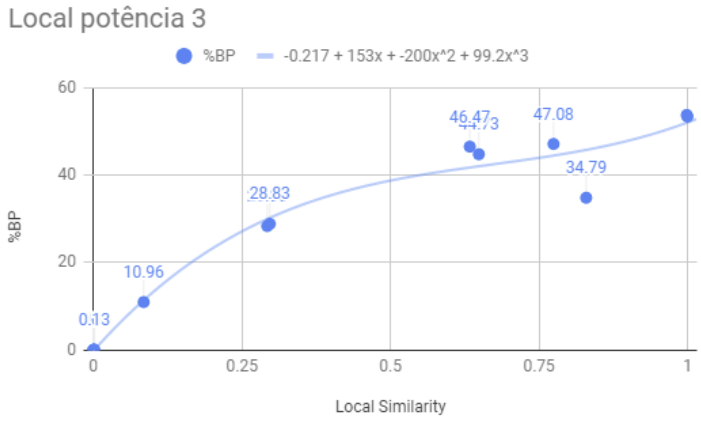


Figura 4.5: Gráficos similaridade local x *block pruning* com a curva polinomial com potência 3.

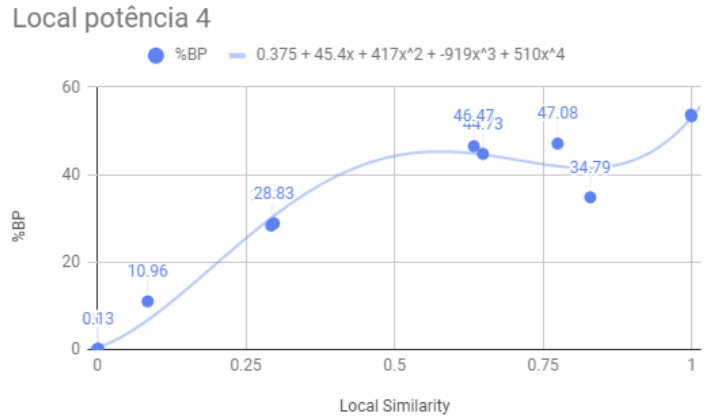


Figura 4.6: Gráficos similaridade local x *block pruning* com a curva polinomial com potência 4.

Identificação	Seq 1	Seq 2	Similaridade ALF	% BP Real	%BP Estimado – Local pot4	Erro (%)
adenovirus	35k	35k	0.651	37.68%	44.69%	+ 7,01%
gordonia	94k	94k	0.832	38.14%	41.90%	+ 3,76%
pseudomas	280k	311k	0.010	0.36%	0.91%	+0,55%
streptococcus	79k	69k	0.001	0.72%	0.45%	-0,27%
mycobacterium	155k	132k	0.004	0.00%	0.56%	+0,56%

Tabela 4.7: Valores de *block pruning* gerados utilizando os gráficos

Identificação	t(s) estimado BP Real	t(s) estimado BP Gerado	Erro (%)
adenovirus	19,349	17,635	- 8.854
gordonia	133,125	126,794	- 4.755
pseudomas	1810,98	1803,20	- 0.429
streptococcus	119,81	120,071	+ 0.210
mycobacterium	439,872	437,957	- 0.435

Tabela 4.8: Tempos gerados com a equação de regressão proposta (Equação 4.1) e com o *block pruning* real.

Capítulo 5

Conclusão

Este trabalho de graduação fez a análise e a avaliação da utilização de métodos de comparação entre sequências biológicas que não se baseiam no alinhamento das sequências (*alignment-free*) visando entender se faz sentido a utilização desses métodos para a descoberta da porcentagem de *block pruning* entre as sequências. Vários algoritmos diferentes e que utilizam técnicas diferentes foram analisados e o algoritmo chamado N2 da ferramenta ALF-N2 foi o escolhido para uma análise mais profunda por causa dos resultados aderentes.

Após os resultados iniciais do ALF-N2, foram feitos testes com as mesmas sequências alterando o tamanho das palavras (k-mers) que o algoritmo utiliza. Esses testes adicionais se mostraram muito bons já que o tempo de execução aumentou de alguns segundos para pouco mais de alguns minutos e os resultados ficaram cada vez mais próximos da %BP encontrada na ferramenta MASA. Isso reforça a ideia inicial de utilizar técnicas para estimar a %BP antes da execução do alinhamento propriamente dito entre as sequências visando acelerar esse processo. Portanto, os resultados foram bem promissores e mesmo com alguns falsos positivos encontrados, a elaboração de uma correlação entre os valores pode ser obtida sem um grande esforço. Foi verificado que o melhor tamanho de palavra ocorreu com o *k-mer* igual a 12, com tempo de execução reduzido e ao mesmo tempo os melhores resultados para serem comparados com a %BP.

Após isso, foram criadas regressões que fazem a correlação entre a similaridade e o *block pruning* gerado no MASA. Com isso, utilizou-se os valores de similaridade gerados no ALF-N2 nas regressões para gerar a %BP. Os valores de *block pruning* obtidos com as regressões foram bem interessantes e ficaram dentro de uma variação aceitável com relação aos valores reais para a regressão polinomial potência 4, obtida no presente trabalho. O valor de %BP obtido com a regressão foi utilizado na equação de predição do tempo de execução do alinhamento, que depende da %BP. Os valores gerados de *block pruning* levaram a bons resultados quando comparados com os valores reais, com erro máximo de

8,85%. Sendo assim, os objetivos do presente trabalho de graduação foram plenamente atingidos pois com a estratégia aqui proposta é possível gerar a %BP sem a execução do alinhamento.

Como trabalhos futuros sugerimos utilizar a estratégia proposta no presente trabalho de graduação na ferramenta MASA, para descobrir se existe realmente uma melhora de desempenho quando se conhece os valores de *block pruning* previamente. Também seria interessante adaptar a estratégia aqui proposta para a execução do MASA com múltiplas GPUs, onde será muito importante a estimativa da %BP antes da execução, para fins de balanceamento de carga.

Referências

[1] Myers, Eugene W. e Webb Miller.: *Optimal alignments in linear space.* Comput. Appl. Biosci., 1988. ix, 8, 9

[2] Andrzej Zielezinski, Susana Vinga, Jonas Almeida Wojciech M. Karlowski: *Alignment-free sequence comparison: benefits, applications, and tools.* Zielezinski et al. Genome Biology (2017) 18:186, 2017. ix, 1, 9, 10, 11, 12, 13

[3] Bernhard Haubold, Fabian Klötzl, Peter Pfaffelhuber: *andi: Fast and accurate estimation of evolutionary distances between closely related genomes.* Bioinformatics, Volume 31, Issue 8, 15 April 2015, Pages 1169–1175,, 2015. ix, 13, 14, 29

[4] Edans F. O. Sandes, Guillermo Miranda, E. Ayguade Xavier Martorell G. Teodoro e Alba C. M. A. De Melo: *Masa: A multiplatform architecture for sequence aligners with block pruning.* ACM Transactions on Parallel Computing, 2016. ix, 2, 19, 23, 24, 29

[5] Sandes, E. F. O.: *Algoritmos paralelos exatos e otimizações para alinhamento de sequências biológicas longas em plataformas de alto desempenho.* Tese (Doutorado), Departamento de Ciência da Computação, Universidade de Brasília, Brasília, DF,, 2015. ix, 19, 21, 24, 25, 26

[6] Edans F O Sandes, George L M Teodoro, Maria Emilia M T Walter Xavier Martorell Eduard Ayguade Alba C M A Melo: *Formalization of block pruning: Reducing the number of cells computed in exact biological sequence comparison algorithms.* The Computer Journal, Volume 61, Issue 5, 1 May 2018, Pages 687–713,, 2018. ix, 22, 34

[7] Mount, David W.: *Sequence and genome analysis.* New York: Cold Spring, 2004. 1, 4, 5

[8] Jonathan Goeke, Marcel H. Schulz, Julia Lasserre e Martin Vingron: *Estimation of pairwise sequence similarity of mammalian enhancers with word neighbourhood counts.* Bioinformatics (2012), 2012. 1, 16, 17, 18

[9] Brian D. Ondov, Todd J. Treangen, Páll Melsted Adam B. Mallonee Nicholas H. Bergman Sergey Koren e Adam M. Phillippy: *Mash: fast genome and metagenome distance estimation using minhash.* Genome Biology, 2016. 1, 15, 16

[10] Vinga, Susana: *Information theory applications for biological sequence analysis.* Briefings in Bioinformatics, Volume 15, Issue 3, May 2014. 1, 2

[11] Melo, Marco Antonio C. de Figueiredo Jr. Edans F. de Oliveira Sandes Genaina N. Rodrigues George L. M. Teodoro Alba Cristina M. A. de: *Masa-opencl: Parallel pruned comparison of long dna sequences with opencl.* Concurrency Computat Pract Exper., 2018. 2

[12] Figueirêdo Júnior, Marco Antônio Caldas de: *Masa-opencl: Parallel pruned comparison of long dna sequences with opencl.* Concurrency Computat Pract Exper. 2018;e5039., 2018. 2, 27, 28

[13] Nicholas M. Luscombe, Dov Greenbaum e Mark Gerstein.: *What is bioinformatics? a proposed definition and overview of the field.* Methods of information in medicine, 2001. 4

[14] Needleman, Saul B. e Christian D. Wunsch: *A general method applicable to the search for similarities in the amino acid sequence of two proteins.* Journal of Molecular Biology, 1970. 5

[15] M. S. Waterman., T. F. Smith e: *Identification of common molecular subsequences.* Journal of Molecular Biology, 1981. 6, 7

[16] Gotoh., Osamu: *An improved algorithm for matching biological sequences.* Journal of Molecular Biology,, 1982. 8

[17] Hirschberg., D. S.: *A linear space algorithm for computing maximal common subsequences.* Commun. ACM,, 1975. 8, 9

[18] Susana Vinga, Jonas Almeida: *Alignment-free sequence comparison—a review.* Bioinformatics, Volume 19, Issue 4, March 2003. 10

[19] Kullback S, Leibler RA: *On information and sufficiency.* The annals of mathematical statistics, 1951. 13

[20] Susana Vinga, Rodrigo Gouveia-Oliveira, Jonas S. Almeida: *Comparative evaluation of word composition distances for the recognition of scop relationships.* Bioinformatics, Volume 20, Issue 2, January 2004. 13

[21] Höhl M, Ragan MA: *Is multiple-sequence alignment required for accurate inference of phylogeny?* Systematic Biology, 2007. 13

[22] GenBank e WGS Statistics.: *Genbank and wgs statistics.* http://www.ncbi.nlm.nih.gov/genbank/statistics, Accessed 2019. 15

[23] AZ., Broder: *On the resemblance and containment of documents.* Compression and Complexity of Sequences, 1997. 15

[24] Konstantinidis KT, Tiedje JM.: *Genomic insights that advance the species definition for prokaryotes.* Proc Natl Acad Sci U S A., 2005. 15, 16

[25] Fan H, Ives AR, Surget Groba Y Cannon CH.: *An assembly and alignment-free method of phylogeny reconstruction from next-generation sequencing data.* BMC Genomics., 2015. 15

[26] Pruitt KD, Tatusova T, Brown GR Maglott DR: *Ncbi reference sequences (refseq): current status, new features and genome annotation policy.* Nucleic Acids Res, 2012. 15

[27] Zinzen, R.P. et al.: *Early and late periodic patterns of even skipped expression are controlled by distinct regulatory elements that respond to different spatial cues.* Cell, Nature. 16

[28] Kantorovitz, M.R. et al. (2007): *Motif-blind, genome-wide discovery of cis-regulatory modules in drosophila and mouse.* Dev. Cell,, 2009. 16

[29] Döring A, Weese D, Rausch T Reinert K: *Seqan an efficient, generic c++ library for sequence analysis.* BMC Bioinformatics, 2008. 17

[30] Thomas-Chollier, M. et al.: *Rsat 2011: regulatory sequence analysis tools.* Nucleic Acids Res.,, 2011. 18

[31] Carpenter, J.E. et al.: *Assessment of the parallelization approach of d2-cluster for high-performance sequence clustering.* J. Comput. Chem.,, 2002. 18

[32] Kantorovitz, M.R. et al. (2007): *A statistical method for alignment-free comparison of regulatory sequences.* Bioinformatics, 2007. 18

[33] Reinert, G. et al.: *Alignment-free sequence comparison (i): Statistics and power.* J. Comput. Biol., 2009. 18

[34] E. Stone, D. Gohara e G. Shi: *Opencl: A parallel programming standard for heterogeneous computing systems.* Computing in science and engineering vol.12, 2010. 19

[35] Nvidia: *Cuda c programming guide.* NVIDIA Corporation, 2014. 19

[36] A. Duran, E. Ayguade, R.M. Badia J. Labarta L. Martinell X. Martorell e J. Planas: *Ompss: A proposal for programming heterogeneous multicore architectures.* Parallel Processing Letters, 2011. 19

[37] Sandes, E. F. O.: *Comparação paralela de sequências biológicas longas utilizando unidades de processamento gráfico (gpus).* Tese (Mestrado), Departamento de Ciência da Computação, Universidade de Brasília, Brasília, DF,, 2011. 19

[38] Edans Flavius O. Sandes, Alba Cristina M.A. de Melo: *Cudalign: using gpu to accelerate the comparison of megabase genomic sequences.* Proceedings of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 2010. 20

[39] Pfister, G.F.: *In search of clusters: the coming battle in lowly parallel computing.* Prentice-Hall, Inc, 1995. 20

[40] A. C. M. A Melo, E. F. O. Sandes e: *Retrieving smith-waterman alignments with optimizations for megabase biological sequences using gpu.* IEEE Transactions on Parallel and Distributed Systems,, 2013. 20, 21, 22, 25

[41] A. Driga, P. Lu, J. Schaeffer D. Szafron K. Charter e I. Parsons: *Fastlsa: A fast, linear-space, parallel and sequential algorith for sequence alignment.* Algorithmica, 2006. 20

[42] Edans F. O. Sandes, Guillermo Miranda, Alba Cristina de Melo Xavier Martorell and Eduard Ayguade: *Cudalign 3.0: Parallel biological sequence comparison in large gpu clusters.* 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing,, 2014. 22, 23

[43] Oliveira Sandes, Guillermo Miranda, Xavier Martorell Eduard Ayguade George Teodoro Edans Flavius de e Alba Cristina Magalhaes Melo.: *Cudalign 4.0: incremental speculative traceback for exact chromosome-wide alignment in gpu clusters.* IEEE Transactions on Parallel and Distributed Systems, 2016. 23

[44] Figueiredo Jr., Edans F. de Oliveira Sandes, Genaina N. Rodrigues George L. M. Teodoro Alba Cristina M. A. de Melor Marco Antonio C. de: *Masa-opencl: Comparação paralela de sequências biológicas longas em gpu.* Dissertação apresentada como requisito parcial para conclusão do Mestrado em Informática, 2015. 25

[45] Höhl M, Rigoutsos I, Ragan MA.: *Pattern-based phylogenetic distance estimation and tree reconstruction.* Evol Bioinform Online, 2006. 32

[46] Höhl M, Ragan MA.: *Is multiple-sequence alignment required for accurate inference of phylogeny?* Syst Biol., 2007. 32