Aprendendo Histórias: Aprendizado Computacional para Classificar Hiper-histórias Metabólicas

Trabalho de Conclusão de Curso apresentado ao
Departamento de Ciência da Computação do
Instituto de Matemática e Estatística da
Universidade de São Paulo

Larissa de Oliveira Penteado
Orientador: Roberto Marcondes César Jr
Coorientador: Ricardo Luiz de Andrade Abrantes

São Paulo, 2017

Este trabalho contou com apoio financeiro e institucional de Iniciação Científica da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP). Processo: 2017/04250-2. Apoio do projeto Temático FAPESP 2015/22308-2 e projeto CAPES STIC AMSUD Processo - 99999.007564/2015-05. Além de contar com a colaboração da professora Marie-France Sagot, da Universidade de Lyon.

Agradecimentos

Gostaria de começar agradecendo ao Professor Roberto e ao Ricardo pela dedicação e orientação durante a realização deste trabalho. Também pela colaboração da Professora Marie-France Sagot (Directeur de recherche - INRIA), da Universidade de Lyon.

Agradeço aos meus pais por sempre terem colocado minha educação acima de tudo, espero um dia corresponder aos esforços que já fizeram por mim. A Amanda, minha irmã, por estar sempre ao meu lado me apoiando.

Preciso agradecer ao Lucas por seu meu grande amigo e companheiro e estar comigo em todos os momentos, principalmente nestes dois últimos anos.

Ao Rodrigo por sua amizade e ajuda durante o curso, eu devo também um muito obrigada.

Por fim, agradeço a Deus, por abençoar o trabalho de minhas mãos e me dar a capacidade de desenvolver este TCC.

Resumo

Com os avanços tecnológicos das últimas décadas e a maior disponibilidade de dados de genômica, proteômica e metabolômica dos organismos, fez-se possível um estudo mais aprofundado sobre as vias de reações presentes nos mesmos. Redes metabólicas são o conjunto de reações que ocorrem no organismo, já o perfil metabolômico apresenta a variação dos metabólitos entre dois estados (antes e depois de algum evento). A partir destas redes e perfis é possível obter hiper-histórias metabólicas que são um conjunto de reações que explicam o fluxo de metabólitos entre os estados. Quando aplicado à rede da levedura, o método adotado na pesquisa deste TCC devolve um número grande (aproximadamente 10^5 soluções) de hiper-histórias, o que torna difícil extrair informações manualmente destes dados. Torna-se interessante, deste modo, a aplicação de técnicas de aprendizagem de máquina que ajudem a agrupar estas soluções para que seja possível melhor analisá-las e compreendê-las. O estudo e aplicação de algoritmos de classificação (de *clustering*) para esta tarefa são os objetivos principais deste trabalho.

Palavras-chave: *Clustering, Aprendizado de Máquina, Redes metabólicas, Perfil Metabolômico, Hiper-histórias Metabólicas.*

Abstract

Due to last decades' technological advances and a greater organisms' genomic, proteomic and metabolic data availability, it is now possible to study more carefully the reaction pathways that are present within such organisms. Metabolic networks are a set of reactions that take place within the organism. The metabolomic profile presents the variation of metabolite between two steady-states (before and after an event). By combining these networks and profiles, we can obtain metabolic hyperstories that are a set of reactions that explain the metabolite flux between two states. When applied to the yeast's network, the method applied in this TCC's research outputs a great number of hyperstories (approximately 10^5 solutions), which makes it difficult to extract information from these data manually. Therefore, applying machine learning techniques to cluster such solutions is interesting, in order to better analize and understand them. The study and application of clustering algorithms to accomplish this task are the main objectives of this work.

Keywords: *Clustering, Machine Learning, Metabolic Networks, Metabolomic Profiles, Metabolic Hyperstories.*

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

1.1 Motivação

Quando estuda-se o metabolismo de organismos e processos celulares específicos, é interessante observar o comportamento do metaboloma: conjunto de diferentes metabólitos presentes em uma célula, tecido, órgão ou organismo [19]. No momento em que são expostos a diferentes condições fisiológicas, como alterações no meio em que vivem,como escassez de alimento, choque térmico, exposição a compostos tóxicos, radiação, entre outros, os organismos apresentam mecanismos de resposta a tais estresses, que provocam mudanças nas concentrações desses metabólitos. A esta etapa em que estão ocorrendo as mudanças é dado o nome de estado transiente. Para compreender estas variações, existem dois conceitos de modelagem importantes: o perfil metabolômico que representa os dados de concentração de metabólitos antes e depois de algum evento (medidos em estado estacionário –*steady-state*), e a rede metabólica que apresenta o conjunto das reações que ocorrem no organismo. [13]

Na literatura, pode-se encontrar muitos exemplos do uso desta ferramenta, como em [15], no qual é realizado um perfil metabolômico

obtido a partir da exposição da levedura *S. cerevisiae* ao cádmio (que é uma substância tóxica) e realizada uma análise estatística para compreender as mudanças observadas.

O problema de buscar uma explicação aos dados de metabolômica é conhecido na literatura pelo nome de *metabolite set enrichment analysis* [30]. Os métodos para *metabolite set enrichment analysis* utilizam vias metabólicas anotadas, ou seja, conhecidas na literatura. Porém, ao estudar novos fenômenos, é interessante poder identificar vias alternativas ou possíveis novas interpretações biológicas, ainda não conhecidas ou anotadas.

Para tal pode-se utilizar o conceito de histórias metabólicas, que são um conjunto de reações extraídas da rede metabólica, que explicam o fluxo de matéria entre os metabólitos apresentados no perfil matabolômico. Existem, atualmente, algumas técnicas para se obter histórias metabólicas de redes. Dentre elas, destacam-se um que modela as redes por meio de grafos dirigidos [1, 17, 5] e outra que o faz utilizando hipergrafos, as histórias são chamadas de *hiper-histórias metabólicas*. Este último método será melhor explicado em outra seção (a 2.2) deste trabalho.

Ao realizar-se este tipo de modelagem, e combinando à rede metabólica o perfil metabolômico do organismo, pode-se predizer quais conjuntos de reações possivelmente foram responsáveis pelas mudanças observadas durante a resposta fisiológica à alterações de suas condições basais.

Aplicando a técnica *Totoro* [12] à rede e perfil metabólicos da levedura, *S. cerevisiae*, quando exposta a cádmio (Cd^{2+}) obtém-se um conjunto de soluções composto por mais de 10^5 hiper-histórias metabólicas, o que dificulta muito a extração de conclusões sobre as mesmas. Deste modo, é relevante estudar e aplicar métodos de manipulação destas soluções obtidas de modo que fique mais inteligível sua análise e compreensão. Dentre estes métodos, são par-

ticularmente interessantes técnicas de Aprendizagem de Máquina (*Machine Learning*) que ajudem a agrupar as hiper-histórias.

1.2 Objetivos

O objetivo deste TCC foi o desenvolvimento de um novo método de análise de hiper-histórias metabólicas utilizando técnicas não-supervisionadas de aprendizagem de máquina.

1.3 Organização do trabalho

No capítulo 2 desta monografia, iremos desenvolver uma breve contextualização dos conceitos utilizados neste trabalho no que concerne aos aspectos de modelagem e de algoritmos. No próximo capítulo, o 3, será descrito o *pipeline* da metodologia desenvolvida para o tratamento e particionamento dos dados. No capítulo 4, apresentaremos os resultados obtidos e suas análises. Finalmente, no último capítulo, o 5, serão expostas as conclusões finais sobre os resultados obtidos e o trabalho em geral.

Capítulo 2

Teoria

Neste capítulo, serão apresentados alguns conceitos teóricos relevantes a este trabalho.

2.1 Teoria de Hipergrafos

Esta seção trata da introdução de alguns conceitos sobre hipergrafos que serão utilizados para a extração de hiper-histórias metabólicas. Estas definições foram extraídas de [9, 13, 10].

Definição 1: Um hipergrafo é um par $\mathbb{H} = (\mathbb{V}, \mathbb{A})$, onde $\mathbb{V}$ é um conjunto finito de vértices e $\mathbb{A}$ é um conjunto finito de hiperarcos. Um hiperarco $a \in \mathbb{A}$ é um par ordenado (X, Y) tal que X e Y são subconjuntos disjuntos não-vazios de $\mathbb{V}$. O conjunto X (resp. Y) é dito origem(a) (resp. destino(a)).

Definição 2: Um sub-hipergrafo $\mathbb{H}' = (\mathbb{V}', \mathbb{A}')$ é um hipergrafo com $\mathbb{V}' \subseteq \mathbb{V}$ e $\mathbb{A}' \subseteq \mathbb{A}$.

Definição 3: Um hipercaminho P_{st}, de tamanho q, num hipergrafo $\mathbb{H} = (\mathbb{V}, \mathbb{A})$, é uma sequência de vértices e hiperarcos $P_{st} = (v_1 = s, a_{i1}, v_2, a_{i2}, ..., a_{iq}, v_{q+1} = t)$, em que $s \in origem(a_{i1})$, $t \in destino(a_{iq})$ e $v_j \in origem(a_{ij-1}) \cap destino(a_{ij})$.

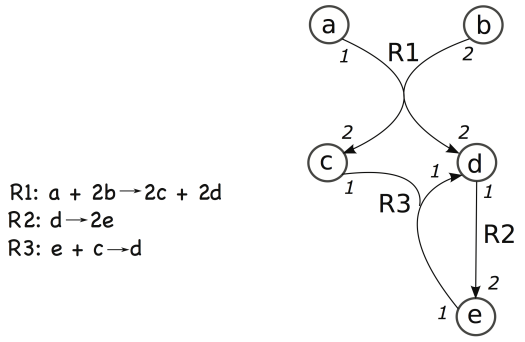
2.2 Modelagem de redes utilizando hipergrafos

Nas redes metabólicas é representado um conjunto de reações entre metabólitos que ocorrem em um organismo. No perfil metabólico, representa-se as mudanças de concentração dos metabólitos, que participam destas reações, entre dois estados estacionários. Assim, é possível modelar estas informações através de um hipergrafo $\mathbb{H} = (\mathbb{V}, \mathbb{A})$, no qual $\mathbb{V}$ é o conjunto de vértices que representam os metabólitos e $\mathbb{A}$ representa os hiperarcos que correspondem às reações que ocorrem entre eles. Ou seja, existe o hiperarco que sai de (a, b) e chega em c se, e somente se, existe a reação $a + b \rightarrow c$. Esta modelagem pode ser vista na Figura 2.1.

Nesta seção, será descrita a modelagem e extração de hiperhistórias utilizada pelo método *TOTORO* (*TOpological analysis of Transient metabOlic RespOnse*) desenvolvido em [13]. As definições e conceitos apresentados a seguir foram extraídas deste trabalho citado (*ibidem*).

O conjunto de vértices $\mathbb{V}$ é dividido em vértices Pretos $\mathbb{B}$ –aqueles cuja concentração do metabólito mudou entre os dois estados– e Brancos $\mathbb{W}$ –os que não tiveram sua concentração ou alterada ou medida. Estes conjuntos satisfazem que $\mathbb{V} = \mathbb{B} \cup \mathbb{W}$ e $\mathbb{B} \cap \mathbb{W} = \emptyset$.

Os vértices Pretos, por sua vez, podem ser classificados como Verdes –vértices que representam metabólitos cuja concentração aumentou entre as duas condições– e Vermelhos –aqueles cuja con-



a) Reações b) Hipergrafo

Figura 2.1: Em a) têm-se um conjunto de reações que formam uma rede metabólica. Em b) têm-se as reações de a) modeladas como um hipergrafo, onde cada aresta representa uma reação e os pesos atribuídos à cada aresta são os números estequiométricos.

centração diminuiu. É válido que $\mathbb{B} = Vermelhos \cup Verdes$ e $Vermelhos \cap Verdes = \emptyset$.

Existe também uma divisão para os vértices Brancos. Eles podem ser do tipo (puramente) Brancos –sua concentração não foi alterada entre os dois estados– e Cinzas –aqueles que a concentração não foi medida. De tal forma que as relações $\mathbb{W} = Cinzas \cup Brancos$ e $Cinzas \cap Brancos = \emptyset$ são satisfeitas. Todas estas relações estão representadas na tabela 2.1.

	Pretos		Brancos	
Concentração	Verdes	Vermelhos	Brancos	Cinzas
	Aumentou	Diminuiu	Não alterada	Não medida

Tabela 2.1: Classificação dos vértices da rede metabólica, segundo a variação de concentração dos metabólitos entre dois estados.

É dito que um hiperarco é *positivo* (+), se ele representa uma reação pela qual ocorreu aumento no fluxo de matéria. Do contrário, o hiperarco é *negativo* (-), o que significa que houve

uma diminuição do fluxo de matéria naquela reação. Com estas definições, pode-se formalizar as chamadas *regras de coerência* que definem um conjunto minimal de hiperarcos necessários para que os vértices sejam coerentes.

Regras de coerência para os vértices:

- Brancos:
  - Um hiperarco que chega a este vértice e um que sai devem ter sinais iguais.
  - Dois hiperarcos que chegam (ou saem deste) a este vértice têm sinais opostos.
- Verdes:
  - Um hiperarco que chega a este vértice tem sinal positivo.
  - Um hiperarco que sai deste vértice tem sinal negativo.
- Vermelhos:
  - Um hiperarco que chega a este vértice tem sinal negativo.
  - Um hiperarco que sai deste vértice tem sinal positivo.

Utilizando esta modelagem e as definições acima, é possível definir uma hiper-história metabólica.

Definição 4: Seja $\mathbb{H} = (\mathbb{V}, \mathbb{A})$ um hipergrafo dirigido com $\mathbb{V} = \mathbb{B} \cup \mathbb{W}$ e $\mathbb{B} \cap \mathbb{W} = \emptyset$. Uma hiper-história metabólica é o sub-hipergrafo extraído $\mathbb{H}' = (\mathbb{V}', \mathbb{A}')$ de $\mathbb{H}$ e uma função de sinal associada a $\mathbb{A}'$ tais que $sinal : \mathbb{A}' \mapsto \{-, +\}$, com $\mathbb{A}' \subseteq \mathbb{A}$, $\mathbb{V}' = \mathbb{B} \cup \mathbb{W}'$, com $\mathbb{W}' \subseteq \mathbb{W}$ e todos os vértices são coerentes.

Extração de hiper-histórias pelo *TOTORO*

Será descrito agora como o *TOTORO* encontra hiper-histórias metabólicas em hipergrafos. De acordo com a definição de vértices coerentes, pode-se notar que se a reação direta for positiva (resp. negativa) num hipergrafo é equivalente, em termos de coerência, à reação reversa negativa (resp. positiva). Isto porque, as regras citadas são simétricas. Assim, segue a definição de hipergrafo transformado.

Definição 5: Seja $\mathbb{H}_t = (\mathbb{V}, \mathbb{A}_t)$ um hipergrafo dirigido transformado de $\mathbb{H} = (\mathbb{V}, \mathbb{A})$, tal que $\forall a \in \mathbb{A}, a \in \mathbb{A}_t$ e $\bar{a} \in \mathbb{A}_t$, com $\bar{a}$ tal que origem($\bar{a}$) = destino(a) e destino($\bar{a}$) = origem(a).

Pode-se definir, com estes conceitos, uma hiper-história metabólica em termos de hiper-caminhos (definição 2.1) em hipergrafos.

Definição 6: Seja $\mathbb{H}_t = (\mathbb{V}, \mathbb{A}_t)$ um hipergrafo dirigido transformado, com $\mathbb{V} = \mathbb{B} \cup \mathbb{W}$ e $\mathbb{B} \cap \mathbb{W} = \emptyset$. Uma hiper-história metabólica será um conjunto de hiperarcos $\mathbb{A}' \subseteq \mathbb{A}_t$ com $\mathbb{V}' = \mathbb{B} \cup \mathbb{W}'$ e $\mathbb{W}' \subseteq \mathbb{W}$, tal que as seguintes propriedades são satisfeitas:

1. Não há um hiperarco e seu reverso em $\mathbb{A}'$.
2. Vértices Verdes não podem ser fontes.
3. Vértices Vermelhos não podem ser sorvedouros.
4. Vértices (puramente) Brancos não podem ser fontes ou sorvedouros.
5. Para todo hiperarco $a \in \mathbb{A}'$ temos que:

- Para todo $x \in origem(a)$, existe um hipercaminho de um vértice Vermelho até o vértice x.
- Para todo $y \in destino(a)$,existe um hipercaminho de y a um vértice Verde.

Do item(4) da definição acima, constata-se que vértices (puramente) Brancos que são fonte ou sorvedouro não podem estar na hiper-história, contudo vértices Cinzas que são ou fonte ou sorvedouro –chamados de vértices Cinzas isolados– podem, uma vez que não há informações sobre eles. Indicando ainda, se estiverem presentes, que seria interessante tentar medir a variação de concentração destes, se possível.

Ao buscar por hiper-histórias, deseja-se soluções que possuam um conjunto minimal ou mínimo de arcos. A primeira diz respeito a uma solução da qual não consegue-se remover nenhum elemento e continuar tendo uma solução. É pertinente encontrar soluções deste tipo, pois elas são hiper-histórias que representam mudanças locais no metabolismo. O que concorda com a hipótese de que o organismo não mudou toda a sua rede de metabolismo para responder às condições não-basais. A solução mínima, em contrapartida, busca por hiper-histórias compostas por um número mínimo de hiperarcos.

Implementação do *TOTORO*

No início é feito um pré-processamento da rede metabólica que está representada em um arquivo SBML e um arquivo BV com informações dos vértices que são Pretos. Este é feito utilizando a linguagem *Python*. Para formalizar o problema de encontrar hiper-histórias em lógica disjuntiva, foi utilizada a linguagem de programação *Prolog* e o paradigma ASP (*Answer Set Programming*). Por fim, para enumerar todas as hiper-histórias foi empregado o

Clingo, um resolvedor lógico para ASP (*answer set solver*).

Foi demonstrado em [13] que a partir do *Totoro* é possível inferir quais reações estiveram possivelmente presentes ou reprimidas durante um estado transiente. Estas informações são relevantes ao estudo do metabolismo de diversos organismo, dada a dificuldade[28] de se obter quais reações e mecanismos estão presentes nas respostas destes a estímulos do meio ou internos.

2.3 Algoritmos de *Clustering*

Para realizar o particionamento das hiper-histórias, foram utilizados alguns algoritmos de *clustering* disponíveis em uma biblioteca do *Python* chamada *Scikit-Learn* [3]. Foram eles: *K-Means*, *Mini-Batch K-Means*, *Affinity Propagation* e *Spectral Clustering*.

Os dois primeiros foram escolhidos por serem os mais conhecidos para realizar este tipo de tarefa. Os dois últimos, por sua semelhança com os dados tratados neste TCC, uma vez que ambos modelam os dados de entrada como grafos.

Notação

Definição dos seguintes conceitos:

- Seja $\mathbb{D} = \{d_1, d_2, ..., d_n\}$, o conjunto de dados, em que cada $d_i \in \mathbb{R}^n$ representa um vetor de características.

- Se x, y $\in \mathbb{R}^n$, então dist(x, y) representa a distância entre os vetores x e y.

- Norma euclidiana ou norma-2: x, y $\in \mathbb{R}^n$, então $||x - y||_2 = \sum_{i=1}^{n}(x_i - y_i)^2$
- Para qualquer subconjunto finito $C \subset \mathbb{R}^n$, $|C|$ indica o número de elementos neste subconjunto.

2.3.1 K-Means

Aprendizado não-supervisionado trata do problema de particionar um conjunto de dados em k partes, em que k é um número inteiro, $k > 1$. O critério para esta partição é a proximidade de todos os elementos de um mesmo subconjunto (uma parte) em relação ao centro deste. Trata-se de um problema NP-difícil [16].

Existem algumas heurísticas para tratar o problema que geralmente conseguem convergir a ótimos locais e produzem resultados satisfatórios, como é o caso do Algoritmo de Lloyd [14].

A inicialização dos k centróides de cada uma destas k partes é aleatória e o algoritmo termina o particionamento, após estes convergirem a um ponto fixo, ou seja, quando ao final de uma iteração eles permanecem iguais ao que eram no início da mesma, ou quando atinge-se um número limite de iterações pré-definido.

Uma iteração deste algoritmo pode ser descrita da seguinte forma. Sejam $\mathbb{D}$, o conjunto de dados, $C = \{c_1, c_2, ...c_k\}$ o conjunto de centróides, e $P_1^j, ..., P_k^j$ cada uma das partes, da partição encontrada na iteração j. Na iteração $j + 1$, tem-se que:

1. Para cada $d_i \in D$, encontre c, um centróide que minimiza a distância de d_i a cada um dos centróides, $c = argmin_{c_l \in C} dist(c_l, d_i)$. Se l é o índice do centróide c, então $d_i \in P_l^{j+1}$
2. Reestime os centróides de acordo com o seu novo centro de massa definido pelos pontos a ele associados.

$$c_i = 1/|P_i^{j+1}| * \sum_{p_l \in P_i^{j+1}} p_l$$

Em particular, o *K-Means* implementado na biblioteca *Scikit-Learn* é uma implementação do algoritmo de Lloyd, e a métrica empregada para calcular as distâncias entre os pontos e os centróides é a norma euclidiana. A complexidade do algoritmo é $O(knT)$, em que k é o número de *clusters*, n é o tamanho do conjunto de dados e T corresponde ao número de iterações do algoritmo. T pode ser definido ao chamar este método, sendo que seu valor padrão é 300. Essas informações podem ser vistas na página do *scikit* referente ao *K-Means* [24].

Limitações do algoritmo

É sabido que esta heurística, para resolver o problema de particionamento do conjunto $\mathbb{D}$, apresenta alguns pontos negativos. Por exemplo, é preciso definir *a priori* o número de *clusters*, está sujeito a ótimos locais que podem não ser soluções muito satisfatórias e tem uma escalabilidade ruim para um conjunto grande de dados, já que estes precisam estar sempre na memória principal quando o algoritmo está sendo executado.

2.3.2 Mini-Batch K-Means

A performance de memória do *K-Means* piora quanto maior o tamanho do conjunto de dados, uma vez que todo o conjunto precisa estar na memória enquanto o algoritmo é executado. Como uma alternativa a isto, existe o *mini-batch K-Means*, descrito em [26]. A ideia principal deste método é utilizar um *batch* (lote) que é um subconjunto dos dados de tamanho fixo. Os elementos deste *batch*

são escolhidos aleatoriamente em cada iteração.

Na primeira iteração, escolhe-se aleatoriamente um conjunto, de tamanho k, de centróides. Cada iteração deste algoritmo procede do seguinte modo:

1. Escolha $\mathbb{B} \subset \mathbb{D}$ e $|\mathbb{B}| = m$, onde m é o tamanho da *batch* pré-estabelecido.

2. Para cada $b \in \mathbb{B}$, compute o centróide c mais próximo a b, igual é feito no passo (1) do *K-Means*.

3. Para cada $b \in \mathbb{B}$ e c, associado a este b no passo anterior, calcula-se a taxa de aprendizado para este centróide (η), que é o inverso ao número de elementos b que foram associados a este c. Calcula-se então a atualização do c, fazendo uma combinação convexa entre o c e o b:
$c = (1 - \eta) * c + \eta * b$.

Resultados empíricos demonstraram que o *mini-batch K-Means* consegue convergir a uma partição dos dados com qualidade semelhante à do *K-Means* com a vantagem de conseguir fazê-lo em menos tempo e com menor consumo de memória. Sendo mais adequado, deste modo, para um conjunto grande de dados [2].

2.3.3 Affinity Propagation

Utilizando este método deseja-se particionar um conjunto de dados $\mathbb{D}$, em K partes, utilizando alguma medida de similaridade. Para tal, considera-se que todos os pontos são potenciais exemplares, que são o representante de um certo *cluster* (similares aos centróides do *K-Means*). Assume-se então, que cada um dos pontos é um nó em uma rede e que é possível transmitir mensagens –números reais–

através das arestas até que seja encontrado um conjunto de exemplares que satisfaçam algum critério pré-estabelecido. [8]

Como entrada para este algoritmo é fornecida, além do conjunto de dados, uma matriz que contém valores reais de $s(i,k), i \neq k$, que são as similaridades que medem "o quão adequado é o ponto k ser exemplar do *cluster* contendo i"(*ibidem*). Esta medida pode ser a norma euclidiana, por exemplo. Além disso, também é possível fornecer a preferência, que são as medidas de similaridade para $s(i,i)$. Quanto maior este número, maior a chance de i ser um exemplar. Se esta medida não for concedida, podemos assumir que os pontos têm chances iguais de serem exemplares.

As mensagens trocadas para quaisquer dois pontos i e k, com $i \neq k$ são: *responsabilidade* $r(i,k)$ e *disponibilidade* $a(i,k)$. A primeira é enviada de i a k e indica "o quão apropriado é k ser exemplar do subconjunto que contém i"(*ibidem*). Enquanto isso, $a(i,k)$ é enviada de k a i e "representa o quão bom seria para k ser exemplar de i sabendo que outros pontos tem k como exemplar também"(*ibidem*). Ou seja, para o cálculo da responsabilidade considera-se outros pontos que estão no mesmo subconjunto que i. Por outro lado, para calcular a disponibilidade, são considerados todos os pontos representados por k.

No início do algoritmo têm-se as medidas de similaridade e:

$$a(i,k) = 0$$
$$r(i,k) = s(i,k) - \max_{k' \neq k}\{s(i,k')\}$$

Uma iteração típica do algoritmo ocorre da seguinte maneira:

- $r(i,k) = s(i,k) - \max_{k' \neq k}\{\, a(i,k') + s(i,k')\}$, se $i \neq k$.
- $r(k,k) = s(k,k) - \max_{k' \neq k}\{s(k,k')\}$, caso contrário.
- $a(i,k) \;=\; \min\{\, 0, r(k,k) + \sum_{j \neq k, j \neq i} \max\{\, 0, r(j,k)\}\ \}$, se i $\neq$ k.

$a(k,k) = \sum_{i \neq k} \max\{\, 0, r(i,k)\}$, caso contrário.

- Para todo $i \in \mathbb{D}$, calcula-se $l = argmin_k\{a(i,k) + r(i,k)\}$. Há duas opções, se $l = i$, então i será seu próprio exemplar, caso contrário, l é exemplar para i.

A fim de evitar instabilidades numéricas, quando os valores de $r(i,k)$ e $a(i,k)$ são atualizados, introduz-se um fator de atenuação (*damping factor*) λ, $0 < \lambda < 1$. Então, se o método está numa iteração t, têm-se que:

$$r_t(i,k) = \lambda r_{t-1}(i,k) + (1-\lambda) r_t(i,k)$$
$$a_t(i,k) = \lambda r_{t-1}(i,k) + (1-\lambda) r_t(i,k)$$

O algoritmo termina após um número pré-estabelecido de iterações ou se após 10 iterações não houve mudanças quanto aos exemplares de cada ponto. A complexidade deste algoritmo é $O(n^2T)$, em que $n = |\mathbb{D}|$ e T é o número de iterações até a convergência ser atingida e a quantidade de memória utilizada é $O(n^2)$ [23]. O ponto interessante deste algoritmo é que não é necessário fornecer o número de partes (K) em que deseja-se particionar o conjunto de dados *a priori*, permitindo uma seleção automática de modelo.

Limitações do Algoritmo

Além da complexidade deste algoritmo, outra limitação apresentada é a dificuldade em saber qual a escolha de preferências que pode produzir o melhor particionamento dos dados. Pode haver casos degenerados, em que a função do passo (3) da iteração do algoritmo

tem máximos locais e não conseguimos atingir a convergência, ocorrendo deste modo oscilações na propagação de afinidades, que não podem ser automaticamente corrigidas [29].

2.3.4 Spectral Clustering

Este algoritmo tem por objetivo a partição de um conjunto de dados $\mathbb{D}$ em k partes. Para tal, é feita a transformação deste conjunto para um grafo G não-dirigido, com peso nas arestas, representado por uma matriz de afinidades. Os vértices deste grafo são os pontos em $\mathbb{D}$, já as arestas são construídas seguindo alguma medida de similaridade ou distância. Para particionar o grafo, considera-se o problema do corte balanceado.

Definição do algoritmo, adaptada de [18]:

1. Montar a matriz de afinidade $A \in \mathbb{R}^{\mathrm{nxn}}$ (em que n é o tamanho do conjunto de dados, como definido acima), tal que:
   $A_{ij} = e^{-dist(d_i,d_j)/(2\sigma^2)}$, se $i \neq j$ e
   $A_{ij} = 0$, se $i = j$.

2. Defina a matriz diagonal D, em que o elemento na posição (i, i) é a soma dos elementos da i-ésima linha de A. E também a matriz Laplaciana normalizada, dada por $L = D^{-1/2}AD^{-1/2}$.

3. Encontre $x_1, ..., x_k$, os k maiores autovetores de L (ortogonais), e forme a matriz $X = [x_1x_2...x_n] \in \mathbb{R}^{\mathrm{nxk}}$, em que cada coluna de i é um autovetor x_i.

4. Forme a matriz Y a partir de X, normalizando as linhas desta última, de modo que elas tenham tamanho unitário (ou seja, $Y_i = X_i/||X_i||$ e $||Y_i|| = 1$.

5. Considerando cada linha de Y como um ponto em $\mathbb{R}^{\mathrm{k}}$, particiona-se este conjunto em k *clusters*, através de algum algoritmo de *clustering* (por exemplo, o *K-Means*).

6. Atribuímos então cada ponto $d_i \in \mathbb{D}$ a um *cluster* j se, e somente se, a linha i da matriz Y foi atribuída ao *cluster* j.

Em particular, no algoritmo implementado em [25] o método de particionamento da matriz Y escolhido é o *K-Means*.

Limitações do Algoritmo

É um algoritmo mais lento, uma vez que é preciso calcular os autovetores da matriz Y. Além disso, como é necessário utilizar o *K-Means* como passo do algoritmo, as limitações desse valem para o *Spectral Clustering* também.

2.4 Heurística para a escolha do número de *clusters*

De modo a auxiliar a escolha do número de partes em que o conjunto de dados $\mathbb{D}$ será particionado, existem algumas heurísticas interessantes. A seguir será descrita uma destas heurísticas que foi utilizada neste projeto:

2.4.1 Método da Silhueta

Suponha que após a execução de algum algoritmo de *clustering*, têm-se os seguintes *clusters*: $\{C_1, C_2, ..., C_k\}$. E defina a distância entre um ponto x qualquer até um conjunto $\mathbb{B}$, como sendo:

$$d(x_i, \mathbb{B}) = \frac{1}{|\mathbb{B}|} \sum_{x \in \mathbb{B}} dist(x_i, x)$$

Assim em [22] são definidas duas funções de dissimilaridade. Elas são a média de dissimilaridade de $d_i \in \mathbb{D}$ em relação a todo elemento do *cluster* C, tal que $d_i \in C$, e podem ser de dois tipos:

$$\text{interna: } a_i = d(d_i, C)$$

$$\text{menor dissimilaridade entre } \textit{clusters}\text{: } b_i = \min_{E \neq C} d(xi, E)$$

A medida de silhueta por *cluster* é dada por:

$$s_i = \begin{cases} \dfrac{b_i - a_i}{\max\{b_i, a_i\}}, \ |C| > 1, \\ 0, \ |C| = 1. \end{cases}$$

É possível verificar que $-1 \leqslant s_i \leqslant 1$. Com este valor pode-se extrair algumas conclusões sobre o quão boa foi a escolha de um ponto $d_i \in \mathbb{D}$ ser designado a um certo subconjunto $C_j \in \{C_1, ..., C_k\}$ da partição. Assim, existem três casos:

- Quanto mais próximo s_i for de 1, conclui-se que a dissimilaridade interna entre d_i e os outros pontos de C_j é muito menor que sua dissimilaridade com os pontos dos outros *clusters*. Evidenciando, assim, que a escolha de $d_i \in C_j$ foi a apropriada.
- Quanto mais próximo s_i for de 0, infere-se que as dissimilaridades têm valores próximos, e assim, não pode-se concluir se o ponto foi bem classificado.
- Por fim, quanto mais próximo s_i for de -1, é possível dizer que a dissimilaridade entre d_i e algum outro *cluster* é muito menor que a dissimilaridade interna. Isto indica que a escolha de classificar d_i pertencendo a C_j pode não ter sido muito boa, e o ponto está mal-classificado.

Então, o valor de silhueta de uma partição como descrita acima

é $s(k) = \frac{1}{n}\sum_{i=1}^{n} s_i$. Dado um conjunto de inteiros $\{k_1, ..., k_m\}$, valores para o número de partes em que deseja-se particionar o conjunto $\mathbb{D}$, têm-se que o k a ser escolhido será aquele que obtiver maior valor de s(k), ou seja, $k^{\text{ótimo}} = argmax_{k\in\{k_1,...,k_m\}}\{s(k)\}$

2.5 Redução de Dimensionalidade

Quando lida-se com dados cuja dimensão é grande, ou seja, seus vetores de características têm bastantes *features*, é necessário atentar a alguns problemas que podem ocorrer. Conforme a dimensão dos dados aumenta, mais estes tendem a se tornar esparsos, ou seja, pontos com muitas componentes com valor zero. Isto influencia a medida de distância entre os pontos, que pode tornar-se desprezível. O que gera, deste modo, um obstáculo à aplicação de algoritmos de *clustering*, uma vez que grande parte destes utiliza medidas de distância como métrica para particionar o conjunto de dados.

Diante de tais fatos, torna-se interessante aplicar alguns métodos de redução de dimensionalidade a este conjunto para que apliquemos os algoritmos de *clustering* nestes dados transformados. Alguns estudos mostram que isto melhora a acurácia e tempo de execução de métodos como o *K-Means* [7].

Neste artigo, é apresentada uma técnica que aplica um algoritmo de *Feature Reduction* chamado *Principal Component Analysis* (PCA) ao conjunto de dados e depois aplica o *K-Means*. O PCA consiste em um método para diminuir a dimensionalidade dos dados, projetando-os em um espaço de dimensão menor, de modo que se tenha a maior variância possível entre cada uma das componentes dos pontos.

Capítulo 3

Metodologia

A seguir será apresentado o problema biológico tratado neste trabalho e também o *pipeline* de execução que foi seguido para tentar encontrar uma partição das hiper-histórias metabólicas que forneça informações sobre o mesmo.

3.1 Exposição da levedura *S. cerevisiae* ao Cádmio

A levedura *S. cerevisiae* é um eucarioto (organismo com membrana nuclear) unicelular que pertence ao Reino *Fungi*. São seres vivos largamente estudados, pois todo o seu genoma é conhecido, são não patogênicos e usados em processos para a produção de fármacos. Além disso, sua capacidade de fermentação permite seu uso na indústria alimentícia para a produção de fermentos e bebidas alcoólicas, por exemplo [20].

A levedura é considerada um bom modelo eucariota para o estudo de mecanismos moleculares de resposta a estresse oxidativo. Neste sentido, é interessante entender como organismos respondem

à intoxicação por cádmio que é um metal pesado, tóxico em concentrações baixas e cujo mecanismo de toxicidade ainda não é completamente compreendido [27].

Supõe-se que ao entrar em uma célula, o cádmio(Cd^{2+}) desloca o ferro(Fe^{2+}) e zinco(Zn^{2+}) ligados a proteínas (ou seja, que estão atuando como coenzimas), o que causa a inativação destas. Com a inativação destas enzimas vários processos metabólicos são interrompidos (*ibidem*). Uma evidência que suporta esta suposição é o estresse oxidativo que a levedura apresenta, após exposição a este metal. Porém, ainda não é certo se esta é a única causa da toxicidade do cádmio.

Um dos caminhos metabólicos ligados à desintoxicação do cádmio pela levedura é o da biossíntese de glutationa. Esta enzima antioxidante contém um grupo de tiol capaz de fazer a reação de quelação responsável por sequestrar o íon Cd^{2+}, impedindo-o de ligar-se a outras enzimas, neutralizando assim seu efeito tóxico. A biossíntese de glutationa, no entanto, requer enormes concentrações de enxofre (S). Isto faz com que a célula procure outros meios de obter este elemento, como substituir enzimas ricas em enxofre por isoenzimas que não contém enxofre [17]. Esse tipo de informação é o que deseja-se recuperar a partir da análise da rede e perfil metabólicos da *S. cerevisiae*.

3.2 *Pipeline* de execução

Será apresentado nesta seção o *pipeline* de execução implementado para a obtenção do particionamento do conjunto de hiper-histórias, assim como a descrição dos dados utilizados neste trabalho. Na figura 3.1, é representado um diagrama que esquematiza os passos que o algoritmo segue.

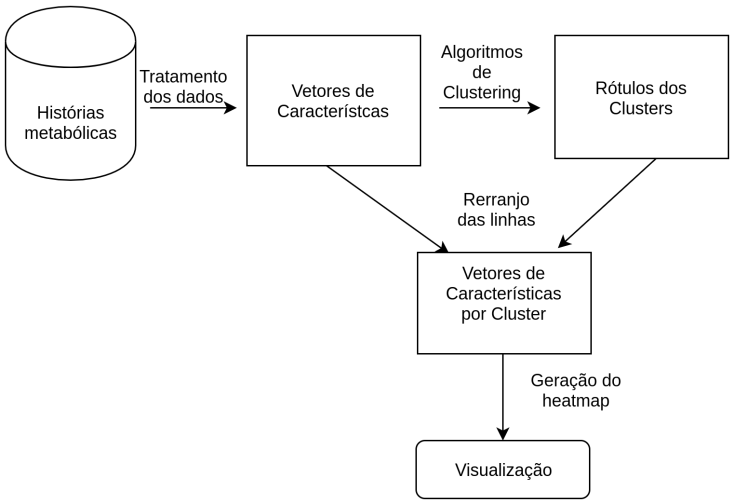


Figura 3.1: Diagrama do *pipeline* de execução.

3.2.1 Obtenção dos dados

Foram utilizados dois conjuntos de dados durante o desenvolvimento deste trabalho. O primeiro é formado pelas hiper-histórias metabólicas obtidas a partir de uma rede metabólica da levedura *S. cerevisiae* e a partir de um perfil metabolômico com 21 vértices Pretos. O segundo contém as hiper-histórias da mesma rede metabólica da levedura, porém com perfil metabolômico composto por 8 vértices Pretos.

Esta rede foi obtida por [17] através do *MetExplore* [5]. Para seu uso no *Totoro*, ela sofreu alguns pré-processamentos como descrito em [13]. Dentre estes estão a remoção de alguns compostos e a separação de cofatores envolvidos numa mesma reação.

Hiper-histórias metabólicas da *S. cerevisiae* com perfil metabolômico com 21 vértices Pretos

O conjunto de soluções obtidas pelo método *Totoro* com esta rede metabólica é formado por 268 hiper-histórias metabólicas. Existem 69 reações representadas pela união de todas estas hiper-histórias. Além disso, foi aplicada a restrição de minimização de arestas (ou seja, reações) durante a execução do programa, pois é interessante obter um conjunto com poucas reações para que fique mais acessível a interpretação das mesmas. A união de todas as hiper-histórias pode ser vista na figura 3.2.

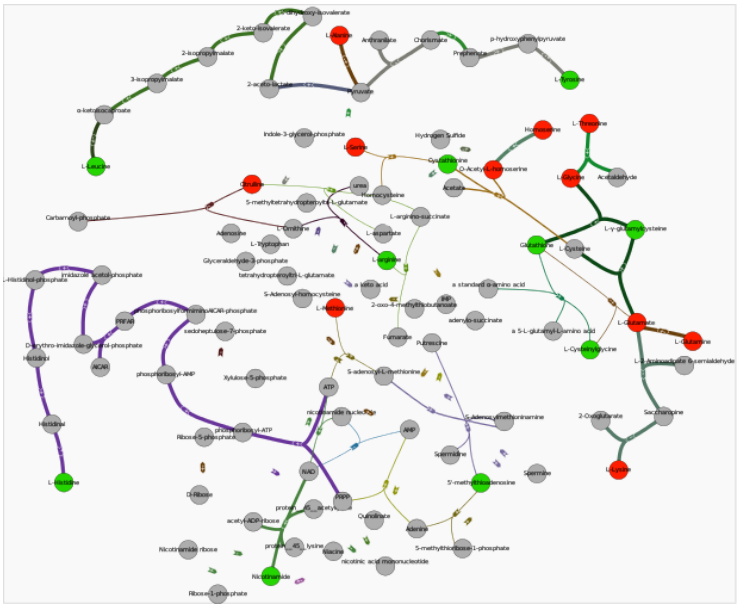


Figura 3.2: Figura mostrando uma antologia obtida combinando as 268 histórias metabólicas, foi utilizado o software DINGHY, disponível em [11]. Como podemos observar, há 21 vértices Pretos (Vermelhos ou Verdes), os outros vértices são cofatores, ou seja, vértices Brancos ou Cinzas.

Hiper-histórias metabólicas da *S. cerevisiae* com perfil metabolômico com 8 vértices Pretos

Este conjunto de soluções nos foi concedido pela autora do [13], Alice Julien-Laferrière, uma vez que a obtenção deste a partir do *Totoro* leva 168 horas. Como o objetivo deste trabalho não é a ob-

tenção destas hiper-histórias, mas o tratamento e manipulação das mesmas, optou-se por esta escolha.

Existem no total 56515391 soluções, quando é permitido que haja vértices cinzas na solução. Estas estão armazenadas em um arquivo com aproximadamente 26GB. A rede metabólica utilizada para obter tais respostas é composta por 8 vértices Pretos. As soluções encontradas eram minimais em relação ao número de arcos, porém não havia restrições em relação ao número de vértices Cinzas ou Brancos presentes , o que resulta nesta enorme quantidade de soluções, uma vez que o hipergrafo é bastante denso.

3.2.2 Representação dos dados

Após aplicar o método *Totoro* à rede metabólica, obtém-se dentre as respostas, um arquivo que é a saída de um programa chamado *Clingo* (disponível em [4]), utilizado durante a execução do *Totoro*. Neste, existem todas as hiper-histórias representadas por meio de um conjunto de números inteiros $\mathbb{H}$ com as seguintes características. Sejam $h \in \mathbb{H}$ uma hiper-história e $id(r) \in \mathbb{Z}$ um identificador de uma reação r, então $id(r) \in h$ se, e somente se, a reação correspondente pertence à hiper-história metabólica h.

Este é um exemplo de uma hiper-história representada no arquivo de saída do *Clingo*:

Answer: 1
rsel(565) rsel(309) sel(84) sel(270) sel(456) sel(104)
Optimization: 6

3.2.3 Tratamento dos dados

O arquivo citado acima é um conjunto de linhas deste tipo, em que os identificadores que representam as reações são os números entre parênteses. Para extraí-los dos arquivos, foi feito uso da seguinte expressão regular (*Regex*): '\d+'. Armazena-se o resultado em vetores de características binários, ou seja, compostos por $0's$(zeros) e $1's$ (uns). Os quais seguem a seguinte regra: sejam h uma hiper-história metabólica encontrada pelo algoritmo *Totoro* e v o vetor de características que representa esta hiper-história. Se há n reações na rede metabólica, o vetor possui n posições, cada uma correspondendo a uma reação. Deste modo, se uma reação r está presente em h, então $v[id(r)] = 1$, onde $id(r)$ é um número inteiro como definido acima. Caso contrário, $v[id(r)] = 0$.

Então o conjunto de hiper-histórias $\mathbb{H}$ é representado por uma matriz em que cada linha corresponde a uma hiper-história h e as colunas são os identificadores das reações presentes na rede metabólica utilizada.

Como os índices das reações não são contíguos, são obtidas matrizes esparsas, com algumas colunas compostas por zeros. Deste modo, optou-se por processar esta matriz, removendo colunas compostas somente por zeros ou por uns. Isto já ajuda na redução de dimensionalidade do problema.

3.2.4 Aplicação dos algoritmos de *Clustering*

Para escolher o número de partes da partição que o *K-Means*, o *Mini-Batch K-Means* e o *Spectral Clustering* irão realizar no conjunto de dados, foi utilizada a heurística de Silhueta, descrita na seção 2.4.1. Com esse número é aplicado então o algoritmo de *clustering* correspondente e obtém-se um vetor com o identificador (um número inteiro) do subconjunto da partição ao qual cada ponto do

conjunto de dados pertence.

Para o *Affinity Propagation*, o algoritmo é executado e este seleciona o número de *clusters*, uma vez que consegue ter uma seleção automática de modelo como explicado na seção 2.3.3. Obtém-se também um vetor, como descrito acima.

Também é possível escolher reduzir a dimensão da matriz de características e utilizar esta nova matriz para servir de entrada a estes algoritmos.

3.2.5 Rearranjo das linhas das matrizes

Com o resultado da aplicação de um algoritmo de *clustering*, cada hiper-história (ou linha) da matriz de dados recebe um rótulo que é o número da parte da partição encontrada pelo algoritmo a que a hiper-história pertence. Assim para deixar as hiper-histórias pertencentes a um mesmo *cluster* organizadas de forma contígua na nossa matriz de dados, rearranja-se as linhas da matriz de acordo com seus rótulos.

Deste modo, as primeiras linhas da matriz de dados contém as hiper-histórias que estão no *cluster* de número 0, as seguintes as que estão no de número 1, e assim por diante. Optou-se por fazer isto, porque assim facilita a visualização de como ficou a partição e também sua manipulação.

3.2.6 Geração do *heatmap* para visualização

Após o rearranjo das linhas da matriz de dados, é interessante visualizar o resultado obtido. Para tal, gera-se um *heatmap* binário utilizando a ferramenta de uma biblioteca do Python chamada *seaborn* [21].

As linhas da imagem gerada são as hiper-histórias, as colunas as

reações da rede metabólica, assim como na matriz de dados. Já as linhas vermelhas são as divisões entre cada *cluster* encontrado. Como pode ser visto no exemplo na figura 3.3.

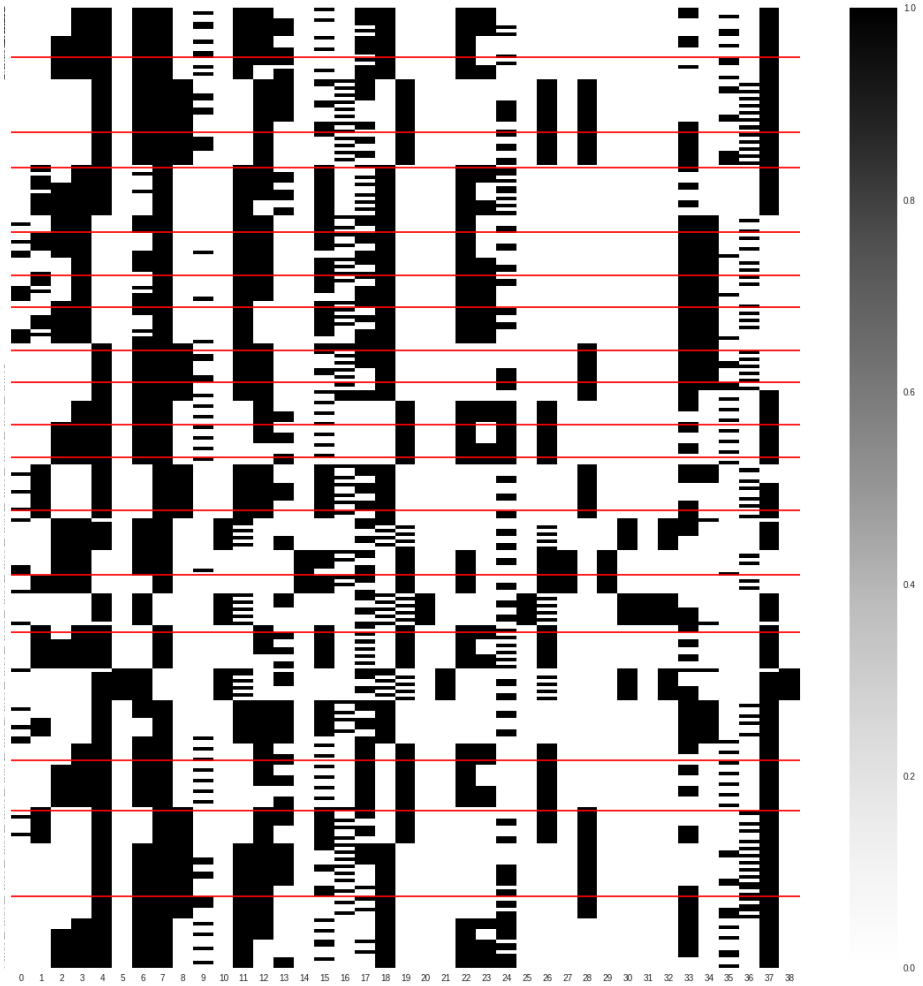


Figura 3.3: Exemplo de *Heatmap*, em que as colunas são reações e as linhas hiper-histórias. As 16 linhas vermelhas horizontais são as separações dos *clusters*.

Capítulo 4

Resultados

Os experimentos descritos no capítulo anterior foram realizados e os resultados são discutidos neste capítulo.

4.1 Resultados para o conjunto de dados com 268 hiper-histórias

A matriz binária de características tem dimensões: 268 linhas e 582 colunas. Após a aplicação da etapa em que remove-se as colunas compostas por apenas ou 1s ou 0s, as dimensões da matriz são: 268 linhas e 39 colunas.

Para a aplicação do método de silhueta para a seleção do k, número de *clusters*, dos algoritmos *K-Means, Mini-Batch K-Means e Spectral Clustering*, foram testados $k \in [2, 20]$. Foram realizados testes com números maiores, mas o *score* do método piorava progressivamente, então esse intervelo de números fez mais sentido.

Devido ao fato dos dados terem esta dimensão grande, optou-se por diminuir sua dimensão utilizando PCA antes de aplicar os algoritmos de particionamento, como explicado na seção 2.5.

Para o *Mini-Batch K-Means*, foi escolhido um *batch* com tamanho 100. As próximas seções apresentam e discutem os diferentes resultados obtidos a partir dos experimentos realizados.

4.1.1 *K-means*

As tabelas 4.1 e 4.2 apresentam o tempo de execução das etapas do *pipeline* para o particionamento do conjunto de dados, acima descrito. Pode-se ver como o resultado com PCA obtém melhor *score* de Silhueta do que o sem PCA.

Sem PCA

Etapa	Tempo (em s)
Ler arquivo com hiper-história e montar matriz binária	0.008222
Tratar matriz binária para retirar colunas com só 1s ou só 0s	0.000751
Estimar k e aplicar algoritmo de *clustering*	0.571311
Rearranjar as linhas da matriz	0.000111
Gerar e salvar *heatmap*	2.150673

Tabela 4.1: Tabela mostrando o tempo gasto para execução de cada etapa do *pipeline* para o *K-Means* sem PCA

Score obtido pelo método da silhueta: 0.660031537157 e o k escolhido foi 2. O *heatmap* gerado pode ser visto na figura 4.1.

Com PCA

Etapa	Tempo (em s)
Ler arquivo com hiper-história e montar matriz binária	0.008188
Tratar matriz binária para retirar colunas com só 1s ou só 0s	0.000778
Estimar k e aplicar algoritmo de *clustering*	0.394036
Rearranjar as linhas da matriz	0.000158
Gerar e salvar *heatmap*	2.199069

Tabela 4.2: Tabela mostrando o tempo gasto para execução de cada etapa do *pipeline* para o *K-Means* com PCA

Score obtido pelo método da silhueta: 0.811081936065 e o k escolhido foi 9. O *heatmap* gerado pode ser visto na figura 4.2.

4.1.2 *Mini-Batch K-Means*

Nas tabelas 4.3 e 4.4, são apresentados os tempos de execução dos passos para executar o *Mini-Batch K-Means* com e sem redução de dimensionalidade por PCA. Assim, como no caso do *K-Means*, quando utiliza-se PCA para pré-processar a matriz de dados, obtém-se um melhor *score* de silhueta.

Sem PCA

Etapa	Tempo (em s)
Ler arquivo com hiper-história e montar matriz binária	0.008861
Tratar matriz binária para retirar colunas com só 1s ou só 0s	0.001295
Estimar k e aplicar algoritmo de *clustering*	0.406360
Rearranjar as linhas da matriz	0.000844
Gerar e salvar *heatmap*	2.170488

Tabela 4.3: Tabela mostrando o tempo gasto para execução de cada etapa do *pipeline* para o *Mini-Batch K-Means* sem PCA

Score obtido pelo método silhueta: 0.581661648841 e o k escolhido foi 2. O *heatmap* gerado pode ser visto na figura 4.3.

Com PCA

Etapa	Tempo (em s)
Ler arquivo com hiper-história e montar matriz binária	0.008096
Tratar matriz binária para retirar colunas com só 1s ou só 0s	0.000751
Estimar k e aplicar algoritmo de *clustering*	0.313257
Rearranjar as linhas da matriz	0.000195
Gerar e salvar *heatmap*	2.215912

Tabela 4.4: Tabela mostrando o tempo gasto para execução de cada etapa do *pipeline* para o *Mini-Batch K-Means* com PCA

Score obtido pelo método silhueta: 0.811081936065 e o k escolhido foi 9. O *heatmap* gerado pode ser visto na figura 4.4.

4.1.3 *Affinity Propagation*

A seguir são apresentadas as tabelas 4.5 e 4.6 contendo o tempo de execução para os passos do *pipeline* apresentado no capítulo anterior para a aplicação do algoritmo *Affinity Propagation* sem e com PCA. Novamente, quando aplica-se PCA antes do particionamento, obtém-se melhores resultados, segundo a heurística apresentada na seção 2.4.1.

Sem PCA

Etapa	Tempo (em s)
Ler arquivo com hiper-história e montar matriz binária	0.008248
Tratar matriz binária para retirar colunas com só 1s ou só 0s	0.000777
Aplicar algoritmo de *clustering*	0.161922
Rearranjar as linhas da matriz	0.000301
Gerar e salvar *heatmap*	2.222236

Tabela 4.5: Tabela mostrando o tempo gasto para execução de cada etapa do *pipeline* para o *Affinity Propagation* sem PCA

Score obtido pelo método silhueta: 0.186079422718 e o k determinado pelo algoritmo foi 21. O *heatmap* gerado pode ser visto na figura 4.5.

Com PCA

Etapa	Tempo (em s)
Ler arquivo com hiper-história e montar matriz binária	0.008268
Tratar matriz binária para retirar colunas com só 1s ou só 0s	0.000881
Aplicar algoritmo de *clustering*	0.133393
Rearranjar as linhas da matriz	0.000198
Gerar e salvar *heatmap*	2.162453

Tabela 4.6: Tabela mostrando o tempo gasto para execução de cada etapa do *pipeline* para o *Affinity Propagation* com PCA

Score obtido pelo método silhueta: 0.495615598764 e o k determinado pelo algoritmo foi 7. O *heatmap* gerado pode ser visto na figura 4.6.

4.1.4 *Spectral Clustering*

As tabelas 4.7 e 4.8 contém os tempos de execução do passos propostos no capítulo anterior para o particionamento do conjunto de dados sem e com pré-processamento do conjunto de dados por PCA, respectivamente. Notou-se que o desempenho deste último foi melhor de acordo com o *score* de silhueta.

Sem PCA

Etapa	Tempo (em s)
Ler arquivo com hiper-história e montar matriz binária	0.008519
Tratar matriz binária para retirar colunas com só 1s ou só 0s	0.000794
Estimar k e aplicar algoritmo de *clustering*	0.615308
Rearranjar as linhas da matriz	0.000169
Gerar e salvar *heatmap*	2.204395

Tabela 4.7: Tabela mostrando o tempo gasto para execução de cada etapa do *pipeline* para o *Spectral Clustering* sem PCA

Score obtido pelo método da silhueta: 0.396329121083 e o k escolhido foi 9. O *heatmap* gerado pode ser visto na figura 4.7.

Com PCA

Etapa	Tempo (em s)
Ler arquivo com hiper-história e montar matriz binária	0.008108
Tratar matriz binária para retirar colunas com só 1s ou só 0s	0.000751
Estimar k e aplicar algoritmo de *clustering*	0.624451
Rearranjar as linhas da matriz	0.000134
Gerar e salvar *heatmap*	2.189706

Tabela 4.8: Tabela mostrando o tempo gasto para execução de cada etapa do *pipeline* para o *Spectral Clustering* com PCA

Score obtido pelo método da silhueta: 0.579555305413 e o k escolhido foi 7. O *heatmap* gerado pode ser visto na figura 4.8.

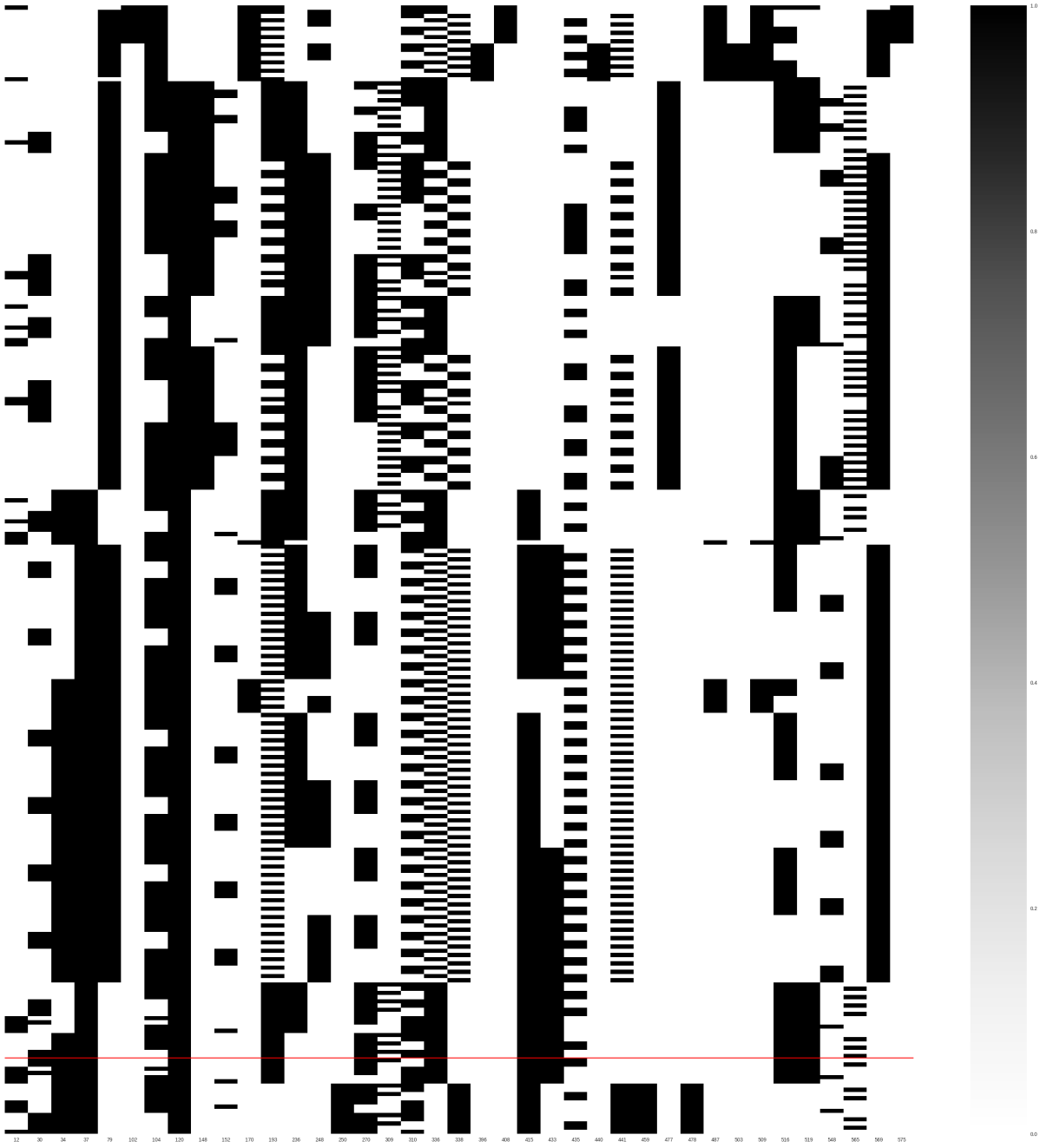


Figura 4.1: *Heatmap K-Means* sem redução de dimensionalidade por PCA, k = 2

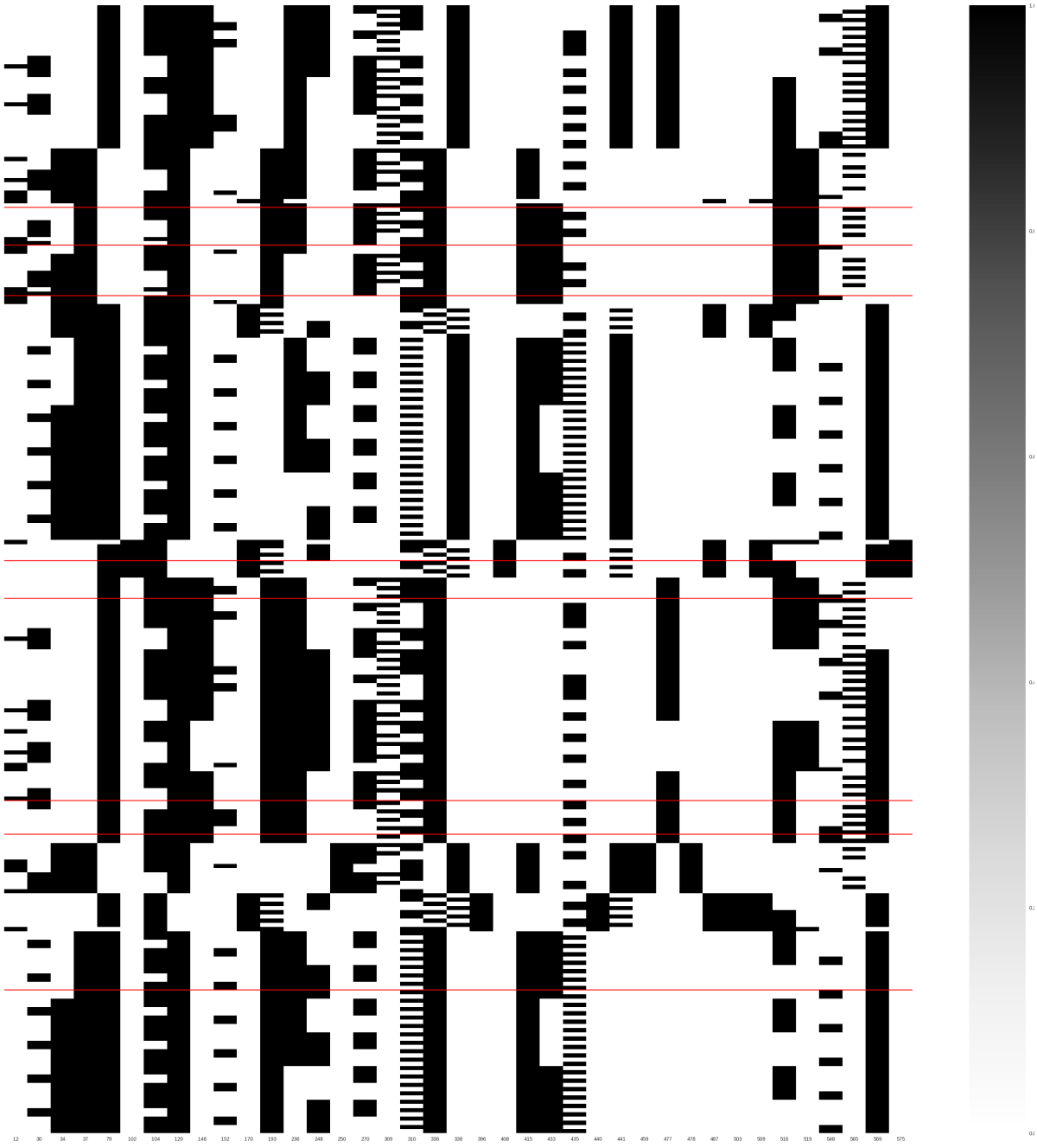


Figura 4.2: *Heatmap K-Means* com redução de dimensionalidade por PCA (n = 5), k = 9

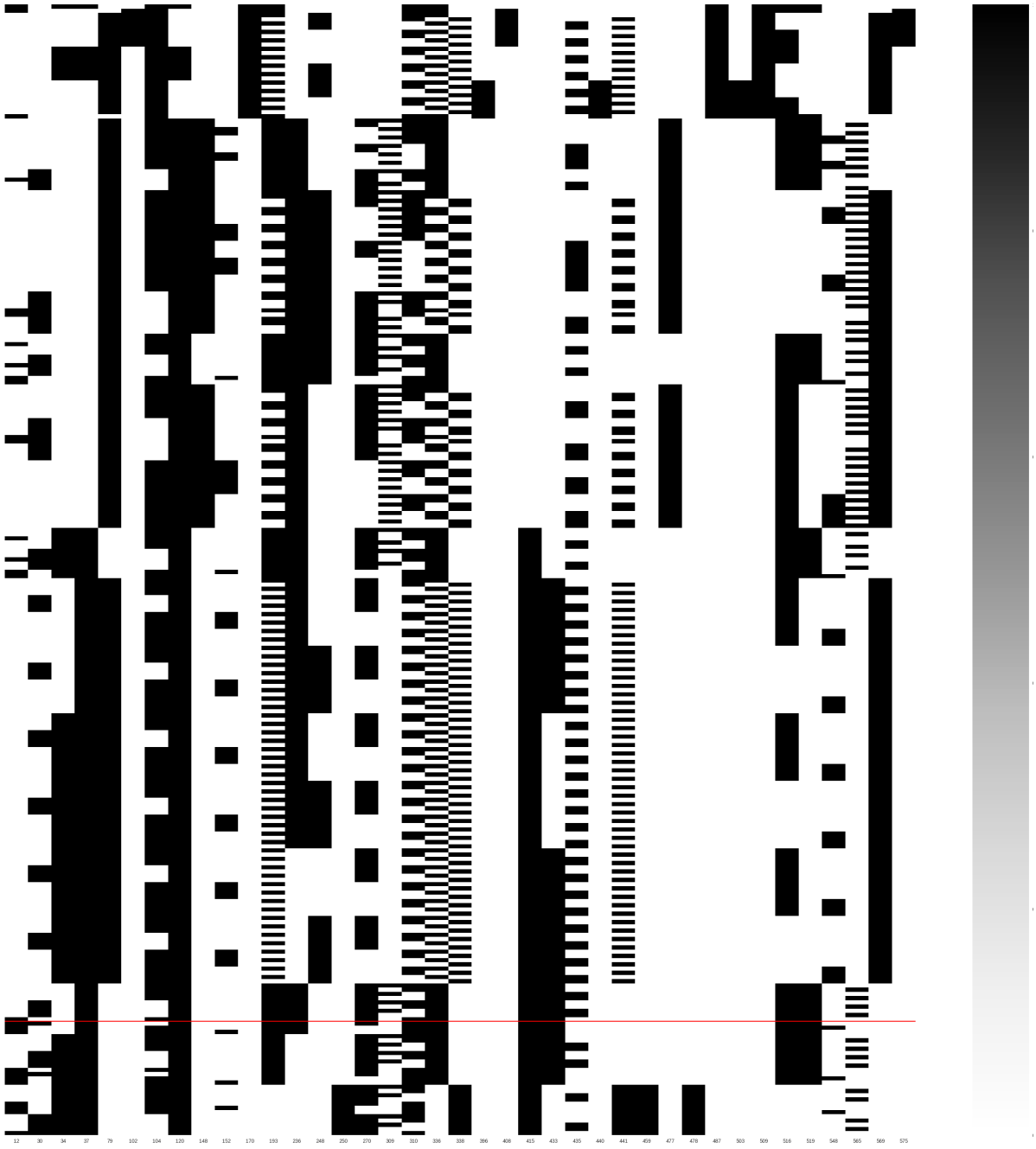


Figura 4.3: *Heatmap Mini-Batch K-Means* sem redução de dimensionalidade por PCA, k = 2

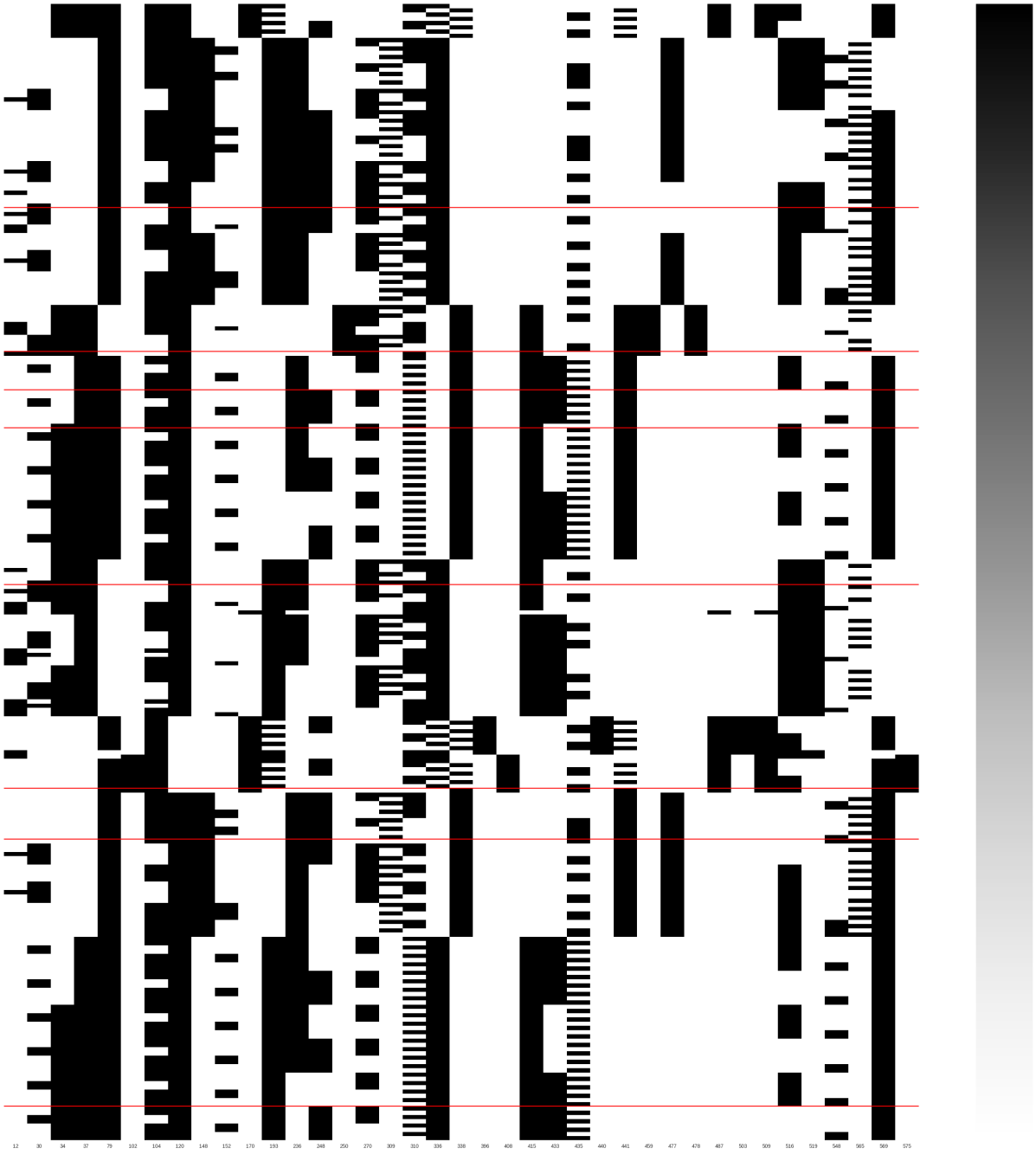


Figura 4.4: *Heatmap Mini-Batch K-Means* com redução de dimensionalidade por PCA (n = 5), k = 9

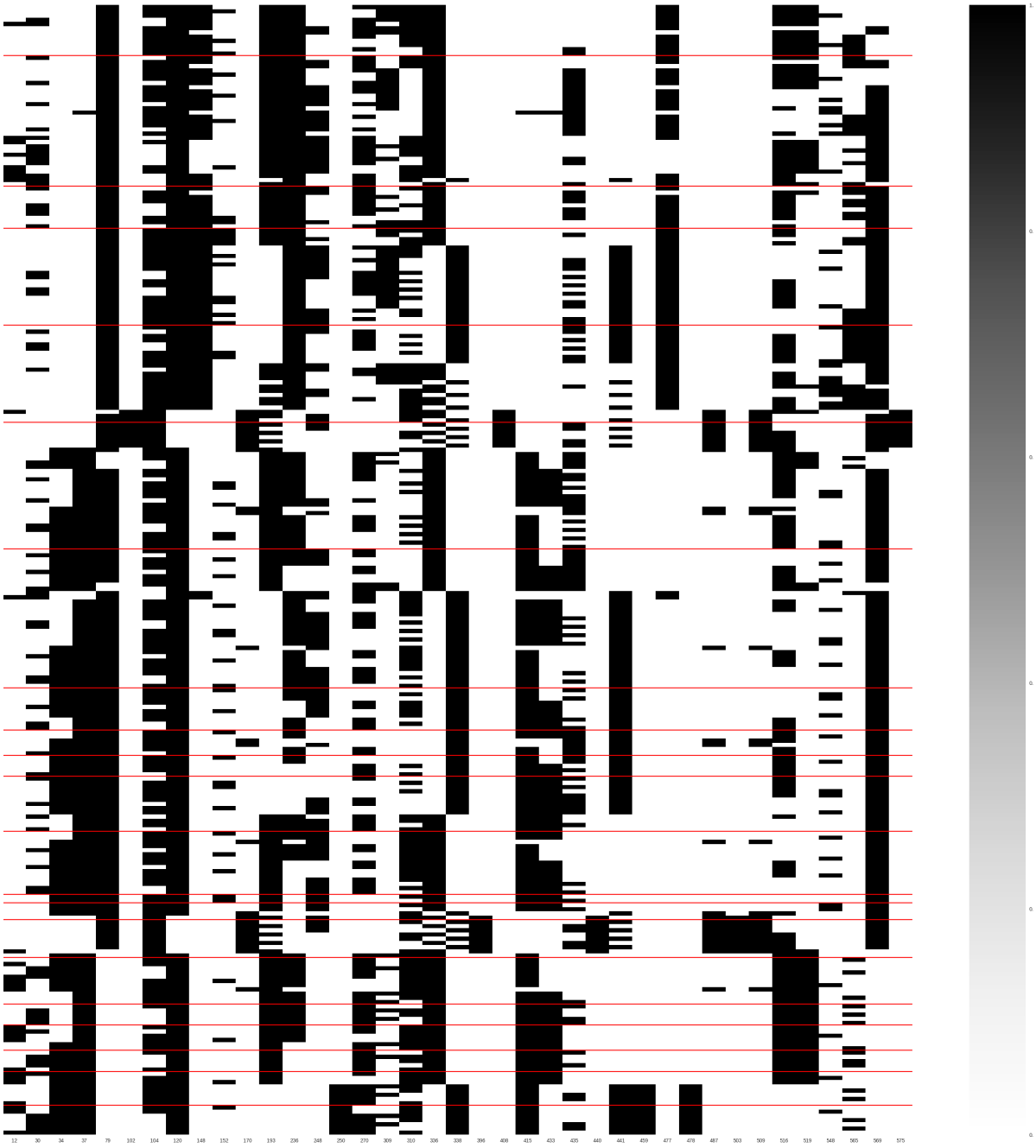


Figura 4.5: *Heatmap Affinity Propagation* sem redução de dimensionalidade por PCA, k = 21

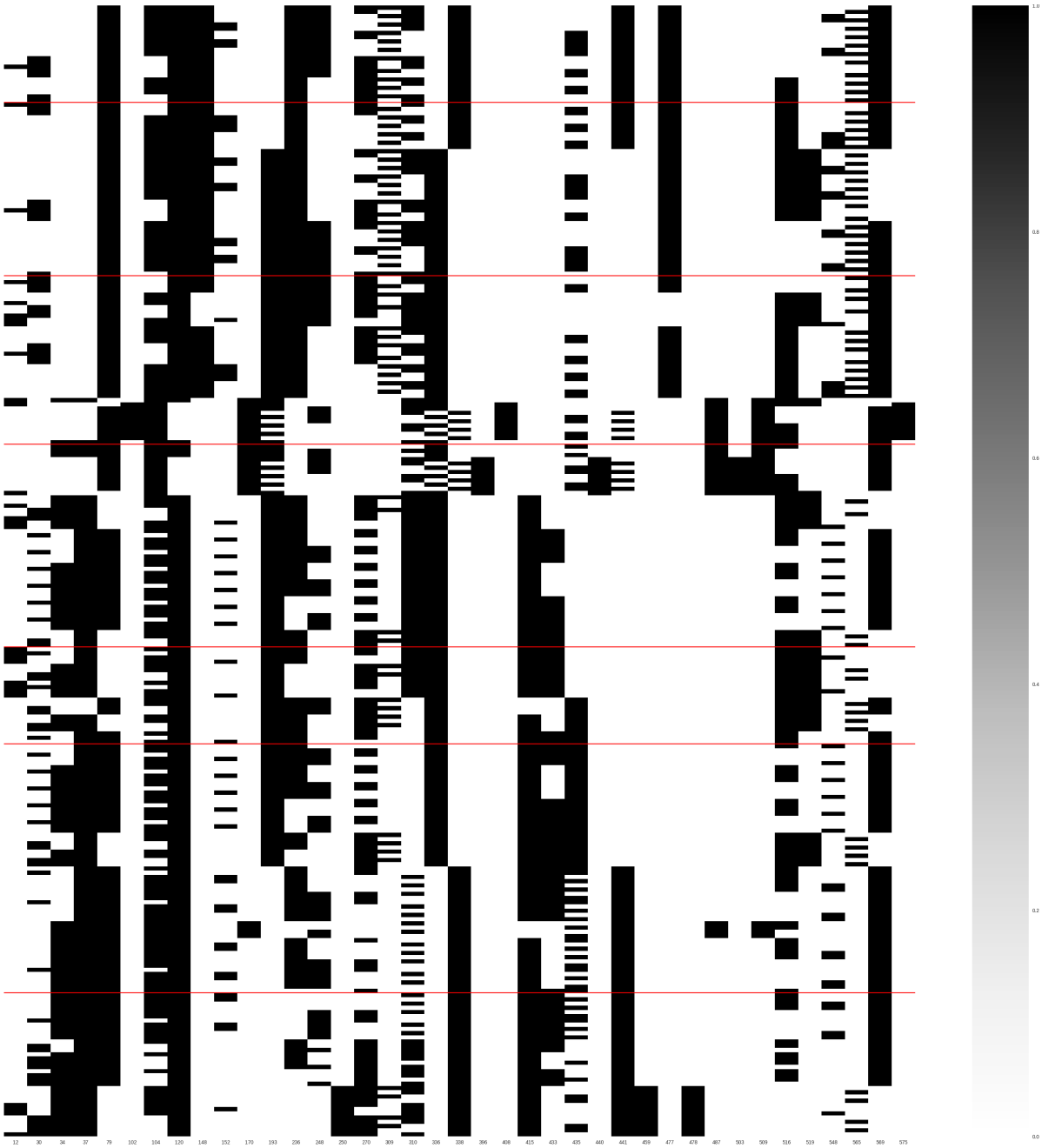


Figura 4.6: *Heatmap Affinity Propagation* com redução de dimensionalidade por PCA (n = 5), k = 7

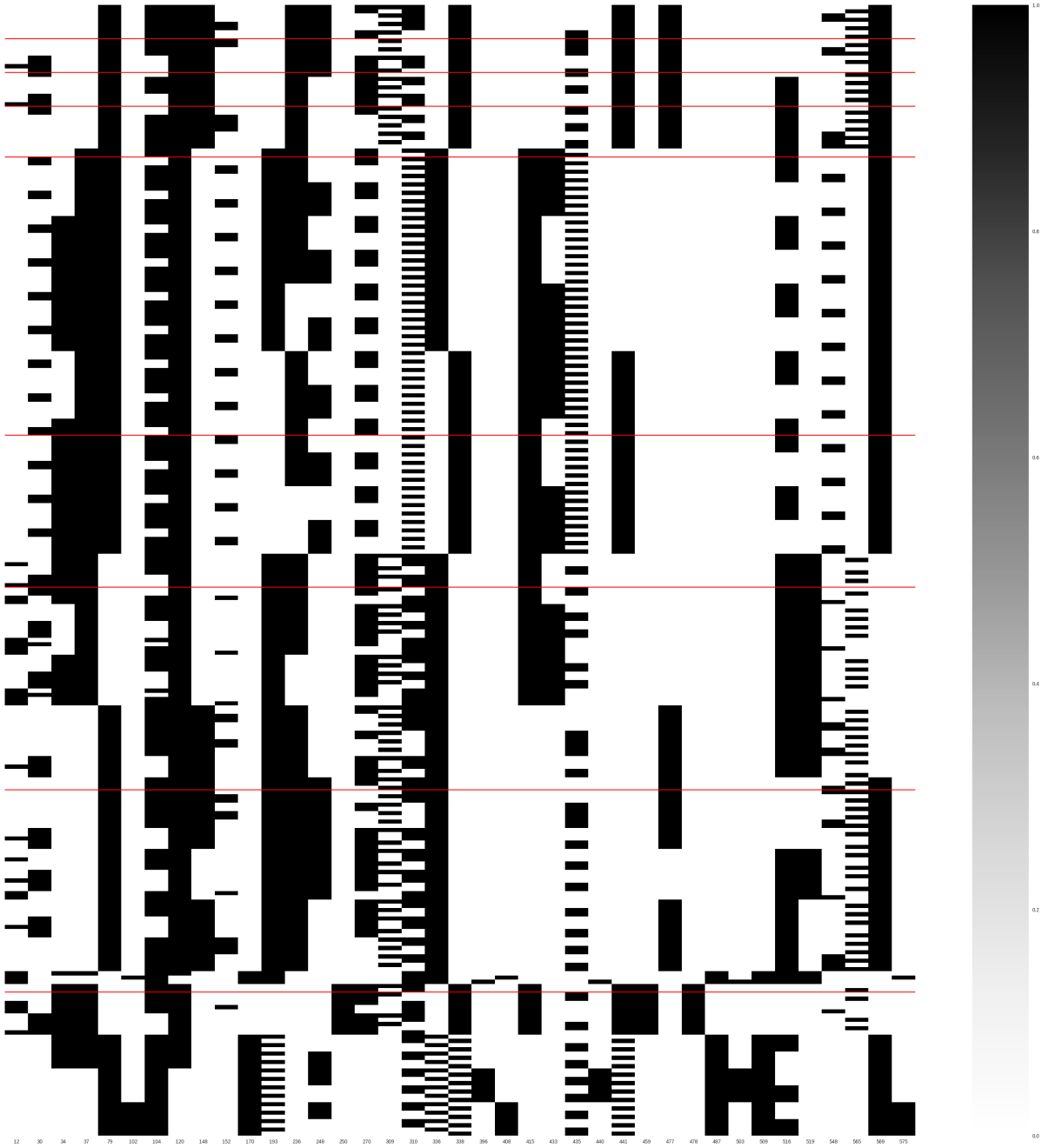


Figura 4.7: *Heatmap Spectral Clustering* sem redução de dimensionalidade por PCA, k = 9

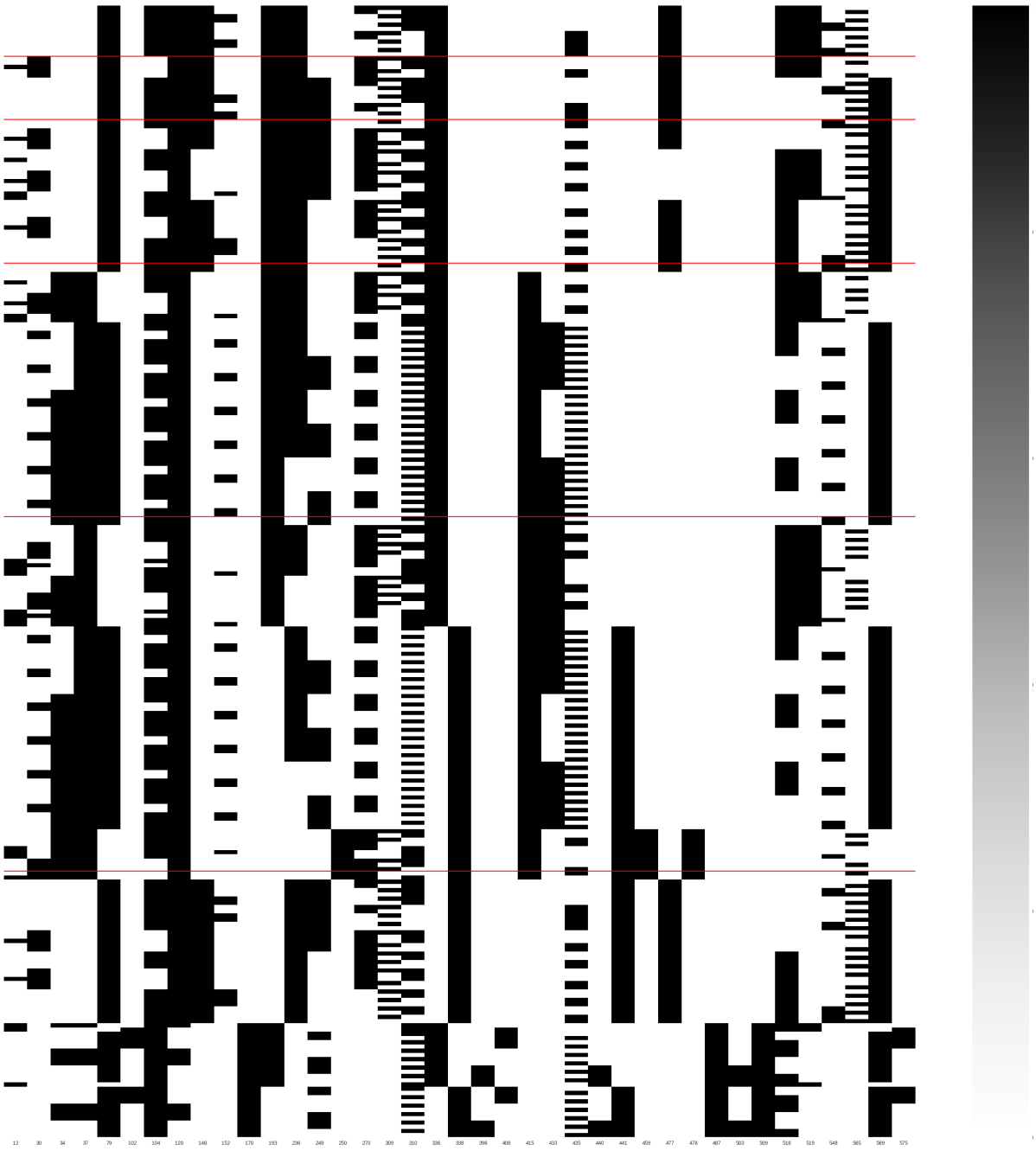


Figura 4.8: *Heatmap Spectral Clustering* com redução de dimensionalidade por PCA (n = 5), k = 6

Análise dos resultados

Alguns resultados confirmaram empiricamente algumas suposições feitas nas seções dos capítulos acima. O tempo de execução do *Mini-Batch K-Means* é menor do que o do *K-Means* para o mesmo conjunto de dados, tendo *score* do método de silhueta bastante semelhante. Além disso, para todos os métodos de *clustering*, após a aplicação da redução de dimensionalidade por PCA ao conjunto de dados, de fato, obteve-se melhores valores de *score* do método de silhueta.

Ademais, foi obtido um resultado interessante, em três algoritmos: *K-Means* sem PCA, *Mini-Batch K-Means* sem PCA e *Affinity Propagation* com PCA. Foi possível recuperar partições em que cada um dos seguintes conjuntos de reações: [102, 408, 575], [250, 459, 478], [396, 440, 503], estavam contidos completamente em um *cluster* da partição. As reações que estes identificadores representam podem ser vistas no Apêndice A.

Conjunto	*K-Means*	*Mini-Batch K-Means*	*Affinity propagation com PCA*
1º	*cluster* 1	*cluster* 1	*cluster* 3
2º	*cluster* 2	*cluster* 2	*cluster* 7
3º	*cluster* 1	*cluster* 1	*cluster* 4

Tabela 4.9: Tabela que apresenta o *cluster* em que está localizado cada conjunto de reações. Em que 1° indica representa o conjunto [102, 408, 575]; 2°, o conjunto [250, 459, 478] e 3° o conjunto [369, 440, 503].

Na tabela 4.9, foi indicado o *cluster* ao qual cada um dos conjuntos de reação (como descrito acima) foi atribuído após a execução destes algoritmos de particionamento. Cada linha representa a localização de [102, 408, 575], [250, 459, 478] e [396, 440, 503], respectivamente,

como é possível ver nas figuras 4.1, 4.3 e 4.6.

Observando a figura 3.2, é possível notar que a principal via de reações que explica os dados de metabolômica utilizados neste trabalho é o da glutationa. Mas, ao identificar-se estes outros subconjuntos, recupera-se informações sobre outros caminhos importantes relacionados à desintoxicação da levedura *S. cerevisiae* em relação à Cádmio(Cd^{2+}).

O primeiro subconjunto de reações está relacionado ao metabolismo do nicotinato, o segundo ao metabolismo do aminoácido purina e o terceiro ao PPP (*Pentose Phosphate Pathway*). Sabe-se que tanto o metabolismo da purina como o do nicotinato/nicotinamida [31, 6] estão ligados à resposta ao estresse oxidativo. Isso mostra que o *pipeline* desenvolvido pode ajudar a compreender os resultados obtidos pelo *Totoro*, identificando grupos de hiper-histórias metabólicas ligadas à diferentes funções biológicas que podem explicar o fenômeno estudado.

4.2 Resultados para o conjunto de dados com 56 515 391 hiper-histórias

Foram realizados experimentos com o *Mini-Batch K-Means*, uma vez que este é mais recomendado para este conjunto grande de dados. O k escolhido para teste foi 50 e o tamanho da *batch* foi de 10000 amostras. O tamanho da matriz binária de características é: 56515391 linhas e 587 colunas. O tamanho da matriz binária de características após a remoção de colunas compostas ou só por 1s ou só por 0s é: 56515391 linhas e 232 colunas.

Sem PCA

Etapa	Tempo (em s)
Ler arquivo com hiper-história e montar matriz binária	2633.254752
Tratar matriz binária para retirar colunas com só 1s ou só 0s	277.014309
Aplicar algoritmo *Mini-Batch K-Means*	372.798458
Prever os *labels*	624.327012
Rearranjar as linhas da matriz	63.135010

Tabela 4.10: Tabela mostrando o tempo gasto para execução de cada passo para calcular a partição do conjunto de dados

Dificuldades

Algumas dificuldades são inerentes à manipulação e uso de dados grandes. O tempo e a quantidade de memória necessárias para a execução dos algoritmos é um grande limitante para este tipo de dado. Diante disso, tivemos dificuldades para executar outros métodos de *clustering* para estes dados. Além disso, não foi possível gerar o *heatmap*, calcular o *score* de silhueta obtido pelos resultados mostrados acima ou mesmo aplicar PCA à matriz de dados. Estes implicam na dificuldade da visualização dos resultados e estimação do k. Por isso, tivemos que escolher um número para k que achamos razoável e testar.

Análise dos resultados

Devido ao tamanho do conjunto de dados e dificuldades acima descritas, a análise dos resultados ainda não foi concluída. Sugerimos isso como trabalhos futuros do TCC.

Capítulo 5

Conclusões

Estudar e compreender o metabolismo dos organismos sempre foi uma tarefa essencial para se entender como os diversos seres vivos funcionam e se comportam, principalmente em situações não-fisiológicas e de estresse.

Diante disso, têm-se desenvolvido, cada vez mais, métodos para extrair informações metabólicas de dados disponíveis. Dentre eles, temos o conceito de hiper-história metabólica, que busca encontrar explicações para o fluxo de matéria após algum evento ocorrido dentre dois estados basais, modelando os dados dos metabólitos e reações por meio de perfis e redes metabólicas.

Entretanto, por vezes têm-se que o número de hiper-histórias encontrado é muito grande. Como é o caso de quando aplica-se este algoritmo ao conjunto de dados da exposição da levedura *S. cerevisiae* ao Cádmio (Cd^{2+}). Assim, é importante para a análise destas hiper-histórias que alguma técnica de particionamento seja aplicada, de modo que possamos extrair informações relevantes dos pontos de vista químico e biológico.

Para tal tarefa, foram aplicados quatro tipos de algoritmos de *clustering –K-Means, Mini-Bach K-Means, Affinity Propagation e*

Spectral Clustering. Com os três primeiros foi possível agrupar reações de diferentes vias metabólicas que estão associadas à resposta ao estresse oxidativo. Isso indica que estes algoritmos podem ser utilizados como ferramenta para a classificação de hiper-histórias metabólicas auxiliando a interpretação de resultados do método *Totoro*.

Em relação ao futuro, seria importante tentar interpretar os resultados obtidos com o conjunto maior de dados, e tentar aplicar técnicas mais eficientes de manipulação, *clustering* e visualização, do ponto de vista de memória e tempo, a este tipo de dado.

Apêndice A

Nome da reações por número no *Heatmap*

12. R15__45__RXN
30. ACETYLHOMOSER__45__CYS__45__RXN
34. NAD+ pyrophosphatase
37. Adenine phosphoribosyltransferase_REV
79. Methionine adenosyltransferase_REV
102. Nicotinate-nucleotide pyrophosphorylase (carboxylating)
104. RXN__45__721
120. RXN3O__45__4120
148. S-ADENOSYLMETHIONINE_IN_NIL_TO_ADENOSYL-HOMO-CYS_IN_NIL_COF_RXN
152. Glycine hydroxymethyltransferase
170. Tryptophan synthase
193. Arginase
236. Nicotinamide-nucleotide adenylyltransferase_REV
248. Spermine synthase_REV
250. AMP deaminase
270. Cystathionine beta-synthase
309. 5-methyltetrahydropteroyltriglutamate–homocysteine S-

methyltransferase
310. RXN__45__6601_BACK
336. Ornithine carbamoyltransferase
338. Argininosuccinate synthase
396. RXN3O__45__131
408. Nicotinate phosphoribosyltransferase
415. 5'-methylthioadenosine phosphorylase_REV
433. ATP_IN_NIL_TO_AMP_IN_NIL_COF_RXN
435. RXN3O__45__112
440. RXN__45__8441
441. Argininosuccinate lyase
459. Adenylosuccinate lyase
477. Adenosylhomocysteinase
478. Adenylosuccinate synthase
487. Ribose-phosphate pyrophosphokinase
503. Ribokinase
509. Transketolase
516. Spermidine synthase
519. Ornithine decarboxylase
548. L-serine dehydratase
565. Homocysteine S-methyltransferase
569. Adenosylmethionine decarboxylase_REV
575. COMPR_Nicotinamidase_Amidase

Bibliografia

[1] Vicente Acuña, Etienne Birmelé, Ludovic Cottret, Pierluigi Crescenzi, Fabien Jourdan, Vincent Lacroix, Alberto Marchetti-Spaccamela, Andrea Marino, Paulo Vieira Milreu, Marie-France Sagot, et al. Telling stories: Enumerating maximal directed acyclic graphs with a constrained set of sources and targets. *Theoretical Computer Science*, 457:1–9, 2012.

[2] Javier Béjar Alonso. K-means vs mini batch k-means: a comparison. 2013.

[3] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[4] clingo. Clingo. `https://potassco.org/clingo/`.

[5] Ludovic Cottret, David Wildridge, Florence Vinson, Michael P. Barrett, Hubert Charles, Marie-France Sagot, and Fabien Jourdan. Metexplore: a web server to link metabolomic experiments and genome-scale metabolic networks. *Nucleic Acids Research*, 38(suppl 2):W132–W137, 2010.

[6] CL Crowley, CM Payne, H Bernstein, C Bernstein, and Denise Roe. The nad+ precursors, nicotinic acid and nicotinamide protect cells against apoptosis induced by a multiple stress inducer, deoxycholate. *Cell death and differentiation*, 7(3):314, 2000.

[7] B Dash, Debahuti Mishra, A Rath, and Milu Acharya. A hybridized k-means clustering approach for high dimensional dataset. *International Journal of Engineering, Science and Technology*, 2(2):59–66, 2010.

[8] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

[9] Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. Directed hypergraphs and applications. *Discrete applied mathematics*, 42(2-3):177–201, 1993.

[10] André Luiz Pires Guedes and Lilian Markenzon. Directed hypergraph planarity. *Pesquisa Operacional*, 25(3):383–390, 2005.

[11] inria. Dinghy. `http://dinghy.gforge.inria.fr`.

[12] Alice Julien-Laferriere. Totoro: Topological analysis of transient metabolic response. Não publicada ainda, com previsão de publicação para o fim de 2016.

[13] Alice Julien-Laferrière. *Models and algorithms applied to metabolism: From revealing the responses to perturbations towards the design of microbial consortia* . Theses, Université Lyon 1 - Claude Bernard, December 2016.

[14] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):881–892, 2002.

[15] Geoffrey Madalinski, Emmanuel Godat, Sandra Alves, Denis Lesage, Eric Genin, Philippe Levi, Jean Labarre, Jean-Claude Tabet, Eric Ezan, , and Christophe Junot. Direct introduction of biological samples into a ltq-orbitrap hybrid mass spectrometer as a tool for fast metabolome analysis. *Analytical Chemistry*, 80(9):3291–3303, 2008. PMID: 18351782.

[16] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is np-hard. In *International Workshop on Algorithms and Computation*, pages 274–285. Springer, 2009.

[17] Paulo Vieira Milreu, Cecilia Coimbra Klein, Ludovic Cottret, Vicente Acuña, Etienne Birmelé, Michele Borassi, Christophe Junot, Alberto Marchetti-Spaccamela, Andrea Marino, Leen Stougie, Fabien Jourdan, Pierluigi Crescenzi, Vincent Lacroix, and Marie-France Sagot. Telling metabolic stories to explore metabolomics data: a case study on the yeast response to cadmium exposure. *Bioinformatics*, 30(1):61–70, 2014.

[18] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[19] Stephen G Oliver, Michael K Winson, Douglas B Kell, and Frank Baganz. Systematic functional analysis of the yeast genome. *Trends in biotechnology*, 16(9):373–378, 1998.

[20] Simon Ostergaard, Lisbeth Olsson, and Jens Nielsen. Metabolic engineering of saccharomyces cerevisiae. *Microbiology and molecular biology reviews*, 64(1):34–50, 2000.

[21] pydata. Seaborn. `https://seaborn.pydata.org/`.

[22] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[23] scikit learn. Affinity propagation scikit-learn. `http://scikit-learn.org/stable/modules/clustering.html#affinity-propagation`.

[24] scikit learn. K-means scikit-learn. `http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`.

[25] scikit learn. Spectral clustering scikit-learn. `http://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html`.

[26] David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178. ACM, 2010.

[27] Karin Vido, Daniel Spector, Gilles Lagniel, Sébastien Lopez, Michel B Toledano, and Jean Labarre. A proteome analysis of the cadmium response in saccharomyces cerevisiae. *Journal of Biological Chemistry*, 276(11):8469–8474, 2001.

[28] Donald Voet, Judith G Pratt Voet, W Charlotte, G Voet Judith, and W Pratt Charlotte. *Fundamentals of biochemistry: life at the molecular level*. Number 577.1 VOE. 2013.

[29] Kaijun Wang, Junying Zhang, Dan Li, Xinna Zhang, and Tao Guo. Adaptive affinity propagation clustering. *arXiv preprint arXiv:0805.1096*, 2008.

[30] J. Xia and D. S. Wishart. MSEA: a web-based tool to identify biologically meaningful patterns in quantitative metabolomic data. *Nucleic Acids Research*, 38(Web Server):W71–W77, may 2010.

[31] Qixiao Zhai, Yue Xiao, Jianxin Zhao, Fengwei Tian, Hao Zhang, Arjan Narbad, and Wei Chen. Identification of key proteins and pathways in cadmium tolerance of lactobacillus plantarum strains by proteomic analysis. *Scientific Reports*, 7(1):1182, 2017.