Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Trabalho de Conclusão de Curso

Implementação do método WGCNA em GPU

Leonardo Costa Santos
Orientador: Prof. Ronaldo Hashimoto

São Paulo
2024

Resumo

Weighted Gene Correlation Network Analysis é uma importante técnica de análise de dados de expressão gênica, como dados de *microarray* ou RNA-seq. Ela pode ser utilizada para estudar as relações entre diferentes conjuntos de genes, encontrar genes correlacionados a sintomas ou outras características clínicas de um organismo, e para calcular biomarcadores e genes alvo para tratamentos.

Neste trabalho, o WGCNA foi implementado em GPU através da plataforma CUDA, com o objetivo de obter maior desempenho que as implementações em CPU disponíveis atualmente.

Abstract

Abstract *Weighted Gene Correlation Network Analysis* is an important technique for the analysis of gene expression data, such as microarray or RNA-seq data. It can be used to study the relations between different sets of genes, finding genes that are correlatated to symptoms or other clinical traits of an organism, as well as calculate biomarkers and target genes for treatments.

In this work, we implemented WGCNA on GPUs through the use of the CUDA platform, with the goal of achieving higher performance than the available CPU implementations.

Conteúdo

Capítulo 1

Introdução

As redes de coexpressão gênicas correspondem a uma estratégia sistêmica para a análise da expressão gênica. As redes são construídas a partir de dados de expressão gênica de diferentes amostras biológicas, como tecidos, células ou fluidos corporais. As redes mostram quais genes são expressos de forma semelhante ou diferente em cada condição experimental ou clínica. Assim, é possível identificar grupos de genes que estão co-regulados ou que têm funções biológicas comuns. Esses grupos podem representar vias metabólicas, processos celulares ou doenças.

Um dos algoritmos mais usados para criar redes de coexpressão gênica é o WGCNA (Weighted Gene Co-expression Network Analysis) [7].

Originalmente, o WGCNA foi desenvolvido em R [7] e disponibilizado para a comunidade científica. Atualmente, é possível também encontrar implementações em Python [11]. No entanto, implementações para CPU podem ser demoradas para grandes conjuntos de dados. Dessa forma, foi encontrada na literatura uma ferramenta em R que utiliza a natureza paralela das arquiteturas GPU (Unidade de Processamento Gráfico) como implementação do método WGCNA [9].

O objetivo deste trabalho é criar uma implementação do WGCNA que utiliza o potencial paralelo da GPU para tratar conjuntos de dados envolvendo grande quantidade de genes.

Capítulo 2

Fundamentos

2.1 Expressão Gênica

Expressão gênica é o nome dado ao conjunto de processos biológicos que transformam a informação contida no DNA em seus produtos finais. Isto inclui os processos de transcrição, tradução de RNA e síntese e modificação de proteínas e de RNAs não codificantes (ncRNAs).

Como todas as células de um organismo possuem o mesmo DNA, as diferenças entre os tipos de células e tecidos são determinadas pelo padrão de expressão de seus genes. Deste modo, cada tipo de célula expressa um subconjunto diferente de genes do genoma do organismo. São essas diferenças de expressão gênica que determinam a estrutura de uma célula e as funções exercidas por ela.

A expressão dos genes de uma célula é controlada por uma série de processos de regulação gênica. Estes processos determinam quais genes são expressos em uma célula e em que quantidades seus produtos são sintetizados. Um exemplo de processo de regulação é a interação de certas proteínas com regiões promotoras de transcrição no DNA, que provocam um aumento no nível de transcrição do gene correspondente.

São estes processos de regulação gênica que são responsáveis por controlar o desenvolvimento da célula, o funcionamento correto de seus processos metabólicos e também o modo como as células reagem à mudanças no meio em que se encontram. Portanto, o estudo dos padrões de expressão gênica e seus mecanismos de regulação é de grande importância para a biologia e biomedicina, permitindo a análise do funcionamento de células e seus mecanismos de sinalização, tal como a identificação de biomarcadores de doenças e possíveis alvos para terapias. [1]

2.2 Redes de Co-expressão

Um conceito importante para o estudo de padrões de expressão gênica de um organismo é o de **co-expressão**. Dois genes são considerados co-expressos quando os níveis de expressão deles estão fortemente relacionados. Um par de genes em que o produto de um promove a expressão do outro, por exemplo, seria um par de genes co-expressos, devido à esta interação.

O conjunto de todos os genes e suas interações de co-expressão em um conjunto de dados forma uma **rede de co-expressão gênica**.

A análise de redes de co-expressão permite a inferência de diversas informações biologicamente relevantes. Redes de co-expressão gênicas podem ser utilizadas para tarefas como identificação e priorização de genes relacionados à doenças, anotação funcional de genes e identificação de genes reguladores. [4]

A construção e análise de redes de co-expressão, em geral, é feita em três passos. Primeiro uma medida de co-expressão entre genes é calculada a partir dos dados de expressão. Esta medida pode ser a correlação de Pearson, como é o caso no WGCNA, ou outra medida de correlação, informação mutua entre genes, entre outras. Em seguida, a rede de co-expressão é construída de modo que cada nó da rede representa um gene, e cada aresta representa a relação de co-expressão, de modo que dois nós estão conectados apenas se os genes correspondentes forem suficientemente co-expressos. O terceiro passo da análise é a aplicação de um algoritmo de clusterização para a identificação de módulos da rede. Um módulo é um grupo de genes que apresenta padrões de expressão parecidos. A partir destes padrões, podemos, por exemplo, tentar inferir relações de regulação ou funções comuns entre os genes do módulo.[4]

2.3 *Weighted Gene Correlation Network Analysis*

2.3.1 Introdução

Uma técnica de análise de dados de expressão gênica bastante utilizada é o *Weighted Gene Correlation Network Analysis* (**WGCNA**)[12]. O WGCNA é utilizado para encontrar agrupamentos de genes que sejam altamente correlacionados entre si, resumir tais agrupamentos usando genes *hub* ou *eigengenes*, e relacionar os grupos ou módulos encontrados tanto entre si quanto com características externas das amostras. Para isso, o WGCNA utiliza redes com pesos, em que as arestas tem valores reais entre 0 e 1 em vez de valor binário, em combinação com o critério de topologia livre de escala.

2.3.2 Construção da rede

Para definir a rede de co-expressão, precisamos calcular alguma medida do qual relacionado um par de genes é entre si. As medidas utilizadas no WGCNA são baseadas na correlação de Pearson entre os níveis de expressão dos genes. Desse modo, genes altamente correlacionados são considerados co-expressos.

Considere uma matriz de dados de expressão gênica $X_{n\times m}$, com m observações de n genes, tal que as colunas de X têm média 0 e variância 1. Então, a matriz de correlação de Pearson de X é dada por

$$C = \frac{1}{m} X X^T.$$

A co-expressão, ou similaridade, entre dois genes i e j pode ser calculada como

$$s_{ij} = |C_{ij}|, \tag{2.1}$$

que dá igual peso a correlações positivas e negativas, e dá baixa similaridade quando não há correlação, ou como

$$s_{ij} = \frac{1 + C_{ij}}{2}, \tag{2.2}$$

que preserva a informação do sinal da correlação, considerando similares os genes com alta correlação positiva e dissimilares os com correlação negativa.

A matriz de adjacência da rede é calculada como uma função da similaridade entre os genes. Em geral, a função utilizada é uma binarização, em que a adjacência $a_{ij} = 1$ se $s_{ij} \geq \tau$ e 0 caso contrário, com $\tau \in [0, 1]$. Porém, o uso desta função gera uma rede de co-expressão sem pesos, perdendo assim informação sobre o grau de similaridade entre os genes. Já no WGCNA, calculamos a adjacência entre os genes i e j como

$$a_{ij} = s_{ij}^{\beta},$$

com $\beta \geq 1$. O parâmetro β é escolhido de forma que a rede obtida tenha uma topologia livre de escala, em que a distribuição de graus da rede segue uma lei de potência. Ou seja,

$$\Pr(k) \propto k^{-\gamma},$$

onde k é o grau do nó na rede e $\gamma \geq 1$.

2.3.3 Clusterização

A partir da rede de co-expressão obtida, calculamos um agrupamento de genes através de um algoritmo de clusterização. O algoritmo usado pelo WGCNA é a clusterização hierárquica.

Para a clusterização é necessário o uso de uma medida de distância entre os pontos do conjunto de dados. Distâncias bastante usadas para clusterização são a distância euclidiana e a similaridade por cosseno. No WGCNA a função de distância utilizada é a *Topological Overlap Dissimilarity Measure*.

Topological Overlap Measure

A *Topological Overlap Measure* (**TOM**) é uma medida de interconectividade entre dois nós de uma rede. Para dois nós i, j de uma rede, a TOM ω_{ij} é dada por:

$$TOM_{ij} = \frac{l_{ij} + a_{ij}}{\min\{k_i, k_j\} + 1 - a_{ij}}$$

em que k_i é o grau do i-ésimo nó da rede, e $l_{ij} = \sum_u a_{iu}a_{uj}$, que, em redes sem pesos, é igual ao número de vizinhos comuns entre os nós i e j.

A partir da medida TOM, calculamos a distância entre dois nós i, j como

$$d_{ij} = 1 - TOM_{ij}$$

A medida TOM foi escolhida pelos autores do WGCNA por resultar em módulos mais distintos que a medida usual, que era $1 - |C_{ij}|$.[12]

Clusterização Hierárquica

A **clusterização hierárquica** é um algoritmo de clusterização que busca construir uma hierarquia de *clusters*, em que pontos em um *cluster* estão todos mais próximos entre si que com pontos em outro *cluster*. A hierarquia resultante é geralmente representada por uma árvore chamada **dendrograma**.

Existem duas estratégias para a clusterização hierárquica, a **aglomerativa**, em que todos os pontos começam em seu próprio *cluster*, e pares de *clusters* são agrupados iterativamente até que todos os pontos sejam incluídos em uma única hierarquia; e a **divisiva**, em que todos os pontos começam no mesmo grupo e este é dividido em grupos menores iterativamente, até que todos os pontos tenham seu próprio grupo. [10]

Para realizar a clusterização hierárquica, precisamos calcular de algum modo a distância não apenas entre pontos do conjunto de dados, mas entre pontos e *clusters* ou entre dois *clusters*. Para tal, existem diferentes estratégias, chamadas de **funções de ligação** (*linkage functions*). Considere os *clusters* α, β, com matrizes de expressão X_α, X_β, e uma função de distância d. Algumas funções de ligação comuns L são: [10]

1. Ligação Simples:
$$L(\alpha, \beta) = \min_{x_i \in X_\alpha, x_j \in X_\beta} d(x_i, x_j)$$

2. Ligação Completa:
$$L(\alpha, \beta) = \max_{x_i \in X_\alpha, x_j \in X_\beta} d(x_i, x_j)$$

3. Ligação Média:
$$L(\alpha, \beta) = \frac{1}{|\alpha||\beta|} \sum_{x_i \in \alpha, x_j \in \beta} d(x_i, x_j)$$

Para obter os módulos a partir do dendrograma, é necessário "cortar"a árvore em alguma altura da hierarquia. O método de corte pode ser por um limiar estático, em que separamos em *clusters* distintos os ramos do dendrograma unidos em uma altura maior que o limiar. O WGCNA também utiliza um método "dinâmico" de corte que, em vez de usar apenas um limiar estático, corta o dendrograma considerando a estrutura dos módulos obtidos.[8]

2.3.4 Análise dos módulos

Os módulos obtidos da clusterização podem ser analisados de diversas formas. Pode-se realizar uma análise de enriquecimento funcional dos módulos da rede, para verificar se os módulos estão correlacionados com alguma função específica. Podemos encontrar os chamados genes *hub* ou os *eigengenes* dos módulos.

Um gene *hub* é um gene do módulo que pode ser utilizado para representar o módulo como um todo. Por exemplo, o gene de *hub* pode ser o gene de maior conectividade intramodular, ou o gene mais similar com o restante do módulo por alguma medida de similaridade, como a correlação ou o TOM.

Eigengenes são vetores calculados a partir dos dados de expressão dos genes de um módulo que pode ser usado como um representante do módulo, de forma similar a um gene *hub*. O *eigengene* de um módulo α é definido como o primeiro componente principal da matriz de expressão gênica X_α referente aos genes do módulo. Os *eigengenes* também podem ser usados para obter os genes *hub*, calculados como o gene com maior correlação com o

eigengene de cada módulo.

Os genes *hub* e *eigengenes* obtidos da rede podem ser usados para diversos tipos de análise. A correlação entre os *eigengenes* de módulos diferentes pode ser usados para inferir funções relacionadas entre seus genes. Podemos calcular a correlação entre os *eigengenes* dos módulos e variáveis externas das amostras coletadas para obter uma correlação entre estas e o pertencimento ao módulo, como na figura 4.3. [6]

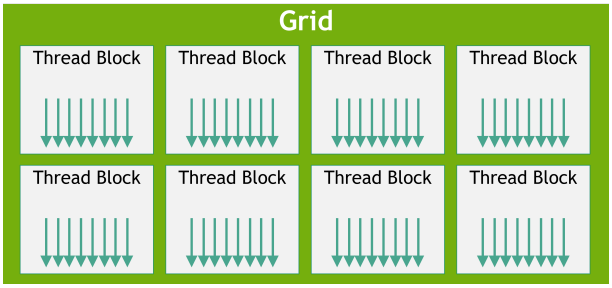
2.4 CUDA

CUDA (Compute Unified Device Architecture) é uma plataforma de computação paralela e uma API desenvolvida pela NVIDIA. Ela permite que os desenvolvedores utilizem GPUs (Unidades de Processamento Gráfico) para processamento geral, oferecendo um grande aumento no desempenho para aplicações computacionais intensivas. [2]

O uso de GPUs torna viável a realização de cálculos científicos e analise de dados em escala significativamente maior. No WGCNA, por exemplo, o cálculo da matriz de correlação e da matriz de TOM são computacionalmente intensivas mas altamente paralelizáveis. Portanto buscamos implementar estes cálculos em GPU, utilizando CUDA C++.

Programação em CUDA é feita através da escrita de *kernels* CUDA. Um *kernel* é uma função em CUDA C++ especialmente marcada para compilação para a GPU. *Kernels* CUDA são executados em executados na GPU por *threads* em paralelo.

As *threads* da GPU são organizadas em **blocos** indexados, que por sua vez são organizados em grades pela GPU, como mostra a figura 2.1. Cada bloco, e cada *thread* em um bloco, possui um índice de até três dimensões. Estes índices podem ser usados para identificar a *thread* e permite que elas calculem os índices de valores que elas devem usar, por exemplo, para acessar os valores de um vetor ou matriz.



Source: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

Figura 2.1: Organização das threads da GPU em CUDA.

As *threads* são escalonadas na GPU em grupos de 32 chamados *warps*, de modo que as *threads* de um mesmo *warp* tenham índices consecutivos. As *threads* de um *warp* são executadas todas simultaneamente, começando do mesmo endereço de instrução.

Um *kernel* CUDA `foo` pode ser chamado a partir de código C++ pela sintaxe `foo<<<b,t>>>`, onde `b` é o número ou dimensão de blocos a ser utilizados, e `t` o número ou dimensão de *threads* por bloco.

A organização das *threads* da GPU em uma hierarquia reflete a organização da memória da GPU. Cada *thread* possui uma memória local, e cada bloco possui uma memória própria compartilhada entre suas *threads*. As *threads* também tem acesso à memória global da GPU. O padrão de acesso à memória pelas *threads* tem um impacto significativo no desempenho do *kernel*. Assim como na CPU, acessos à memoria contígua é mais eficiente que acessos aleatórios, porém com a diferença que, devido ao método de escalonamento das *threads* em *cores* da GPU, são mais eficientes os acessos à endereços espaçados pelo tamanho da grade de blocos a que pertencem. [2] [3]

Capítulo 3

Implementação

Nossa implementação do WGCNA foi feita utilizando CUDA C++ 12, uma extensão da linguagem de programação C++ desenvolvida pela NVIDIA para a programação de suas GPUs para computação geral.

O CUDA foi utilizado para implementar o cálculo das matrizes de correlação dos de expressão, de adjacência e de *Topological Overlap*, o ajuste da distribuição livre de escala, e para a atualização da matriz de distâncias durante a clusterização hierárquica.

Nossa implementação está organizada em três arquivos fonte:

- kernel.cu

  É o arquivo principal. Ele contém a função `main` e todos os *kernels* CUDA que realizam os cálculos necessários para o WGCNA.

  Listing 3.1: Kernels CUDA para implementação do WGCNA

```
__global__ void averagesKernel(/* ... */);
__global__ void standardDeviationKernel(/* ... */);
__global__ void correlationKernel(/* ... */);
__global__ void adjacencyKernel(/* ... */);
__global__ void topologicalOverlapKernel(/* ... */);
__global__ void minDistanceKernel(/* ... */);
__global__ void mergeClustersKernel(/* ... */);
```

- tree.cpp

  Contém a classe `Tree`, uma implementação de uma árvore binária. Esta classe é usada para facilitar a construção do dendrograma durante a clusterização hierárquica. Esta class também implementa um método de corte dinâmico do dendrograma baseado na do pacote WGCNA em R.

Listing 3.2: Interface da classe Tree

```
class Tree {
public:
  explicit Tree(int value);
  Tree(float dist, Tree* left, Tree* right);

  inline bool IsLeaf() const;
  inline int Value() const;

  std::map<int,int> DynamicCut(float baseHeight,
                               int minClusterSize) const;

private:
  void getInOrderDistances(std::vector<float>& dists) const;
  void getLeafValues(std::vector<int>& dists) const;

  static std::vector<int> cut(std::vector<float> const& heights,
                              float level, int minClusterSize);
  static std::vector<int> adaptiveCut(std::vector<float> heights,
                                      int minClusterSize);
};
```

- matrix.cpp

  Contém a classe `Matrix`, que funciona como um *wrapper* para facilitar a criação e gerenciamento de memória para armazenar as matrizes de dados utilizadas na CPU, acesso à indices na matriz, e também a leitura de matrizes a partir de arquivos CSV.

Listing 3.3: Interface da classe Matrix

```
class Matrix
{
public:
  Matrix() = default;
  Matrix(int n) : Matrix(n, n) {}
  Matrix(int m, int n) : _m(m), _n(n) { ... }
  Matrix(int m, int n, float* data) : _m(m), _n(n), _data(data) {}
  ~Matrix();
```

```
  Matrix(Matrix const& other) : Matrix(other._m, other._n) { ... }
  void operator=(Matrix& other) noexcept;

  float& operator()(int i, int j);
  float const& operator()(int i, int j) const;

  bool operator==(Matrix const& other) const;
  bool operator!=(Matrix const& other) const;

  std::pair<int, int> Size() const;

  float* Data();
  float const* Data() const;

  void DebugPrint() const;
};
```

Os *kernels* CUDA implementados são:

- `averagesKernel`, `standardDeviantionKernel` e `correlationKernel`

  que calculam, respectivamente, a média, o desvio padrão e a correlação da matriz de dados de expressão dada como entrada do programa;

- `adjacencyKernel` calcula a matriz de adjacência com *soft thresholding* a partir da matriz de correlação, dando a opção entre os dois tipos de similaridade;

- `topologicalOverlapKernel` calcula a matriz de TOM para a clusterização.

Todos os *kernels* acima são chamados com uma *thread* da GPU por resultado, i.e. uma *thread* por coluna para calcular as médias e desvios padrão e uma *thread* por entrada de cada matriz.

- `minDistanceKernel`

  busca o menor valor na matriz de distancias para a clusterização

- `mergeClustersKernel`

  atualiza a matriz de distancias a cada passo da clusterização hierárquica, utilizando Ligação Média.

Capítulo 4

Resultados

O desempenho obtido em nossa implementação em CUDA é superior à implementação em R para quantidades pequenas e médias. Para quantidades maiores de dados, obtivemos performance comparável à versão em R. Para grandes conjuntos de dados, o tempo de execução em CUDA é afetado pelo aumento na quantidade de operações de leitura e escrita na memória global da GPU, que possuem maior latência, além de possíveis gargalos na transferência de dados entre CPU e GPU

Para os testes e geração das figuras, foram utilizados dados do tutorial de WGCNA criado pelos autores originais do pacote R, que contém dados de expressão gênica coletados dos fígados de fêmeas de ratos. [5]

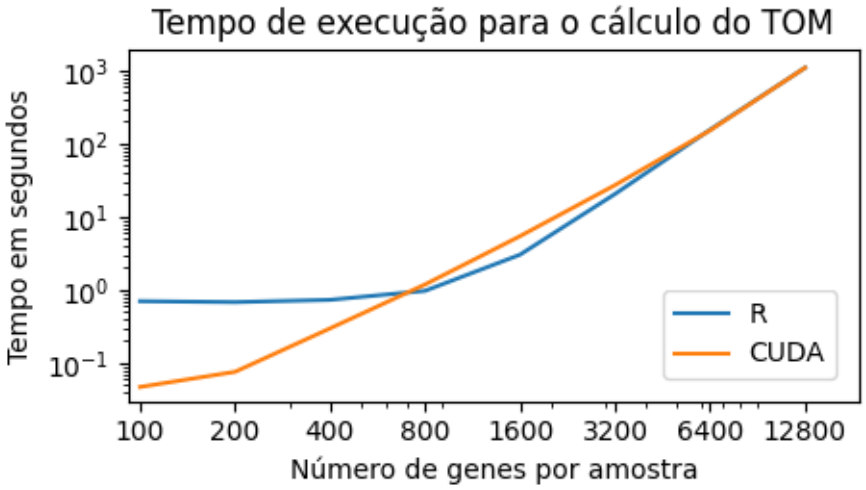


Figura 4.1: Gráfico log-log do tempo de execução para o cálculo da matriz TOM da nossa implementação, e da implementação em R

Os resultados produzidos por nossa implementação são consistentes com os da implementa-

ção original em R, obtendo módulos bem similares. As Figuras 4.2 e 4.3 mostram análises geradas com a saída da nossa implementação.

O dendrograma na Figura 4.2 mostra uma clusterização hierárquica dos *eigengenes* encontrados, que pode ser utilizado para identificar relações entre os *clusters*, ou mesmo *superclusters* de genes.

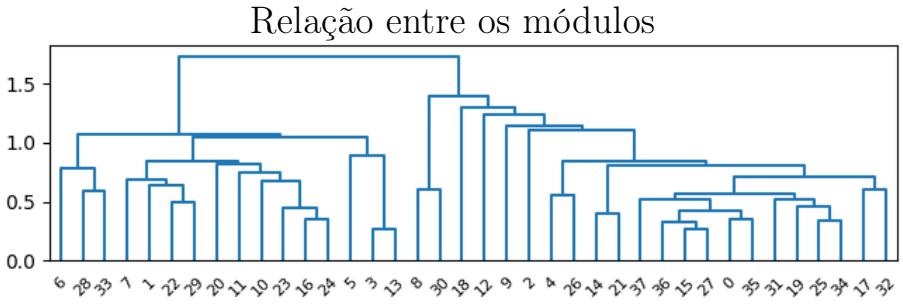


Figura 4.2: Dendrograma dos módulos obtidos

O *heatmap* na figura 4.3 mostra a correlação entre os *eigengenes* e outras variáveis coletadas das ratas no estudo. Pode-se ver que alguns dos *clusters*, como o *cluster* 33, tem correlação relativamente alta com variáveis como o peso e níveis de gordura. Já outros tem alta correlação negativa com estas mesmas variáveis. Isto pode indicar que genes destes *clusters* estão relacionados com o metabolismo de gordura no fígado destes ratos.

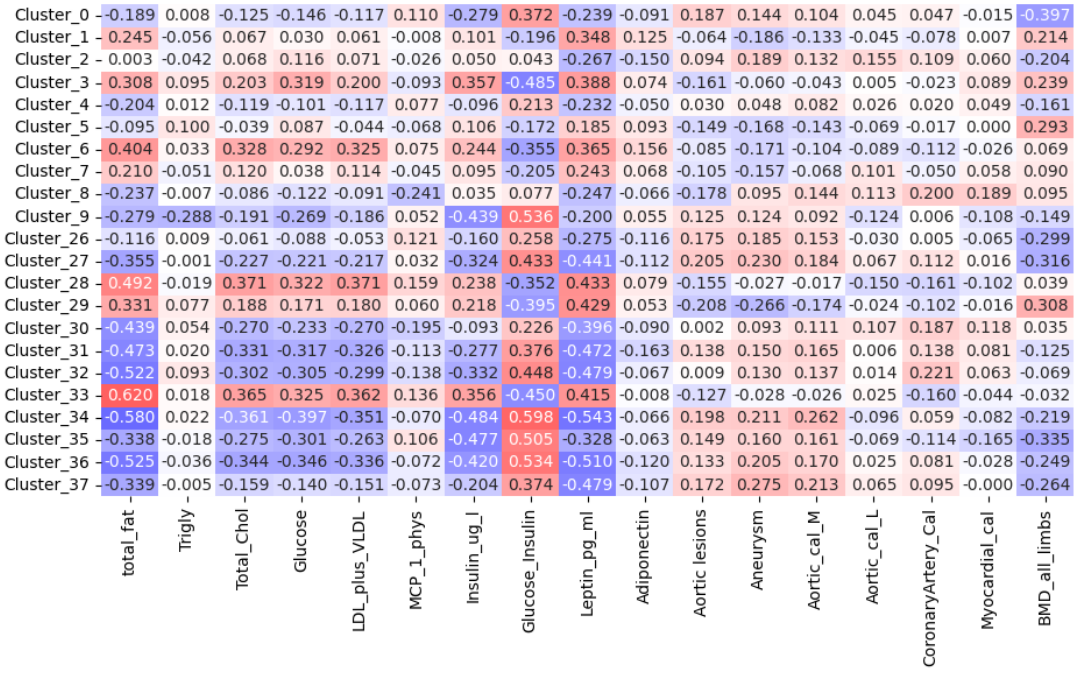


Figura 4.3: Heatmap da correlação entre os *eigengenes* dos módulos e variáveis externas dos dados de expressão

Capítulo 5

Conclusão

Neste trabalho, foi implementado o método WGCNA de análise de dados de expressão gênica. A implementação foi feita em CUDA C++, para paralelização dos cálculos das correlações, da matriz de TOM, das matrizes de distância para a clusterização hierárquica, e para o cálculo dos *eigengenes*. Os resultados obtidos são consistentes com os de outras implementações disponíveis.

Como trabalhos futuros, é possível implementar outros métodos de ligação para a clusterização hierárquica além da ligação média, tal como o uso de outros critérios para a escolha dos parâmetros, e.g. topologias alternativas como a de mundos pequenos. Apesar das limitações de performance do Python para grandes volumes de dados, uma interface baseada em Python ou Julia poderia aproveitar bibliotecas otimizadas para GPUs, mantendo a eficiência da implementação CUDA. Também seria possível estender a ferramenta para calcular outras métricas de rede para complementar a análise de *clusters* do WGCNA.

Referências

[1] Bruce Alberts et al. *Molecular Biology of the Cell: Seventh International Student Edition with Registration Card*. WW Norton & Company, 2022.

[2] *CUDA C++ Programming Guide*. URL: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#kernels (acedido em 15/12/2024).

[3] *CUDA Technical Blog*. URL: https://developer.nvidia.com/blog/cuda-pro-tip-write-flexible-kernels-grid-stride-loops/ (acedido em 15/12/2024).

[4] Sipko van Dam et al. "Gene co-expression analysis for functional classification and gene–disease predictions". Em: *Briefings in Bioinformatics* 19.4 (jan. de 2017), pp. 575–592. ISSN: 1477-4054. DOI: 10.1093/bib/bbw139. eprint: https://academic.oup.com/bib/article-pdf/19/4/575/25193126/bbw139.pdf. URL: https://doi.org/10.1093/bib/bbw139.

[5] Anatole Ghazalpour et al. "Integrating genetic and network analysis to characterize genes related to mouse weight". Em: *PLoS genetics* 2.8 (2006), e130.

[6] Steve Horvath. *Weighted network analysis: applications in genomics and systems biology*. Springer Science & Business Media, 2011.

[7] Peter Langfelder e Steve Horvath. "WGCNA: an R package for weighted correlation network analysis". Em: *BMC bioinformatics* 9 (2008), pp. 1–13.

[8] Peter Langfelder, Bin Zhang e Steve Horvath. "Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R". Em: *Bioinformatics* 24.5 (nov. de 2007), pp. 719–720. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btm563. eprint: https://academic.oup.com/bioinformatics/article-pdf/24/5/719/49051558/bioinformatics_24_5_719.pdf. URL: https://doi.org/10.1093/bioinformatics/btm563.

[9] Meimei Liang et al. "FastGCN: a GPU accelerated tool for fast gene co-expression networks". Em: *PloS one* 10.1 (2015), e0116776.

[10] Frank Nielsen. "Hierarchical Clustering". Em: *Introduction to HPC with MPI for Data Science*. Cham: Springer International Publishing, 2016, pp. 195–211. ISBN: 978-3-319-21903-5. DOI: 10.1007/978-3-319-21903-5_8. URL: https://doi.org/10.1007/978-3-319-21903-5_8.

[11] Narges Rezaie, Farilie Reese e Ali Mortazavi. “PyWGCNA: a Python package for weighted gene co-expression network analysis”. Em: *Bioinformatics* 39.7 (2023), btad415.

[12] Bin Zhang e Steve Horvath. “A general framework for weighted gene co-expression network analysis”. Em: *Statistical applications in genetics and molecular biology* 4.1 (2005).