CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
FACULDADE DE TECNOLOGIA DE BOTUCATU
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

TÚLIO MOREIRA FERNANDES

OTIMIZAÇÃO DE UM ALGORITMO DE REORDENAMENTO DE REDES DE INTERAÇÕES PROTEICAS PELA IMPLEMENTAÇÃO DE UM DICIONÁRIO DE DISPERSÃO

Botucatu-SP
Junho – 2018

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
FACULDADE DE TECNOLOGIA DE BOTUCATU
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

TÚLIO MOREIRA FERNANDES

OTIMIZAÇÃO DE UM ALGORITMO REORDENAMENTO DE REDES DE INTERAÇÕES PROTEICAS PELA IMPLEMENTAÇÃO DE UM DICIONÁRIO DE DISPERSÃO

Orientador: Prof. Dr. OSVALDO CÉSAR PINHEIRO DE ALMEIDA

Artigo apresentado à FATEC - Faculdade de Tecnologia de Botucatu, para obtenção do título de Tecnólogo no Curso Superior de Análise e Desenvolvimento de Sistemas.

Botucatu-SP
Junho – 2018

OTIMIZAÇÃO DE UM ALGORITMO DE REORDENAMENTO DE REDES DE INTERAÇÕES PROTEICAS PELA IMPLEMENTAÇÃO DE UM DICIONÁRIO DE DISPERSÃO

OPTIMIZATION OF AN ALGORITHM TO REORDER PROTEIN INTERACTIONS NETWORKS BY IMPLEMENTING A HASH DICTIONARY

Túlio Moreira Fernandes[1], Osvaldo César Pinheiro de Almeida[2]

1 Graduando em Tecnologia em Análise e Desenvolvimento de Sistemas, Faculdade de Tecnologia de Botucatu.

2 Docente da Faculdade de Tecnologia de Botucatu.

RESUMO

Tabelas de dispersão - ou tabelas *hash* - são amplamente utilizadas como estruturas de dicionário, principalmente por seus métodos rápidos de busca e inserção, que associam palavras umas às outras. O Transcriptograma é uma técnica que permite a análise de expressão gênica na escala de um genoma completo. É possível obter o estado metabólico de uma célula ou tecido reordenando sua rede de interação proteína-proteína (PPI). Este trabalho objetiva melhorar o desempenho de um algoritmo de reordenamento de redes de interações proteicas. Um novo dicionário, de dispersão, foi implementado a fim de acelerar o processo de reordenamento de PPIs, associando e armazenando os nomes das proteínas em uma tabela *hash* mais rapidamente do que a antiga e iterativa lista encadeada. Os resultados são favoráveis à nova estrutura, sendo esta mais eficiente do que a lista encadeada em matéria de tempo dispendido nos reordenamentos e equivalente à estrutura antiga quanto ao consumo de RAM. Tal melhoria se deve especialmente ao *hashing* do dicionário de dispersão.

Palavras-chave: Bioinformática. Espalhamento. Estrutura. Proteína. Transcriptograma.

ABSTRACT

Hash tables are widely used as dictionary structures, mainly by its fast searching and inserting methods that associates words with each others. Transcriptogram is a method that allows the gene expression analysis in whole genome scale. It is possible to obtain the metabolic state of a cell or tissue by reordering its protein-protein interaction network (PPI). This work aims to improve the performance of an algorithm to reorder protein interactions networks. A new hash dictionary structure was implemented to speed up the PPI reordering process, associating and storing the names of the proteins in a hash table more quickly than the old and iterative linked list. The results are favorable to the new structure, being more efficient than the old one in time spent and equivalent in RAM consumption. Such improvement is especially due to dictionary hashing.

Keywords*: Bioinformatics. Hashing. Structure. Protein. Transcriptogram.*

1. INTRODUÇÃO

De acordo com Tenenbaum (2004), compreende-se por tabela de dispersão ou *hash table* uma estrutura de dados estática - isto é, de dimensão predefinida em código - não-sequencial, cujas vantagens, no melhor caso, são a rápida inserção e busca de dados em um vetor, não tendo de percorrê-lo iterativamente para comparar cada valor ao que se busca, nem para encontrar um índice vazio para o armazenamento deste, o que ocorre com estruturas sequenciais – como, por exemplo, uma lista encadeada. Além disso, é destacado o alto nível de aleatorização alcançado no espalhamento dos dados no vetor.

Na técnica de dispersão, a determinação do índice do vetor em que um item – ao qual referiremos como chave - será armazenado se dá de maneira associativa ao próprio valor deste item, através de uma função matemática de espalhamento comumente denominada *hash*. Considerando o melhor caso, onde não ocorra uma colisão – isto é, uma chave tentando ocupar um índice já preenchido -, temos, portanto, que o custo computacional de uma rotina – busca ou inserção – em uma tabela de dispersão é sempre O(1), ou seja, uma operação, já que a associação do valor matemático da chave ao índice de seu armazenamento no vetor dispensa buscas iterativas (MADEIRA, 2010).

Tabelas de dispersão têm aplicações em diversas instâncias da computação que demandam aleatorização e/ou rapidez. Duas notáveis são: DHT - *Distributed Hash Table* – em redes de compartilhamento *peer-to-peer*, a qual visa aleatorizar a distribuição de *trackers* para os nodos da rede (CÉSAR e SOUZA, 2013); *Web caches* compartilhados, também conhecidos como servidores *proxy*, que, além de intermediar o acesso dos clientes aos servidores *web*, utilizam dispersão para armazenar URLs de maneira associativa a seus referentes objetos em disco, visando acelerar o acesso destes pelos clientes, em detrimento dos iterativos ICPs (ROSS, 1997).

Ainda conforme Tenembaum (2004), uma função de dispersão *h* que converte uma chave em seu índice no vetor – ou tabela – pode ser convencionalmente descrita como *h(key) = key % 1000*, considerando um vetor cujas posições vão de 0 a 999. Assim sendo, uma chave de valor 2018 seria armazenada no índice 18, valor correspondente ao resto da divisão do número total de posições do vetor pelo valor da chave. Estruturas de dicionários também são implementadas com tabelas *hash*, ao associar uma palavra (chave) à outra (valor) mediante o valor numérico (*ASCII*) dos caracteres das chaves, estas sendo, agora, *strings*. O código de uma função de espalhamento pode ser elaborado de acordo com a necessidade. Em tese, quanto maior for a dispersão resultante na tabela, menor será a frequência de colisões. As

colisões podem ser tratadas de diversas formas, exceto por espalhamento. Para este trabalho foi escolhido o método de encadeamento aberto, também conhecido por *hashing* fechado.

De acordo com Biazotti (2017), em bioinformática, o reordenamento de uma rede de interação proteica - previamente digitalizada em um arquivo de texto e derivada do genoma, ou seja, do arranjo informacional sequenciado de todos os genes do organismo a ser estudado - é um processo preliminar imprescindível para o prosseguimento da análise dos níveis de expressão gênica de uma célula ou tecido por um pesquisador. Computacionalmente o reordenamento se dá, primeiramente, na leitura do arquivo da rede. Após a leitura, o *software* – Transcriptograma - agrupa as várias interações (linhas) de duas colunas (nomes de proteínas) do arquivo em uma matriz booleana na memória principal. Ocorre, então, o reordenamento dessas interações na rede, que, originalmente dispersas, após este processo estão mais juntas e próximas da diagonal principal da matriz.

Segundo Rybarczyk Filho (2011), tal aglutinação das interações proteicas na diagonal principal permite a formação de picos de interações gênicas, configurando uma condição mais viável para a seleção destas e a condução da análise pelo pesquisador. Após a selecionar os picos interações, ocorre o cruzamento destas com as informações de expressão gênica de um mesmo organismo; estas, por sua vez, obtidas de um banco de dados na internet. Desse cruzamento o pesquisador obtém dados sobre quais genes estão sub ou super expressos e, consequentemente, quais módulos funcionais estão mais ou menos ativos no organismo, podendo indicar, por exemplo, cânceres e outras doenças.

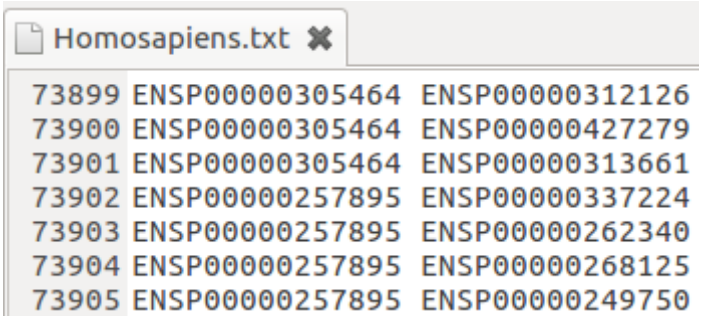
Uma estrutura de dicionário auxilia o algoritmo de reordenamento quanto às associações dos nomes das proteínas (Biazotti, 2017), e o desempenho desta pode impactar diretamente na performance do algoritmo. Este trabalho objetiva melhorar o desempenho de um algoritmo de reordenamento de redes de interação proteica através da implementação de uma nova estrutura de dicionário, sendo esta uma tabela de dispersão com *hashing* aberto – ou encadeado – para o tratamento de colisões, em substituição à singular lista encadeada vigente.

2. MATERIAL E MÉTODOS

2.1. Redes de interação proteína-proteína

Ao abrir o arquivo *Homosapiens.txt* em um editor de texto, podemos ver explicitamente as interações de uma proteína (primeira coluna) com outras (segunda coluna), do organismo *Homo sapiens,* como observável na Figura 1.

Figura1. Ao selecionar da linha 73899 à linha 73905 temos, como exemplo, que a proteína ENSP00000305464 interage com pelo menos outras três proteínas. Da mesma forma, a proteína ENSP00000257895 interage com pelo menos outras quatro (STRING, 2018).



De acordo com Kuentzer (2011), em uma rede de interação proteica os nodos estão associados às proteínas, e, suas arestas, às interações destas umas com as outras. O número de nodos é proporcional à complexidade da rede.

Os arquivos das redes de interações proteicas foram obtidos do banco de dados STRING, cujo catálogo possui, atualmente, 9.643.763 proteínas, de um total de 2.031 organismos.

2.2. Algoritmo de reordenamento

Segundo Biazotti (2017), na existência de interação de uma proteína com ao menos uma outra, o Transcriptograma transcreve tal condição como verdadeira (V) em uma matriz booleana; caso contrário, como falsa (F), como se observa na Figura 2. Em função da eficiência do método, uma aleatorização dos vértices da matriz precede as permutações de reordenamento.

Ainda segundo Biazotti (2017), após a transcrição de todas as interações da rede em uma matriz booleana, a melhor configuração de reordenamento precedente à análise de expressão gênica requer duas condições: todas as interações terem o menor número possível de elementos vizinhos como sendo proteínas que não interagem; o elemento central estar o mais próximo possível da diagonal principal da matriz. O algoritmo designa o elemento verdadeiro central e inicia uma análise iterativa de sua vizinhança. Com base nas duas condições descritas em cálculos de proximidade e probabilidade, é decidido se determinada interação deve ser ou não permutada e com qual. Assim, o algoritmo permuta os elementos em busca de uma grande aglutinação à diagonal principal.

Figura 2: Matriz booleana de interações proteicas reordenada, da pior para a melhor configuração. O algoritmo analisa iterativamente a vizinhança do elemento central (com flechas brancas). Cada elemento em preto representa uma interação (1); os brancos representam proteínas que não interagem (0) (Biazotti, 2017).

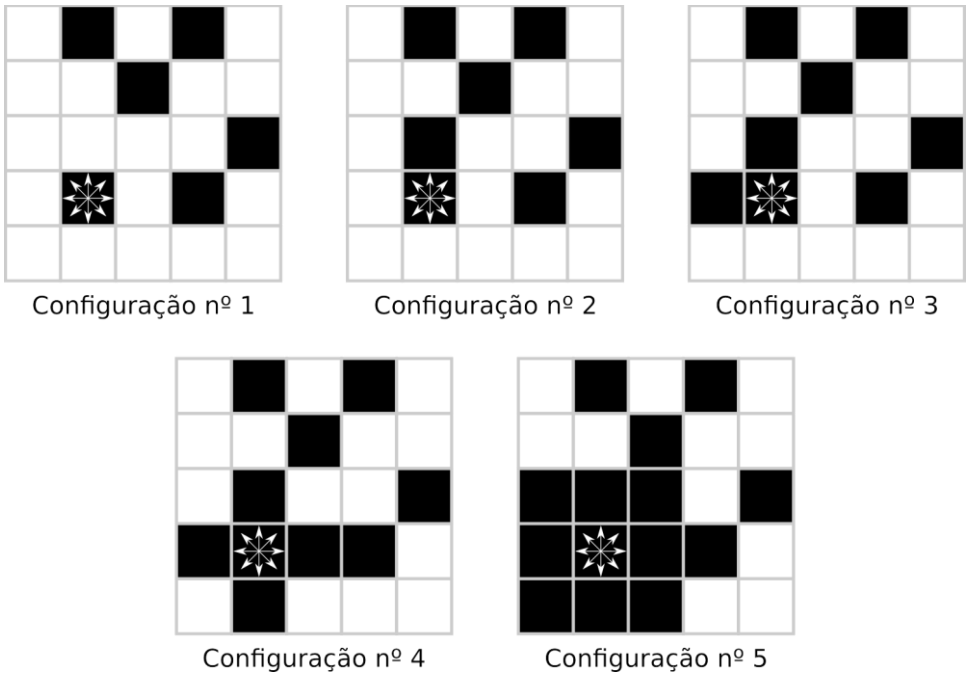


Tabela 1. As quatro redes a serem reordenadas para a produção deste trabalho.

Arquivo	Organismo	Tamanho	N° de linhas
aedes_07.txt	*Aedes aegypti*	188,8 KB	4.840
anoph_07.txt	*Anopheles*	264,1 KB	6.772
Homosapiens.txt	*Homo sapiens*	14,8 MB	463.206
rattus_norvegicus_07.txt	*Rattus norvegicus*	23,6 MB	621.996

2.3. Ferramentas de desenvolvimento

Como ferramenta de desenvolvimento integrado (*IDE*) foi utilizado o *RStudio*, tanto por seu editor suportar a sintaxe e indentação da linguagem utilizada (C) como para gerar gráficos estéticos para análise comparativa. Para a compilação foi utilizado o *GNU C Compiler* (*GCC*).

2.4. Ferramentas e métodos estatísticos

Para mensurar o tempo e uso de memória de cada execução foi utilizado o comando */usr/bin/time* – não confundir com o comando *time* embutido no *bash*. Para a elaboração dos gráficos estéticos foi utilizada a linguagem R, juntamente à biblioteca de plotagem *ggplot2*.

Após uma bateria de dez execuções de reordenamento das redes de interação proteína-proteína dos organismos *Aedes aegypti*, *Anopheles, Homo sapiens* e *Rattus norvegicus* nas duas estruturas de dicionário comparadas neste trabalho, obtemos dois conjuntos de dados: a variação das grandezas de valores de tempo e consumo de *RAM* de cada execução, exibida em diagramas de caixa – *boxplot*; os valores médios das mesmas grandezas em cada das duas estruturas, cujas diferenças serão analisadas percentualmente.

2.5. Dicionário de dispersão

Uma nova estrutura de dados de dicionário foi incorporada ao algoritmo de reordenamento, substituindo a antiga lista encadeada. Seu mecanismo de busca, inserção e remoção de valores se dá através de uma função de espalhamento (*hash*), escrita em linguagem C, cujos parâmetros incluem, além da própria chave (*string*), o tamanho da cadeia de caracteres desta em número inteiro.

Figura 3. O resultado retornado pela função – índice do vetor onde o nome (chave) será armazenado – é gerado por operações matemáticas a partir aos valores *ASCII* de seus caracteres.

```c
static unsigned int hash(char *chave, int tamanho_chave) {
    unsigned int hash = 0;
    int indice;

    for (indice = 0; indice < tamanho_chave; indice++) {
        unsigned char c = chave[indice];
        hash += c;
        hash += (hash << 10);
        hash ^= (hash >> 6);
    }
    hash += (hash << 3);
    hash ^= (hash >> 11);
    hash += (hash << 15);

    return hash;
}
```

3. RESULTADOS E DISCUSSÃO

Gráfico 1. *Aedes aegypti*: tempo médio de 2,05 e 1,98 segundos em lista encadeada e dicionário de dispersão, respectivamente. Melhora de 3,41%. Consumo médio de RAM de 2.500 KB e 2.474 KB. Redução de 1,04%.

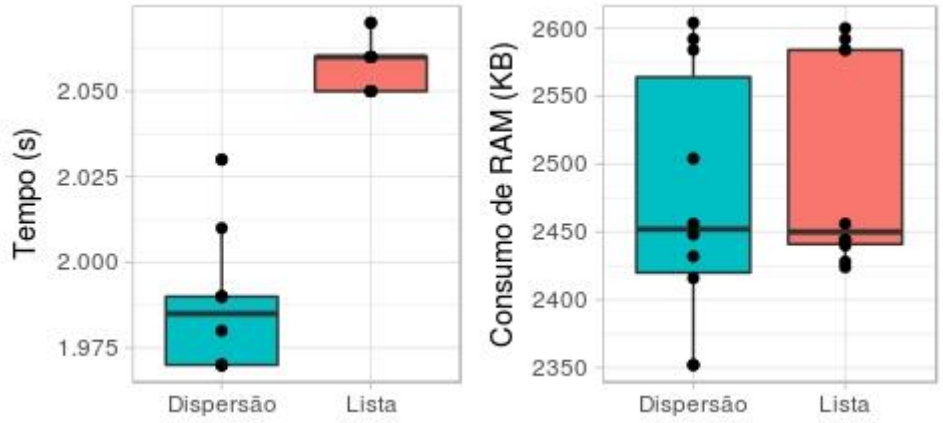


Gráfico 2. *Anopheles*: tempo médio de 2,84 e 2,73 segundos em lista encadeada e dicionário de dispersão, respectivamente. Melhora de 3,87%. Consumo médio de RAM de 2.638 KB e 2.604 KB. Aumento de 1,29%.

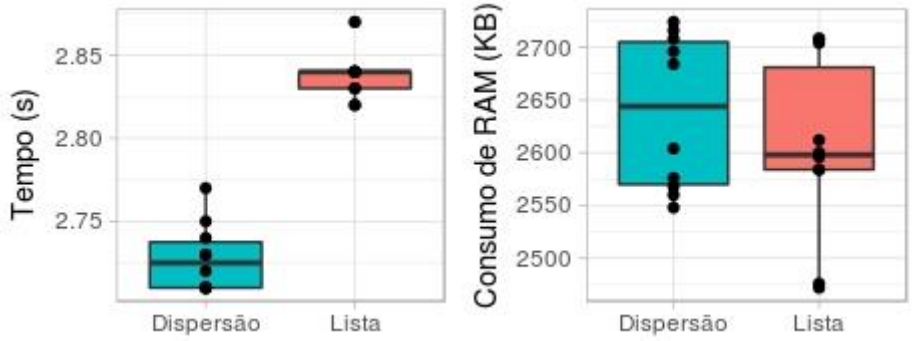


Gráfico 3. *Homo sapiens*: tempo médio de 624 e 457 segundos em lista encadeada e dicionário de dispersão, respectivamente. Melhora de 26,76%. Consumo médio de RAM de 131.903 KB e 131.722 KB. Redução de 0,14%.

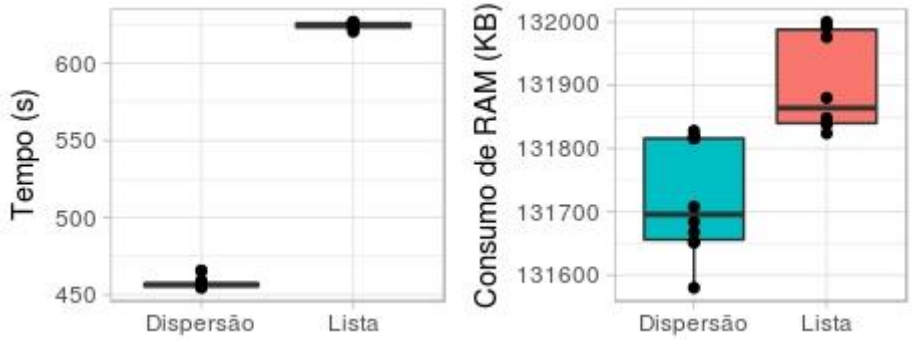
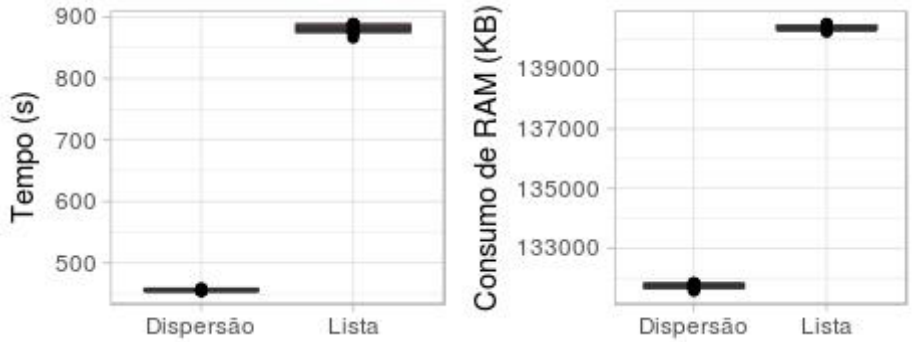


Gráfico 4. *Rattus norvegicus*: tempo médio de 881 e 456 segundos em lista encadeada e dicionário de dispersão, respectivamente. Melhora de 48,24%. Consumo médio de RAM de 140.382 KB e 131.738 KB. Redução de 6,16%.



4. CONCLUSÃO

Podemos concluir que, após a incorporação do dicionário de dispersão ao algoritmo de reordenamento, os resultados mostraram-se favoráveis à nova estrutura. Em comparação à antiga lista encadeada, o dicionário de dispersão reduziu o tempo médio despendido em todos os testes, sendo esta uma diferença aumentada proporcionalmente à complexidade (nodos) e tamanho em disco das redes processadas. Ademais, tal redução nos tempos de reordenamento se deu mantendo um consumo muito equivalente de RAM nas duas estruturas, variando minimamente a uma taxa de 2,16% para mais ou para menos. Conclui-se que o aumento da performance com a nova estrutura de dicionário se deve muito especialmente ao *hashing*.

5. REFERÊNCIAS

BIAZOTTI, A. A. **Desenvolvimento de Ferramenta Computacional para intergração de transcriptomas e redes biológicas: medidas de desempenho global.** Dissertação (Mestrado em Biotecnologia) apresentada à Universidade Estadual Paulista de Botucatu/ SP. 2017. Disponível em: < https://repositorio.unesp.br/handle/11449/152771>. Acesso em: 22 mai. 2018.

CÉSAR, C. D. A. C.; SOUZA, C. A. Um sistema de cache com tabelas hash distribuídas para aplicações peer-to-peer. In: 31°. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 115., 2013, São José dos Campos. **Anais...** São José dos Campos, SP: ITA, 2013. p. 116.

Create Elegant Data Visualisations Using the Grammar of Graphics • ggplot2. Disponível em: <http://ggplot2.tidyverse.org/>. Acesso em: 29 mai. 2018.

GCC, the GNU Compiler Collection - GNU Project - Free Software Foundation (FSF). Disponível em: <https://gcc.gnu.org/>. Acesso em: 29 mai. 2018.

GDB: The GNU Project Debugger. Disponível em: <https://gcc.gnu.org/>. Acesso em: 29 mai. 2018.

KUENTZER, F. A. et al. **Otimização e análise de algoritmos de ordenamento de redes proteicas**. 2014. Dissertação (Mestrado em Ciência da Computação) apresentada à Pontifícia Universidade Católica do Rio Grande do Sul/ RS. 2014. Disponível em: <http://repositorio.pucrs.br/dspace/handle/10923/6663>. Acesso em: 23 mai. 2018.

MADEIRA, D. **Uma estrutura baseada em hash table para buscas otimizadas em octree em gpu.** Tese (Doutorado em Computação) apresentada à Universidade Federal Fluminense/ RJ. 2010. Disponível em: <http://www2.ic.uff.br/PosGraduacao/Dissertacoes/458.pdf>. Acesso em: 25 mai. 2018.

FILHO, J. L. R. **Medidas de performance metabólica usando a expressão gênica de genoma completo.** Tese (Doutorado em Física) apresentada à Universidade Federal do Rio Grande do Sul/ RS. 2011. Disponível em: <https://lume.ufrgs.br/handle/10183/31005>. Acesso em: 22 mai. 2018.

ROSS, K. W. **Hash routing for collections of shared web caches.** Ieee Network, v. 11, n. 6, p. 37-44, 1997.

R: What is R? Disponível em: <https://www.r-project.org/about.html>. Acesso em: 29 mai. 2018.

RSTUDIO. **About – RStudio.** Disponível em: <https://www.rstudio.com/about/>. Acesso em: 29 mai. 2018.

SILVA, S. R. M. **A eficiência do transcriptograma.** Dissertação (Mestrado em Física) apresentada à Universidade Federal do Rio Grande do Sul/ RS. 2013. Disponível em: <https://lume.ufrgs.br/handle/10183/87225>. Acesso em: 22 mai. 2018.

STRING: functional protein association networks. Disponível em: <https://string-db.org/cgi/about.pl>. Acesso em: 30 mai. 2018.

TENENBAUM, A. M.; LANGSAM, Y; AUGENSTEIN, M. J. **Estruturas de dados usando C**. Pearson Makron Books, 2004. p. 595-598.

time(1) - Linux manual page. Disponível em: <http://man7.org/linux/man-pages/man1/time.1.html>. Acesso em: 30 mai. 2018.