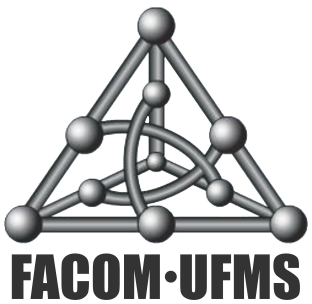
Projeto de Primers via Árvore de Sufixos

Trabalho de Conclusão de Curso

Bacharelado em Ciência da Computação

Nariélly Calista Farias

Faculdade de Computação
Universidade Federal de Mato Grosso do Sul



Orientadora: Prof. Dra. Luciana Montera

Campo Grande, Dezembro de 2010

Sumário

Lista de Figuras **1**

1 Introdução **4**

1.1 Apresentação 4

1.2 Justificativa 5

1.3 Objetivos 6

1.4 Organização 6

2 Conceitos de Biologia Molecular **7**

2.1 Material Genético 7

2.1.1 DNA e RNA 7

2.1.2 Replicação 8

2.1.3 *Primers* 9

2.2 Reação de PCR 10

2.2.1 PCR Múltipla 14

2.3 Projeto de *Primers* 14

2.3.1 Características (Desejáveis) dos *Primers* 14

2.3.2 Projeto de *Primer* para Reação de PCR Múltiplas 19

2.3.3 Ferramentas Computacionais para o Projeto de *Primers* 19

3 Conceitos de Computação **21**

3.1 Introdução 21

3.2 Definições Básicas e Notação 22

3.3 Busca de *Strings* 22

3.3.1 Visão Geral 22

3.3.2 Algoritmo Força Bruta 23

3.4 Árvore de Sufixos 24
3.4.1 Definições Básicas 24
3.4.2 Busca em Árvore de Sufixos 25
3.4.3 Construção de Árvore de Sufixos 27
3.4.4 Árvore de Sufixos Generalizada 28
3.4.5 Contextualização em Relação à PCR Múltipla 28
3.4.6 Relação entre Árvore de Sufixos com o Projeto de Primer 29

4 Implementação **30**
4.1 O que é *libstree*? 30
4.2 O que fornece *libstree*? 30
4.3 Como *libstree* foi utilizada? 31
4.4 Funcionalidades 32

5 Resultados **34**

6 Conclusão e Trabalhos Futuros **38**

Lista de Figuras

2.1 Bases nitrogenadas que compõe o DNA. Baseada em [17]. 8

2.2 A dupla hélice do DNA. Retirada de [1]. 9

2.3 Ciclos de PCR com *primers*. A cada ciclo, uma molécula de DNA dá origem a duas novas moléculas idênticas à molécula original. 12

2.4 Atuação da enzima DNA polimerase na replicação do DNA. (a) A polimerase é capaz de estender o *primer1* criando uma fita complementar ao molde. (b) A polimerase não é capaz de estender o *primer2* - uma extensão não ocorre a partir da extremidade $5'$. Baseada em [14]. 13

2.5 *Primer* com *repeats* (ATC). Os *matches* ocorridos no *annealing* entre o *primer* e a fita molde são representados pelo símbolo |, enquanto que os *mismatches* são representados por $*$. Baseada em [14]. 15

3.1 Exemplo de comparação de sequências $s = AGTCATCTCTCGAC$ e $p = TCTCTCGA$ através do algoritmo força-bruta. Dizemos que ocorre um casamento entre as sequências s e p na posição $d = 5$. 23

3.2 Árvore de sufixos para a sequência $xabxa\$$. 25

3.3 Árvore de sufixos para a sequência $xabxa$ (sem o símbolo especial \$). . . . 26

3.4 Exemplo de construção da árvore de sufixos para a sequência $xabxa\$$. . . . 27

3.5 Exemplo da relação entre árvore de sufixos com o projeto de primer. 29

5.1 Gráfico de desempenho da geração da árvore de sufixos. 35

5.2 Gráfico de desempenho da busca da maior *substring* comum. 36

5.3 Gráfico de desempenho da busca de todas as *substrings* comums entre min=5 e max=10. 36

5.4 Gráfico de desempenho da busca de todas as *substrings* específicas entre min=5 e max=10. 37

Resumo

O presente trabalho aborda o problema de busca de *primers* via construção de uma árvore de sufixos para sequências de DNA. A solução do problema via árvore de sufixos é viável, uma vez que, com árvore de sufixos, é possível identificar regiões específicas e regiões em comum entre sequências formadas por um mesmo alfabeto. O objetivo principal é a criação de uma ferramenta para auxiliar o projeto de *primers* para reações de PCR que envolvem, ao mesmo tempo, múltiplas sequências de DNA, chamadas de reação de PCR múltiplas, e cujo objetivo é a amplificação de uma ou mais sequências específicas de um conjunto de sequências. Em um experimento de PCR múltipla, a escolha dos *primers* não é mais dependente apenas da composição da sequência que se deseja amplificar, mas também das demais presentes na reação, visto que os *primers* selecionados devem ser específicos para a sequência alvo.

O resultado obtido foi um *software* cuja funcionalidade é a determinação de regiões possíveis para o projeto de *primers* em sequências de DNA. Mais especificamente, dado um conjunto de $n \geq 2$ de sequências de DNA passadas como entrada e um conjunto de restrições para guiar a escolha de *primers*, o usuário terá como saída trecho(s) de DNA(s) onde a busca por pares de *primers* pode ocorrer. Esse tipo de aplicação deve ser capaz de permitir o manuseio de milhares de bases passados para a consulta, portanto algumas falhas acontecidas ao longo da execução do projeto foram relacionadas a biblioteca utilizada na implementação, devido a sua limitação na quantidade de bases que esta tem a capacidade de processar.

Palavras chaves: Árvore de Sufixos, Bioinformática, PCR, Projeto de *Primer*.

Abstract

The present work boards primers search problem for construction of a suffixes tree for DNA sequences. The problem solution for suffixes tree is viable, once, with suffixes tree, is possible to identify specific regions and regions in common between sequences formed by a same alphabet. The main goal is the creation of a tool to assist primers project for PCR's Reactions who involve, at the same time, DNA multiple sequences, called PCR's multiple reaction, and whose goal is the amplification of one or more specific sequences of a sequences set. In a PCR's multiple experiment, primers choice is not more dependent just of the sequence composition that is wished to amplify, but also of the too much presents in the reaction, since primers selected should be specific for the sequence target.

The obtained result was a software whose functionality is the determination of possible regions for primers project in DNA sequences. More specifically, given a set of $n \geq 2$ of sequences of passed DNA as entrance and a restrictions set to guide primers choice, the user will have as DNA exit extract where the search for pairs of primers can occur. That kind of application should be able to allow the handling of thousands of bases pasts for the consultation, therefore some failures happened along the project execution were related to library used in the implementation, due to its limitation in the bases quantity that this has the capacity of prosecuting.

Keywords: Suffix tree, Bioinformatics, PCR, *Primer* Design.

Capítulo 1

Introdução

1.1 Apresentação

A Bioinformática é uma nova área de pesquisa que une conhecimentos da área de Ciência da Computação, Biologia Molecular, Bioquímica, entre outras. O termo Bioinformática tem sido utilizado para descrever os processos computacionais usados para dar apoio às pesquisas relacionadas com material genético (DNA, RNA e proteínas).

Em relação às pesquisas com material genético, uma das principais áreas de atuação da Bioinformática, o trabalho do biólogo e pesquisador da área começa com o sequenciamento de um trecho de DNA (Ácido Desoxirribonucléico) de um organismo de interesse. O sequenciamento pode ser feito por diferentes técnicas, mas, em geral, as etapas iniciais do processo básico envolve a obtenção de uma amostra de DNA do organismo de interesse e a produção de várias cópias desta amostra. As cópias são submetidas as máquinas sequenciadoras para que sua composição seja determinada.

A produção destas cópias pode ser feita pela técnica de *Polimerase Chain Reaction*, ou simplesmente PCR (técnica *in vitro*). A PCR permite aumentar milhões ou bilhões de vezes a quantidade inicial de DNA. Para isso, é necessária em uma mesma reação uma pequena amostra do DNA que se deseja copiar, nucleotídeos livres, uma enzima do tipo DNA Polimerase e inicializadores de polimeração, os *primers*.

O processo do projeto dos *primers* é uma etapa chave para uma PCR bem sucedida, necessitando do conhecimento prévio da sequência de DNA alvo, onde se deseja que esses inicializadores sejam ligados. O projeto de *primers* envolve a otimização de um conjunto de parâmetros como: temperatura, tamanho, composição e conteúdo GC; a busca manual é possível, porém nem sempre trivial, por isso existe um grande número de *softwares* para projeto de *primers* específicos a fim de que os *primers* se unam à sequência de DNA de interesse da melhor maneira possível.

A reação de PCR ocorre em equipamentos chamados termocicladores que, por simples variações de temperatura (aquece a 95°C e resfria a cerca de 60°C, várias vezes), permite que a reação ocorra.

A técnica de PCR tem provocado grande impacto nas principais áreas da Biologia Molecular: mapeamento gênico, clonagem e sequenciamento de DNA, e detecção da ex-

pressão gênica. Atualmente, a PCR é utilizada para o diagnóstico de doenças genéticas, bem como na detecção de material genético presente em pequenas quantidades na amostra em análise. Como exemplo, é possível a detecção e identificação de material genético de vírus como o HIV ou o vírus da hepatite, assim como a detecção de organismos geneticamente modificados em produtos alimentícios.

Dada a importância e o uso do experimento de PCR em diversas técnicas de Biologia Molecular e da relativa dificuldade em se determinar *primers*, mais especificamente o projeto de *primers* em reações de PCR múltiplas, este trabalho apresenta uma abordagem para o problema via árvore de sufixos e implementação de um *software* baseado em uma biblioteca já existente. Com este trabalho, espera-se contribuir com o aperfeiçoamento da ciência e pesquisa em bioinformática.

1.2 Justificativa

O problema abordado neste projeto de pesquisa visa a busca de *primers* via construção de uma árvore de sufixos para sequências de DNA. A solução do problema via árvore de sufixos é viável, uma vez que, com árvore de sufixos, é possível identificar regiões específicas e regiões em comum entre sequências formadas por um mesmo alfabeto.

O objetivo principal é a criação de uma ferramenta para auxiliar o projeto de *primers* para reações de PCR que envolvem, ao mesmo tempo, múltiplas sequências de DNA, chamadas de reação de PCR múltiplas, e cujo objetivo é a amplificação de uma ou mais sequências específicas de um conjunto de sequências. Em um experimento de PCR múltipla, a escolha dos *primers* não é mais dependente apenas da composição da sequência que se deseja amplificar, mas também das demais presentes na reação, visto que os *primers* selecionados devem ser específicos para a sequência alvo.

A realização deste trabalho se justifica perante a necessidade de uma ferramenta específica para o projeto de *primers* para reações de PCR múltiplas, bem como reações baseadas em PCR. A idéia de sua realização surgiu em uma conversa com o pesquisador Dr. Flábio Ribeiro Araújo da Embrapa Gado de Corte, quando, durante uma conversa, percebemos que não existe (ao menos não é do seu/nosso conhecimento) uma ferramenta com o objetivo especificado.

Este trabalho permitirá que os pesquisadores façam projeto de *primers* que serão usados em experimento e PCR múltiplas. Em reações deste tipo é importante que se respondam perguntas como:

- existe uma sequência específica (que não ocorre em nenhuma outra) de uma dada sequência n.
- quais são as *substrings* comuns a duas ou mais sequências de DNA presentes em uma reação de PCR múltipla?

Como resposta, o programa irá fornecer ao usuário como saída, trecho(s) de DNA(s) onde a busca por pares de *primers* pode ocorrer.

1.3 Objetivos

Em experimentos de PCR múltiplas, que envolvem, ao mesmo tempo, múltiplas sequências de DNA, a escolha dos *primers* não é mais dependente apenas da composição da sequência que se deseja amplificar, mas também das demais presentes na reação, visto que os *primers* selecionados devem ser específicos para a(s) sequência(s) alvo. Em reações deste tipo, pode-se, entre outros, objetivar a amplificação de uma ou mais sequências específicas, presentes na reação.

Assim, como objetivo deste projeto de pesquisa, temos o estudo teórico do problema de projeto de primer para PCR múltipla. Deste estudo, resultará a implementação de um *software* cuja funcionalidade específica é a seguinte: dado um conjunto de n sequências de DNA, ($n \geq 2$) e alguns critérios como parâmetros da busca como, por exemplo,

- uma *substring* específica de uma ou mais sequências do conjunto;
- uma *substring* em comum entre duas ou mais *substrings* do conjunto;

o usuário terá como saída trecho(s) de DNA(s) onde a busca/projeto de *primers* pode ocorrer.

1.4 Organização

O Capítulo 2 apresenta informações teóricas básicas, necessárias ao entendimento do problema tratado, em sua parte inicial. Em seguida, apresentamos uma especificação e detalhamento do problema abordado, incluindo um estudo da técnica de PCR, das características dos *primers*, do projeto de *primers* e de ferramentas computacionais para o projeto de *primers*.

No Capítulo 3 apresentamos a solução teórica encontrada, apontando para possíveis formas de resolver o problema. Neste capítulo, apresentamos informações teóricas necessárias para o entendimento da estrutura de dados adotada na solução do problema do projeto de *primers*, a árvore de sufixos.

O Capítulo 4 aborda a solução computacional adotada para solucionar o problema e os módulos do programa implementados.

No Capítulo 5 são apresentadas as etapas deste projeto de pesquisa, mostrando como o mesmo foi conduzido.

O Capítulo 6 mostra os resultados obtidos neste projeto de pesquisa, a discussão dos resultados, quais as dificuldades encontradas, os testes realizados para validar o sistema computacional proposto e também algumas sugestões para trabalhos futuros.

Por fim, o Capítulo 7 apresenta a conclusão.

Capítulo 2

Conceitos de Biologia Molecular

Este capítulo contêm informações retiradas de [26], [18], [14].

2.1 Material Genético

2.1.1 DNA e RNA

Ácidos nucléicos são polímeros lineares de nucleotídeos, unidos por ligação fosfodiéster, que compõem o material genético de todo organismo vivo. Existem dois tipos de ácidos nucléicos: ácido desoxirribonucléico (DNA) e ácido ribonucléico (RNA). Cada nucleotídeo é formado de um grupo fosfato, um açúcar (pentose) e uma base nitrogenada. No DNA existem quatro tipos de bases nitrogenadas: adenina (A), guanina (G), timina (T) e citosina (C). No RNA existem a adenina (A), guanina (G), uracila (U) e citosina (C). As principais diferenças entre o RNA e o DNA são: no RNA tem-se a presença da uracila (U) em vez da timina(T); a pentose no RNA é a ribose e no DNA, é a desoxirribose; e o RNA é uma fita simples e o DNA é uma fita dupla. A Figura 2.1 mostra um esquema da estrutura/composição das bases nitrogenadas.

A cadeia de nucleotídeos possui uma orientação química. Em uma fita de DNA ou RNA, numa das extremidades há um grupo fosfato ligado ao C_5 (carbono $5'$) do açúcar (extremidade $5'$) e na outra há uma hidroxila ligada ao C_3 (carbono $3'$) do açúcar (extremidade $3'$). Assim, na extremidade $5'$ da cadeia, um grupo fosfato está presente e, na extremidade $3'$, um grupo OH. Convencionou-se escrever e ler a sequência nucleotídica da esquerda para a direita, no sentido $5' \rightarrow 3'$. A representação de uma cadeia polinucleotídica é feita apenas através das letras das bases nitrogenadas que a compõe como, por exemplo, a representação da primeira fita do DNA da Figura 2.2 seria $5'$ $AGTACG$ $3'$.

O DNA é composto de duas cadeias polinucleotídeas que formam uma dupla hélice em torno de um eixo central. Uma fita se une à outra por meio de pontes de hidrogênio formadas entre pares de nucleotídeos. A ilustração da dupla hélice do DNA pode ser vista na Figura 2.2. Os pares de bases são: adenina que se une à timina, e citosina que se une à

guanina. A ligação entre as bases A e T dá-se pela formação de duas pontes de hidrogênio e entre C e G por três pontes de hidrogênio, o que faz com que sua ligação seja mais forte. O comprimento de uma sequência de DNA é descrito em pares de bases (pb), quilobases (1000 pb), megabases (1 milhão pb) etc [7].

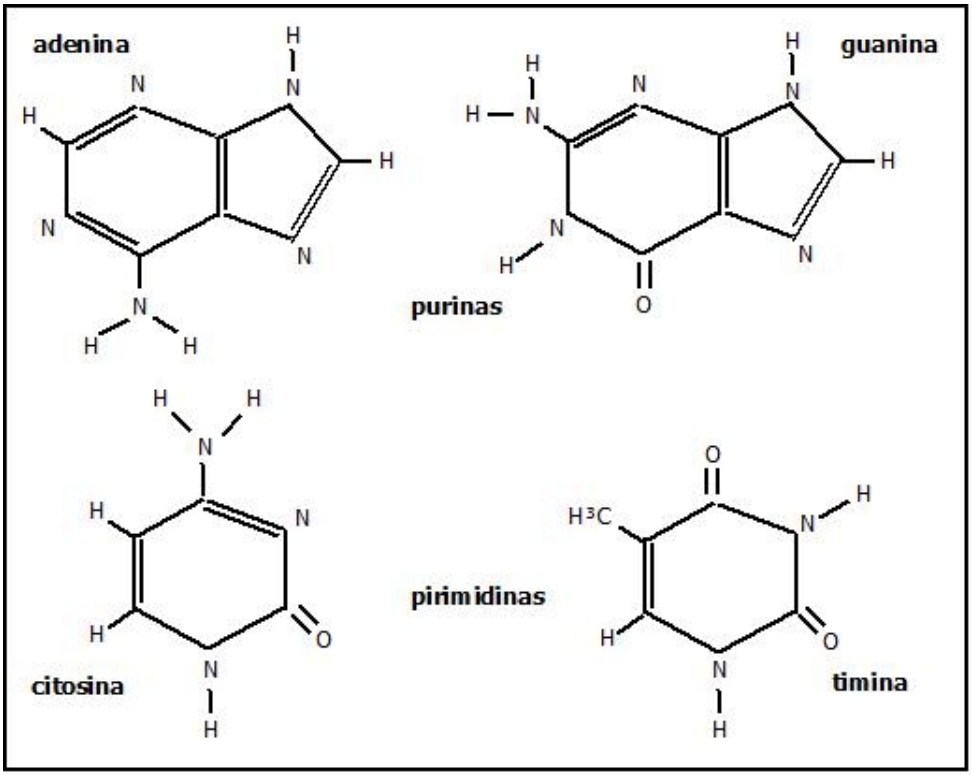


Figura 2.1: Bases nitrogenadas que compõe o DNA. Baseada em [17].

O RNA é uma molécula de ácido nucléico formada por uma só cadeia polimérica. Ele é sintetizado a partir do molde de DNA e é utilizado na expressão da informação genética. O RNA se divide em 3 tipos: RNA ribossômico (RNAr), RNA transportador (RNAt) e RNA mensageiro (RNAm), dependendo de sua atuação dentro da célula.

2.1.2 Replicação

A replicação é o processo pelo qual uma molécula de DNA se duplica, dando origem a duas moléculas idênticas à molécula inicial. Para que esse processo ocorra, há a necessidade de um conjunto de proteínas específicas. A seguir é descrito o processo simplificado de replicação do DNA *in vitro*.

Para que ocorra a replicação de uma molécula de DNA faz-se necessário separar as duas fitas de nucleotídeos da molécula, pois cada uma dessas fitas irá servir como molde para a produção da nova fita. A síntese de novas fitas é feita pela enzima DNA polimerase. Para

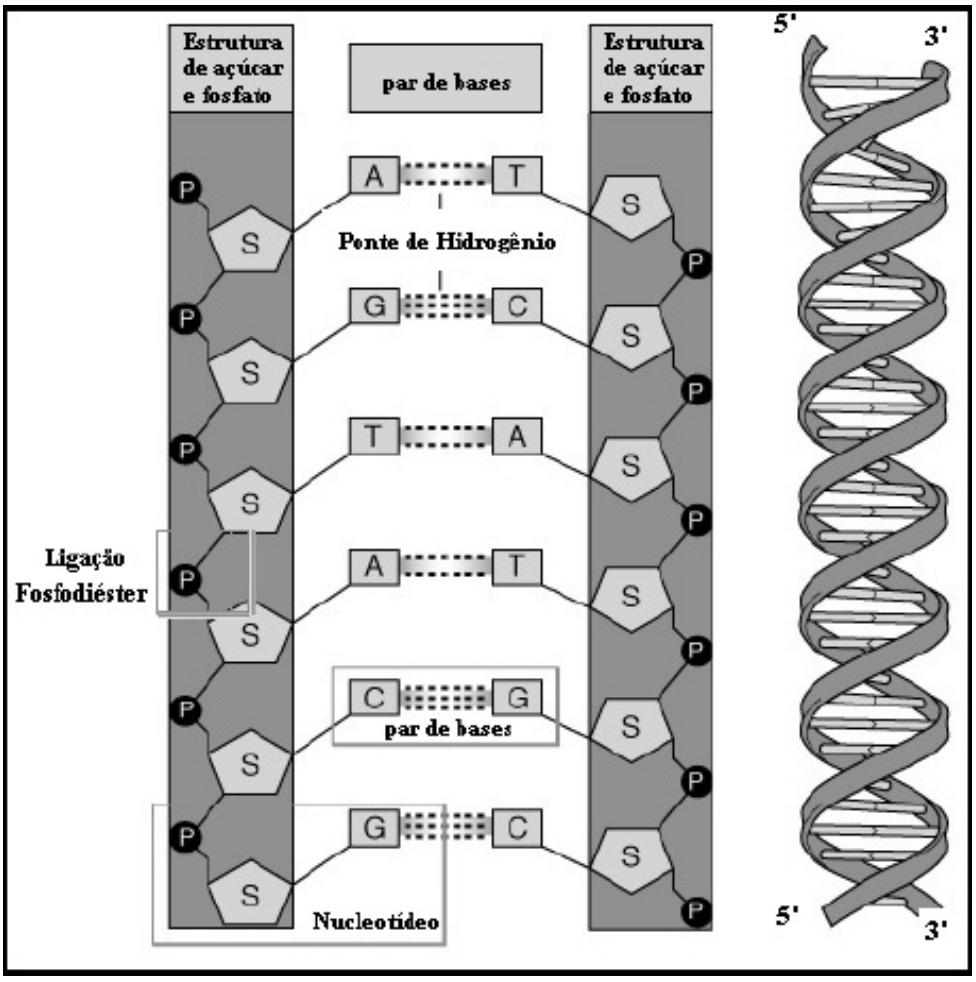


Figura 2.2: A dupla hélice do DNA. Retirada de [1].

que esta enzima atue na replicação é necessária a presença de *primers* que são sequências iniciadoras do processo de síntese e correspondem a curtas cadeias de nucleotídeos complementares à sequência que se deseja replicar, dita sequência alvo. Em uma mesma reação, *primers* e sequência alvo (na forma de fita simples) se unem em um processo que se baseia no princípio de complementaridade das bases ($A-T$ e $C-G$).

2.1.3 *Primers*

Primer, ou iniciador, é uma pequena cadeia de DNA de fita simples, onde, a partir de sua extremidade $3'$, a enzima DNA polimerase iniciará a incorporação de nucleotídeos tomando como base uma fita de DNA molde (a qual o *primer* se acopla), formando uma nova fita de DNA. *Primers* são sequências iniciadoras do processo de síntese e correspondem a curtas cadeias de nucleotídeos complementares à região inicial (*primer forward*) da primeira fita molde e à região final (*primer reverse*) da segunda fita molde da sequência que se deseja amplificar, dita sequência alvo. Veja a atuação do *primer forward* e *primer*

reverse na Figura 2.3 na etapa 2 (*annealing*). O *primer forward* sintetiza o DNA em direção ao *primer reverse* e vice-versa. *Primers* são produzidos *in vivo* ou *in vitro*. *In vivo*, os *primers* são produzidos por uma RNA polimerase especial chamada Primase [18]. *Primers* são utilizados para reações de PCR e para sequenciamento. O projeto de *primer* é uma das mais importantes etapas para que se tenha uma reação de PCR bem sucedida [2].

2.2 Reação de PCR

A PCR, ou *Polimerase Chain Reaction* (Reação em Cadeia da Polimerase), é uma técnica que possibilita a reprodução *in vitro* de milhares, milhões ou mesmo bilhões de cópias de um determinado fragmento de DNA ou RNA. Através dessa técnica, uma sequência particular de interesse pode ser amplificada por síntese, pela polimerase de DNA, tornando-se majoritária na amostra de DNA. Assim, a sequência amplificada pode ser utilizada para outros fins, como por exemplo, uma sequência de DNA que codifica para um gene e que essa sequência deva ser submetida a uma gama de experimentos que permitam a investigação de sua função e estrutura, entre outros. Para viabilizar tais experimentos, quantidade suficiente dessa sequência deve estar disponível [14]. O termo amplificação é utilizado para caracterizar experimentos com vistas ao aumento (em número) de uma molécula e ocorre por meio de diversos ciclos de extensão por polimerase.

A PCR é um método de amplificação (de criação de múltiplas cópias) de DNA sem o uso de um organismo vivo. Para que as enzimas do tipo polimerase atuem é necessária a presença de três componentes básicos em uma reação de amplificação: *primers*, moldes e os quatro nucleotídeos.

Uma reação de PCR ocorre pela execução cíclica de três etapas distintas, chamadas de desnaturação, *annealing* e extensão, descritas a seguir:

- Desnaturação: Este processo faz com que a fita dupla de uma molécula de DNA se separe em duas fitas simples. A desnaturação de uma molécula é obtida pelo seu aquecimento a temperaturas próximas de 94°C, por cerca de 1 minuto.

- *Annealing*: Em uma mesma reação, e sob temperatura ideal - denominada temperatura de *annealing* - *primers* e sequência alvo se unem por complementaridade entre suas bases. Este processo ocorre sob temperaturas próximas de 55°C por 1 a 2 minutos.

- Extensão: Uma vez formado o complexo - *primer* e sequência alvo - as enzimas do tipo polimerase ligam-se ao grupo OH da extremidade do carbono C_3 do *primer* (extremidade $3'$ OH) para dar início ao processo de síntese. Para que a nova molécula seja sintetizada, a DNA polimerase utiliza a sequência alvo como molde e, segundo o princípio da complementaridade entre as bases, adiciona novos nucleotídeos aos *primers*, sendo estes nucleotídeos complementares aos nucleotídeos da fita molde. Em geral, a extensão ocorre sob temperaturas próximas de 72°C durante 2 a 5 minutos. Ao final da extensão, uma nova molécula, complementar à sequência molde, é produzida.

O número de ciclos, a temperatura de *annealing*, o tempo de cada ciclo e outros componentes de PCR variam de acordo com o objetivo e condições utilizadas [18]. A Figura 2.3 ilustra as três etapas descritas anteriormente.

A amplificação de uma molécula em laboratório ocorre por meio de diversos ciclos de extensão por polimerase. No primeiro ciclo, a molécula de DNA alvo é desnaturada resultando em duas sequências molde onde os *primers* se pareiam. Os *primers* são estendidos pela polimerase e resultam em duas novas moléculas de DNA idênticas à molécula original. No segundo ciclo, as duas moléculas resultantes do ciclo anterior são desnaturadas, dando origem a quatro moldes que, após o *annealing* com os *primers* e a extensão, dão origem a quatro novas moléculas de DNA idênticas à molécula original. Assim, após um número n de ciclos de PCR (desnaturação, *annealing* e extensão) tem-se, idealmente 2^n moléculas de DNA idênticas à molécula original. Devido à necessidade do grupo $3'\ OH$ livre na sequência do *primer* para que a enzima DNA polimerase se acople, a síntese de uma nova molécula só ocorre no sentido $5' \rightarrow 3'$. Caso o *primer* pareie com a sequência de DNA molde de forma que o sentido de extensão seja $3' \rightarrow 5'$, a polimerase não atua e, portanto, a extensão não ocorre. Veja a Figura 2.4.

Outro fator importante a ser considerado no processo de síntese *in vitro* de DNA são as mutações que uma reação de PCR pode introduzir nas sequências produzidas. Nucleotídeos não complementares ao molde podem ser inseridos durante a síntese da nova fita de DNA, e estes erros são propagados nos ciclos seguintes da reação, resultando em sequências distintas das originais. Os erros ou mutações ocorridos durante a síntese de DNA são também chamados substituições.

Diversos são os parâmetros que afetam a eficiência da reação de PCR, entre eles: temperatura de *annealing* (Ta) e tempo de *annealing*, temperatura de desnaturação (Tm) e tempo de desnaturação, concentração de $MgCl_2$ e demais componentes presentes na reação, concentração e qualidade dos *primers*, entre outros.

Podemos destacar três parâmetros que podem ser utilizados para medir a qualidade das sequências resultantes de uma reação de PCR: especificidade, eficiência e fidelidade.

A especificidade é a medida para saber se a reação amplifica apenas a região alvo e não outras regiões da molécula, o que ocasionaria a amplificação de trechos não desejados. Ela é muito influenciada pelo *primer* utilizado na reação, principalmente seu tamanho. A concentração de *primers* também é uma variável importante, sendo que baixas concentrações aumentam a especificidade, enquanto altas concentrações facilitam o *annealing* não específico [2].

A eficiência trata do volume de produto em relação ao tempo da reação de PCR. A eficiência pode ser alta se o produto amplificado for pequeno e se os ciclos da PCR forem bem ajustados, permitindo um *annealing* e extensão rápidas. Um aumento na concentração de íons Mg_{++} também pode aumentar a eficiência da reação, embora aumentos exagerados tendem a causar reações menos específicas [2].

Por fim, a fidelidade, ou exatidão, verifica se o produto amplificado é uma cópia idêntica à original. A fidelidade está relacionada ao tipo da polimerase utilizada na PCR, uma vez que diferentes polimerases apresentam diferentes taxas de erro. Por exemplo, DNA polimerase possui fidelidade igual a $2.7 * 10^{-5}$.

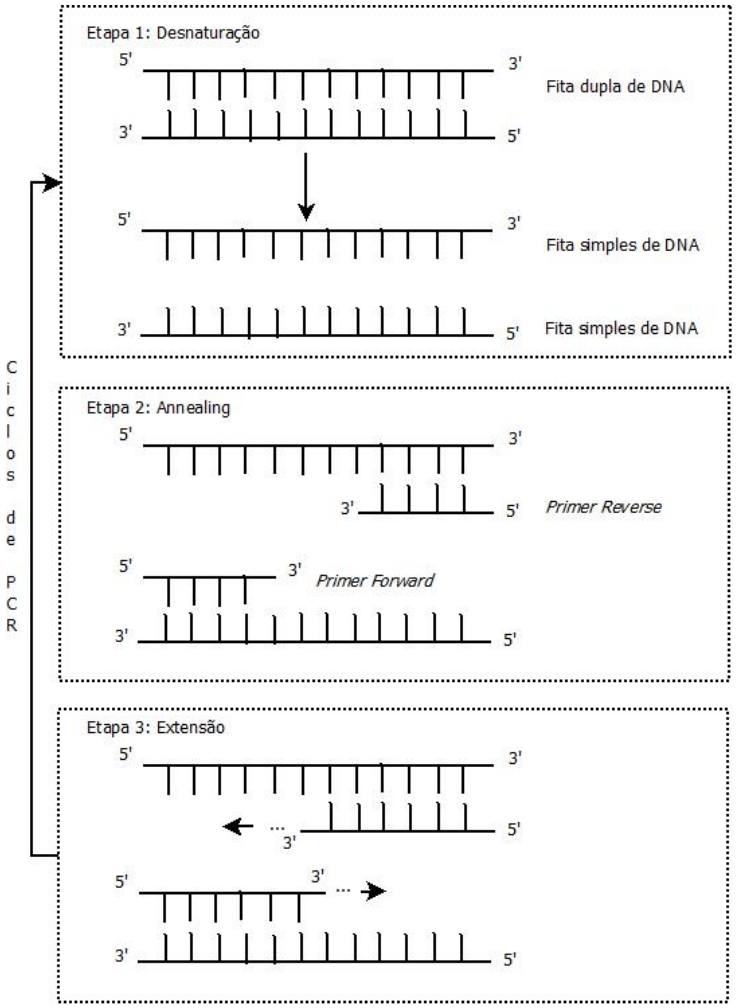


Figura 2.3: Ciclos de PCR com *primers*. A cada ciclo, uma molécula de DNA dá origem a duas novas moléculas idênticas à molécula original.

Há alguns tipos de reações de PCR com processos e finalidades diferentes. Dentre elas, destacam-se a PCR Competitiva, *Real time* PCR, RAPD-PCR e *Multiplex* PCR, descritos a seguir:

- PCR Competitiva [4]- Além do DNA molde, é adicionado à reação um outro trecho de DNA da sequência com tamanho e concentração conhecidos (controle), cujas extremidades são complementares também aos *primers* que irão amplificar a sequência alvo. O resultado é a amplificação de dois trechos de DNA: a de interesse e a controle. Esta última, levando-se em conta a quantidade inicial e dados sobre a eficácia da reação serve de padrão para a quantificação do DNA alvo. Em resumo, se conhecemos a quantidade final do fragmento controle e as condições da reação, podemos dizer o quanto de DNA alvo foi amplificado. Esta técnica é utilizada em *kits* diagnósticos como, por exemplo, para determinação de carga viral de HIV.
- *Real time* PCR [10], [12] - A PCR de tempo real é um tipo de PCR quantitativo que

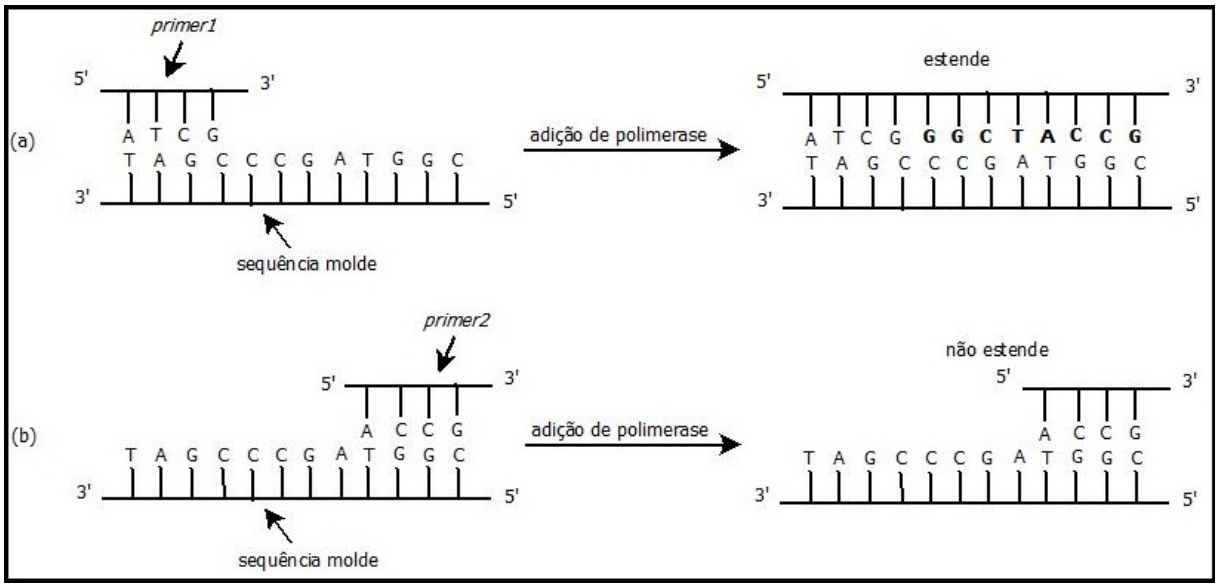


Figura 2.4: Atuação da enzima DNA polimerase na replicação do DNA. (a) A polimerase é capaz de estender o *primer1* criando uma fita complementar ao molde. (b) A polimerase não é capaz de estender o *primer2* - uma extensão não ocorre a partir da extremidade 5′. Baseada em [14].

mede a quantidade de DNA ou de RNA em uma amostra. O tempo real é usado geralmente para determinar a expressão do RNA de um gene, e sua expressão nivela (número de cópia de RNA) durante determinadas condições. A PCR de tempo real pode ser usado para comparar amostras normais às amostras da doença, dando uma idéia a respeito das mudanças da expressão que ocorrem com patogénese. A PCR de tempo real devido a sua sensibilidade é usado igualmente na deteção dos micróbios patogénicos no sangue tal como vírus. Para se detectar o aumento de produto de uma PCR ao longo de cada ciclo, é preciso marcar o DNA amplificado com algum tipo de molécula fluorescente (Exemplo: *SYBR TM Green* e *TaqMan TM*). A contaminação é motivo de grande preocupação para laboratórios que empregam métodos como PCR e a pouca chance de contaminação com *Real time* PCR se dá porque o tubo de reação não precisa ser aberto ao seu final (já que a detecção ocorre on-line, durante os ciclos de amplificação).

- RAPD-PCR [25] [24]: Para certos tipos de análise, a reação de PCR específica apresenta um grande fator limitante: o seu uso em larga escala (ex., vários *locos*) requer o conhecimento dos nucleotídeos que compõem as duas extremidades da sequência de DNA que se deseja amplificar. No início da década de 90 apresentou-se uma variação na técnica de PCR conhecida como RAPD ou AP-PCR. Esta variação foi desenhada para contornar o problema do conhecimento prévio da sequência de DNA que se deseja amplificar, possibilitando a utilização da técnica em organismos onde nenhum conhecimento de sequência de DNA existia. Esta variação possibilita que ocorra amplificação ao acaso de segmentos de DNA no genoma.

Destaca-se no contexto deste trabalho a *Multiplex* PCR, ou reação de PCR múltipla, descrita em maiores detalhes na seção seguinte.

2.2.1 PCR Múltipla

Multiplex PCR é uma variante da PCR que permite a amplificação simultânea de vários fragmentos, em uma mesma reação, usando um ou mais pares de *primers*. Desde sua primeira descrição [6], esse método tem sido aplicado em muitas áreas de teste de DNA, incluindo mutações e polimorfismos, ou ensaios quantitativos e PCR de transcrição reversa.

Em *Multiplex* PCR, mais de um segmento genômico pode ser amplificado numa única reação, cada um com seu par de *primers* específico, o que pode simplificar experimentos como o de investigação de paternidade, onde vários marcadores genômico devem ser analisados. De uma maneira geral, reações de PCR múltiplas podem ser caracterizadas como reações onde estão presentes um conjunto de n sequências e se deseja amplificar apenas um subconjunto $k < n$ de sequências.

Uma das questões fundamentais relacionadas a execução de PCRs múltiplas é o projeto de *primers*, uma vez que questões como número e especificidade dos *primers* são mais complicados de se resolver, como descrito na seção 2.3.

2.3 Projeto de *Primers*

O projeto de *primer* é uma das mais importantes etapas para que se tenha uma reação de PCR bem sucedida, uma vez que são eles os iniciadores do processo de amplificação juntamente com as enzimas do tipo polimerase. Existem muitos pontos que devem ser levados em consideração para se projetar um *primer* de boa qualidade. Os mais relevantes são apresentados na seção seguinte.

2.3.1 Características (Desejáveis) dos *Primers*

Dentre as características dos *primers* que influenciam diretamente a reação de PCR estão a especificidade e o comprimento do *primer*, a temperatura de *melting* (Tm), a temperatura de *annealing* (Ta), a especificidade no *annealing* do *primer*, a composição de bases e da extremidade $3'$ dos *primers*. Assim sendo, a determinação e escolha de um *primer* ou par de *primers* a ser utilizado em um experimento específico requer uma série de cuidados. Não existem valores específicos pré-definidos para os parâmetros envolvidos no projeto de *primers*. Contudo, intervalos de valores para esses parâmetros são de senso comum, como descritos mais adiante.

A seguir, alguns dos parâmetros/características que devem ser consideradas durante o projeto de *primers* são apresentados.

Estrutura Interna

A composição dos *primers* deve ser tal que, preferencialmente, não exista *runs* e *repeats*.

Um *repeat* é definido como a ocorrência repetida de uma *substring* do *primer* em posições consecutivas da sua sequência. *Repeats* devem ser evitados uma vez que eles podem favorecer o *annealing* entre o *primer* e o DNA alvo em posições não desejadas (evento também chamado de *misprimer*), como mostrado na Figura 2.5.

Runs são definidos como a repetição consecutiva de uma única base na sequência do

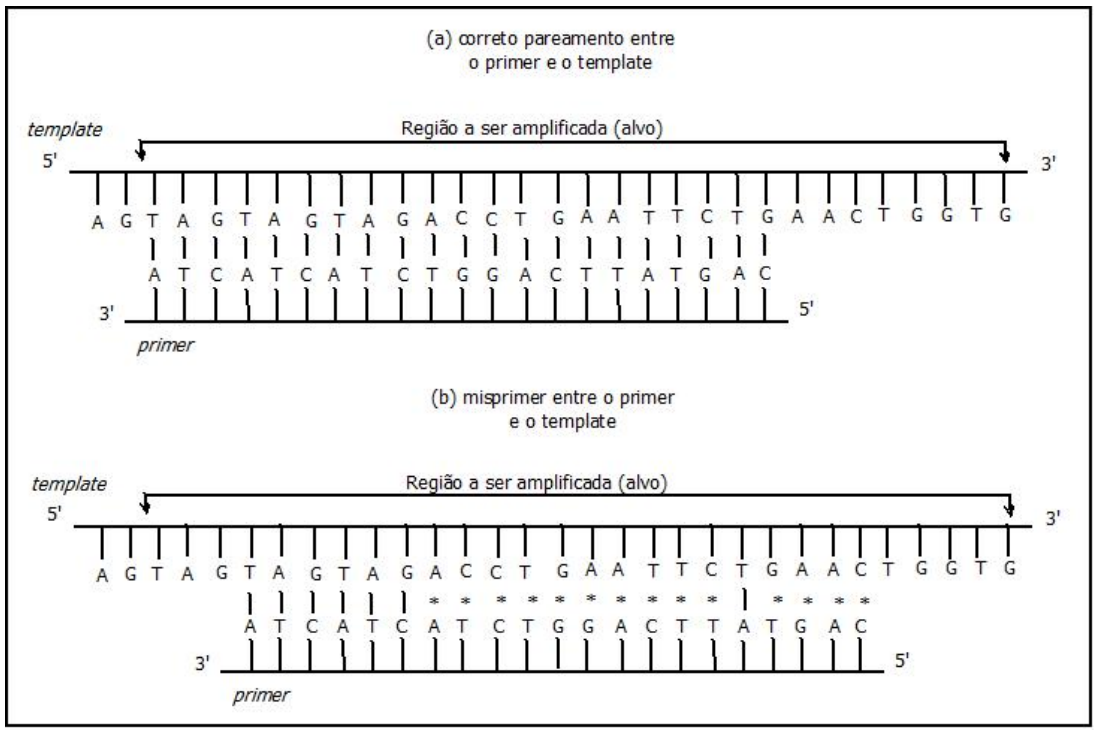


Figura 2.5: *Primer* com *repeats* (ATC). Os *matches* ocorridos no *annealing* entre o *primer* e a fita molde são representados pelo símbolo |, enquanto que os *mismatches* são representados por $*$. Baseada em [14].

primer. A presença de *runs* também favorece o *misprimer*.

Primers trabalham em pares - *forward primer* e *reverse primer*. Se os *primers forward* (F) e *reverse* (R) apresentarem trechos complementares entre si, pode ocorrer o *annealing* entre o *primer* e a fita molde, formando uma estrutura secundária chamada *hetero-dimer*. Caso o *annealing* ocorra entre dois *primers* F ou dois *primers* R, ocorre a formação de uma estrutura secundária denominada *self-dimer*. *Primers* longos podem se auto-parear, resultando em estruturas secundárias denominadas *hairpins*.

Primers podem parear entre si, ao invés de se parearem com o DNA molde. Nestes casos, a eficiência da PCR pode ser comprometida uma vez que a concentração de *primers* disponíveis ao *annealing* com o DNA molde diminui.

Especificidade e Tamanho do *primer*

Um *primer* é específico se ele se pareia com a fita molde apenas na região específica para a qual foi projetado. *Primers* não específicos acarretam a produção (amplificação) de fragmentos de DNA não correspondentes à região alvo. Portanto, deve, idealmente, existir somente um sítio de *annealing* no DNA molde onde ocorra a ligação do *primer*, o que significa que as sequências dos *primers* são específicas na sequência do DNA molde.

O comprimento do *primer* influencia a especificidade, bem como as temperaturas de *melting* e *annealing*. Quanto maior o comprimento do *primer*, maior a possibilidade deste ser específico; da mesma forma, maiores serão as temperaturas de *melting* e *annealing*.

Portanto, o *primer* deve ser comprido o suficiente para ser único na sequência e reduzir a probabilidade da sequência ser encontrada em locais não alvos na sequência a ser amplificada. Entretanto, *primers* muito grandes são muito caros para fazer; pareiam mais lentamente; e aumentam a probabilidade de ocorrência de formação de estruturas secundárias.

Em resumo, o tamanho de um *primer* influencia:

- a sua especificidade (*primers* maiores são mais específicos);
- custo de produção (quanto maior o *primer*, mais elevado é o seu custo);
- estabilidade da formação *fita molde-primer* (quanto maior o *primer*, mais fortemente ele estará unida a fita molde devido ao maior número de pontes de hidrogênio resultantes desta ligação);
- formação de estruturas secundárias (*primers* maiores são mais propensos à formação de estruturas secundárias).
- Quanto maior o comprimento do *primer*, maiores serão as temperaturas de *melting* e *annealing*.

Não existe um tamanho fixo ótimo para um *primer*. De forma geral, o comprimento do *primer* não deve ser inferior a 15 bases para assegurar a especificidade. Normalmente os *primers* devem ter entre 18 e 24 nucleotídeos. O estabelecimento deste intervalo de valores para o tamanho de um *primer* se baseia no fato de se buscar *primers* específicos que apresentem temperatura de *annealing* dentro da faixa considerada como a mais adequada, como descrito mais adiante.

Composição e Extremidade $3'$

As bases que compõem a sequência de um *primer* afetam a especificidade do *annealing* e as temperaturas de *melting* e *annealing*.

A quantidade de bases Citosina (C) e Guanina (G) de um *primer* influenciam fortemente a temperatura na qual a reação de *annealing* deve ocorrer basicamente pelo fato do *annealing* destas bases ocorrer devido à formação de três pontes de hidrogênio enquanto que o *annealing* entre as bases A e T ocorre devido à formação de duas pontes

de hidrogênio. Quanto maior o número de bases C e G, mais fortemente ligados estarão o *primer* e a fita molde. Geralmente, a quantidade de $(G + C)$ deve estar em torno de $40-60\%$ para assegurar temperaturas de *melting* e *annealing* adequadas à reação de PCR. Porém, as temperaturas de *melting*, *annealing* e a estabilidade no *annealing* são também afetadas por outros fatores, discutidos logo adiante no texto.

A preferência por uma base C ou G na extremidade $3'$ do *primer* é justificada por ser esta a extremidade na qual a enzima polimerase inicia a extensão do *primer*. Como o *annealing* $C - G$ é mais estável (devido ao maior número de pontes de hidrogênio formadas entre essas duas bases), espera-se que a polimerase inicie o processo de síntese mais eficientemente neste caso.

Temperatura de *Melting* (Tm)

A temperatura de *melting* (Tm) é definida como a temperatura na qual metade dos fragmentos de DNA está na forma desnaturada, ou seja, não pareados, e a outra metade está pareada.

A Tm é dependente da composição do DNA, de modo que quanto maior o conteúdo de $C + G$ no DNA maior a Tm necessária.

Existem diversas aproximações para o cálculo da Tm de um *primer*, sendo possível, de uma maneira geral, dividí-las em três classes distintas:

1. Básicas (consideram apenas a composição do *primer*);
2. Dependentes do Sal (consideram a concentração de sal na reação onde os pareamentos ocorrem); e
3. Baseadas na Termodinâmica da reação (utilizam-se do modelo *Nearest Neighbor*).

As equações (2.1), (2.2) e (2.3) apresentam exemplos de fórmulas básica, dependentes do sal e baseadas em termodinâmica para o cálculo da Tm, propostas por Wallace e seus colaboradores [22], Howley e seus colaboradores [11] e Rychlik e seus colaboradores [15], respectivamente.

$$Tm = 2 * (|A| + |T|) + 4 * (|C| + |G|) \tag{2.1}$$

$$Tm = 100,5 * \frac{|C| + |G| - 6,4}{|A| + |T| + |C| + |G|} - \frac{820}{|A| + |T| + |C| + |G|} + 16,6 * \log[Na^+] \tag{2.2}$$

$$Tm = \frac{\Delta H}{\Delta S + R * \ln \frac{\gamma}{4}} - 273,15 + 16,6 * \log[Na^+] \tag{2.3}$$

Nas equações 2.2 e 2.3, $[Na^+]$ é a concentração de sal na reação. Na equação 2.3, $R = 1,987 cal/°\text{C} * mol$ é a constante molar dos gases e γ é a concentração do *primer* na solução, medida em nM. Nas equações 2.1 e 2.2, $|A|, |T|, |C|$ e $|G|$ representam, respectivamente, o número de bases, Adenina, Timina, Citosina e Guanina presentes nos *primers*. Como pode ser observado na equação 2.3, a fórmula para o cálculo da Tm baseada no modelo *Nearest Neighbor* (NN) utiliza-se dos conceitos de variação de entalpia (ΔS) e entropia (ΔH). A entalpia pode ser definida como a quantidade de energia disponível na forma de calor, em uma determinada reação. A variação da entalpia de um sistema é o calor liberado ou absorvido quando uma transformação ocorre sob pressão constante. A entropia é uma grandeza termodinâmica que aparece geralmente associada ao que se denomina "grau de desordem"de um sistema termodinâmico. Com a entropia procura-se "medir"a parte da energia que não pode ser transformada em trabalho em transformações termodinâmicas. A variação de entropia do sistema é dada pelo quociente entre a energia transferida para o sistema sob a forma de calor e a temperatura absoluta constante na qual este se encontra. Valores para (ΔH) e (ΔS) entre os pares de bases vizinhos em uma região de pareamento podem ser encontrados em [3], [19], [16] e [5].

Temperatura de *annealing* (Ta) e Estringência no *annealing* do *primer*

A temperatura de *annealing* (Ta) é a temperatura na qual ocorre o *annealing* entre *primer* e a fita molde. A estringência determina a especificidade no produto de DNA a ser amplificado, ou seja, se o *primer* se pareia com a fita molde apenas na região específica para a qual foi projetado.

Como existe uma tripla ligação entre as bases G e C e uma dupla ligação entre A e T, a temperatura de *annealing* sofre alteração devido à quantidade de pares de bases $G-C$ e $A-T$. Quanto maior for a quantidade de GC em um *primer*, maior será a temperatura de *annealing*.

Existem, basicamente, duas fórmulas para calcular a Ta de um *primer*:

- *primers* com 20 ou menos pares de bases

$$Ta = Tm - 5°\text{C}. \tag{2.4}$$

- *primers* com mais de 20 pares de bases

$$Ta = 62,3°\text{C} + 0,41°\text{C}(\%GC) - 500/tamanho - 5°\text{C}, \tag{2.5}$$

A estringência determina a especificidade do *primer* em relação ao trecho de DNA a ser amplificado. A Ta é o fator mais significante que afeta a estringência no *annealing* do *primer*. Analisando a Ta conclui-se:

- Quanto menor → menor estringência → *primer* tende a parear em outras regiões diferentes da região alvo para o qual foi projetado.
- Quanto maior → maior estringência → *primer* tende a parear apenas na região da sequência alvo para o qual foi projetado.

Resumo - Critérios para Projeto de *Primer*

1. Especificidade: assegurar o *annealing* correto dos *primers* com a fita molde;
2. Comprimento: entre 18-24 nucleotídeos.
3. Composição de bases: a quantidade de $(G + C)$ deve estar em torno de $40 - 60\%$;
4. Tm: entre $55 - 80°C$ é desejável;
5. Minimizar estruturas secundárias: *hairpins, self* e *hetero-dimers*.

2.3.2 Projeto de *Primer* para Reação de PCR Múltiplas

Este projeto de pesquisa busca auxiliar na escolha de trecho(s) de DNA(s) onde a busca/projeto de *primers* deve ocorrer.

O projeto de *primer* para reação de PCR múltiplas é mais complexo porque a seleção de *primers* para estas reações consiste em: dado um conjunto de n sequências de DNA, $(n \geq 2)$, determinar algumas informações úteis para o pesquisador, por exemplo, uma *substring* específica de uma ou mais sequências do conjunto, uma *substring* em comum entre duas ou mais *substrings* do conjunto, entre outras.

Em experimentos de PCR múltiplas, a escolha dos *primers* não é mais dependente apenas da composição da sequência que se deseja amplificar, mas também das demais presentes na reação, visto que os *primers* selecionados devem ser específicos para a(s) sequência(s) alvo.

2.3.3 Ferramentas Computacionais para o Projeto de *Primers*

Existem várias ferramentas computacionais que auxiliam na tarefa de projeto de *primers*. Exemplos de ferramentas disponíveis na internet é o *Web Primer*, *Gene Runner*, *Primer3*, *Primer-BLAST*, entre outras.

A ferramenta *Web Primer*[1] permite dois tipos de entrada de dados: uma delas é o identificador no *GenBank* da sequência a ser amplificada, e a outra entrada é a sequência de DNA. Esta ferramenta provê *primers* para duas finalidades distintas, sequenciamento e PCR.

A ferramenta *Gene Runner*[2] provê vários serviços, como projeto de *primers*, alinhamento de sequências, tradução para proteínas, e outros. Para o projeto de *primers* esta ferramenta não é muito eficiente, pois não possui muitas opções para restringir os *primers*.

A ferramenta *Primer3*[3] é um programa eficiente de desenho de *primers* para PCR que permite um controle considerável sobre a natureza dos *primers*, incluindo o tamanho do produto desejado, o tamanho do *primer* e o intervalo da Tm, e a presença/ausência de uma extremidade de $3' - GC$.

[1] disponível no endereço http://www.yeastgenome.org/cgi-bin/*web-primer*
[2] disponível no endereço http://www.generunner.com
[3] disponível no endereço http://biotools.umassmed.edu/bioapps/primer3_www.cgi

A ferramenta *Primer-BLAST*[4] foi desenvolvido no NCBI para ajudar os usuários a fazer *primers* específicos para a sequência alvo para PCR. Ele usa *Primer3* para projetar *primers* para a PCR e, em seguida, realiza a comparação destes *primers* via *BLAST* para uma base de dados especificada pelo usuário. Os resultados da comparação via *BLAST* são então analisados automaticamente para evitar a escolha de pares de *primers* que possam causar amplificação de outras sequências que não seja a sequência alvo.

[4]disponível no endereço http://www.ncbi.nlm.nih.gov/tools/*primer*-blast/

Capítulo 3

Conceitos de Computação

Os conceitos apresentados neste capítulo foram retiradas de [20], [8], [23], [21] e [13].

3.1 Introdução

Técnicas computacionais para a manipulação de cadeias de caracteres sempre fizeram parte do que se considera importante em Ciência da Computação. Em particular, a comparação de sequências vem se tornando, cada vez mais, um problema computacional de grande interesse.

O interesse pelo problema de comparação de sequências tem aumentado principalmente por causa de duas vertentes distintas de aplicações. A primeira delas está relacionada à quantidade de informação disponível na *web* e à forma como essa informação pode ser recuperada eficientemente. A segunda vertente está relacionada aos inúmeros avanços da Biologia Molecular, em particular no desenvolvimento de técnicas de sequenciamento de DNA, o que, por sua vez, produz um grande volume de dados.

A comparação de sequências biológicas, sejam de DNA ou de proteínas, é um dos principais problemas em Biologia Computacional, área de pesquisa destinada à solução computacional de problemas em Biologia Molecular.

Há interesse em comparar sequências de DNA e proteínas de diversas maneiras, como por exemplo:

- Comparar uma sequência de DNA de centenas de bases com um genoma de milhões de bases para verificar se aquele trecho ocorre no genoma; e
- Comparar sequências de DNA (ou proteína) de algumas centenas de pares de bases entre si para identificar as semelhanças/diferenças.

Certamente, o campo de aplicações que necessitam de técnicas para a comparação de sequências não se limita aos exemplos citados, mas eles já fornecem motivação suficiente para o estudo de algoritmos para o problema de comparação de sequências, também conhecido como *string-matching* ou *pattern-matching*.

3.2 Definições Básicas e Notação

Nesta seção são descritos os principais conceitos e notações utilizados ao longo do texto, incluindo conceitos relacionados à complexidade de algoritmos. Estas informações foram retiradas de [20].

Símbolos, ou caracteres, são os elementos básicos de uma *string*. Um conjunto fixo de símbolo é chamado de **alfabeto**. Uma *string* s de um determinado alfabeto é uma sequência de símbolos pertencentes a este alfabeto. Por exemplo, **aeb** e **adecba** são *strings* do alfabeto $\Sigma = \{a, b, c, d, e\}$. Uma **sequência** ou **cadeia** é uma sucessão ordenada de símbolos de um alfabeto.

O **tamanho** ou **comprimento** de uma sequência s, denotado por $|s|$, é o número de símbolos em s. A cadeia vazia tem tamanho zero e é denotada ϵ. Os símbolos em uma cadeia s são indexados de 1 até $|s|$, sendo que o símbolo que ocupa a i-ésima posição em s é representado por $s[i]$.

Uma *substring* s' de s é uma *string* formada por um grupo consecutivo de símbolos adquiridos da *string* s. Uma *substring* pode ser obtida retirando zero ou mais símbolos a partir do início ou fim da *string* s.

Um **prefixo** de uma *string* s é uma *substring* de s da forma $s[1 \ldots i]$ para $1 \leq i \leq |s|$ (ϵ é prefixo de s). Assim sendo, o prefixo i de s é igual a $s[1 \ldots i]$.

Um **sufixo** de uma *string* s é uma *substring* de s da forma $s[i \ldots |s|]$ para $1 \leq i \leq |s|+1$ (ϵ é sufixo de s). Assim sendo, o sufixo i de s é igual a $s[i \ldots |s|]$. Denotaremos esse sufixo como **sufixo i de s**.

Neste texto, frequentemente nos referiremos a sequências biológicas. Tais sequências podem ser de DNA, onde o alfabeto é $\{A, C, G, T\}$.

3.3 Busca de *Strings*

3.3.1 Visão Geral

Um dos problemas mais comuns envolvendo *strings* é a procura de ocorrências de um determinado padrão em um texto. Este problema de comparação entre *strings* pode ser formulado como segue.

Sejam s e p duas sequências de tamanhos n e m símbolos, respectivamente, com $m \leq n$. A sequência s é chamada **texto** e a sequência p é chamada **padrão**. Dizemos que p ocorre em s na posição $d + 1$ se $s[d + 1 \ldots d + m] = p[1 \ldots m]$ para algum d, com $0 \leq d \leq n - m$. O índice d é chamado um deslocamento de p sobre s. Nesta seção queremos resolver o problema da comparação exata de duas sequências (*string matching problem* - SM):

Problema SM(s, p): *dada uma sequência s, com $|s| = n$, e uma sequência p, com $|p| = m$, com $m \leq n$, encontrar a primeira ocorrência de p em s.*

Este problema é algumas vezes estendido para retornar todas as ocorrências de p

dentro de s. Diversos algoritmos existem para solucionar o problema SM, entre eles: Procura por Força Bruta (*Brute Force Searching*), Knuth-Morris-Pratt e Boyer-Moore; e baseado nestes algoritmos e também em estruturas de dados tipo *hashing*, surgiram uma diversidade de outras variações. Na seção seguinte tem-se a descrição geral do algoritmo de força bruta.

3.3.2 Algoritmo Força Bruta

No modo intuitivo, ou força bruta, o problema é abordado simplesmente tentando comparar p com *substrings* de s em posições sucessivas da *string* texto. O algoritmo força bruta consiste em verificar, em todas as posições no texto entre 0 e $n - m$, se uma ocorrência do padrão existe ou não. Isto é feito movendo o padrão com o texto uma posição à direita.

Algoritmo Força Bruta

Passo 1. Alinha o padrão com o início do texto;

Passo 2. Comparar cada caractere do padrão com o respectivo caractere no texto enquanto todos os caracteres sejam coincidentes (busca bem sucedida) ou uma combinação errônea seja detectada;

Passo 3. Enquanto o padrão não for encontrado e o texto não for percorrido integralmente, realinhar o padrão uma posição a direita e repetir o Passo 2;

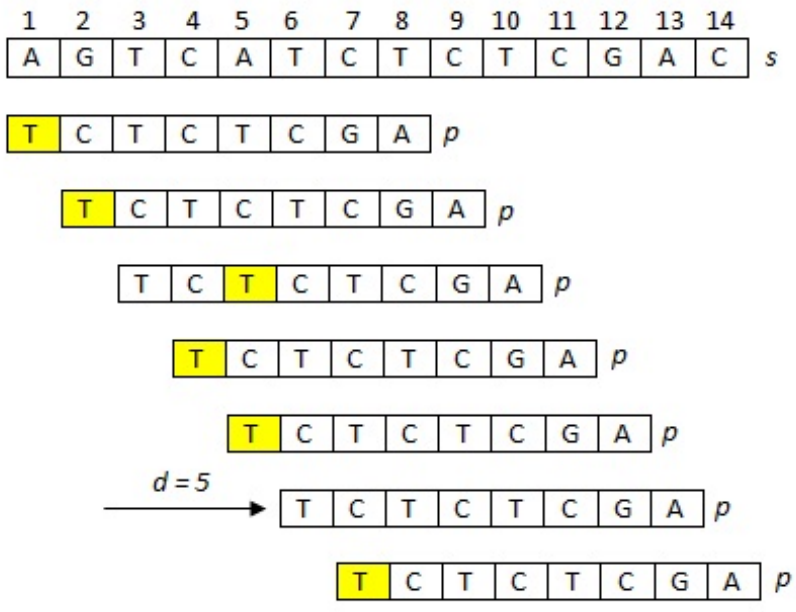


Figura 3.1: Exemplo de comparação de sequências $s = AGTCATCTCTCGAC$ e $p = TCTCTCGA$ através do algoritmo força-bruta. Dizemos que ocorre um casamento entre as sequências s e p na posição $d = 5$.

Formalmente, um algoritmo ingênuo para comparação de duas sequências s e p é uma sequência de passos onde em cada um deles queremos saber se $s[d+1 \ldots d+m] = p[1 \ldots m]$, para todo deslocamento $d = 0, \ldots, n-m$. Ou seja, dado um deslocamento d, comparamos símbolo a símbolo uma *substring* de s contendo m símbolos que inicia-se na posição $d+1$ e termina na posição $d+m$, para algum d válido, com a sequência p. Se $s[d+1 \ldots d+m] = p[1 \ldots m]$, então p ocorre em s na posição $d+1$. Um exemplo de funcionamento deste algoritmo é mostrado na Figura 3.1.

Observe que, no pior caso, quando o padrão não ocorre no texto, o algoritmo força bruta para comparação de sequências baseado na idéia acima tem tempo de execução $O(mn)$.

3.4 Árvore de Sufixos

Uma árvore de sufixos (*ST - Suffix Tree*) de uma sequência s com tamanho n é uma árvore com n folhas numeradas de 1 a n. Qualquer nó interno da árvore possui no mínimo dois filhos. Cada aresta da árvore é rotulada com uma *substring* não vazia. Rótulos de arestas que saem do mesmo nó devem começar com diferentes caracteres. Para cada folha i da árvore, a concatenação de todas os rótulos das arestas da raiz até a folha resulta no sufixo da sequência s que começa na posição i.

Uma árvore de sufixos é uma árvore que armazena os sufixos de uma sequência de símbolos. Essa estrutura foi inicialmente proposta para a solução de problemas de casamento exato de padrões, mas suas funcionalidades vão além disso, incluindo compressão de texto.

Árvore de sufixos são estruturas extremamente eficientes, sendo capazes de representar todas as *substrings* de uma *string* σ de tamanho n em uma árvore compacta com complexidade de espaço $O(n)$. Seu principal atrativo é o fato de poder ser construída sequencialmente em tempo linear e, uma vez construída, permitir consultar se um determinado padrão ω de tamanho m é uma *substring* de σ em $O(m)$ passos, ou ainda retornar todas as k ocorrências do padrão ω no texto σ em um tempo $O(m+k)$, independente do tamanho de σ.

Uma descrição bastante detalhada, incluindo diversas aplicações de árvores de sufixos, pode ser encontrada em [8].

As definições, bem como os algoritmos necessários à contextualização da utilização de árvore de sufixos para o projeto de *primers* para reação de PCR múltiplas serão apresentados a seguir.

3.4.1 Definições Básicas

Seja $s[1 \ldots n]$ uma sequência de n símbolos de um alfabeto Σ, onde $s[n] = \$$ é um símbolo especial que não ocorre em qualquer outra posição de s. Uma árvore de sufixos T_s para s é uma árvore enraizada, com n folhas e no máximo $n-1$ nós internos, tal que:

1. as arestas de T_s são orientadas no sentido raiz $\rightarrow$ folhas e são rotuladas com *sub-*

strings de s;

2. cada nó interno tem pelo menos dois filhos;
3. quaisquer arestas distintas que saem de um mesmo nó estão rotuladas com *substrings* de prefixos diferentes;
4. cada folha está rotulada com um inteiro i, $1 \leq i \leq n$, que representa um índice de s; e
5. a concatenação dos rótulos das arestas de um caminho da raiz até uma folha i corresponde ao sufixo de s que começa na posição i.

A Figura 3.2 mostra a árvore de sufixos para a sequência $s = xabxa\$$. Note que a árvore tem $n = 6$ folhas, numeradas de 1 a 6, de tal modo que o caminho da raiz até a folha i representa o sufixo de s que começa em $s[i]$, $i = 1, \ldots, 6$.

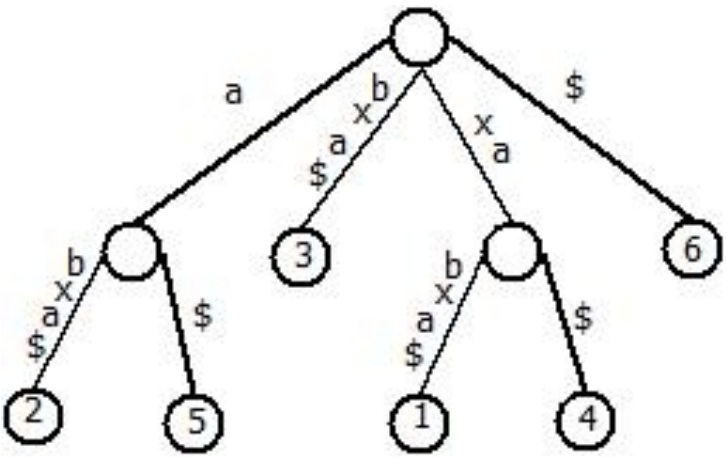


Figura 3.2: Árvore de sufixos para a sequência $xabxa\$$.

A presença do símbolo especial \$ se faz necessária pelo fato de ser impossível representar todos os sufixos de uma sequência que tenha dois ou mais sufixos que compartilham um mesmo prefixo. A Figura 3.3 mostra a árvore de sufixos para a sequência $s = xabxa$. Note que os sufixos 4 e 5 não estão explicitamente representados na árvore. Assim, para garantir que todos os sufixos de s sejam explicitamente representados, faz-se uso do símbolo especial \$.

3.4.2 Busca em Árvore de Sufixos

A aplicação clássica de árvores de sufixos é a busca de todas as ocorrências de uma sequência p de tamanho m em um texto s de tamanho n, dada a árvore de sufixos T_s. A idéia parte do princípio de que qualquer sufixo $s[i \ldots m]$ de s deve estar representado

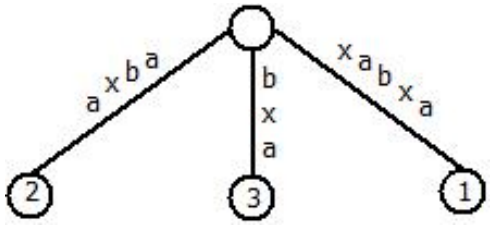


Figura 3.3: Árvore de sufixos para a sequência $xabxa$ (sem o símbolo especial \$).

na árvore, através de um caminho único da raiz de T_s até a folha rotulada com i. Por outro lado, qualquer *substring* $s[i \ldots j]$ de s, $i \leq j \leq m$, é um prefixo do sufixo $s[i \ldots m]$ e, portanto, deve rotular a parte inicial desse caminho único. Assim, para descobrir se há ocorrências de p em s, deve-se percorrer esse caminho, comparando os símbolos de p com os símbolos que rotulam as arestas do caminho, até que p seja integralmente encontrado ou nenhum casamento seja mais possível. No primeiro caso, todas as folhas na subárvore abaixo do nó onde a busca terminou rotulam posições onde p ocorre em s. No segundo caso, p não ocorre em s.

O caminho a ser percorrido na busca é único porque quaisquer duas arestas que saem de um vértice interno de T_s são rotuladas com *substrings* de s que obrigatoriamente começam com símbolos diferentes, pela própria definição da árvore de sufixos.

Na Figura 3.2 podemos ver que a busca por $p = xa$ terminará em um vértice interno, cuja subárvore tem as folhas rotuladas com 1 e 4, significando, portanto, que p ocorre nessas posições. Note que essa busca, mesmo que com resultado positivo, não necessariamente termina em um vértice interno. Ela pode terminar em uma folha, ou mesmo no interior de uma aresta. No caso da busca terminar em uma folha, tem-se apenas uma ocorrência de p em s, enquanto que se a busca terminar antes de alcançar algum vértice qualquer (interno ou folha), esse vértice determina a posição ou as posições de ocorrência.

No caso em que a busca termina sem que p seja integralmente avaliado, tem-se a situação onde p não ocorre em s. Essa busca também pode terminar em um vértice ou no interior de uma aresta.

Uma vez determinado que p ocorre em s, nas posições determinadas pelos rótulos das folhas abaixo do (ou exatamente no) vértice onde a busca terminou, basta executar um algoritmo qualquer de percurso na subárvore enraizada por esse vértice, coletando os rótulos das folhas. Assim, pode-se determinar as ocorrências de p em s em tempo $O(m + k)$, onde k é o número de ocorrências e m é o número de símbolos em p.

A construção da árvore T_s para uma sequência s de tamanho n, como veremos adiante, pode ser feita em tempo $O(n)$. Dessa forma, temos um algoritmo que gasta $O(n)$ para o pré-processamento (construção da árvore) e depois pode-se fazer várias buscas, cada uma com o tempo $O(m + k)$.

3.4.3 Construção de Árvore de Sufixos

Weiner [23] propôs, em 1973, o primeiro algoritmo linear no tamanho de s para a construção da árvore de sufixos T_s. Alguns anos mais tarde McCreight [13] também propôs um algoritmo linear, mais econômico em termos de espaço. Em 1995, Ukkonen [21] apresentou um algoritmo também linear, com as vantagens do algoritmo de McCreight, só que mais facilmente explicável.

O algoritmo de Ukkonen, apesar de mais simples do que os anteriores, ainda requer uma extensa e detalhada explicação, que será omitida neste texto. Uma excelente descrição do algoritmo linear de Ukkonen pode ser obtida em [8]. A fim de exemplificar o processo de construção de uma árvore de sufixos, um algoritmo simples, porém de tempo $O(n^2)$, é descrito a seguir e na Figura 3.4. A idéia geral consiste em incluir em uma árvore inicialmente contendo apenas o sufixo $s[1 \ldots n]$ e em seguida ir adicionando os outros sufixos $s[i \ldots n]$ de s, para $i = 2, \ldots, n$.

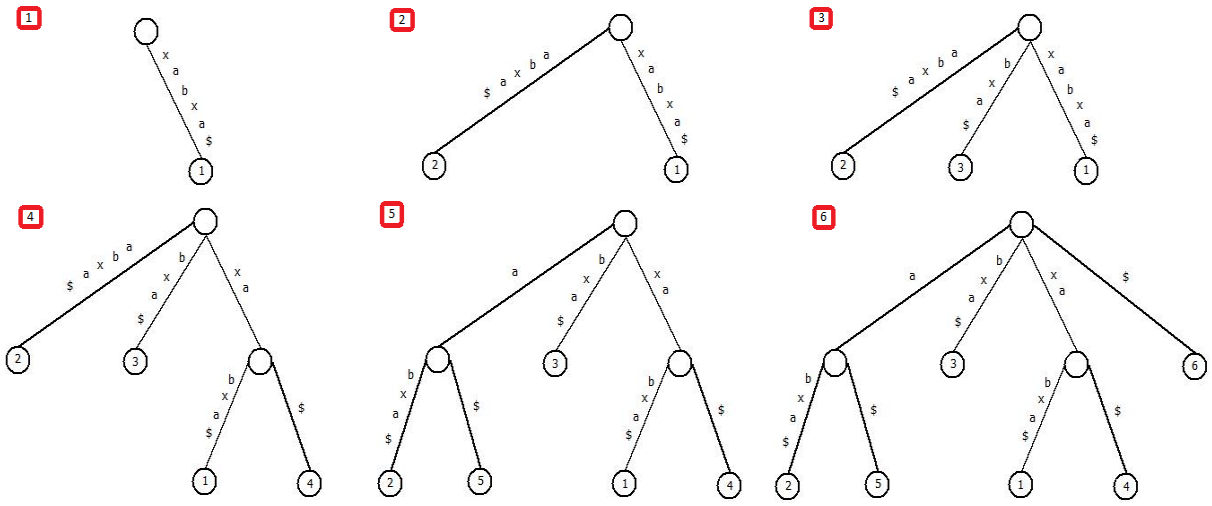


Figura 3.4: Exemplo de construção da árvore de sufixos para a sequência $xabxa\$$.

O passo inicial consiste na construção de uma árvore $T_s{}^1$ contendo apenas a raiz, uma aresta rotulada com o sufixo $s[1 \ldots n]$ e uma folha rotulada com 1. Seja então $T_s{}^i$ a árvore intermediária contendo todos os sufixos $s[j \ldots n]$ de s, com $1 \leq j \leq i$.

A cada passo do algoritmo constrói-se a árvore $T_s{}^{i+1}$, acrescentando o sufixo $s[i+1 \ldots n]$ em $T_s{}^i$, da seguinte forma. Faz-se uma busca de $s[i+1 \ldots n]$ em $T_s{}^i$. Seja C o caminho percorrido em $T_s{}^i$ na busca por s[i+1...n]. Note que C sempre termina antes que $s[i+1 \ldots n]$ seja encontrado integralmente, uma vez que não há sufixos que sejam prefixos de outros sufixos maiores. Seja então $s[r]$ o último símbolo encontrado no caminho C, $i+1 \leq r < n$, ou seja, o sufixo $s[i+1 \ldots n]$ tem duas partes: a primeira parte, s[i+1...r], foi encontrada em C e a segunda parte, $s[r+1 \ldots n]$, não foi encontrada. Essa segunda parte deve ser inserida na árvore imediatamente após o símbolo de $s[r]$ em C. Duas situações podem ocorrer:

- a primeira situação ocorre quando, logo após $s[r]$ em C, temos um nó interno v. Neste caso o algoritmo insere uma nova aresta (v, w) em ${T_s}^i$, rotulada com a sequência $s[r+1 \ldots n]$, e w é uma nova folha rotulada com $i+1$;
- a segunda situação ocorre quando $s[r]$ e $s[r+1]$ estão na mesma aresta de ${T_s}^i$. Neste caso um novo nó interno v é inserido exatamente entre $s[r]$ e $s[r+1]$. Uma nova folha w também é criada e rotulada com $i+1$, e uma nova aresta (v, w) é criada com rótulo $s[r+1 \ldots n]$.

O algoritmo descrito acima é $O(n^2)$, uma vez que são feitas n inserções de novos sufixos, esses de tamanhos $n, n-1, \ldots, 1$; e cada inserção tem o custo do tamanho do sufixo, já que consiste no caminho que esse sufixo percorreria na árvore. Estamos supondo que Σ tem tamanho fixo.

3.4.4 Árvore de Sufixos Generalizada

Uma árvore de sufixos generalizada (GST - *Generalized Suffix Tree*) [8] é uma árvore de sufixos que combina todos os sufixos de um conjunto de *strings*. Sejam s_1 e s_2 duas sequências. Para construção de uma GST para essas duas sequências, primeiro é feita a construção de uma ST para a sequência s_1. A partir da raiz desta árvore, deve-se comparar s_2 com os caminhos na árvore, até que uma diferença ocorra. Neste ponto deve-se adicionar os caracteres restantes de s_2 na árvore. Quando s_2 for inteiramente processada, a árvore combinará os sufixos das duas sequências. Se N é a soma dos tamanhos de todas as sequências, a árvore GST terá no máximo N folhas, e pode ser construída em tempo $O(N)$.

3.4.5 Contextualização em Relação à PCR Múltipla

No contexto deste projeto de pesquisa são consideradas reações de PCR que envolvem, ao mesmo tempo, múltiplas sequências de DNA, chamadas de reação de PCR múltiplas. Em reações dete tipo, pode-se, entre outros, objetivar a amplificação de uma ou mais sequências específicas, presentes na reação.

Em experimentos de PCR múltiplas, a escolha dos *primers* não é mais dependente apenas da composição da sequência que se deseja amplificar, mas também das demais presentes na reação, visto que os *primers* selecionados devem ser específicos para a sequência alvo.

Formalmente, o problema de seleção de *primers* para reações de PCR múltiplas envolve o seguinte problema: dado um conjunto S de N sequências construídas sobre o alfabeto $\Sigma = \{A, C, G, T\}$, determinar, caso exista, uma *substring* t de $s_i \in S$, para $1 \leq i \leq N$, que não seja *substring* de nenhuma outra sequência $s_j \in S$, para $1 \leq j \leq N$ e $j \neq i$ e que respeite características pré-definidas.

Algumas variações do problema citado no parágrafo anterior serão também consideradas, entre elas:

- determinar, caso exista, uma *substring* t de $s_i \in S$ para $1 \leq i \leq N$;
- determinar, caso exista, uma *substring* t de s_i para $\forall s_i \in S$;

3.4.6 Relação entre Árvore de Sufixos com o Projeto de Primer

Considere 3 genomas $G1$, $G2$ e $G3$ presentes em uma mesma solução. Deseja-se amplificar, com um mesmo par de *primers* e em uma mesma reação de PCR, trechos em todos os 3 genomas.

Dado uma árvore de sufixos generalizada representando as sequências $G1$, $G2$ e $G3$ é possível encontrar todas as *substrings* comuns a $G1$, $G2$ e $G3$ que possuem um tamanho mínimo (para que o projeto de primers seja possível). Caso sejam encontradas ao menos duas *substrings* comuns e estas estejam distantes uma das outras (em todos os genomas) por um mínimo de k *bp*, sendo k o tamanho mínimo exijido dos fragmentos dos genomas a serem amplificados, pode-se realizar o projeto de *primers* nessas *substrings*.

Como exemplo, considere a Figura 3.5 onde foram encontrados entre $G1$, $G2$ e $G3$ as *substrings* comuns $ATCCGGTACGTACGTC$ e $CGTACGTTATGGTA$.

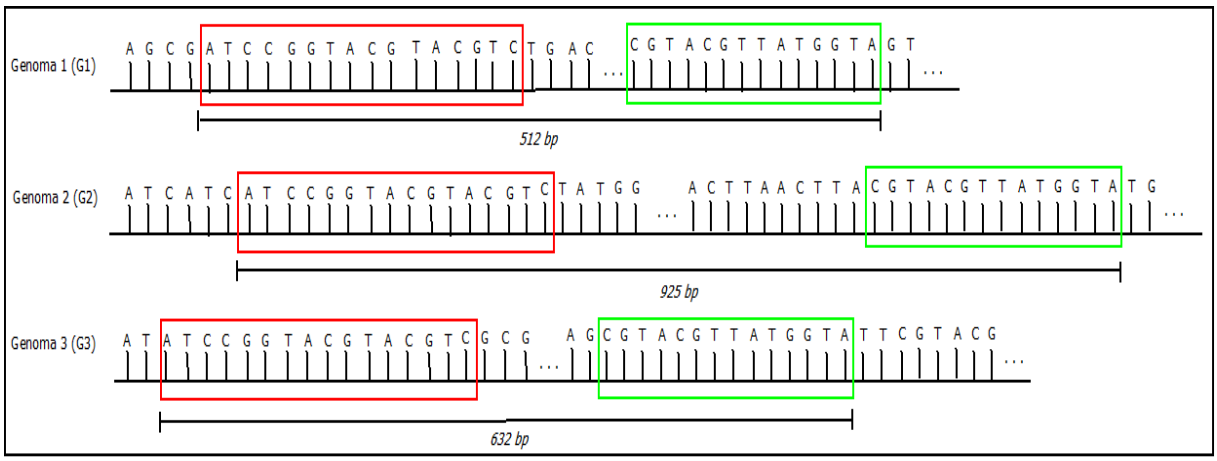


Figura 3.5: Exemplo da relação entre árvore de sufixos com o projeto de primer.

Capítulo 4

Implementação

A implementação foi baseada, em sua maior parte, na utilização da biblioteca *libstree*[1] e alguns outros métodos acrescentados a esta para atender algumas funcionalidades específicas ao problema tratado, que serão descritas a seguir.

4.1 O que é *libstree*?

A biblioteca *libstree* é um pacote que contém uma implementação de uma árvore de sufixos generalizada, escrito em C. *Libstree* é uma biblioteca livre sob os termos da licença BSD, baseia-se em Linux, FreeBSD e OpenBSD.

A geração de árvores de sufixo em *libstree* é altamente eficiente e implementado utilizando o algoritmo de Ukkonen. Isso significa que *libstree* constrói árvores de sufixos em tempo linear ao comprimento das cadeias.

4.2 O que fornece *libstree*?

Libstree pode lidar com múltiplas cadeias na árvore de sufixos, incluindo a inserção dinâmica e remoção de cadeias. Ele oferece vários meios de obter informações sobre os nós da árvore, como busca em profundidade, busca em largura, percurso sobre todas as folhas da árvore, como encontrar a maior *substring* comum e a maior *substring* repetida, entre outros métodos. Esta biblioteca também fornece uma documentação[2] da API, para auxiliar o usuário usar a *libstree*.

[1] disponível no endereço http://www.icir.org/christian/downloads/

[2] disponível no endereço http://www.icir.org/christian/*libstree*/manual/index.html

4.3 Como *libstree* foi utilizada?

Libstree opera com *strings*. *Strings* são instâncias do tipo LST_String, e árvores de sufixo são instâncias do tipo LST_STree. Cada cadeia pertence a uma classe *string*.

Como o aplicativo terá que lidar com várias sequências de caracteres de cada vez, é necessário um local para armazenar este conjunto de sequências de caracteres. *Libstree* fornece o tipo $LST\_StringSet$ para esta funcionalidade e também para construir uma árvore de sufixos para um determinado $LST\_StringSet$.

Principais métodos da biblioteca *libstree* utilizados:

- $LST\_String*\ lst\_string\_new(void*\ data,\ u\_int\ item\_size,\ u\_int\ num\_items)$ - Função que aloca um espaço de memória e copia os dados para essa área, em seguida, retorna uma nova *string*.
- $u\_intlst\_string\_get\_length(LST\_String*\ string)$ - Função retorna o número de caracteres na *string*.
- $lst\_string\_print\ const\ char*(LST\_String*\ string)$ - Função que retorna um ponteiro para uma sequência de caracteres que representa LST_String especificada.
- $LST\_StringSet*\ lst\_stringset\_new(void)$ - Função que cria um novo *stringset*, local para armazenar um conjunto de sequências de caracteres. *Stringset* é a maneira como várias sequências de caracteres são passadas para um outro método.
- $void\ lst\_stringset\_add(LST\_StringSet*\ set, LST\_String*\ string)$ - Função que adiciona uma *string* para o conjunto especificado.
- $void\ \ lst\_stringset\_foreach(LST\_StringSet*\ \ set, LST\_StringCB\ \ callback, void*\ user\_data)$ - Função que, para cada *string* no conjunto *set*, é chamada a função *callback*, passando *user_data*, que são paramêtros que o usuário pode passar para *callback*. A função *callback* é criada pelo usuário, com funcionalidade especificada por ele, mas deve possuir a seguinte assinatura:
  - $void\ \ (*LST\_StringCB)\ \ (LST\_String\ \ *string, void\ \ *data)$ - onde $(*LST\_StringCB)$ é o nome dado à função pelo usuário como, por exemplo, *string*_cb, $(LST\_String\ *string)$ é a *string* recebida como paramêtro e data é algum outro parâmetro que o usuário deseja passar para a função, ou $NULL$, caso contrário.
- $LST\_STree*lst\_stree\_new(LST\_StringSet*\ strings)$ - Função que cria uma árvore de sufixos a partir do conjunto de *strings* recebido como parâmetro.
- $void\ lst\_stree\_free(LST\_STree*\ tree)$ - Função que libera toda a memória alocada pela LST_STree especificada.
- $LST\_String\ *\ lst\_node\_get\_string(LST\_Node*\ node, int\ max\_depth)$ - Função que obtém a *string* contendo todos os caracteres encontrados quando a percurso vai da raiz até o nó passado por parâmetro.

- *void lst_alg_leafs(LST_STree* tree, LST_NodeVisitCB callback, void* data)* - Função que itera sobre todas as folhas da árvore, chamando a função *callback* para cada folha visitada. A função *callback* é criada pelo usuário, com funcionalidade especificada por ele, mas deve possuir a seguinte assinatura:

  – *void (*LST_NodeVisitCB) (LST_Node *node, void *data)* - onde *(*LST_NodeVisitCB)* é o nome dado à função pelo usuário como, por exemplo, node_cb, *(LST_Node *node)* é o nó folha recebido como paramêtro e data é algum outro parâmetro que o usuário deseja passar para a função, ou *NULL*, caso contrário.

- *LST_StringSet* lst_alg_longest_common_substring(LST_STree* tree, u_int min_len, u_int max_len)* - Função que procura a maior *substring* comum na árvore. Para limitar o tamanho da *string*, pode-se passar um valor apropriado para *max_len*, ou 0 para obter a maior *string*(s) possível. Para obter a maior *substring* comum de pelo menos um número determinado de caracteres, usar *min_len*, ou 0 para não limitar o tamanho.

4.4 Funcionalidades

As implementações do pacote *libstree* não foram alteradas. Entretanto, novos códigos foram inseridos a fim de implementar funcionalidades adicionais necessárias ao tratamento do problema de busca por regiões onde *primers* podem ser projetados. Baseado nas implementações do pacote *libstree* e nos novos códigos inseridos, seguem as funcionalidades do programa que foram criadas:

- Encontrar um segmento específico, cujo tamanho está em um determinado intervalo *(MIN, MAX)*, que:

  - ocorre em uma sequência escolhida em relação as demais;

  - ocorre em uma sequência qualquer, ou seja, buscar a(s) sequência(s) específica(s) de um organismo em relação ao demais envolvidos na reação;

  - ocorre em uma sequência dada em relação a um subconjunto das demais (considere o conjunto sendo estritamente menor que o conjunto original, exceto a sequência escolhida);

- Encontrar um segmento específico de tamanho x que:

  - ocorre em uma sequência escolhida em relação as demais;

  - ocorre em uma sequência qualquer, ou seja, buscar a(s) sequência(s) específica(s) de um organismo em relação aos demais envolvidos na reação;

  - ocorre em uma sequência dada em relação a um subconjunto das demais (considere o conjunto sendo estritamente menor que o conjunto original, exceto a sequência escolhida);

- Encontrar o menor segmento específico que:

  - ocorre em cada uma das sequências do conjunto;

  - ocorre em um subconjunto de sequências (considere o conjunto sendo estritamente menor que o conjunto original);

- Encontrar um segmento comum, cujo tamanho está em um determinado intervalo *(MIN, MAX)*, que:

  - ocorre em todas as sequências do conjunto;

  - ocorre em um subconjunto de sequências (considere o conjunto sendo estritamente menor que o conjunto original);

- Encontrar um segmento comum de tamanho x que:

  - ocorre em todas as sequências do conjunto;

  - ocorre em um subconjunto de sequências (considere o conjunto sendo estritamente menor que o conjunto original);

- Encontrar o maior segmento comum que:

  - ocorre em todas as sequências do conjunto;

  - ocorre em um subconjunto de sequências (considere o conjunto sendo estritamente menor que o conjunto original);

- Encontrar o maior segmento comum, cujo tamanho está em um determinado intervalo *(MIN, MAX)*, que:

  - ocorre em todas as sequências do conjunto;

  - ocorre em um subconjunto de sequências (considere o conjunto sendo estritamente menor que o conjunto original);

- Encontrar uma sequência comum a um subconjunto que seja específica em relação a outro subconjunto.

Capítulo 5

Resultados

Usando uma máquina com 8 processadores Intel(R) Xeon(R) 2.13GHz, 20 GB de memória RAM foram executados vários experimentos com o algoritmo, principalmente testando o tempo de execução com sequências de tamanhos variados. Para os testes foram considerados os genomas dos organismos: *Mycobacterium bovis BCG str. Tokyo 172*[1], *Mycobacterium bovis AF2122/97*[2] e *Mycobacterium bovis BCG str. Pasteur 1173P2*[3], cujos tamanhos, em pb, são 4.371.711, 4.345.492 e 4.374.522, respectivamente. Entretanto, descobrimos que a *libstree* não consegue lidar com arquivos maiores do que 4.000 caracteres cada. Por isso, foram considerados apenas os 4.000 primeiros caracteres dos genoma dos organismos *Mycobacterium bovis BCG str. Tokyo 172*, *Mycobacterium bovis AF2122/97* e *Mycobacterium bovis BCG str. Pasteur 1173P2*.

As figuras citadas abaixo são gráficos baseados no número de bases da sequência nucleotídica x tempo (segundos) que mostram um resultado mais extensivo demonstrando a desempenho do algoritmo na:

- construção de uma árvore de sufixos na Figura 5.1;
- busca da maior *substring* comum na Figura 5.2;
- busca de todas as *substrings* comums entre min=5 e max=10 na Figura 5.3;
- busca de todas as *substrings* específicas entre min=5 e max=10 na Figura 5.4.

Os tempos gastos dependem não só do tamanho do genoma, mas também da composição de cada um deles.

As consultas que podem ser efetuadas já foram descritas anteriormente e, no programa, elas estão disponíveis em um menu onde o usuário pode ir destinando sua pesquisa conforme sua necessidade. Em caso de dúvidas, existe no menu uma opção de ajuda onde pode-se obter informações de como cada consulta deve ser efetuada.

Em linhas gerais, a execução do programa é feita digitando-se:

[1] disponível em ftp://ftp.ncbi.nih.gov/genbank/genomes/Bacteria/Mycobacterium_bovis_BCG_Tokyo_172/
[2] disponível em ftp://ftp.ncbi.nih.gov/genbank/genomes/Bacteria/Mycobacterium_bovis/
[3] disponível em ftp://ftp.ncbi.nih.gov/genbank/genomes/Bacteria/Mycobacterium_bovis_BCG_Pasteur_1173P2/

./projetoPrimersAS

Depois da execução, o seguinte menu aparece:

Projeto de *Primers* via Árvore de Sufixos
(1) Gerar árvore
(2) *Substring* comum
(3) *Substring* específica
(4) Sequência comum a um subconjunto e específica em relação a outro subconjunto
(5) Ajuda
(6) Sair
Escolha uma opção para consulta:

Dependendo da opção, é solicitado que o usuário entre com os genomas para a consulta. Por exemplo:
$numero\_arquivos < nome\_arquivos \ldots >$
$2\ CR380947.fna\ BX248333.fna$

Em cada opção, o usuário poderá direcionar sua consulta conforme as funcionalidades descritas na seção 4.4.

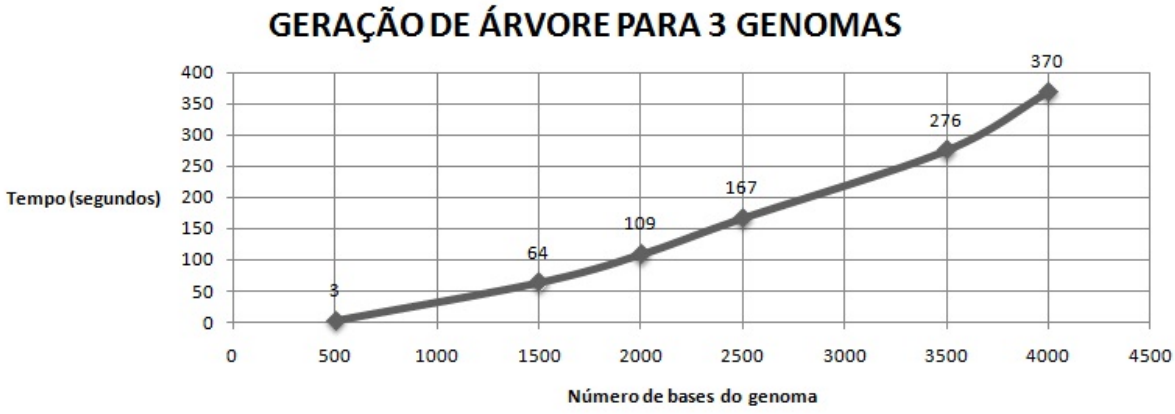


Figura 5.1: Gráfico de desempenho da geração da árvore de sufixos.

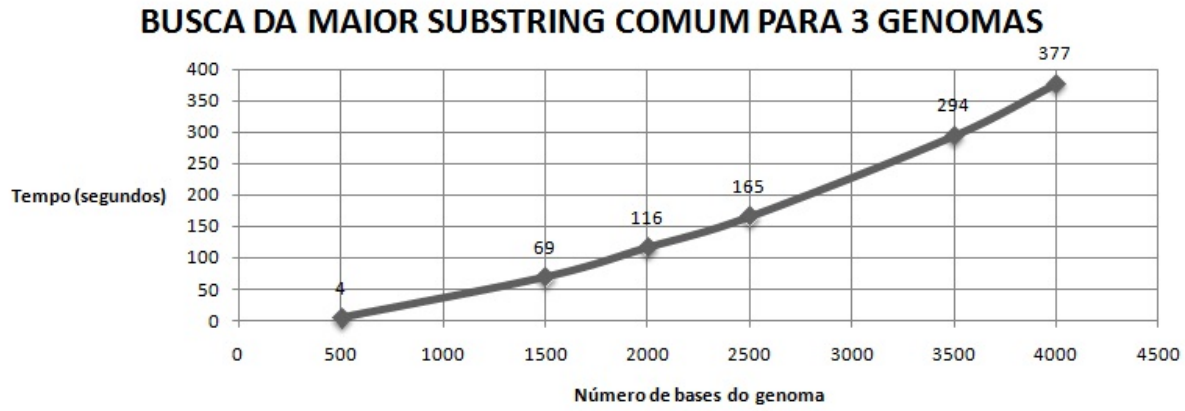


Figura 5.2: Gráfico de desempenho da busca da maior *substring* comum.

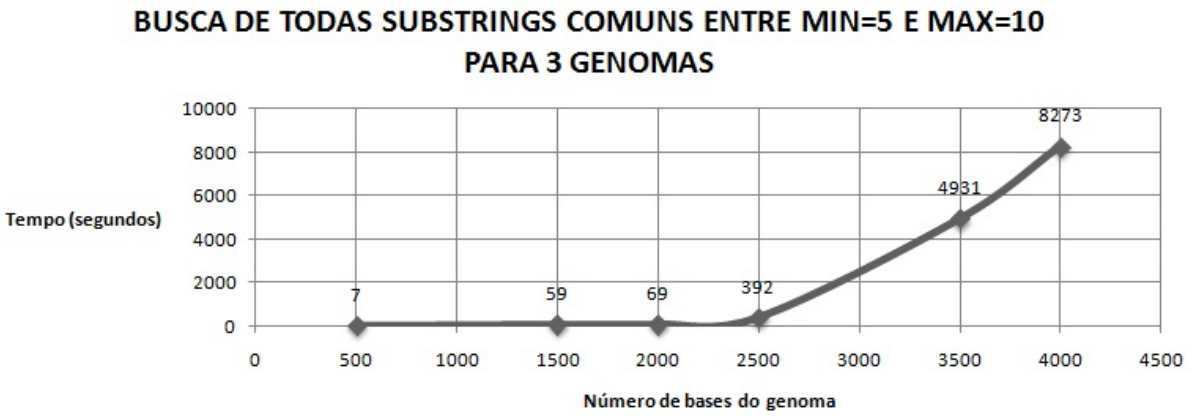


Figura 5.3: Gráfico de desempenho da busca de todas as *substrings* comums entre min=5 e max=10.

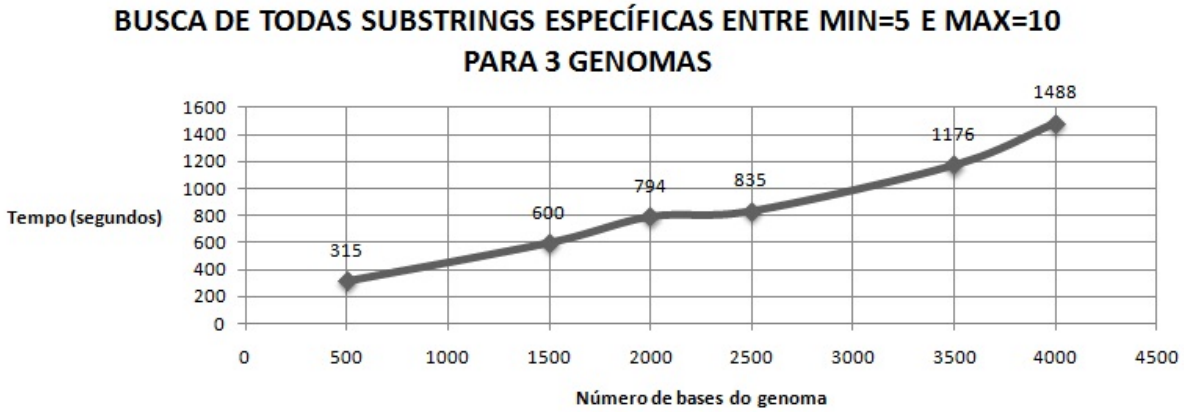


Figura 5.4: Gráfico de desempenho da busca de todas as *substrings* específicas entre min=5 e max=10.

Capítulo 6

Conclusão e Trabalhos Futuros

Com o surgimento dos Projetos Genoma, as pesquisas na área de Bioinformática vêm crescendo muito nos últimos anos, gerando resultados de suma importância para o conhecimento de como a vida funciona da maneira mais exata possível.

Para a análise destes dados, surgiram inúmeras ferramentas computacionais que são responsáveis pelo grande avanço da Bioinformática, porém não existe uma ferramenta com o objetivo específico para o projeto de *primers* para reações de PCR múltiplas.

Devido à relativa dificuldade em se determinar *primers*, mais especificamente o projeto de *primers* em reações de PCR múltiplas, este trabalho apresenta uma abordagem para o problema via árvore de sufixos e implementação de um *software*. Dentre vários benefícios, essa aplicação permite ao pesquisador direcionar suas consultas, tentando garantir como saída trechos onde os *primers* devem ser projetados para serem usados em experimento e PCR múltiplas.

Como a primeira parte do projeto era estudar conceitos da área de Biologia Molecular necessários ao entendimento do problema de projeto de *primer*, foi feito um levantamento de informações no qual resultou em um conhecimento melhor do problema que iríamos trabalhar.

Em seguida, destinamos ao estudo e descrição de experimento que se baseiam em reações de PCR múltiplas. Simultaneamente, realizou-se uma formalização do problema de projeto de *primer* para reações de PCR múltiplas e suas variantes. Com estas informações focamos o estudo em uma estrutura de dados para solucionar este problema. Assim, decidimos por utilizar árvore de sufixos, com o estudo teórico especificamente focados na construção e busca.

Após esta fase inicial de aquisição de conhecimento, iniciou-se a fase de desenvolvimento. Uma pesquisa também foi realizada para definição de como iríamos construir árvore de sufixos que utilizaríamos no programa e decidimos por utilizar a biblioteca *libstree*, devido a eficiência na geração de árvores de sufixo e a documentação disponível da biblioteca. Com a biblioteca para criação das árvores de sufixos já escolhida, foram modeladas quais consultas seriam realizadas pelo programa. Após a implementação, foram realizados diversos testes e os respectivos melhoramentos necessários.

Assim, esse tipo de aplicação deve ser capaz de permitir o manuseio de milhares de bases passados para a consulta. Portanto algumas falhas acontecidas ao longo da execução

do projeto foram relacionadas a biblioteca *libstree*, devido à sua limitação na quantidade de bases que esta tem a capacidade de processar, que apesar de ter se demonstrado como uma poderosa ferramenta para se trabalhar com árvore de sufixos, apresentou alguns problemas que interferiram de alguma maneira no resultado final do projeto.

De um modo geral, o trabalho alcançou o objetivo proposto de desenvolver uma ferramenta para determinar trechos onde o projeto de *primers* (considerando reações de PCR múltiplas) deve ocorrer. Entretanto, resta resolver a limitação em relação ao tamanho do genoma manipulado pela biblioteca utilizada (*libstree*) que não permitiu que o problema fosse de forma satisfatória resolvido, se considerarmos que o tamanho dos genomas dos organismos, mesmo os mais simples, é de ordem de milhares de pares de bases.

Como trabalhos futuros propomos um estudo mais avançado sobre a biblioteca *libstree*, a fim de resolver sua limitação em relação ao tamanho das sequências de entrada, bem como da implementação de novas consultas que possam direcionar ainda mais o processo de projeto de *primers*. Pode-se também direcionar os estudos/esforços no sentido da criação de um *software* que aborde o problema de projeto de *primer* como um todo, visto que nosso trabalho aborda apenas uma fase inicial deste processo, que consiste na determinação de trechos onde os *primers* devem ser projetados.

Referências Bibliográficas

[1] ACCESS Excellence. DNA - A More Detailed Description, Acesso em: 14 out. 2003.

[2] ALKAMI Biosystems. *Alkami Quick Guide for PCR*. United States of America: Guanabara Koogan, 1999.

[3] ALLAWI, H. T.; SANTALUCIA. J. Jr. Thermodynamics and NMR of internal G·T mismatches in DNA. *Biochemistry*, 1997.

[4] BAPTISTA, P.V.; Osório-Almeida. *Genética Molecular - Protocolos trabalhos práticos 2*. Sci. Amer. Books, W.H. Freeman and Co., N.Y., 2003.

[5] BRESLAUER, K. J.; FRANK, R.; BLOCKER, H.; MARKY, L. A. Predicting DNA duplex stability from the base sequence. *Proc Natl Acad Sci USA*, 1986.

[6] (CHAMBERLAIN, J.L., BUSH, R. HAMMETT, A.L.). Non-Timber Forest Products: The other forest products. *Forest Products Journal*, 1998.

[7] GIBAS, Cynthia; JAMBECK, Per. *Desenvolvendo Bioinformática: ferramentas de software para aplicações em biologia*. Rio de Janeiro: Campus, 2001.

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

[9] HIGUCHI, R., DOLLINGER, G., WALSH, P.S., GRIFFITH, R. Amplificação e detecção simultâneas de seqüências específicas do ADN. *Biotecnologia*, 1992.

[10] HOFMANN-LEHMANN, R., SWENERTON, R.K., LISKA, V., LEUTENEGGER, C.M., LUTZ, H., MCCLURE, H.M., RUPRECHT, R.M. Sensitive and robust one-tube real-time reverse transcriptase-polymerase chain reaction to quantify SIV RNA load: comparison of one- versus two-enzyme systems. *AIDS Research and Human Retroviruses*, 2000.

[11] HOWLEY, P.M.; ISRAEL, M.F.; LAW, M-F.; MARTIN, M.A. A rapid method for detecting and mapping homology between heterologous DNAs. *Journal of Biological Chemistry*, 1979.

[12] LEUTENEGGER, C.M., Klein, D., HOFMANN-LEHMANN, R., MISLIN, C., HUMMEL, U., BONI, J., BORETTI, F., GUENZBURG, W.H., LUTZ, H. Rapid feline immunodeficiency virus provirus quantitation by polymerase chain reaction using

the TaqMan fluorogenic real-time detection system. *Journal of Virological Methods*, 1999.

[13] MCCREIGHT, E.M. A space-economical suffix tree construction algorithm. *J.ACM*, 1976.

[14] MONTERA, L.; NICOLETTI M.C. The PCR primer design as a metaheuristic search process. Master's thesis, Universidade Federal de São Carlos, 2008.

[15] RYCHLIK, W.; SPENCER, W.J.; RHOADS, R.E. Optimization of the annealing temperature for DNA amplification in vitro. *Nucleic Acids Research*, 1990.

[16] SANTALUCIA, J. Jr.; ALLAWI, H. T.; SENEVIRATNE, P. A. Improved Nearest-Neighbor parameters for predicting DNA duplex stability. *Biochemistry*, 1996.

[17] SCARIOT, F. Sistema Web para projeto e análise de primers. Master's thesis, Universidade do Vale do Itajaí, 2004.

[18] SILVA, F. H. *Módulo: Biologia Molecular*. I Escola Brasileira de Inteligência Artificial e Bioinformática - InBio - São Carlos-UFScar, 2001a.

[19] SUGIMOTO, N.; NAKANO, S.; YONEYAMA, M.; HONDA, K. Improved thermodynamic parameters and helix initiation factor to predict stability of DNA duplexes. *Nucleic Acids Research*, 1996.

[20] TELLES, G.P., ALMEIDA, N.F., MARTINEZ, F.H.V. Algoritmos e heurísticas para comparações exata e aproximada de sequências. *XXIV Jornadas de Atualização em Informática*, 2005.

[21] UKKONEN, E. On-line costruction of suffix-trees. *Algorithmica*, 1995.

[22] WALLACE, R.B.; SHAFFER, J.; MURPHY, R.F.; BONNER, J.; HIROSE, T.; ITAKURA, K. Hybridization of synthetic oligodeoxyribonucleotides to φX 174 DNA: the effect of single base pair mismatch. *Nucleic Acids Research*, 1979.

[23] WEINER, P. Linear pattern matching algorithms. *In Proc. of the 14th IEEE Symp. on Switching and Automata Theory*, 1973.

[24] Welsh & McClelland. Amplified Polymorphism - Polymerase Chain Reaction, 1990.

[25] Williams et al. Random Amplified Polymorphic DNA, 1990.

[26] ZAHA, A. *Biologia Molecular Básica*. Porto Alegre: Mercado Aberto, 1996.