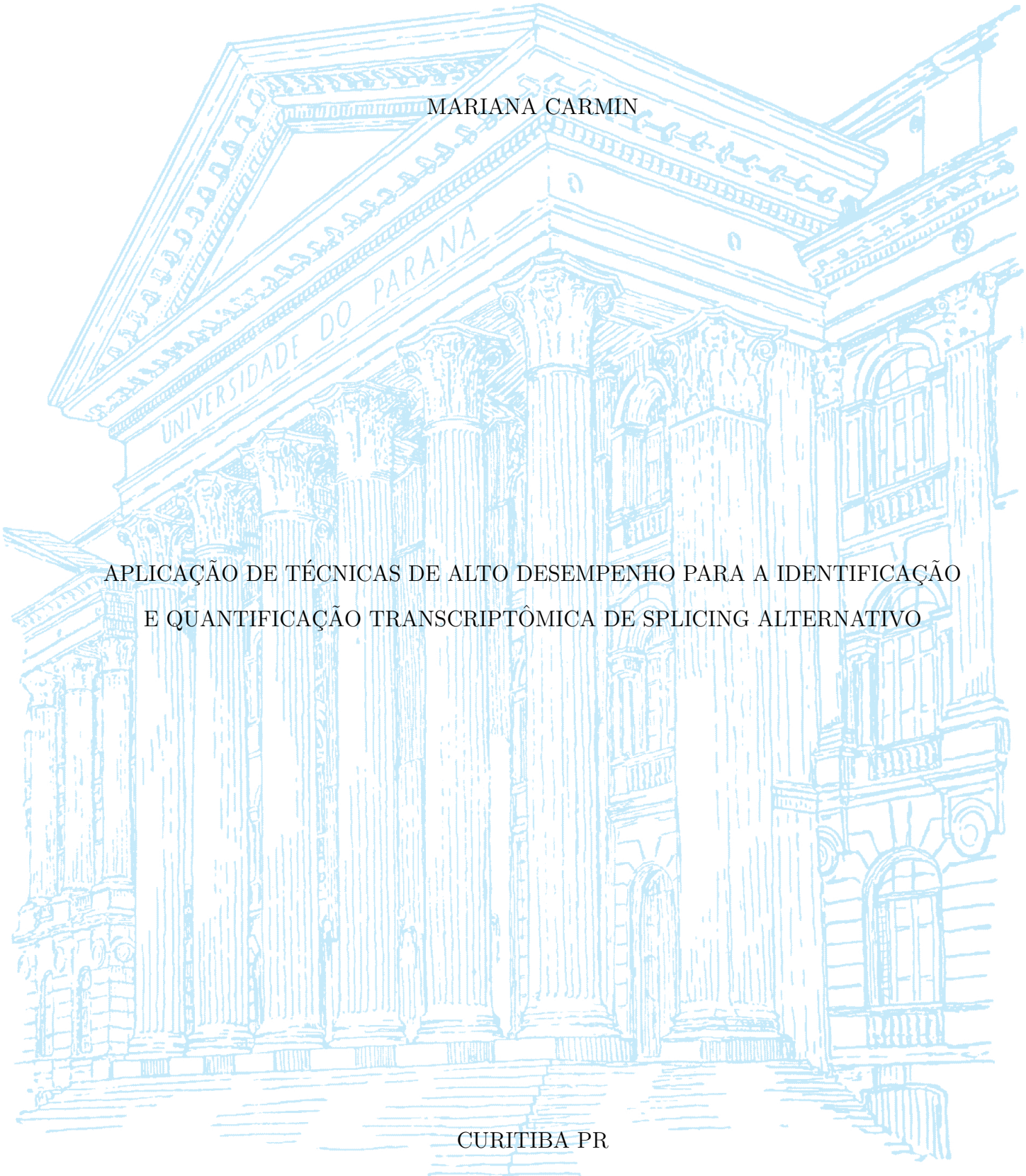
UNIVERSIDADE FEDERAL DO PARANÁ

MARIANA CARMIN



APLICAÇÃO DE TÉCNICAS DE ALTO DESEMPENHO PARA A IDENTIFICAÇÃO E QUANTIFICAÇÃO TRANSCRIPTÔMICA DE SPLICING ALTERNATIVO

CURITIBA PR

2019

MARIANA CARMIN

APLICAÇÃO DE TÉCNICAS DE ALTO DESEMPENHO PARA A IDENTIFICAÇÃO E QUANTIFICAÇÃO TRANSCRIPTÔMICA DE SPLICING ALTERNATIVO

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Informática Biomédica, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Informática Biomédica.*

Orientador: Marco Antonio Zanata Alves.

CURITIBA PR

2019

[noautoscale]0-iniciais/aprovacao.pdf

RESUMO

No ácido desoxirribonucleico (DNA) de eucariotos, organismos que possuem seu material genética dentro do núcleo, existem partes codificantes e partes não codificantes. Quando um ácido ribonucleico (RNA) é gerado ele porta tanto as partes codificantes, que geram proteínas, quanto partes não codificantes. O processo regulatório natural de retirada das partes não codificantes (introns) do RNA é chamado de *splicing*. Dessa maneira, podemos ter o *splicing* alternativo ao formar diferentes arranjos a partir da mesma molécula de RNA, possibilitando assim que um único gene codifique para múltiplas proteínas . Atualmente, os dados provenientes de técnicas de sequenciamento profundo, por exemplo RNA-Seq, permitem reconhecer os diferentes transcritos gerados por eventos de *splicing* alternativo (Hooper, 2014). A tarefa de quantificação transcriptômica é altamente repetitiva, possuindo independência entre os dados. A quantificação da expressão de um gene é independente de quaisquer outros genes, por isso é passível do uso de computação paralela. Neste trabalho, propomos a criação de uma ferramenta capaz de ler um arquivo do tipo Gene Transfer File (GTF) e gerar um outro arquivo GTF contendo todos os eventos de *splicing* encontrados. A ferramenta AStalavista foi utilizado como modelo para o desenvolvimento deste trabalho. Para o desenvolvimento utilizamos a linguagem de programação C por prover alto desempenho além de permitir diversas formas para a paralelização do programa. Para a detecção dos eventos são analisadas todas as combinações, dois a dois, dos transcritos do mesmo gene. Neste trabalho foram analisados eventos de uso alternativo de exon, sítios doadores alternativos, sítios aceptores alternativos e retenção de introns. Para cada evento foi atribuído um código diferente, neste código também são mostrados os exons envolvidos no evento de splicing. Esses códigos são gravados no arquivo GTF de saída, que também contém informações sobre o gene analisado, seu cromossomo e posições cromossômicas de ocorrência. Neste trabalho foi usada a anotação genômica de três organismos *Zebrafish* (Danio rerio), *Mouse* (Mus musculus) e *Human* (Homo sapiens). Comparando com o trabalho base, os resultados preliminares mostram que para o mesmo conjunto de teste, os resultados gerados apresentaram um aumento de desempenho de pelo menos 5× no tempo de execução dependendo da arquitetura do computador utilizado.

Palavras-chave: Splicing alternativo, Computação paralela, bioinformática

ABSTRACT

In deoxyribonucleic acid (DNA) of eukaryotes, organisms that have their genetic material within the nucleus, there are coding and non-coding parts. When a ribonucleic acid (RNA) is generated, it carries both coding parts, the ones that generate proteins and non-coding parts. The natural regulatory process of removing non-coding parts (introns) from RNA is called splicing. In this way, we can have alternative splicing, in this case forming different transcripts from the same RNA molecule. This fact enabling a single gene to code for multiple proteins. Currently, data from deep sequencing techniques, for example RNA-Seq, allow us to recognize the different transcripts generated by alternative splicing events (Hooper, 2014). The task of transcriptomic quantification is highly repetitive, having independence between data. Quantifying the expression of a gene is independent of any other genes, so it is possible to use parallel computing. In this work, we propose the creation of a tool capable of reading a Gene Transfer File (GTF) file and generating another file containing all found splicing events. The AStalavista tool was used as a model for the development of this work. In our development we use the programming language C because it provides high performance and allows several ways to parallelize the program. For event detection are analyzed all combinations, two by two, of the transcripts that belongs to the same gene. In this work we analyzed events of alternative exon use, alternative donor sites, alternative acceptor sites and intron retention. Each event has been assigned a different code, this code also shows the exons involved in the splicing event. These codes are written to the output file, which also contains information about the analyzed genome, its chromosome, and chromosome positions. In this work we used genomic annotation of three organisms Zebrafish (Danio rerio), Mouse (Mus musculus) and Human (Homo sapiens). Comparing with the base work, the preliminary results show that for the same test set, the generated results showed a performance increase of at least 5× depending on the computer architecture used.

Keywords: Alternative splicing, high processing computing, bioinformatics

LISTA DE FIGURAS

LISTA DE TABELAS

Glossário

AB3C Associação Brasileira de Bioinformática e Biologia Computacional. 31

asgal *Alternative Splicing Graph ALigner*. 21

CPU *Central Processing Unit*. 22

DNA Ácido desoxirribonucleico. iii, iv, 14

GPU *Graphics Processing Unit*. 21

HPC *High-performance computing*. 18

IEEE Instituto de Engenheiros Eletricistas e Eletrônico. 20

miso *Mixture of Isoforms*. 21

mRNA RNA mensageiro. 14

NCBI *National Center for Biotechnology Information*. 20

RNA Ácido ribonucleico. iii, iv

rRNA RNA ribossomal. 14

tRNA RNA transportador. 14

UFPR Universidade Federal do Paraná. 27

SUMÁRIO

1 INTRODUÇÃO

O projeto genoma humano, iniciado em 1990 e que teve sua conclusão em 2003, consistiu em uma grande iniciativa de parcerias internacionais. Seu objetivo era sequenciar completamente o genoma humano, permitindo o conhecimento de todos os seus nucleotídeos.

Com a conclusão do projeto genoma humano, foi mostrado que a quantidade de genes codificantes, ou seja, genes que são transformados em proteína é um pouco maior que 30.000 [Lander, 2001]. O ser humano produz uma quantidade de proteínas próxima de 100.000. Assim, nota-se que a quantidade de genes codificantes e as proteínas produzidas não representam uma relação de um para um.

Esse fenômeno é explicado por diversos fatores, um deles é o mecanismo que permite que sejam produzidas diferentes proteínas a partir do mesmo gene, denominado *splicing* alternativo [Modrek e Lee, 2002]. No *splicing* alternativo os exons e introns que compõem um transcritos, ou partes desses exons e introns, são removidos ou adicionados gerando transcritos diferentes (alternativos), que podem ou não, possuir tamanhos distintos [Goldstrohm et al., 2001].

O aumento dos dados transcriptômicos, que são oriundos por exemplo de RNASeq (técnica que utiliza sequenciamento de última geração), possibilitou o estudo do perfil de expressão de *splicing* alternativo em diferentes amostras. Dessa forma, permitindo a análise em diferentes tecidos e condições [Hooper, 2014].

Ao mesmo tempo, a evolução nas tecnologias de integração levaram as indústrias de processadores a investirem no uso de processadores formados de múltiplos núcleos de processamento para alavancar o desempenho das atuais famílias de processadores.

Realizar essa análise em várias amostras pode demorar horas ou até mesmo dias, dependendo da quantidade de sequências [Li et al., 2018]. Além disso, muitas análises são realizadas em computadores pessoais, o que torna muito custosa a tarefa de analisar um grande volume de dados.

Neste trabalho propomos a criação de uma ferramenta computacional otimizada, baseada no *software* AStalavista [Foissac e Sammeth, 2007], para quantificação de eventos de splicing. Utilizando técnicas de computação de alto desempenho para garantir uma ferramenta que disponibilize os resultados, de um grande volume de dados, em um tempo viável e em um computador pessoal.

2 CONCEITOS PRELIMINARES

Neste capítulo serão explanados os conceitos base necessários para o entendimento desse trabalho, tanto do ponto de vista biológico quanto computacional.

2.1 BASE NITROGENADA

As bases nitrogenadas fazem parte da constituição do DNA. Elas são compostos cíclicos que possuem nitrogênio. Se uma base, possuir um anel duplo será classificada como purina e se possuir anel simples será classificada como pirimidina [Rodwell, 2016]. Como ilustrado na Figura 2.1, do lado esquerdo temos uma purina que possui anel duplo e do lado direito uma pirimidina com anel simples.

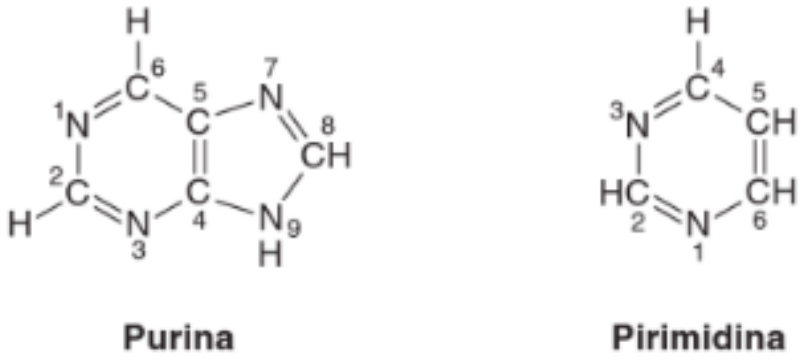


Figura 2.1: Estrutura química de uma purina do lado esquerdo, com anel duplo e de uma pirimidina do lado direito, com anel simples. Ao lado esquerdo, uma purina destacando seus elemento químicos nitrogênio (N), carbono (C) e hidrogênio (H), ao lado direito, uma pirimidina com os mesmo três elementos químicos. [Murray et al., 2010].

As bases nitrogenadas que compõem o DNA e o RNA são classificadas da seguinte forma:

1. **Purinas:** Adenina e Timina.
2. **Pirimidinas:** Citosina, Uracila e Guanina.

O DNA possui em sua composição adeninas, citosinas, timinas e guaninas, já o RNA possui adeninas, citosinas, uracilas e guaninas. Os dois se diferem na base nitrogenada uracila, que só é encontrada no RNA, substituindo as timinas presentes no DNA [Rodwell, 2016].

2.2 NUCLEOTÍDEOS

As bases nitrogenadas são um dos componentes dos nucleotídeos, os precursores de ácidos nucleicos. Além das bases nitrogenadas, a formação de um nucleotídeo é feita por outros dois compostos: Um açúcar pentose; Uma ligação fosfato.

Os três compostos unidos formam um nucleotídeo e se organizam como mostrado na Figura 2.2 .

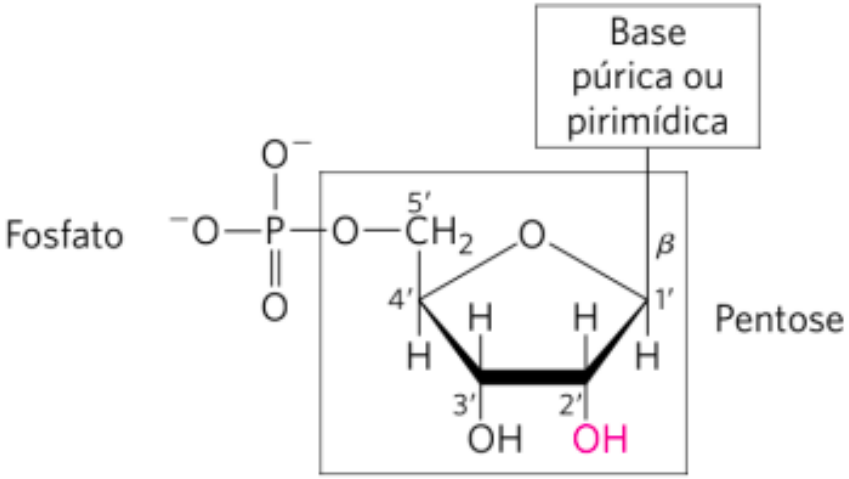


Figura 2.2: Estrutura de um nucleotídeo contendo uma base nitrogenada, podendo ser uma purina ou pirimidina, um fosfato e um açúcar pentose. A pentose, destacada ao centro em um quadrado, mostra as posições de cada elemento químico, a base nitrogenada ligada a posição 1' e o fosfato ligado ao carbono na posição 5' [Nelson e Cox, 2018].

Para formar a cadeia de ácido desoxirribonucleico (DNA) ou ácido ribonucleico (RNA) os nucleotídeos são unidos por ligações fosfodiéster, onde um grupo 5'-fosfato de uma unidade nucleotídica é ligado ao próximo nucleotídeo em sua extremidade 3'-hidroxila [Nelson e Cox, 2018]. As coordenadas 5' e 3' são usadas para denotar o sentido das operações de tradução, transcrição e replicação. A operação de transcrição será explicada nas próximas sessões.

2.3 ÁCIDO DESOXIRRIBONUCLEICO (DNA)

A estrutura do ácido nucleico (DNA) foi postulada por James Watson e Francis Crick em 1953 [Watson et al., 1953]. Um modelo com duas cadeias de DNA helicoidais enroladas em torno do mesmo eixo para formar uma dupla hélice [Nelson e Cox, 2018]. Neste modelo cada base nucleotídica pareá-se com uma base da fita oposta, obedecendo o pareamento de Adenina (A) com Timina (T) e Citosina (C) com Guanina (G), como é mostrado em na Figura 2.3.

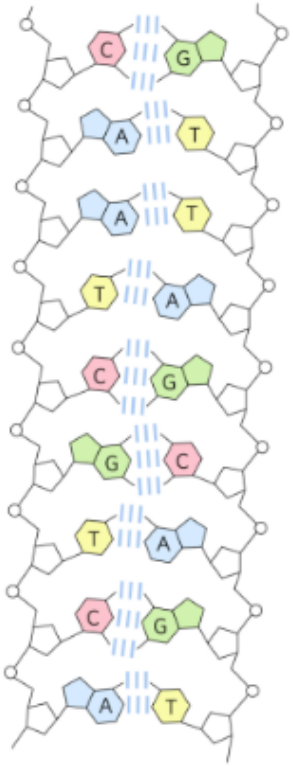


Figura 2.3: Estrutura de uma molécula de DNA, dupla fita constituída de nucleotídeos, cada nucleotídeo contendo uma base nitrogenada, podendo ser uma timina (T), citosina (C), guanina (G) ou adenina(A). As ligações e pareamento entre bases nitrogenadas é mostrado, respeitando o pareamento de "A"com "T"e "C"com "G". [Nelson e Cox, 2018].

O DNA detém o código genético inteiro, ou seja, detém a informação capaz de garantir o funcionamento do organismos. O DNA se encontra no núcleo celular e para ser transportado para fora do núcleo é transformado em uma molécula de RNA [Nelson e Cox, 2018].

2.4 ÁCIDO RIBONUCLEICO (RNA)

Os ácidos ribonucleicos (RNA) formam o principal canal de transmissão de informação do DNA até a formação de proteínas. Apesar do RNA ser muito similar ao DNA, ele possui algumas diferenças, como não possuir a base nitrogenada Timina e no seu lugar possuir Uracila e ser formado por uma fita simples [Conn e Stumpf, 1980]. Os RNAs são elementos que desempenham diversas funções dentro da célula, como realizar a regulação de expressão gênica e transmissão de informação, suas funções estão ligadas ao seu tipo [Conn e Stumpf, 1980]. Os tipos de RNA mais comumente estudados são:

1. RNA transportador (tRNA): transporta aminoácidos no momento da tradução.
2. RNA mensageiro (mRNA): codifica uma sequência de aminoácidos.
3. RNA ribossomal (rRNA): compõe os ribossomos, que são as organelas responsáveis por traduzir o mRNA.

2.5 AMINOÁCIDOS

São conhecidos, atualmente, na literatura 20 aminoácidos. Esses aminoácidos unidos por ligações covalentes formam polímeros, esses polímeros são classificados como proteínas.

A asparagina, o primeiro aminoácido descoberto, foi encontrado em 1806 e o último dos 20 a ser encontrado, a treonina, foi descoberta em 1938. Os nomes dados aos aminoácidos são derivados da fonte em que foram isolados, por exemplo, o glutamato foi isolado pelo primeira vez no glúten do trigo [Nelson, 2014].

2.6 PROTEÍNAS

Proteínas são as macromoléculas, moléculas orgânicas de elevada massa molecular, com maior abundância no meio celular, elas controlam quase todos os processos que ocorrem em uma célula. Por isso quando um processo biológico é alvo de estudos, uma ou mais proteínas são analisadas.

Existem inúmeras proteínas em apenas uma célula, resultado de diferentes combinações dos 20 aminoácidos existentes, essas macromoléculas desempenham funções completamente diferentes.

Usando essas diferentes construções de proteínas os organismos podem gerar como produto final enzimas, hormônios, anticorpos, transportadores, fibras musculares, proteínas das lentes dos olhos, penas, teias de aranha, chifres de rinocerontes, proteínas do leite, antibióticos, venenos de cogumelos e uma infinidade de moléculas diferentes dependendo do tipo do organismos e da sua necessidade no momento da produção [Nelson, 2014].

As proteínas são o produto final do processo descrito como o dogma central da biologia molecular.

2.7 DOGMA CENTRAL DA BIOLOGIA MOLECULAR

Formulado em 1958 o dogma central da biologia molecular mostra o caminho que a informação percorre do DNA até sua transformação em proteína. Foi postulado que o DNA, que se encontra no núcleo de células eucariotas, serve como molde para a produção de RNA. O RNA por sua vez migra para fora do citoplasma, onde é decodificado para a formação de uma cadeia de aminoácidos [Crick, 1958]. Esse fluxo de informação segue o esquema exemplificado em 2.6.

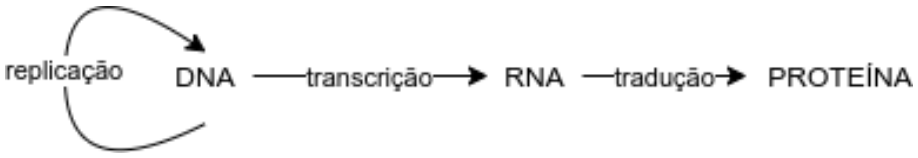


Figura 2.4: Fluxo de informação do dogma central da biologia molecular, mostrando as etapas desde o DNA até a formação de proteína. A replicação, duplicação da molécula de DNA, transcrição, síntese de uma molécula de RNA a partir de uma molécula de DNA e tradução, decodificação da molécula de RNA em proteína. Fonte: Autoria própria.

A replicação é responsável por gerar novas fitas de DNA. Nesse processo, as enzimas, um dos produtos finais formadas por proteínas, são responsáveis por usar cada uma das fitas da dupla hélice para sintetizar uma fita complementar.

Já na transcrição uma das fitas de DNA é usada para sintetizar uma fita de RNA. Essa fita de RNA será traduzida em cadeias de aminoácidos fora do núcleo celular, no caso de eucariotos [Crick, 1958].

A seguir será apresentada com mais detalhes a etapa de transcrição, por possuir uma relevância maior para o entendimento desse trabalho.

2.7.1 Transcrição

O processo de transcrição consiste na síntese de uma cadeia de RNA a partir de uma fita molde de DNA.

Enzimas são responsáveis por reconhecer a posição de ligação com a cadeia de DNA, além de reconhecer quais das fitas da dupla hélice deve ser usada como molde.

A dupla hélice do DNA é temporariamente aberta, com a finalidade de que a enzima percorra a fita, adicionando os nucleotídios, um a um, na fita de RNA.

Como ilustrado na Figura 2.5, a cadeia de DNA está sendo aberta e sintetizada pela RNA polimerase produzindo uma nova cadeia de RNA.

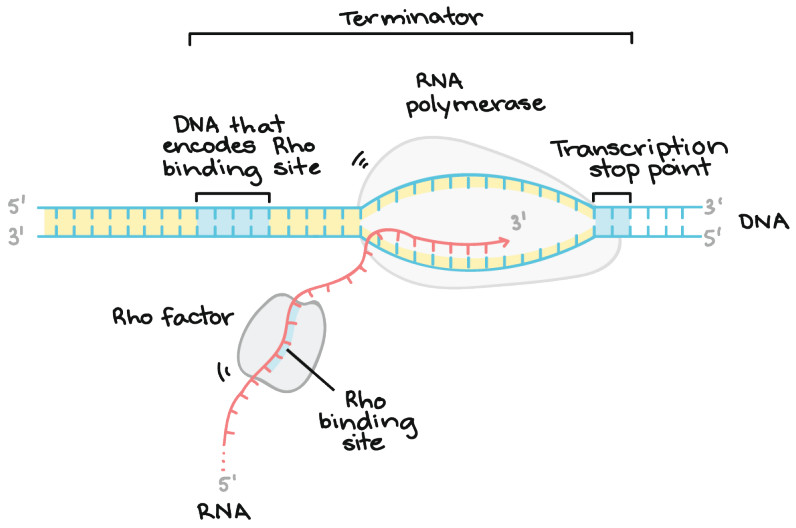


Figura 2.5: Exemplificação do processo de transcrição do DNA para uma molécula de RNA. Molécula de RNA semiaberta e uma enzima realizando o processo de síntese do RNA, a partir dos nucleotídeos do DNA, repeitando os pontos de inicio e fim da transcrição, contidos na molécula de DNA.

O processo de transcrição termina quando a enzima encontra um *códon* de parada. Quando isso ocorre ela se desconecta da fita de DNA e libera a fita de RNA recém sintetizada no núcleo [Karp, 2011].

A fita de RNA deve atravessar a membrana do núcleo, para que no citoplasma passe pelo processo de tradução.

2.8 *SPLICING*

A maioria dos genes humanos possuem exons separados por introns, ou seja, partes codificantes e partes não codificantes. Quanto o gene é transcrito para uma sequência de RNA, ele porta tanto as partes que codificam para proteína quanto as partes que não codificam.

Essa partes não codificantes, apesar de possuírem sequências para o processamento correto do RNA, devem ser removidas antes da fase de tradução. O processo de

remoção das partes não codificantes e junção das partes codificantes é chamado de splicing [Rowen et al., 2002].

2.9 *SPLICING* ALTERNATIVO

Em eucariotos, um dos principais mecanismos ligados a variedade no transcriptoma é o *splicing* alternativo.

Neste mecanismos, dependendo das condições em que a célula se encontra ou o seu nível de desenvolvimento são geradas combinações diferentes de RNA madura. Esse RNA maduro pode ser do tipo mensageiro ou não codificante.

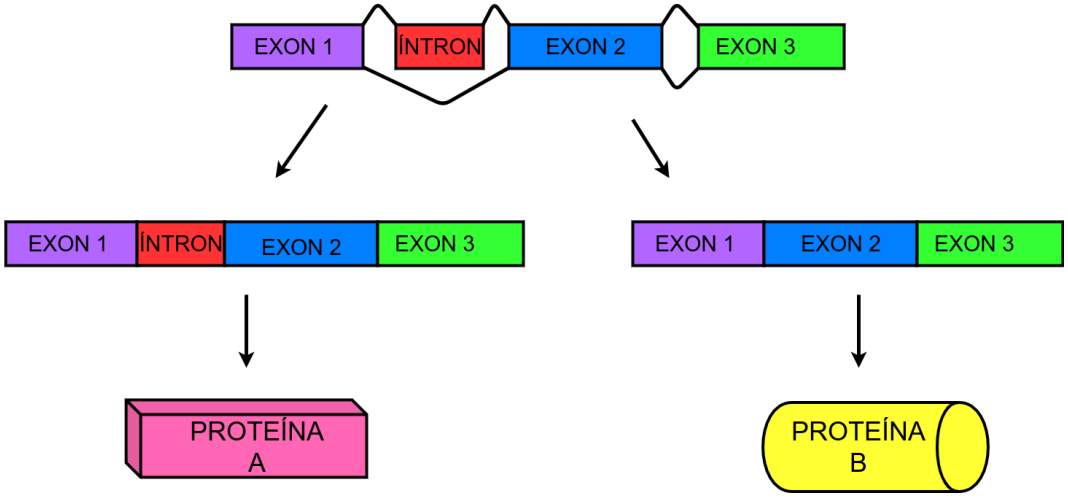


Figura 2.6: Processo de *splicing* alternativo, uma molécula de RNA, parte superior, antes do processamento, portando três exons e um íntron e logo abaixo duas moléculas que sofreram o processo de *splicing* alternativo, a esquerda uma molécula portando tanto os três exons quanto o íntron, originando uma proteína do tipo "A"e a esquerda uma molécula portando apenas os exons, originando uma proteína do tipo "B". Fonte: Autoria própria.

O processo de *splicing* alternativo é responsável por aumentar a diversidade de proteínas. Conferindo a capacidade de um gene gerar mais do que um transcrito maduro [Modrek e Lee, 2002].

Estima-se que 90% dos genes humanos sofre processo de *splicing*. Na literatura, atualmente, é reconhecida a alta frequência e importância desse evento [Lareau et al., 2004].

O processo de *splicing* vem sendo utilizado para diversos estudos, como os evolutivos, respondendo perguntas de como qual seria a origem desse mecanismo e como ele atua no processo de diversidade biológica e especiação [Lareau et al., 2004].

2.10 QUANTIFICAÇÃO DE *SPLICING* ALTERNATIVO

O processo de quantificação de *splicing* alternativo poder ser realizado utilizando alguns métodos. A primeira classificação é de acordo com o modo de deteccção que pode ser baseado em:

1. **Detecção de eventos Alternative Splicing (AS):** O método baseado em detectar eventos AS pode utilizar a técnica de *de novo assembly* ou anotação gênica

2. **Reconstrução de isoforma:** Similar ao método anterior também poder utilizar a técnica de *de novo assembly* ou anotação gênica [Denti et al., 2018].

Para o desenvolvimento desse trabalho foi utilizada a técnica de reconstrução de isoforma com a sub técnica de anotação gênica.

2.11 LINGUAGEM INTERPRETADA

Em linguagens interpretadas existe apenas um código fonte. Quando há a execução, um interpretador é chamado e executa as funções descritas no código fonte.

Uma vantagem existente nesse tipo de arquivo é a possibilidade de execução em diversos tipos de arquiteturas e plataformas, uma vez que o código fonte é apenas interpretado. Este fato apesar de ser um vantagem por permitir a execução em diferentes máquina, este tipo de abordagem não permite que o código se molde as arquiteturas de forma a prover uma execução mais otimizada.

Um exemplo de linguagem interpretada é a linguagem Python e a linguagem Java que possui uma abordagem híbrida, sendo tanto interpretada quanto compilada [Lana, 2018].

2.12 LINGUAGEM COMPILADA

Já em linguagens compiladas, quando a execução é chamada um programa, denominado compilador, é chamado. O compilador é responsável por transformar o código fonte em linguagem de máquina, ou seja, em binário (zeros e uns).

Vale ressaltar que a compilação otimiza o código para que o código rode da forma mais otimizada possível. A linguagem de programação C, assim como a linguagem C++ são exemplos de linguagens interpretadas [Lana, 2018].

2.13 COMPUTAÇÃO DE ALTO DESEMPENHO

Computação de alto desempenho ou *High-performance computing* (HPC) é um conceito usado desde a década de 40, dessa época até os anos 90, os computadores usados para essa finalidade eram máquinas especializadas que possuíam processadores dedicados para a realização de aplicações científicas ou de engenharia [STRINGHINI, 2018].

Nos anos 90 a industria de semicondutores aumentava a frequência de operações, gerando um crescente aumento do desempenho dos processadores. Porém, nos anos 2000 foram enfrentadas cada vez mais dificuldades no processo de aumentar a frequência das operações, a solução para continuar aumentando o desempenho dos processadores foi acoplar múltiplos núcleos computacionais em um processador. São chamados processadores *multi-core* e seus núcleos compartilham recursos, como por exemplo, memórias *cache* [STRINGHINI, 2018].

Com essas arquiteturas paralelas, aplicações com alta demanda computacional, como simulações climática, reações químicas e *pipelines* de bioinformática, podem ser executados em menor tempo, se os recursos da máquina forem usados de forma otimizada e coordenada.

Visando aumentar a quantidade e a qualidade dos processadores em arquiteturas paralelas, alguns fatores devem ser reformulados:

1. **Arquitetura:** deve ser capaz de integrar núcleos inter-operáveis.

2. **Entrada e saída:** além do processamento, a velocidade da transmissão dos dados deve ser levada em consideração

3. **Memória:** deve preocupar-se com cache compartilhada e possíveis sobrecargas de interconexões.

4. **Programação:** deve ser repensada para usufruir da disponibilidade de inúmeros núcleos.

5. **Consumo Energético:** para garantir a viabilidade, deve ser pensado em maneiras de economia em *hardware* e eficiência em programações paralelas.

2.14 PROGRAMAÇÃO PARALELA

A programação usada do cenário de computação de alto desempenho, ou seja, usando múltiplos cores é chamada de computação paralela. Nesse modelo são abstraídos detalhes sobre a plataforma, de modo a expressar paralelismo em alto nível.

Fica a cargo do programador encontrar as áreas potencialmente paralelas em seu programa, bem como a independência dos dados. O modelo fornece as primitivas para que a paralelização ocorra [Foster, 1999].

Um programa é dito como paralelo quando é constituído de vários blocos de código sequencial, cujos blocos podem ser executados de modo concorrente. Esse processo de paralelização pode ocorrer de em um mesmo processador, utilizando o modelo de memória compartilhada ou utilizando o processador de diversas máquinas, no modelo chamado de troca de mensagem.

O modelo de troca de mensagem, funciona com várias máquinas conectadas via rede, realizando a comunicação por meio de mensagens. O modelo MPI (*Message Passing Interface*) utiliza esse tipo de técnica.

Já no modelo de memória compartilhada, os processos compartilham a CPU e a memória, desta forma o acesso aos dados deve ser feito visando os dados que estão em cache para otimizar as latência de acesso. Os modelos que usam esse tipo de técnica são o OpenMP e o PThreads[Foster, 1999].

2.15 CONCLUSÃO

Nota-se o avanço dos estudos na área de *splicing* alternativo, com o crescimento dos estudos na área surgem experimentos ainda mais complexos e com um volume de dados cada vez maior. Devido a essa tendência e a independência dos dados analisados, a otimização de programas usados para a quantificação de *splicing* alternativo se faz necessária. Uma vez que a quantificação é o resultado usado para a analise diminuir o tempo computacional torna viável o crescimento da complexidade dos estudos.

3 ESTADO DA ARTE

Neste capítulo, serão abordadas as estratégias usadas para o levantamento bibliográfico, bem como os resultados obtidos com as pesquisas feitas. Além disso, uma descrição dos trabalhos correlatos a este recentemente publicados na área e os que são amplamente usados pelos bioinformatas.

3.1 ESTRATÉGIA DE BUSCA

Para o levantamento bibliográfico deste trabalho foram realizadas buscas em três bases distintas: IEEE, PUBMED e SpringerLink. As palavras-chaves usadas envolviam *parallel computing*, RNASeq, *splicing* alternativo e quantificação. A aplicação de demais critérios ficou condicionada a base usada para a busca e é descrito nas seções posteriores.

Os resultados foram filtrados de acordo com a lista de critérios em 3.1. Primeiro foram analisados os critérios de rejeição, na ordem mostrada na tabela, caso o artigo analisado não apresentasse nenhum critério de rejeição era anotado como aceito. Caso apresentasse eram analisadas as exceções de aceitação, caso o artigo se enquadrasse mesmo possuindo critérios de rejeição era marcado como aceito [1].

Critérios de rejeição
Data anterior a 2010
Não apresenta uma ferramenta computacional
Ferramenta desenvolvida para uma única espécie
Exceções para aceitação
Survey de ferramentas computacionais
Técnica desenvolvida para uma espécie mas reproduzível para demais

Tabela 3.1: Lista de critérios usados para a seleção ou exclusão de um artigo encontrado na busca.

1. **IEEE** : A plataforma Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE) possui foco maior na parte de ciência da computação, esta foi selecionada por representar uma das maiores plataformas de busca na área da computação.

   No IEEE não foram encontrados muitos trabalhos e dentre os encontrados apenas um foi selecionado. Os termos usados para a busca foram : *parallel computing*, RNASeq e *splicing* [2]

2. **PUBMED** : O portal PUBMED, plataforma dentro do NCBI (*National Center for Biotechnology Information*), apresenta a maior concentração de publicações biomédicas, a seleção deste portal foi ocorreu por essa razão.

   Os termos usados foram: *parallel computing*, RNASeq e *splicing* [3]. Este foi o portal com a maior quantidade de trabalhos marcados como selecionado.

[1] O levantamento sistemático completo pode ser visualizado no link: https://docs.google.com/spreadsheets/d/1whFhtboExATERaon8jm-MZE0_YBdAM2lnsbXPUox67k/edit?usp=sharing

[2] O resultado da busca pode ser observado em: ,https://ieexplore.ieee.org/search/searchresult.jsp?action=searchnewsea $truequeryText = (((\%22parallel\%20computing\%22)\%20AND\%20RNA-seq)\%20AND\%20splicing)$

[3] O resultado da busca pode ser observado em: https://rnajournal.cshlp.org/search?fulltext=alternative+splicingsubm

3. **SpringerLink** : O portal SpringerLink foi selecionado por possuir publicações tanto da área computacional quanto médica. Os termos usados foram: *splicing*, RNASeq, *parallel computing* e quantificação [4]Sendo este o portal que apresentou a maior quantidade de resultados com as palavras-chaves usadas.

3.2 RESULTADO OBTIDOS

Os trabalhos encontrados nas plataformas listadas acimas passaram por uma análise com base nos critérios da Tabela 3.1 e em uma leitura superficial do artigo.

Como resultado foi obtido uma lista com treze artigos que serviram como base para o desenvolvimento deste trabalho.

3.3 TRABALHOS CORRELATOS

Neste seção serão expostos brevemente os trabalhos correlatos mais relevantes. Para isso foram selecionados as mais novas publicações na área e os trabalhos mais amplamente usados, ou seja, os trabalhos que foram mais citados como ferramenta em experimentos de artigos encontrados.

1. **Paean:** Neste trabalho é proposto um método de quantificação transcriptômica usando GPU. Os dados usados para a quantificação são dados de RNASeq e a quantificação é realizada com base em tabelas de junções [Li et al., 2018]. Esse é um dos trabalhos mais novos publicados na área de computação sobre o assunto.

2. **asgal:** Neste trabalho a abordagem consiste em criar um gráfico de *splicing* a partir do genoma de referência e da anotação gênica e realizar o alinhamento da nova amostra com esse gráfico criado para a detecção de novos eventos [Denti et al., 2018]. Esse é um dos trabalhos mais novos publicados na área de biomédica sobre o assunto.

3. **miso:** Este é um dos trabalhos mais citados como ferramenta usada para a analise de *splicing* em amostras nos artigos encontrados. Consiste em um modelo estatístico para a estimar a quantidade de expressão de isoformas e de exons alternativos [Katz et al., 2010].

4. **AStalavista:** Neste trabalho a detecção dos eventos de *splicing* ocorre em arquivos de anotação gênica. Os transcritos são comparados dois a dois para a detecção de eventos e são unidos eventos semelhantes e removidos repetições [Foissac e Sammeth, 2007].

Na Tabela 3.2 podemos comparar as técnicas utilizadas pelos quatro trabalhos correlatos, considerando qual é o método utilizada, de reconstrução de isoformas ou detecção de eventos AS. Além disso, qual a técnica utilizada no artigo, se é baseada em anotação gênica ou *de novo assembly* e se possui algum tipo de paralelismo.

[4]O resultado da busca pode ser observado em: https://link.springer.com/search?date-facet-mode=betweenfacet-language=%22En%22showAll=falsefacet-end-year=2019facet-start-year=2015facet-content-type=%22Article%22query=parallel+computing+AND+RNA-seq+AND+splicing+AND+quantification

Título do trabalho	**Método usado**	**Técnica usada**
Paean	Reconstrução de isoforma	*De novo assembly*
asgal	Detecção de eventos AS	Baseado em anotação gênica
miso	Reconstrução de isoforma	*De novo assembly*
AStalavista	Reconstrução de isoforma	Baseado em anotação gênica

Tabela 3.2: Lista de critérios usados para a seleção ou exclusão de um artigo encontrado na busca.

3.4 CONCLUSÃO

Os trabalhos encontrados são desenvolvidos para CPU, a tendencia observada, ao longo do tempo, foi o aumento da complexidade estatística envolvida com os modelos. Os trabalhos mais recentes visam prover modelos que possam inferir ainda mais informações a partir dos dados.

Foi encontrados apenas um trabalhos que realizam a paralelização dessa tarefa, porem é realizada em GPU.

Podemos notar que existe uma demanda por gerar um algoritmo ainda mais rápido, que seja capaz de quantificar amostras de RNA.

4 TRABALHO PROPOSTO

Este trabalho propõe o desenvolvimento de uma ferramenta de quantificação de *splicing* alternativo. O trabalho AStalavista [Foissac e Sammeth, 2007], foi utilizado como base para o desenvolvimento, por ser um trabalho que possui versões executáveis disponíveis, permitindo a comparação dos resultados. Além disso, o trabalho AStalavista, aplica o método escolhido para a quantificação, baseado em anotação gênica.

A proposta é criar um programa com melhor desempenho, utilizando técnicas de computação paralela, sendo capaz de analisar um grande volume de dados de amostras.

4.1 FORMATAÇÃO DA ENTRADA

O arquivo de entrada do programa é um arquivo do tipo *Gene transfer format* (GTF). Este arquivo contém informações de anotação gênica.

Um exemplo é mostrado na Figura 4.1. Cada linha representa um leitura realizada, podendo ser um exon, *start codon*, *stop códon*, CDS, entre outros . A primeira coluna representa o cromossomo ligado aquela amostra, a segunda coluna a técnica usada para o sequenciamento ou a base de dados ou projeto associado ao dado e a terceira descreve qual o tipo de dado, um gene, exon, códon. As duas colunas seguintes representam o inicio e fim.

Este arquivo foi utilizado como entrada, similar ao AStalavista, pois dessa forma a é possível a verificação e validação dos eventos encontrados.

```
4   ensembl gene    17308   18211   .   -   .   gene_id "ENSDARG00000102141"; gene_version "2";
4   ensembl transcript  17308   18211   .   -   .   gene_id "ENSDARG00000102141"; gene_version "2";
transcript_id "ENSDART00000171737"; transcript_version "2";
4   ensembl exon    18134   18211   .   -   .   gene_id "ENSDARG00000102141"; gene_version "2";
transcript_id "ENSDART00000171737"; transcript_version "2"; exon_number "1"; transcript_source
"ensembl"; transcript_biotype "protein_coding"; exon_id "ENSDARE00001173708"; exon_version "2";
```

Figura 4.1: Exemplo de arquivo GTF contendo na primeira linha um gene, na segunda linha um de transcrito e na terceira linha um de exon. O gene foi sequenciado pelo projeto Ensembl, e está localizado da posição 17308 à 18211 do cromossomo 4 e possuí o id ENSDARG00000102141. Já o transcrito, logo abaixo, e que pertence ao mesmo gene, como nota-se pelo gene_id, está localizado na posição 17308 à 18211 do cromossomo 4 e possui id ENSDART00000171737. Por fim, o exon na linha 3 está localizado na posição 18134 à 18211 do cromossomo 4 e corresponde ao primeiro exon do transcrito. Fonte: Autoria própria.

Como é observado na Figura 4.1, há três exemplos de dados, gene, *transcript* e exon, os três são encontrados no cromossomo 4 e foram sequenciados com a técnica de ensembl. Na 4ª e 5ª coluna são registradas as posições cromossômicas de cada um dos elementos, neste caso, em que posição do cromossomo 4 se encontra o gene ENSDARG00000102141, que é de 17308 à 18211.

Logo abaixo do gene temos os transcritos, podemos notar que o transcrito está associado ao gene descrito, pois o campo gene_id apresenta o mesmo valor. Além disso também são trazidas as informação do id do transcrito, esse id é usado para verificar quais exons, linha seguinte, pertencem a ele.

4.2 CODIFICAÇÃO DOS EVENTOS

Similar ao trabalho de [Foissac e Sammeth, 2007], neste trabalho foram usadas codificações para cada evento de *splicing* encontrado. As codificações são as seguintes:

1. **1-2ˆ,0:** Usado quando um evento de *exon skipping* é encontrado
2. **1ˆ,2ˆ:** Usado quando um evento de sítio doadores alternativos é encontrado
3. **1-,2-:** Usado quando um evento de sítio aceptores alternativos é encontrado
4. **1ˆ2-,0:** Usado quando um evento de retenção de intron é encontrado

4.3 FORMATAÇÃO DA SAÍDA

A saída do programa escrita em um arquivo do tipo GTF. Neste arquivo, cada linha representa um evento de *splicing*, mostrando quais foram os transcritos associados aquele evento, o gene e seu tipo, como ilustrado na Figura 4.2.

```
4       Undefined       as_event        43645   44106   .       +       .       transcript_id
"ENSDART00000159640,ENSDART00000170197/ENSDART00000175348"; gene_id "4:31259-45642W"; flanks "43645-,44106^"; structure
"0,1^2-"; splice_chain ",43773^43846-"; sources "Undefined,Undefined"; NMorMRNA "null"; degree "4"; dimension "2_2";
4       Undefined       as_event        44430   44771   .       +       .       transcript_id "ENSDART00000170197/
ENSDART00000175348,ENSDART00000159640"; gene_id "4:31259-45642W"; flanks "44430-,44771^"; structure "0,1^2-"; splice_chain
",44516^44616-"; sources "Undefined,Undefined"; NMorMRNA "null"; degree "4"; dimension "2_2";
4       Undefined       as_event        45166   45396   .       +       .       transcript_id "ENSDART00000159640/
ENSDART00000170197,ENSDART00000175348"; gene_id "4:31259-45642W"; flanks "45166-,45396]"; structure "0,1^2-"; splice_chain
",45291^45376-"; sources "Undefined,Undefined"; NMorMRNA "null"; degree "4"; dimension "2_2";
4       Undefined       as_event        122531  122728  .       +       .       transcript_id
"ENSDART00000172255,ENSDART00000186461"; gene_id "4:121709-125402W"; flanks "122531-,122728^"; structure "0,1^2-";
splice_chain ",122583^122663-"; sources "Undefined,Undefined"; NMorMRNA "null"; degree "4"; dimension "2_2";
4       Undefined       as_event        188736  191603  .       -       .       transcript_id
"ENSDART00000192054,ENSDART00000169187/ENSDART00000169667"; gene_id "4:184863-193762C"; flanks "191603^,188736-";
structure "0,1-2^"; splice_chain ",189516-189433^"; sources "Undefined,Undefined"; NMorMRNA "null"; degree "4"; dimension
"2_2";
4       Undefined       as_event        308812  310007  .       +       .       transcript_id "ENSDART00000067489/
ENSDART00000131419/ENSDART00000145428,ENSDART00000174021"; gene_id "4:308688-317971W"; flanks "308812^,310007^"; structure
"1-,2-"; splice_chain "309913-,309934-"; sources "Undefined,Undefined"; NMorMRNA "null"; degree "4"; dimension "2_2";
4       Undefined       as_event        313172  313941  .       +       .       transcript_id "ENSDART00000067489/
ENSDART00000131419/ENSDART00000174021,ENSDART00000145428"; gene_id "4:308688-317971W"; flanks "313172^,313941^"; structure
"1-,2-"; splice_chain "313807-,313917-"; sources "Undefined,Undefined"; NMorMRNA "null"; degree "4"; dimension "2_2";
4       Undefined       as_event        350294  353526  .       +       .       transcript_id
"ENSDART00000132625,ENSDART00000067486"; gene_id "4:331351-353924W"; flanks "350294-,353526^"; structure "0,1^2-";
splice_chain ",350706^350998-"; sources "Undefined,Undefined"; NMorMRNA "null"; degree "4"; dimension "2_2";
4       Undefined       as_event        365492  367537  .       +       .       transcript_id "ENSDART00000103645/
ENSDART00000139832,ENSDART00000163436"; gene_id "4:357810-386168W"; flanks "365492^,367537^"; structure "1-,2-";
splice_chain "367413-,367419-"; sources "Undefined,Undefined"; NMorMRNA "null"; degree "4"; dimension "2_2";
4       Undefined       as_event        367537  367985  .       +       .       transcript_id
"ENSDART00000103645,ENSDART00000163436"; gene_id "4:357810-386168W"; flanks "367537^,367985^"; structure "1-,2-";
splice_chain "367854-,367863-"; sources "Undefined,Undefined"; NMorMRNA "null"; degree "4"; dimension "2_2";
```

Figura 4.2: Exemplo de arquivo de saída contendo os eventos encontrados e seus respectivos códigos e posições GTF. Possuindo dez eventos, cada evento destacado em uma linha. Na primeira linha temos um evento do tipo retenção de íntron que ocorre na posição 43645 à 44106 do cromossomo 4, envolvendo três transcritos (ENSDART00000159640, ENSDART00000170197 e ENSDART00000175348). Fonte: Autoria própria.

Como mostrado na primeira linha do arquivo da Figura 4.2, o evento encontrado se encontra no cromossomo 4, primeira coluna do arquivo, acontece entre as posições 43645 e 44106 nos transcritos ENSDART00000159640, ENSDART00000170197 e ENSDART00000175348. Os transcritos ENSDART00000170197 e ENSDART00000175348 apresentam as mesmas isoformas, já que estão separados por um "/", mas possuem isoformas distinta do transcrito ENSDART00000159640. O campo de *structure* denota que o tipo de evento encontrado é do tipo *intron retention* pois apresenta o código 0,1ˆ2-.

4.4 EVENTOS ANALISADOS

Existe uma vasta gama de eventos de *splicing* alternativo. Os cinco principais são de *exon skipping* (uso alternativo de exon), *alternative 5 splice sites* (sítios doadores [5']

alternativos), *alternative 3 splice sites* (sítios aceptores [3'] alternativos), *intron retention* (retenção de intron) e *mutually exclusive exons* (exons mutuamente excludentes).

Os eventos analisados por este trabalho estão entre os cindo principais porem o evento de *mutually exclusive exons* não foi abordado, isso se deve ao fato do mesmo evento não ser buscado pelo AStalavista impossibilitando a validação dos eventos encontrados na execução desse trabalho Os seguintes eventos foram então analisados:

1. ***Exon skipping:*** O evento de *exon skipping* consiste em suprimir a expressão de um dos exons, ou seja, quando analisamos duas amostras uma vai possuir um exon em uma posição em que outra amostra não possui, como mostrado na Figura 4.3 (a).

2. ***Intron retention:*** O evento de *exon skipping* consiste em suprimir a expressão de um dos exons, ou seja, quando analisamos duas amostras uma vai possuir um exon em uma posição em que outra amostra não possui, como mostrado na figura 4.3 (b).

3. ***Alternative donnor site:*** O evento de *alternative donnor site* consiste em suprimir a expressão de um dos exons, ou seja, quando analisamos duas amostras uma vai possuir um exon em uma posição em que outra amostra não possui, como mostrado na figura 4.3 (c).

4. ***Alternative acceptor site:*** O evento de *exon skipping* consiste em suprimir a expressão de um dos exons, ou seja, quando analisamos duas amostras uma vai possuir um exon em uma posição em que outra amostra não possui, como mostrado na figura 4.3 (d).

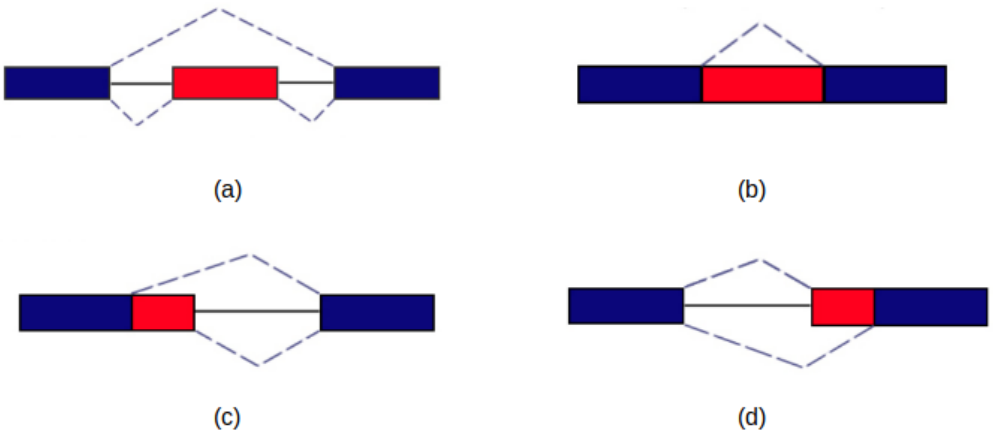


Figura 4.3: Ilustração dos quatro eventos analisados, *exon skipping* em (a) mostrando os dois caminhos possíveis um contendo o exon em vermelho do transcrito em destaque e um portando apenas os exons em azul, *intron retention* em (b) processo onde o íntron em vermelho pode ou no ser retido no transcrito,*alternative donnor site* em (c) e *alternative acceptor site* em (d). Fonte: Modificado de [Ast, 2004].

4.5 LINGUAGEM DE PROGRAMAÇÃO E PARALELIZAÇÃO

A linguagem de programação escolhida para o desenvolvimento deste trabalho foi a linguagem de programação C. Esta escolha foi tomada por ser uma linguagem de alto desempenho com bibliotecas que permitem a paralelização do código.

Uma primeira versão do trabalho foi desenvolvida utilizando a linguagem Python, sem nenhum tipo de paralelização.

A versão do trabalho desenvolvida em C foi paralelizada a técnica de paralelização baseada em memória compartilhada, utilizando a biblioteca OpenMP. Uma primeira versão do programa foi desenvolvida de modo sequencial. A partir dessa versão foi realizada a paralelização do código utilizando duas abordagens:

1. **Leitura do arquivo GTF:** A tarefa que mais consome tempo computacional é a leitura do arquivo e armazenamento em uma estrutura de dados. A estrutura desenvolvida nesse trabalho foi pensada de modo a otimizar o processo de descoberta de eventos.

   Para paralelizar a leitura do arquivo, foi realizada a divisão por blocos de linhas do arquivo. Cada *thread* fica responsável por processar e armazenar essas linhas do seu bloco.

   A divisão dos blocos é realizada de acordo com a quantidade de genes e quantidade de *threads* utilizadas. Desta forma, o primeiro passo consiste em dividir o arquivo em blocos de genes, iniciando na posição de início do gene e terminando na última linha correspondente aquele gene.

   A quantidade de blocos de genes atribuídos a cada *thread* depende da quantidade de *threads* utilizadas. Por exemplo, em um arquivo com 32500 genes cada *thread* fica responsável por 8125 genes, ou seja, responsável por processar todas as linhas correspondentes a esses 8125 genes.

2. **Paralelização da descoberta de eventos:** Os eventos são encontrados comparando, dois a dois, os transcritos do mesmo gene. Apesar da comparação ser de dois a dois, os transcritos com eventos de *splicing* semelhantes são agrupados para gerar apenas uma linha no evento de saída.

   Além disso, repetições de eventos, como por exemplo, evento de *exon skipping* encontrado na posição 123 à 432 do transcrito1 comparado ao transcrito2 e evento de *exon skipping* encontrado na posição 123 à 432 do transcrito2 comparado ao transcrito1, devem ser removidos.

   Como a descoberta de eventos é uma tarefa independente de gene para gene a paralelização dessa etapa foi realizada atribuindo, para cada núcleo paralelamente, a descoberta dos eventos referentes a um gene.

5 VALIDAÇÃO E TESTES

5.1 ORGANISMOS MODELO

Para realizar os testes deste trabalho foram utilizados três organismos. Os três foram obtidos na base de dados pública Ensembl:

1. **Zebrafish (Danio rerio)** [1]**:** Arquivo de anotação gênica com 446,8 MB.
2. **Mouse (Mus musculus)** [2]**:** Arquivo de anotação gênica com 805,5 MB
3. ***Human* (Homo sapiens)** [3]**:** Arquivo de anotação gênica com 1,2 GB.

Dentro da base de dados ensembl, foram selecionados os organismos amplamente usados na literatura e com diversos estudos publicados e catalogados. Os três organismos selecionados apresentam um acréscimo de complexidade e também de tamanho em seus arquivos.

5.2 ARQUITETURA

Os testes descritos neste trabalho foram realizados em uma servidor do departamento de informática da UFPR. A máquina usada possui 40 núcleos físicos . Os testes de *speedup* foram realizados com um intervalo de threads que vai de 1 à 70.

5.3 ASTAVAVISTA E VERSÕES SEQUENCIAIS

Nesta sessão são trazidos os tempos médios encontrados na execução do programa AStalavista para os três organismos selecionados. Os tempos encontrados na versão sequencial do código em C e versão sequencial do código em python.

Os tempos foram calculados com base na média de 100 execuções. Podemos notar, como mostrado na Figura 5.1, que o tempo de execução do programa desenvolvido em C quando comparado ao programa AStalavista é reduzido. Como o código fonte do AStalavista não está disponível não podemos afirmar quais mudanças no código geraram tal melhora, porém uma parte da melhora acredita-se que seja associada a linguagem de programação utilizada, uma vez que o programa AStalavista utiliza a linguagem Java.

[1]Disponível em: https://www.ensembl.org/Danio$_{r}erio/Info/Index$
[2]Disponível em: https://www.ensembl.org/Mus$_{m}usculus/Info/Index$
[3]Disponível em: https://www.ensembl.org/Homo$_{s}apiens/Info/Index$

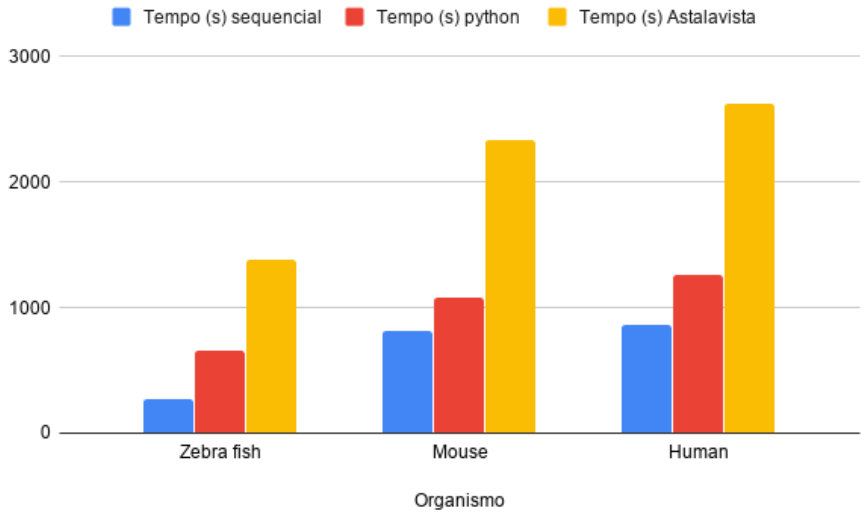


Figura 5.1: Tempo de execução, em segundos, do trabalho AStalavista comparado com a versão sequencial desenvolvida em linguagem C e em python. Em amarelo o tempo de execução do programa AStalavista, para os três organismos selecionados, em vermelho o tempo de execução do algoritmo desenvolvido em python e em azul o tempo de execução do algoritmo desenvolvido na linguagem C.

Na Figura 5.1 podemos notar que o código desenvolvido em C apresenta o melhor desempenho quanto comparado ao trabalho base e ao código inicial desenvolvido em python.

Com relação a memória utilizada por casa instância do problema, a Figura 5.2 mostra qual foi a memória total alocada para cada organismos além da comparação com o tamanho da sequência.

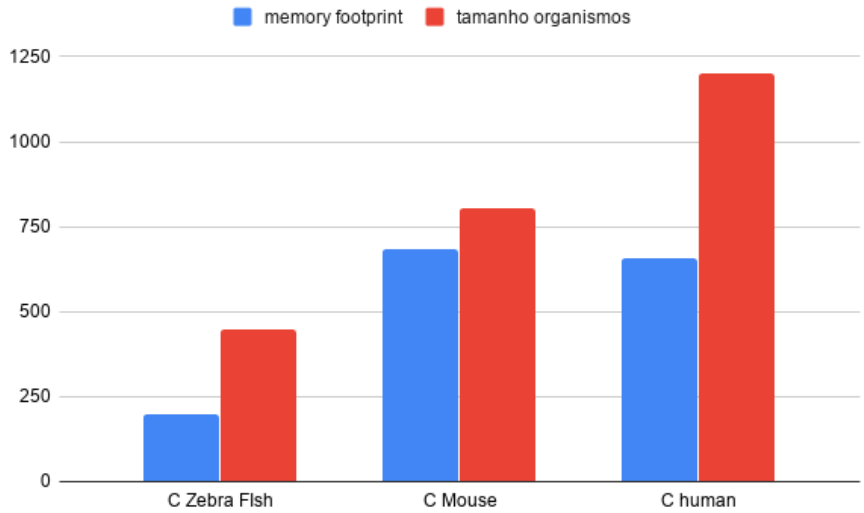


Figura 5.2: Memória usada em MB durante a execução do programa comparada ao tamanho das arquivos de entrada em MB. Nas duas primeiras colunas a esquerda, em vermelho o tamanho do *Zebra Fish* e em azul a quantidade de memória utilizada. Nas duas colunas do meio, em vermelho o tamanho do arquivo de entrada para o *mouse* em MB e em azul a quantidade de memória utilizada e as colunas a direita, em vermelho o tamanho do arquivo de entrada do *human* e em azul a quantidade de memória utilizada.

5.4 TESTE DE ESCALABILIDADE FORTE

A escalabilidade forte é caracterizada como a manutenção da eficiência conforte o número de núcleos usados aumenta, contudo sem haver um aumento do tamanho do problema, ou seja, sem aumento nos dados de entrada.

Foram realizados testes nos três organismos selecionados, variando o número de threads, de 1 a 70. Cada execução foi repetida 100×, para geração do tempo médio usado no cálculo do *speedup*, juntamente com o tempo médio calculado para a execução sequencial para cada organismo.

Como é mostrado na Figura 5.3 o problema não apresenta escalabilidade forte pois não possui uma melhora no tempo de execução proporcional ao aumento no número de núcleos usados, como observado na vigura 5.3

O teste de escalabilidade fraca não foi realizado pois como os dados de entrada são dados biológicos não haveria como simular dados N vezes maiores dos que os usados para os testes.

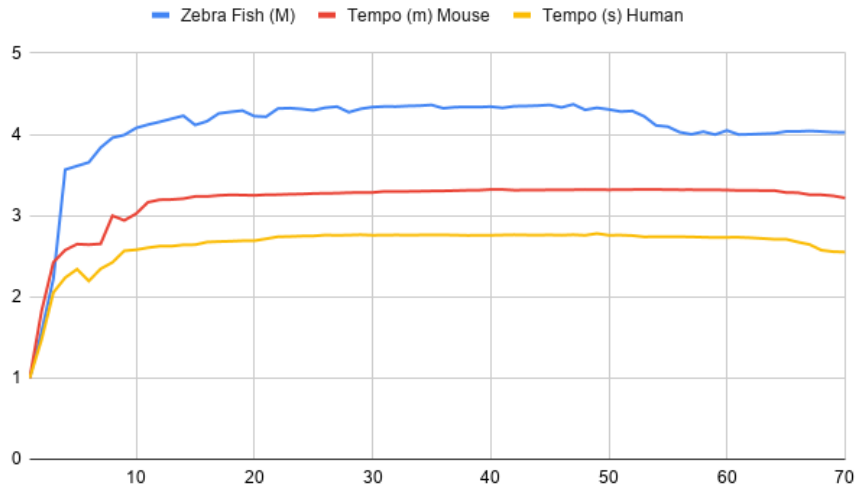


Figura 5.3: *Speedup* observado em diferentes números de *threads* nos três organismos selecionados. Em azul o valor de *speedup* ao longo da variação de *threads* para o *Zebra fish*, em vermelho a variação para o *mouse* e em amarelo a variação para o *human*.

5.5 DISCUSSÃO

A melhora notada no tempo de execução entre o trabalho base e o trabalho desenvolvido, atribui-se a linguagem utilizada enquanto o trabalho base utiliza a linguagem de programação Java, o trabalho desenvolvido utiliza a linguagem C. Enquanto a linguagem Java é uma linguagem de programação interpretada a linguagem C é uma linguagem compilada.

Quanto a melhora adquirida com a paralelização, para o número de *threads* usadas os resultados mostram um crescimento do *speedup*. Ocorre uma estagnação dos ganhos após 15 *threads* e um leve declínio ao fim, vale ressaltar que a máquina usada nos testes possuí 40 núcleos físicos. Além disso, notamos também que o *speedup* máximo atingido, próximo a 4,5 é muito menor que o esperado para 15 *threads*, com uma eficiência nesse caso de apenas 30%. Ou seja, estamos utilizando menos da metade do potencial teórico dos núcleos de processamento.

5.6 CONCLUSÃO

Podemos notar que comparado a implementação base houve uma grande melhora na eficiência da resolução do problema apesar disso a eficiência atingida foi de apenas 30%.

Além disso, o uso de memória para a resolução do programa se mostrou satisfatório para o tamanho dos organismos utilizados.

6 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho visou a construção de uma aplicação capaz realizar a leitura de um arquivo de dado transcriptômico, independente da espécie, e quantificação dos eventos de: *exon skipping*, *intron retention*, *alternative donnor site* e *alternative acceptor site*. Para isso foram aplicadas técnicas de paralelização de código baseada em memória compartilhada usando a biblioteca OpenMP. Tendo como resultado uma aplicação capaz de quantificar dados transcriptômicos em tempo viável, sendo executada em um computador pessoal.

Podemos observar pelos resultados que houve uma significativa melhora no desempenho da aplicação desenvolvida, apesar de não apresentar uma melhora proporcional ao aumento de threads utilizadas.

Durante o desenvolvimento deste trabalho, uma proposta foi escrita para a NVidia, em conjunto com o professor Eduardo Cunha de Almeida, com a intenção de concorrer a doação de GPUs, a proposta foi selecionada e aceita. O resultado deste trabalho foi apresentado na modalidade pôster no I Workshop de Bioinformática. Além disso, também foram apresentados outros projetos no 15º Congresso Internacional da AB3C, X-Metting, em modalidade oral e pôster.

Dentre os trabalhos futuros que podem surgir a partir deste, sugerimos a ampliação da gama de arquivos aceitos como entrada, além do arquivo GTF, usado neste trabalho existem outros tipos de arquivos de dados transcriptômicos, como por exemplo, o arquivo FASTA, que armazena sequências de nucleotídeos.

Além disso, é possível propor a criação de um arquivo de entrada que seja otimizado para a leitura, reduzindo a necessidade de múltiplas leituras para o armazenamento dos dados.

REFERÊNCIAS

[Ast, 2004] Ast, G. (2004). How did alternative splicing evolve? *Nature Reviews Genetics*, 5(10):773.

[Conn e Stumpf, 1980] Conn, E. E. e Stumpf, P. K. (1980). *Introdução à bioquímica*. Editora Blucher.

[Crick, 1958] Crick, F. H. (1958). On protein synthesis. Em *Symp Soc Exp Biol*, volume 12, página 8.

[Denti et al., 2018] Denti, L., Rizzi, R., Beretta, S., Della Vedova, G., Previtali, M. e Bonizzoni, P. (2018). Asgal: aligning rna-seq data to a splicing graph to detect novel alternative splicing events. *BMC bioinformatics*, 19(1):444.

[Foissac e Sammeth, 2007] Foissac, S. e Sammeth, M. (2007). Astalavista: dynamic and flexible analysis of alternative splicing events in custom gene datasets. *Nucleic acids research*, 35(suppl_2):W297–W299.

[Foster, 1999] Foster, I. (1999). *Designing and Building Parallel Programs*. MIT Press.

[Goldstrohm et al., 2001] Goldstrohm, A. C., Greenleaf, A. L. e Garcia-Blanco, M. A. (2001). Co-transcriptional splicing of pre-messenger rnas: considerations for the mechanism of alternative splicing. *Gene*, 277(1-2):31–47.

[Hooper, 2014] Hooper, J. E. (2014). A survey of software for genome-wide discovery of differential splicing in rna-seq data. *Human genomics*, 8(1):3.

[Karp, 2011] Karp, G. (2011). *Biologia celular y molecular: conceptos y experimentos (6a*. McGraw Hill Mexico.

[Katz et al., 2010] Katz, Y., Wang, E. T., Airoldi, E. M. e Burge, C. B. (2010). Analysis and design of rna sequencing experiments for identifying isoform regulation. *Nature methods*, 7(12):1009.

[Lana, 2018] Lana, H. C. (2018). *Projetos Maker: Arduino Eletrônica Robótica Automação residencial*. Novatec Editora.

[Lander, 2001] Lander, E. S. (2001). Initial sequencing and analysis of the human genome. international human genome sequencing consortium. *Nature*, 409:860–921.

[Lareau et al., 2004] Lareau, L. F., Green, R. E., Bhatnagar, R. S. e Brenner, S. E. (2004). The evolving roles of alternative splicing. *Current opinion in structural biology*, 14(3):273–282.

[Li et al., 2018] Li, J., Guan, J., Qian, J., Feng, Y., Yao, R., Fan, R. e Wang, Z. (2018). Paean: A parallel transcriptome quantification tool combining gene expression and alternative splicing events using gpu. Em *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, páginas 676–679. IEEE.

[Modrek e Lee, 2002] Modrek, B. e Lee, C. (2002). A genomic view of alternative splicing. *Nature genetics*, 30(1):13.

[Murray et al., 2010] Murray, R. K., Bender, D. A. e Botham, K. M. (2010). *Harper: bioquímica ilustrada.* McGraw-Hill.

[Nelson e Cox, 2018] Nelson, D. L. e Cox, M. M. (2018). *Princípios de Bioquímica de Lehninger-7.* Artmed Editora.

[Nelson, 2014] Nelson, David L.Cox, M. (2014). *Princípios de Bioquímica de Lehninger.*

[Rodwell, 2016] Rodwell, V. W. (2016). Nucleotídeos. *Bioquímica Ilustrada de Harper*, página 339.

[Rowen et al., 2002] Rowen, L., Young, J., Birditt, B., Kaur, A., Madan, A., Philipps, D. L., Qin, S., Minx, P., Wilson, R. K., Hood, L. et al. (2002). Analysis of the human neurexin genes: alternative splicing and the generation of protein diversity. *Genomics*, 79(4):587–597.

[STRINGHINI, 2018] STRINGHINI, Denise; C ACERES, E. N. (2018). *Minicursos do WSCAD 2018.* Sociedade Brasileira de Computação.

[Watson et al., 1953] Watson, J. D., Crick, F. H. et al. (1953). Molecular structure of nucleic acids. *Nature*, 171(4356):737–738.