UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

AGRUPAMENTO DE SEQUÊNCIAS DE MIRNA UTILIZANDO APRENDIZADO NÃO-SUPERVISIONADO BASEADO EM GRAFOS

VIVIANI AKEMI KASAHARA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Inteligência Artificial.
Orientadora: Drª. Maria do Carmo Nicoletti

São Carlos - SP
Agosto/2016

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária UFSCar
Processamento Técnico
com os dados fornecidos pelo(a) autor(a)

Kasahara, Viviani Akemi
K19a Agrupamento de sequências de miRNA utilizando aprendizado não-supervisionado baseado em grafos / Viviani Akemi Kasahara. -- São Carlos : UFSCar, 2016.
152 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2016.

1. Agrupamento de miRNA. 2. Mineração de dados. 3. Teoria dos grafos. 4. Algoritmo espectral. 5. Aprendizado não-supervisionado. I. Título.

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a defesa de Dissertação de Mestrado da candidata Viviani Akemi Kasahara, realizada em 02/08/2016.

Profª Drª. Maria do Carmo Nicoletti
(UFSCar)

Prof. Dr. Ricardo José Ferrari
(UFSCar)

Prof. Dr. José Augusto Baranauskas
(DCM - FFCLRP - USP)

Certifico que a sessão de defesa foi realizada com a participação à distância do membro Prof. Dr. José Augusto Baranauskas. Depois das arguições e deliberações realizadas, o participante à distância está de acordo com o conteúdo do parecer da comissão examinadora redigido no relatório de defesa da aluna Viviani Akemi Kasahara.

Profª Drª. Maria do Carmo Nicoletti
Coordenadora da Comissão Examinadora
(UFSCar)

AGRADECIMENTO

Meus agradecimentos a todos que colaboraram para que este trabalho se tornasse real e completo. Aos professores e colegas do Departamento de Computação que puderam me ensinar e compartilhar novos conhecimentos. Aos professores que fizeram parte da minha banca examinadora e colaboraram com suas sugestões e críticas construtivas.

Ao apoio oferecido por minha família para que eu pudesse continuar com minha dupla jornada entre o meu trabalho em São Paulo e o curso de mestrado em São Carlos.

À Profª. Drª. Maria do Carmo Nicoletti por toda dedicação, orientação e ajuda oferecida em todos esses anos de convívio e compartilhamento de ideias. Isso foi essencial para que eu pudesse começar e evoluir com este trabalho.

RESUMO

A análise de agrupamento é uma organização de coleção de padrões em grupos, baseando-se na similaridade das propriedades pertencentes aos dados. A técnica de agrupamento pode ser utilizado em muitas áreas de conhecimento como biotecnologia, visão computacional, recuperação de documentos, entre outras. Uma área interessante da biologia envolve o conceito de microRNAs (miRNAs), que são moléculas não-codificadas de RNA com aproximadamente 22 nucleotídeos e que desempenham um papel importante na regulação dos genes. O agrupamento de sequências de miRNA podem ajudar em sua exploração e entendimento, pois as sequências que pertencem ao mesmo grupo possuem uma função biológica similar. Esse trabalho explora e investiga sete algoritmos de agrupamentos não-supervisionados baseados em grafos que podem ser divididos em três categorias: algoritmos baseados em região de influência, algoritmos baseados em árvore *spanning* minimal e algoritmo espectral. Para avaliar a contribuição dos algoritmos propostos, os experimentos conduzidos utilizaram os dados das famílias de miRNAs disponíveis no banco de dados denominado miRBase. Os resultados dos experimentos foram apresentados, analisados e avaliados usando índices de validação de agrupamento e análise visual.

Palavras-chave: agrupamento, agrupamento de miRNA, mineração de dados, aprendizado não-supervisionado, teoria dos grafos, agrupamento baseado em grafos, algoritmo baseados em região de influência, algoritmos baseados em árvore *spanning* minimal, algoritmo espectral, miRNA.

ABSTRACT

Cluster analysis is the organization of a collection of patterns into clusters based on similarity which is determined by using properties of data. Clustering techniques can be useful in a variety of knowledge domains such as biotechnology, computer vision, document retrieval and many others. An interesting area of biology involves the concept of microRNAs (miRNAs) that are approximately 22 nucleotide-long non-coding RNA molecules that play important roles in gene regulation. Clustering miRNA sequences can help to understand and explore sequences belonging to the same cluster that has similar biological functions. This research work investigates and explores seven unsupervised clustering algorithms based on graphs that can be divided into three categories: algorithm based on region of influence, algorithm based on minimum spanning tree and spectral algorithm. To assess the contribution of the proposed algorithms, data from miRNA families stored in the online miRBase database were used in the conducted experiments. The results of these experiments were presented, analysed and evaluated using clustering validation indexes as well as visual analysis.

Keywords: *clustering, miRNA clustering, data mining, unsupervised learning, graph theory, graph based clustering algorithms, algorithm based on region of influence, algorithm based on minimum spanning tree, spectral algorithm, miRNA.*

LISTA DE FIGURAS

Lista de Tabelas

Sumário

Capítulo 1

Introdução

1.1 Contextualização

O estudo e a pesquisa envolvendo métodos e técnicas que viabilizam o aprendizado automático por computadores é uma das caracterizações mais populares da área de Aprendizado de Máquina (AM), inserida na grande área de Inteligência Computacional.

A pesquisa em AM acontece em duas frentes principais: a primeira é a da investigação teórica e criação de algoritmos, que permite a formalização e o estabelecimento de resultados gerais, especificando condições necessárias para o aprendizado acontecer, definindo seus limites de atuação e evidenciando as restrições que devem ser impostas com vistas aos resultados que se quer obter. A segunda frente de pesquisa se dá por meio do uso dos vários modelos de aprendizado automático, implementados como sistemas automatizados, nos quais os mais variados domínios de conhecimento são utilizados nas mais diversas tarefas de aprendizado.

Os chamados algoritmos de agrupamento são uma das possíveis maneiras de viabilizar o denominado aprendizado não supervisionado. Dentre os muitos algoritmos de agrupamentos propostos na literatura, aqueles caracterizados como baseados em grafo foram o foco principal deste trabalho de pesquisa.

1.2 Motivação e Objetivo

O trabalho teve por principal motivação levantar e explorar um grupo de algoritmos de agrupamento, no que diz respeito às suas particularidades e contribuições. O objetivo foi investigar a colaboração de algoritmos de agrupamento baseados em grafos (AABG) em tarefas de agrupamento, por meio de (1) um levantamento dos principais algoritmos existentes e suas respectivas caracterizações; (2) levantamento das particularidades gerais desta classe de algoritmos de agrupamento e daquelas particulares a cada um deles; (3)

identificação das características de domínios de dados que recomendam (ou não) o uso de algoritmos baseados em grafos em tarefas de agrupamento, (4) desenvolvimento de um sistema computacional que permitisse a experimentação com agrupamentos baseados em grafos e (5) uso de tal sistema em uma área específica de conhecimento, aquela de microRNAs, com o objetivo de caracterizar famílias de microRNAs, via agrupamento.

1.3 Organização do Documento

Este documento está organizado como segue:

Capítulo 1: apresenta a área de conhecimento na qual a pesquisa se insere, elencando as principais motivações e objetivo do trabalho, bem como descreve a organização do documento.

Capítulo 2: apresenta uma breve introdução ao assunto de pesquisa *i.e.*, algoritmos de agrupamento baseados em grafos (AABG), seguida por uma revisão bibliográfica dos principais algoritmos encontrados.

Capítulo 3: focaliza a apresentação e discussão de cinco AABGs que usam o conceito de árvore *spanning* minimal (ASM). Os cinco AABGs abordados no trabalho foram: ASME, ASMEH, ASM-PI, ASM-TXE e SFASM. Esses algoritmos foram utilizados e avaliados nos experimentos que fazem o uso das sequências de microRNAs, com o objetivo de agrupá-las em suas respectivas famílias.

Capítulo 4: são apresentados e detalhados os algoritmos baseados em grafos que utilizam o conceito de *região de influência* para indução de agrupamentos. Seis diferentes regiões de influência foram consideradas e experimentadas em avaliações de eficácia do algoritmo na tarefa de agrupamento das diversas sequências de microRNAs em suas respectivas famílias.

Capítulo 5: apresenta e discute um AABG divisivo, denominado *Ncut*, cuja motivação principal para a sua proposta foi a de ser bastante promissora e de fácil implementação.

Esse algoritmo também foi utilizado nos experimentos de agrupamento de microRNAs, e seus resultados, foram detalhados e avaliados.

Capítulo 6: introduz os chamados índices de validação, importantes quando da avaliação da 'qualidade' do agrupamento criado por algoritmos de agrupamento, que foram utilizados nos experimentos conduzidos com os algoritmos escolhidos.

Capítulo 7: apresenta os conceitos fundamentais relacionados a microRNAs, bem como informações sobre uma de suas principais bases de dados (miRBase), a partir da qual foram extraídos os dados utilizados nos experimentos.

Capítulo 8: descreve e discute diversos experimentos que utiliza o algoritmo de agrupamento denominado ASME, bem como os índices de validação, apresentados no Capítulo 6. Também é exibida e discutida a representação das sequências de microRNAs selecionadas nos experimentos, e seus relacionamentos, por meio da visualização de seus respectivos grafos.

Capítulo 9: apresenta e discute alguns experimentos complementares que utilizam os seguintes algoritmos de agrupamentos: ASMEH, ASM-PI, ASM-TXE, SFASM, Região de Influência e *Ncut*. O objetivo foi verificar e comparar os resultados de cada algoritmo, de acordo com suas respectivas características.

Capítulo 10: apresenta as conclusões do trabalho conforme os algoritmos propostos e seus resultados nos experimentos de agrupamento de microRNAs.

Anexo A: tem por principal foco a padronização do formalismo notacional empregado na descrição de conceitos, resultados e algoritmos, bem como a apresentação das definições básicas necessárias para o entendimento do que é apresentado em cada capítulo.

Um diagrama com a estruturação dos algoritmos considerados neste documento (e respectivos capítulos(C)/seções(S) em que são tratados) está apresentado na Figura 1.1.

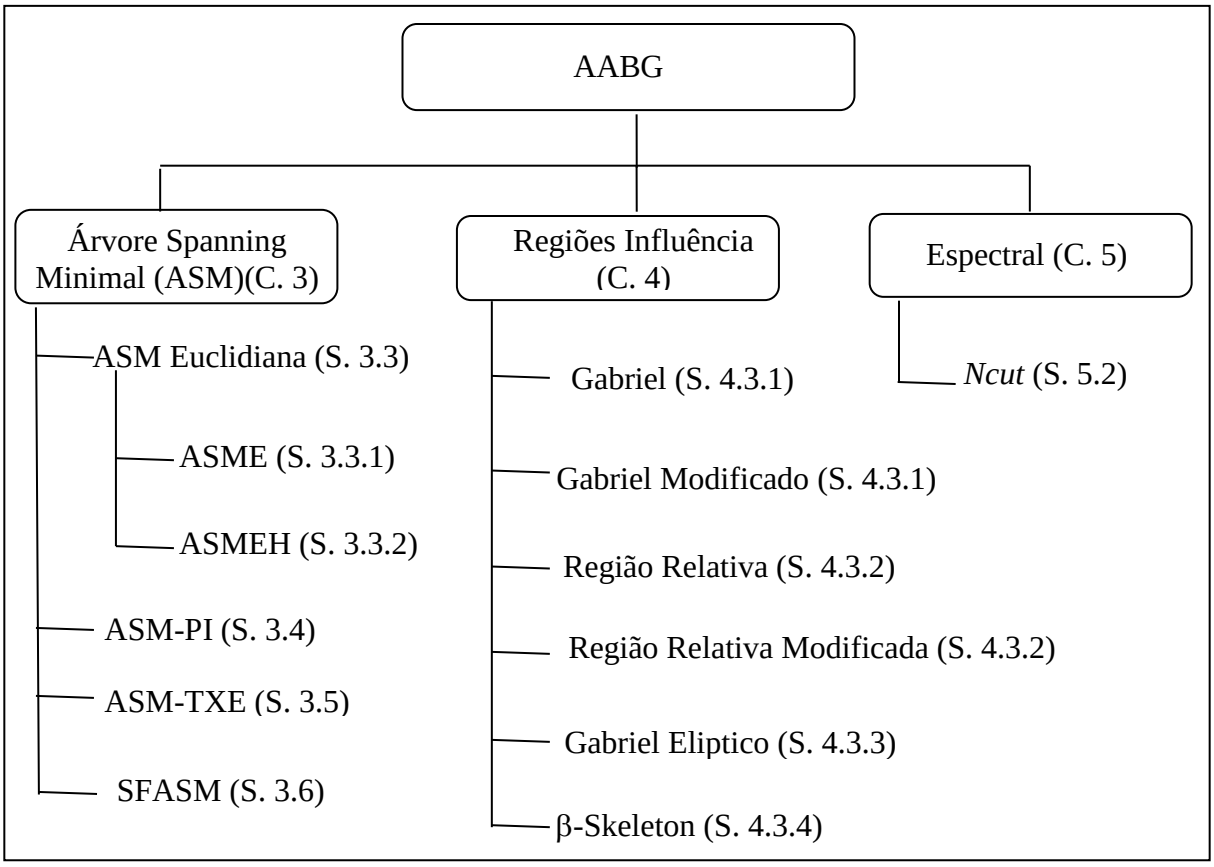

Figura 1.1 Organização de apresentação/discussão dos algoritmos considerados no trabalho.

Capítulo 2

Aprendizado de Máquina e Algoritmos de Agrupamento

2.1 Considerações Iniciais

Categorizar dados (*i.e.*, atribui-los a diferentes categorias/grupos) é uma habilidade humana e um processo fundamental para a transformação de um volume de dados obtidos sobre determinado problema (situação, evento, etc.), em conhecimento. Tal categorização, se for corretamente interpretada, poderá servir como base para tomadas de decisão em relação a vários aspectos do problema. Categorização (ou agrupamento) é comumente utilizada em aplicações do mundo real, particularmente aquelas relacionadas a reconhecimento de padrões em análise e segmentação de imagens, análise linguística de texto, detecção de doenças e muitas outras, em vários domínios de conhecimento.

2.2 Breve Introdução a Aprendizado de Máquina e Algoritmos de Agrupamento

O chamado *aprendizado indutivo de máquina* é o modelo de aprendizado automático mais bem sucedido e o que mais tem sido implementado, utilizando as mais variadas técnicas. Para viabilizar o aprendizado indutivo é imperativo que um conjunto de instâncias, que representam os conceitos a serem aprendidos, esteja disponível. Esse conjunto de instâncias é denominado *conjunto de treinamento*. Conjuntos de treinamento geralmente são descritos por vetores de pares *atributo–valor_de_atributo* e, dependendo da situação, de uma *classe* associada.

A classe de cada padrão de dado que participa do conjunto de treinamento é, na maioria dos casos, determinada por um especialista humano da área de conhecimento descrita pelos dados. O fato da classe participar da descrição do padrão e da técnica de

aprendizado fazer uso dela, caracteriza a técnica como uma *técnica de aprendizado supervisionado*.

Em muitas situações do mundo real, entretanto, a classe à qual cada dado pertence é desconhecida e/ou não existe um especialista humano capaz de, com base na descrição dos valores de atributos que descrevem o dado, estabelecer sua classe. Técnicas de AM que fazem uso de conjuntos de dados que não têm uma classe associada são conhecidas como técnicas de *aprendizado não-supervisionado*. Uma das técnicas de aprendizado não-supervisionado mais difundidas e utilizadas é chamada de *agrupamento* (*clustering*).

Jain e colaboradores em [Jain *et al.*, 1999] definem informalmente agrupamento como a organização de uma coleção de dados (geralmente representados como vetores de medidas ou, então, um ponto em um espaço multidimensional) em grupos, com base na similaridade. Intuitivamente, dados que pertencem a um mesmo grupo de um agrupamento são mais similares entre si do que a dados que pertencem a outros grupos do agrupamento.

Seja CD = $\{d_1, d_2, ..., d_n\}$ um conjunto contendo n dados m-dimensionais, i.e., cada um deles descrito por m atributos. Ou seja, considere $d_x \in$ CD com $x \in [1, n]$, então os atributos de d_x são definidos por $d_x = [d_{x_1}, d_{x_2}, ..., d_{x_m}]$. Um k-agrupamento de CD, notado por AG_k, pode ser definido com uma partição do conjunto de dados CD em k conjuntos (grupos), i.e., $AG_k = \{G_1, G_2, ..., G_k\}$. Considerando que um agrupamento dos dados do conjunto CD é uma partição de CD, então as seguintes três condições devem ser verificadas:

(1) $G_i \neq \emptyset$, i = 1, ..., k
(2) $\bigcup_{i=1}^{k} G_i = CD$
(3) $G_i \cap G_j = \emptyset$, $i \neq j$ e i, j = 1, ..., k

Assume-se que os dados agrupados em G_i (i = 1, ..., k) sejam "mais similares" entre si do que dados que pertencem a grupos distintos G_i e G_j, $i \neq j$ e i, j = 1, ..., k.

Uma vez que o conceito de similaridade é parte integrante do processo de agrupamento, a definição de uma medida de similaridade entre dois pontos de dados, extraídos de um mesmo espaço de atributos, é fundamental a qualquer procedimento de agrupamento. Na maioria dos casos, a definição da medida de similaridade é determinante

para a indução de um agrupamento que seja representativo e que espelhe a real natureza da organização dos dados.

Como apontado em [Jain *et al.* 1999], devido à diversidade de tipos de atributos e de suas respectivas unidades de medida, a medida de distância deve ser cuidadosamente escolhida. Via de regra é mais comum calcular a *dissimilaridade* entre dois pontos de dados usando uma medida de distância definida no espaço de atributos.

Neste trabalho o foco foi em medidas de distância usadas para pontos de dados descritos por atributos com valores contínuos; a mais popular para atributos contínuos é a *distância euclidiana*.

Considere dois pontos de dados d_x e d_y pertencentes a um espaço m-dimensional, notados respectivamente por $d_x = [d_{x_1}, d_{x_2}, ..., d_{x_m}]$ e $d_y = [d_{y_1}, d_{y_2}, ..., d_{y_m}]$. A distância ($dist_{Euc}$) entre dois pontos de dados d_x e d_y (ou, alternativamente, entre d_y e d_x) é dada pela Equação (2.1). A distância euclidiana entre os pontos d_x e d_y representa o comprimento do segmento de reta que os conecta.

$$dist_{Euc}(d_x, d_y) = \sqrt[2]{(d_{x_1} - d_{y_1})^2 + (d_{x_2} - d_{y_2})^2 + \cdots + (d_{x_m} - d_{y_m})^2} \qquad (2.1)$$

Outro tipo de distância utilizada nos experimentos, denominada *city-block*, foi utilizada nesse trabalho com o objetivo de verificar seu impacto nos resultados dos AABG. Considere os mesmos dois pontos de dados d_x e d_y. A distância *city-block* ($dist_{cb}$) entre d_x e d_y (ou, alternativamente, entre d_y e d_x) é dada pela Equação (2.2).

$$dist_{cb}(d_x, d_y) = |(d_{x_1} - d_{y_1})| + |(d_{x_2} - d_{y_2})| + \cdots + |(d_{x_m} - d_{y_m})| \qquad (2.2)$$

Na literatura pode ser encontrado um número considerável de algoritmos de agrupamento bem como de *índices de validação*, que permitem uma avaliação dos resultados produzidos por um tal algoritmo. As várias possibilidades levam à necessidade de duas decisões relevantes:

- Escolha do melhor algoritmo de agrupamento para um determinado conjunto de dados. Cada procedimento poderá exigir parâmetros de entrada de usuário, utilizará uma técnica específica e adotará um critério para medir a similaridade ou a dissimilaridade dos dados. Na dependência de valores de parâmetros (entre outros),

os algoritmos podem produzir diferentes grupos, relativos a um mesmo conjunto de dados de entrada.

- Qual (ais) índice(s) de validação usar e como interpretar seus resultados.

O objetivo desse trabalho foi abordar esses dois aspectos, escolha de algoritmos de agrupamentos baseados em grafos descritos nas Seções 2.2.1 e 2.2.2 e avaliação dos algoritmos com a utilização dos índices com validação interna e relativa, introduzidos na Seção 2.2.3 e detalhados e analisados no Capítulo 6. Os AABGs selecionados estão detalhados nos capítulos 3, 4 e 5, respectivamente. O domínio de conhecimento escolhido para a experimentação com os AABG é apresentado no Capítulo 7. Os capítulos 8 e 9 descrevem os experimentos realizados e o Capítulo 10 apresenta as conclusões do trabalho.

Como sugerido em [Grira *et al.* 2005], quando da escolha de um algoritmo de agrupamento (seja AABG ou não), os seguintes aspectos devem ser lembrados e considerados:

- O tamanho do conjunto de dados pode influenciar o desempenho do algoritmo. Alguns algoritmos podem fazer uso de apenas uma parte do conjunto de dados, tal como o *BIRCH* [Zahn 1996], que constrói uma árvore apenas com um subconjunto dos dados. Neste caso, a precisão do resultado do algoritmo pode ser afetada. Outros algoritmos adotam uma abordagem paralela, em que os processos que os implementam são divididos em subprocessos, que podem ser executados de maneira independente aumentando, dessa forma, o desempenho do algoritmo como um todo.
- O algoritmo deve ser escalável; uma variação no número de atributos que descrevem os dados não deve afetar o desempenho do algoritmo de maneira perceptível.
- O algoritmo deve realizar um tratamento adequado de conjuntos de dados que possuam ruídos, ou seja, deve ter a capacidade de detectar a presença de grupos isolados sem que os ruídos influenciem na sua eficência.
- Muitos algoritmos não são capazes de tratar de maneira eficiente grupos sobrepostos, ou seja, situações em que um dado pode ser categorizado em mais de uma classe ou grupo.

- O algoritmo deve ser capaz de lidar com grupos que tenham densidades e formatos distintos e que essa diversidade não impacte o agrupamento formado.
- O número de parâmetros de entrada para que o algoritmo possa ser executado deve ser o menor possível. Existem casos em que a eficiência do algoritmo fica comprometida, devido à dificuldade do usuário em fornecer um valor de parâmetro conveniente para a situação considerada.
- O algoritmo deve ser capaz de lidar com os mais variados tipos de dados (qualitativos e quantitativos).
- A mudança na sequência em que os dados são processados não deve gerar resultados divergentes. Ou seja, o algoritmo deve ser consistente de forma tal que, dado um conjunto de dados, independentemente da ordem em que seus dados são processados, produzirá o mesmo resultado em todas as execuções.

Para a validação do agrupamento obtido por meio de um algoritmo, existem inúmeras técnicas que, geralmente, são agrupadas e categorizadas como [Jain *et al.* 1999]:

- *Validação externa*: o resultado de um algoritmo de agrupamento é comparado com um agrupamento modelo. Ou seja, verifica se os grupos formados estão de acordo com algum conhecimento prévio representado pelo agrupamento modelo.
- *Validação interna:* no qual avalia se o resultado possui uma estrutura apropriada para os dados em questão. A estrutura avaliada compreende densidade, grau dos vértices, medida de similaridade ou dissimilaridade e depende do índice de validação.
- *Validação relativa*: são comparados os resultados de agrupamentos obtidos usando um ou mais algoritmos para um mesmo conjunto de dados, mas com diferentes valores de parâmetros de entrada.

2.2 Algoritmos de Agrupamento Hierárquicos e Particionais Subsidiados por Teoria dos Grafos

Na literatura podem ser encontrados inúmeros algoritmos de agrupamento, que fazem uso dos mais variados formalismos matemáticos e estatísticos. Como descrito em

[Theodoridis & Koutroumbas 1998] [Jain 2010], algoritmos de agrupamentos podem ser divididos em várias categorias; duas delas são de particular interesse para este trabalho: aquela dos *algoritmos hierárquicos* e a dos *algoritmos particionais*. A categoria de *algoritmos hierárquicos* é geralmente abordada como duas sub-categorias, a dos *algoritmos hierárquicos aglomerativos* e a dos *algoritmos hierárquicos divisivos*, brevemente apresentadas a seguir.

Os *algoritmos hierárquicos aglomerativos* produzem uma sequência de agrupamentos, cada um deles com um número menor de grupos que o anterior. O agrupamento produzido no passo k é baseado no agrupamento produzido no passo k− 1, no qual é feito uma junção de dois grupos. Os principais representantes dos algoritmos aglomerativos são os algoritmos *único-link* e *link-completo*. Esses algoritmos são apropriados para a recuperação de grupos alongados (como é o caso do algoritmo *único-link*) e grupos compactos (como é o caso do algoritmo *link-completo*). Algoritmos aglomerativos podem ser abordados e divididos em duas subcategorias:

- *algoritmos baseados em teoria de matrizes*
- *algoritmos baseados em teoria dos grafos*

Os *algoritmos hierárquicos divisivos* processam os dados na direção oposta àquela dos algoritmos aglomerativos, ou seja, produzem uma sequência de agrupamentos, cada um deles com um número maior de grupos que o anterior. O agrupamento produzido no passo k é baseado no agrupamento produzido no passo k− 1, no qual é feita a divisão de um dos grupos do agrupamento, em dois.

A categoria dos *algoritmos de otimização ou particionais* tem como objetivo evidenciar agrupamentos otimizados como resultado da partição do conjunto de dados segundo um critério pré-estabelecido. A otimização ocorre a partir da definição de uma função de custo *f*, a qual deve ser calculada a cada iteração do algoritmo, conduzindo, dessa maneira, a composição dos grupos de um agrupamento, por meio da busca da solução que otimiza *f*. Os agrupamentos formados não seguem uma estrutura hierarquizada. O principal representante dessa categoria de algoritmos é o simples, eficiente e popular algoritmo denominado *k-means*, descrito em várias referências bibliográficas e, particularmente, em [Jain 2010]. O principal inconveniente desse algoritmo é requerer, como parâmetro de

entrada, a informação do número máximo de grupos (k) que o agrupamento induzido deverá ter. Muitas vezes esse valor é desconhecido e a tendência de informar ao algoritmo um valor arbitrário qualquer compromete sua efetividade.

A categoria dos algoritmos hierárquicos é eficiente para evidenciar grupos de direntes formatos; tais algoritmos, contudo, demandam uma alta complexidade de tempo e espaço, enquanto que os métodos de otimização ou particionais possuem uma melhor performance para a análise de grande volume de dados [Jain *et al.* 1999].

Como comentado anteriormente, esse trabalho de pesquisa teve como foco os algoritmos de agrupamento hierarquicos ou algoritmos de agrupamento particionais baseados em Teoria dos Grafos, referenciados como AABG (*Algoritmos de Agrupamento Baseados em Grafos*).

Determinados tipos de dados bem como suas relações podem ser representados por meio de uma estrutura de grafo. Dados com essa característica são encontrados em vários domínios de dados reais como em sequências biológicas e em análise de redes. O primeiro passo para o agrupamento de tais dados é pré-processar os dados disponíveis, de maneira a extrair os dados de interesse (e.g., de uma base de dados) e moldá-los à uma estrutura de grafo, geralmente representada por uma matriz de adjacência. A partir dessa representação, o método de agrupamento pode, por exemplo, utilizar os conceitos de árvore (Definição 4 do Anexo A) ou caminho (Definição 10 do Anexo A) para evidenciar os grupos que podem ser intrinsicamente representados por subgrafos (Definição 2 do Anexo A). A conectividade de um grafo pode estar associada a medidas, tais como distância euclidiana, número de arestas, quantidade de caminhos diferentes ou, simplesmente, a presença ou não da aresta entre dois vértices.

2.2.1 Considerações sobre Algoritmos Hierárquicos Subsidiados por Teoria dos Grafos

Os algoritmos de agrupamento considerados hierárquicos são aqueles que categorizam os dados em níveis hierárquicos, gerando, a cada nível, um agrupamento. Os agrupamentos evidenciados pelos algoritmos hierárquicos podem ser organizados como uma estrutura denominada dendrograma, ou seja, um diagrama organizado na forma de

árvore. A partir do dendrograma é possível detectar e realizar o corte necessário para a identificação do agrupamento de interesse.

Algoritmos hierárquicos não requerem o conhecimento a priori do número de grupos que o agrupamento a ser induzido deve ter, porém o problema da definição da condição de parada do algoritmo nem sempre é facilmente resolvido.

O algoritmo hierárquico aglomerativo mais simples é o *único-link* [Sneath & Sokal 1973], que utiliza uma estratégia *bottom-up.* O agrupamento inicialmente considerado é tal que cada um de seus grupos é um *singleton*, i.e., um grupo cujo único elemento é apenas um dos dados. A cada passo do processo iterativo é realizada a junção de grupos usando o critério da proximidade entre eles; o processo se repete até que um determinado critério de parada seja satisfeito.

O *link-completo* [King 1967] emprega o mesmo procedimento adotado pelo *único-link*. A diferença entre eles ocorre na escolha do critério de proximidade na junção dos grupos. Enquanto o *único-link* requer a mínima distância entre os grupos, o *link-completo* exige a máxima distância entre os grupos. A distância entre os grupos é definida pela distância dos pontos de dados pertencentes a cada um dos grupos dada pela Equação (2.3). Vários algoritmos foram propostos com base nos dois mais populares AABGs mencionados anteriormente: *único-link* e *link-completo*.

$$dist(G_i, G_j) = \{dist(d_x, dy) \mid d_x \in G_i \text{ e } d_y \in G_j\} \qquad (2.3)$$

Outro método de agrupamento eficiente que utiliza a técnica hierárquica divisiva e baseada em grafos, é representado pelos algoritmos que, inicialmente, constroem uma *árvore spanning* minimal (ASM) (Definição 7 do Anexo A) a partir do conjunto de dados. O processo tem continuidade por meio da remoção, na ASM, das arestas com maiores pesos, que induz uma partição da ASM em subárvores, cada uma tratada como um grupo do agrupamento representado pela ASM inicial [Zahn 1971] [Jain et al. 1999]. Os algoritmos baseados em ASM são geralmente de fácil implementação e muito eficientes na formação de agrupamentos em que os dados contemplam formatos diversos.

Em [Toussaint 1980] é utilizado o *grafo de vizinhança relativo* (RNG), baseado nas estruturas de ASM e na triangulação de Delaunay, com o objetivo de facilitar a representação por meio de um modelo que agrega informações relevantes sobre o conjunto

de dados. Dois algoritmos são apresentados para obtenção do RNG e possuem diferença em performance de execução por não adotarem a mesma técnica.

Métodos hierárquicos divisivos que se utilizam de conceitos da Teoria dos Grafos removem arestas do grafo com vistas à separação de um determinado grupo. Dois problemas inerentes à essa abordagem são (1) a formação de grupos isolados e (2) a dificuldade envolvida na escolha da condição de parada do algoritmo.

2.2.2 Considerações sobre Algoritmos Particionais Subsidiados por Teoria dos Grafos

Os algoritmos de otimização ou particionais utilizam uma técnica para encontrar todos os grupos simultaneamente, de forma que sua estrutura não seja obrigatoriamente hierárquica. O principal algoritmo que representa essa categoria, o *k-means* [MacQueen 1967], utiliza o conceito de *centro de grupo (ou centróide)* para inicializar seu processo de agrupamento (Definição 13 do Anexo A). No início do processo, os centros são escolhidos aleatoriamente de acordo com o número de grupos informado pelo usuário (parâmetro k). A partir daí, com os centros dos grupos definidos, cada dado é associado ao centro mais próximo, utilizando a função *f* do quadrado da distância euclidiana. Os centros dos grupos são iterativamente recalculados e os dados são redistribuídos de forma a minimizar *f* [Jain 2010]. O algoritmo é simples e eficiente para dados com baixa dimensionalidade; porém é bastante sensível a um aumento no volume de dados bem como a um aumento na dimensionalidade dos dados envolvidos, situações que podem comprometer o desempenho do algoritmo. Também é bastante sensível aos k centros iniciais, aleatoriamente escolhidos.

A principal dificuldade no uso desse tipo de algoritmo é relativa ao valor do parâmetro k, que estabelece o número de grupos que o agrupamento induzido deverá possuir, uma vez que uma estimativa plausível para k exige que o usuário tenha uma boa compreensão do domínio do problema e dos dados que o representa. Outro incoveniente é que, para encontrar uma solução ótima, o algoritmo deve ser executado diversas vezes. Os passos envolvidos em um algoritmo particionado podem ser generalizado como detalhado em Algoritmo 2.1.

```
Procedure Algoritmo_Particional(CD, f, k)

{entrada: CD: Conjunto de dados CD = {d1, d2,..., dn}, |CD| = n,
          k: número de grupos e
          f: função objetivo.
saída:    AG: Agrupamento AG = {G1, ..., Gk} – um conjunto contendo as k
          componentes conexas do grafo construído pelo algoritmo - cada
          componente conexa é um grupo do agrupamento obtido.}

begin
  AG ← ∅
  AG ← Escolha_Aleatoria(CD, k)  {escolhe aleatoriamente k dados de CD, que
                                  representam os centros dos k grupos de AG}
  for all di ∈ CD e di ∉ AG do
    begin
      Gj ← Encontra_Grupo_Proximo(di, AG, f)  {encontrar um grupo Gj ∈ AG,
                                               j ∈ [1, k], para di, de forma a otimizar f }
      Gj ← Gj ∪ {di}
      AG ← Atualiza_Agrupamento(AG, Gj)  {atualiza AG com o grupo Gj alterado}
    end
end
```

Algoritmo 2.1. Descrição do algoritmo particional ou de otimização genérico baseado no *k-means*.

Os algoritmos de agrupamentos particionais baseados em um grafo G = (V, E) tem por objetivo dividir o conjunto de dados (i.e., vértices V) nos conjuntos V_1 e V_2 de tal forma que $V_1 \cup V_2 = V$ e $V_1 \cap V_2 = \varnothing$. A partição é realizada por meio de um processo de remoção de arestas $e_i \in E$, denominado corte (Definição 20 do Anexo A) [Shi & Malik 2000].

Recentemente a técnica espectral de algoritmos de agrupamento particionais baseados em grafos tem-se mostrado eficiente e eficaz na solução de problemas, principalmente, no domínio de segmentação de imagem e de análise de grandes volumes de dados, como aqueles provenientes de redes sociais e bioinformática. A teoria espectral de grafos estuda as propriedades de um grafo por meio de suas representações matriciais e seus respectivos espectros (Definição 17 do Anexo A), isto é, os autovalores (Definição 15 do Anexo A) das matrizes às quais está associado.

Essa técnica é de fácil implementação e utiliza métodos da Álgebra Linear para simplificar a solução do problema dos agrupamentos baseados na teoria dos grafos. Os passos para a realização do agrupamento utilizando a técnica espectral são descritos no Algoritmo 2.2. É importante observar que o procedimento particiona o grafo no máximo

em duas partes, evidenciando um ou dois grupos por vez, conforme os sinais dos valores dos autovetores que indicam se a partição deve ocorrer ou não. Para a formação de k grupos, é necessário que o algoritmo seja executado iterativamente até que k grupos sejam produzidos.

```
Procedure Algoritmo_Espectral(CD)

{entrada: CD: Conjunto de dados CD = {d1, d2,..., dn}, |CD| = n.
saída:    AG: Agrupamento AG = {G1, ..., Gk} ─ um conjunto contendo as k
          componentes conexas do grafo construído pelo algoritmo - cada
          componente conexa é um grupo do agrupamento obtido.}

begin
   G = (V, E) ← Constroi_Grafo(CD)  {contrói grafo ponderado e conectado}
   L ← Constroi_Matriz_Laplaciana(G) {Definição 18 ou 19 do Anexo A}
   X ← Encontra_Autovetor(L)
   AG ← Atribui_Dados_Grupos(CD, X)
end
```

Algoritmo 2.2 Descrição do algoritmo genérico baseado em espectros de grafos.

O passo que evidencia os grupos por meio dos autovetores é considerado divergente, ou seja, existem diversas formas de escolha do autovetor e nenhuma delas é estabelecida como referência para utilização nos algoritmos espectrais. O que mais se aproxima da solução ótima é a utilização do segundo autovetor da matriz laplaciana (Definição 18 e Definição 19 do Anexo A) [Andrew *et al.* 2002]. Os autovetores são vetores indicadores utilizados na escolha do grupo para o dado em avaliação. Diversas técnicas podem ser utilizadas para a atribuição dos dados a cada um dos grupos existentes. Por exemplo, em [Luxburg 2007] é descrita uma técnica que utiliza n autovetores para a posterior atribuição dos dados aos grupos por meio do algoritmo de *k-means*.

Os algoritmos espectrais diversificam na utilização dos tipos de matrizes laplacianas, ou seja, as matrizes não-normalizadas e normalizadas. O algoritmo *Ncut,* detalhado em [Shi & Malik 2000], propõem e utiliza uma matriz laplaciana normalizada para realizar o processo e agrupamento em segmentação de imagens e é apresentado em detalhe no Capítulo 5.

2.2.3 Sobre Índices para a Avaliação de Resultados de Algoritmos de Agrupamento

A avaliação e qualificação do resultado produzido por um algoritmo de agrupamento é um processo complexo devido, principalmente, à dificuldade em definir as propriedades que qualificariam tal resultado como um agrupamento ótimo. Essas propriedades devem estar em concordância com o domínio do problema e, no caso de grafos, envolve a característica de conectividade, conforme descrito em [Schaeffer 2007]. Cada grupo deve estar conectado, isto é, dado um grafo G = (V, E) com d_1, d_2 e $d_3 \in V$. Se d_1 e d_2 pertencem ao mesmo grupo então há um caminho (Definição 10 do Anexo A) que conecta d_1 a d_2 e vice-versa. Se d_1 e d_3 pertencem a diferentes grupos do agrupamento formado, então não existe um caminho de d_1 e d_3 (ou vice-versa).

Considere um conjunto de dados CD e um grupo formado a partir de CD, denominado G_1. Pode-se conceituar arestas internas de G_1, $\deg_{int}^{G_1}$, como arestas que conectam qualquer $d_i \in G_1$ a quaisquer outros vertices ou dados $d_j \in G_1$. E, arestas externas de G_1, $\deg_{\text{ext}}^{G_1}$, como arestas que conectam qualquer $d_i \in G_1$ a qualquer outro dado $d_z \notin G_1$. Um grupo considerado ótimo possui $\deg_{\text{ext}}^{G_1} = 0$ e um grupo considerado razoável possui $\deg_{\text{int}}^{G_1} > \deg_{\text{ext}}^{G_1}$. Porém nem sempre será possível atribuir um dado d_i a um grupo G_m, com m = 1, ..., k, devido à presença de dados isolados, ou seja, dados que não possuem nenhuma semelhança com dados de qualquer outro grupo; Ou d_i poderá pertencer a mais de um grupo, como no caso de classificação temática onde um determinado assunto pode pertencer a diferentes categorias. Os algoritmos de agrupamentos que utilizam a técnica *fuzzy* são apropriados para esses casos em que um vértice está relacionado, em diferentes níveis, a mais de um grupo. A abordagem *fuzzy*, entretanto, não é foco desse trabalho, uma vez que agrupamentos, nesse trabalho, foram conceitualmente definidos como partições do conjunto de dados.

Em [Jain 2010] está descrita uma análise comparativa entre 3 algoritmos hierárquicos (*MST, Complete-Link, JP*) e 4 algoritmos particionados ou de otimização (*FORGY, ISODATA, WISH, CLUSTER*) baseados no algoritmo *k-means*. Os resultados obtidos evidenciaram que não existe um melhor algoritmo, de forma que os métodos que utilizam as mesmas técnicas obtiveram resultados similares. Muitos algoritmos, como o *k-means*,

também possuem limite computacional para processar grandes quantidades de dados para encontrar a melhor solução.

A definição de um teorema que unifique as propriedades necessárias para que um agrupamento seja considerado ótimo é discutida em [Kleinberg 2002]. Considere uma função *f*, um conjunto de dados CD e a distância dist entre dois dados d_i e $d_j \in$ CD. A partição do conjunto CD é definida por $\Gamma = f(\text{CD}, \text{dist})$. Um agrupamento ótimo pode ser definido por meio das seguintes propriedades:

- *Invariância de escala:* Para qualquer medida de distância d e qualquer valor constante $\beta > 0$ tem-se que $f(\text{CD}, \text{dist}) = f(\text{CD}, \beta\text{dist})$, ou seja, a variação na escala da distância deve produzir o mesmo resultado.
- *Riqueza:* A função *f* é capaz de produzir qualquer agrupamento possível a partir do conjunto de dados CD e qualquer função dist corretamente definida.
- *Consistência:* Considere as funções de distâncias dist_1 e dist_2 e dois dados d_i e d_j que pertencem a diferentes grupos de Γ. Considere também que $\text{dist}_1(d_i, d_j) \geq \text{dist}_2(d_i,d_j)$, então $f(\text{dist}_1) = \Gamma$ e $f(\text{dist}_2) = \Gamma$. Ou seja, a expansão ou contração da função de distância não altera os grupos formados.

A conclusão apresentada em [Kleinberg 2002] é que não existe uma função f capaz de satisfazer as três propriedades definidas anteriormente e, portanto, suas condições devem ser relaxadas, principalmente a propriedade de riqueza, para que os algoritmos possam satisfazer, pelo menos parcialmente, as características descritas. No caso de agrupamentos baseados em grafos essas três propriedades precisam ser adaptadas, como por exemplo, no caso da riqueza, que pode ser interpretada como a capacidade de produzir todos os subgrafos possíveis a partir de um conjunto de dados.

A técnica da observação, por meio da visualização dos agrupamentos formados também contribui para a análise de suas estruturas e possibilita uma melhor interpretação dos resultados. Porém no caso de conjunto com grande quantidade de dados, essa técnica torna-se inviável para o grafo como um todo, ou seja, é necessário selecionar um subgrafo que represente uma parte relevante dos dados que se deseja analisar visualmente.

Outra forma de avaliar a qualidade dos resultados de algoritmos de agrupamentos e, particularmente, dos AABGs, é utilizar índices para medir e verificar a eficiência do algoritmo. Os índices são utilizados para comparar a estrutura de diferentes agrupamentos formados a partir da execução da técnica adotada por cada algoritmo. Existem muitos índices propostos na literatura, sendo que vários deles são de difícil interpretação ou comparação [Boutin & Hascoet 2004].

Em [Halkidi *et al.* 2002] alguns dos diversos índices relativos são avaliados e o resultado dessa avaliação é que não existe um índice universal, ou seja, um único índice que avalia qualquer tipo de agrupamento, independentemente da aplicação. É necessário compreender as propriedades e aspectos do conjunto de dados para a escolha do índice que melhor condiz com os dados. Essa escolha não é trivial e diversos trabalhos foram realizados com foco na comparação dos índices, como descrito em [Vendramini *et al.* 2010], em que são avaliados e detalhados 40 diferentes métodos utilizando algoritmos de agrupamentos hierárquicos e particionados.

Os índices podem ter sua eficiência afetada, principalmente nos casos de dados com ruídos ou, então, devido à sobreposição de grupos. Tais índices servem de base para uma avaliação dos agrupamentos por meio de um modelo matemático que visa quantificar as características estruturais dos grupos formados, tais como o número de arestas internas e externas. Não são utilizados parâmetros que consideram o domínio do problema, portanto não é possível definir, por meio dos índices, se um dado agrupamento está em conformidade com a área/aplicação específica da qual o conjunto de dados foi extraído.

Capítulo 3

AABGs Baseados em Árvore *Spanning* Minimal

3.1 Considerações Iniciais

Algoritmos de agrupamento caracterizados como baseados em grafos são capazes de detectar grupos de dados que se apresentam em vários possíveis formatos, pelo menos nos casos em que estão bem separados. Essa característica é compartilhada apenas por poucos algoritmos de agrupamento.

No que segue, cinco algoritmos de agrupamento baseados em árvores *spanning* minimais (ASM) serão apresentados e descritos, buscando elencar seus pontos fortes e fracos, bem como as principais características dos dados que, de certa forma, promovem o bom desempenho de cada um deles.

Alguns algoritmos de agrupamento baseados em grafos (como aquele apresentado na Seção 3.3) esperam, como entrada, uma árvore *spanning* que representa o conjunto de dados. Para o uso desses algoritmos, no entanto, o conjunto de dados a ser agrupado deve ser processado por um algoritmo que, a partir dos dados fornecidos, construa uma árvore *spanning* que os conecta. Dois desses algoritmos são revistos na Seção 3.2.

3.2 Indução da Árvore *Spanning* Minimal-(ASM)

Muitas questões de otimização podem ser modeladas no problema de encontrar, em um grafo ponderado, um certo tipo de subgrafo que seja conectado, que tenha o mesmo conjunto de vértices que o grafo original e cujo peso associado seja minimal (ou maximal). Não é difícil ver que tal subgrafo deve ser uma árvore *spanning* do grafo. Para isso ser possível, obviamente, o grafo original deve ser conectado.

Algoritmos que induzem a árvore *spanning* minimal (ASM) a partir de um grafo conectado geralmente são inicializados considerando apenas o conjunto de vértices e, então, a cada iteração, acrescentam uma aresta à árvore sendo construída. A escolha de qual

aresta acrescentar, dentre as disponíveis, dá origem aos diversos algoritmos que se propõem a induzir uma árvore *spanning* minimal a partir de um grafo conectado.

Os dois algoritmos mais conhecidos e utilizados para encontrar ASMs de um grafo são os algoritmos de Prim [Prim 1957] e de Kruskal [Kruskal 1965]. Os dois algoritmos se diferenciam na maneira como constroem as árvores. Basicamente, o algoritmo de Kruskal expande um conjunto de subárvores para formar a ASM e o algoritmo de Prim expande uma única subárvore por meio da adição de arestas a ela até obter a ASM final. É importante observar que diferentes árvores minimais podem ser formadas a partir de um único grafo, pois os algoritmos fazem uma escolha arbitrária caso exista mais de uma aresta com o mesmo menor peso, num determinado passo.

A estratégia utilizada na implementação dos algoritmos de Prim e Kruskal é conhecida como gananciosa pois ambos, a cada passo, escolhem uma aresta com o menor peso possível, sem refazer a decisão tomada. De forma geral, esses algoritmos gananciosos possuem as seguintes características:

- a ASM deve ser encontrada de maneira ótima e, para isso, há o auxílio de um conjunto de candidatos, ou seja, um conjunto de arestas potenciais. A solução ótima é determinada por meio de uma seleção local ótima, isto é, a aresta escolhida nem sempre é a aresta com o menor peso possível do grafo, mas é a aresta com o menor peso num determinado passo do algoritmo.
- os algoritmos utilizam dois conjuntos: um com as arestas avaliadas e rejeitadas, e outro com as arestas avaliadas e selecionadas.
- há uma função de seleção responsável por determinar a aresta que deve ser escolhida, ou seja, a aresta que ainda não foi analisada e com o menor peso possível.
- há uma função que verifica se a ASM formada é a solução do problema, isto é, se a ASM contém o mesmo número de vértices que aquele do grafo dado.
- há uma função que verifica se a aresta candidata escolhida é viável, ou seja, se a inclusão da aresta no conjunto de resultado não forma um ciclo (dependendo do algoritmo) com as outras arestas já selecionadas.

Considere um conjunto de dados CD com n dados. Um grafo G_i é construído tendo por conjunto de vértices V = CD = $\{d_1, d_2, ..., d_n\}$ e conjunto de arestas $E_i = \{e_1, e_2, ..., e_m\}$. O Algoritmo 3.1 descreve os passos do algoritmo de Prim para a geração da ASM a partir de G_i; o algoritmo termina quando n − 1 arestas forem escolhidas, em que n é o número de vértices do grafo original.

```
Procedure ASM_Prim(Gi, r)

{entrada: Gi: Grafo inicial Gi = (CD, Ei), com conjunto de dados
              ou vértices CD = {d1, d2,..., dn}, |CD| = n e o conjunto de arestas Ei e
          r: Qualquer dado ou vértice inicial r, representa a raiz da ASM.
 saída:   GASM: ASM representada por GASM = (V, E), em que V é o conjunto de vértices
              e E o conjunto de arestas da ASM.}

begin
   V ← ∅                        {árvore ASM começa com conjunto de vértices vazio}
   E ← ∅                        {árvore ASM começa com conjunto de arestas vazio}
   for all di ∈ CD do
      begin
         Peso_Vertice(di) ← ∞  {Em cada vértice é armazenado o menor peso encontrado
                                em cada iteração}
         Antecessor(di) ← null  {Antecessor é um vetor que armazena o caminho de
                                construção da ASM}
      end
    Peso(r) ← 0              {começa com o vertice inicial r}
    while |V| ≠ |CD| do
      begin
         u ← Encontra_Min(CD)  {função que seleciona o vértice que possui o menor peso}
         V = V ∪ {u}
         If u ≠ r then
           E ← E ∪ {(u, Antecessor(u))}       {insere a aresta na ASM}
         for all di ∈ CD and di ∉ V and di adjacente a u do
           begin
              if Peso_Aresta(u, di) < Peso_Vertice(di) then
                 begin
                    Antecessor(di) ← u
                    Peso_Vertice(di) ← Peso_Aresta(u, di) {em di é armazenado o menor peso
                                       dentre todas as arestas incidentes a di e adjacentes a u}
                 end
           end
      end
end
```

Algoritmo 3.1 Descrição do Algoritmo de Prim.

A partir de um dado vértice inicial r selecionado em CD, o algoritmo segue escolhendo e acrescentando arestas na árvore, de forma a mantê-la sempre acíclica, mínima e obviamente conexa. Durante a sua execução, o algoritmo de Prim sempre mantém dois

conjuntos de vértices: aqueles que são vértices-extremidade de arestas já escolhidas V, e aqueles que não são vértices extremidade de arestas escolhidas CD. Uma próxima aresta a ser selecionada deve ser tal que una dois vértices, um deles de CD e o outro de V, e, deve ter o mínimo peso. O algoritmo termina quando os n vértices de CD forem selecionados e inseridos em V.

O algoritmo de Kruskal também tem como objetivo a construção da ASM a partir de um conjunto de dados. Considere um conjunto de dados CD com n dados, um grafo G_i é construído com seu conjunto de vértices $V = CD = \{d_1, d_2, ..., d_n\}$ e seu conjunto de arestas $E_i = \{e_1, e_2, ..., e_m\}$. A principal característica do algoritmo de Kruskal, descrito em Algoritmo 3.2, é a de selecionar a aresta de menor peso sem levar em conta se a estrutura sendo formada mantém (ou não) a conectividade.

```
Procedure ASM_Kruskal(Gi)

{entrada: Gi: Grafo inicial Gi = (CD, Ei), com conjunto de dados ou vértices
              CD = {d1, d2,..., dn}, |CD| = n e o conjunto de arestas Ei.
 saída:   GASM: ASM representada por GASM = (V, E), onde V é o conjunto de
              vértices e E o conjunto de arestas da ASM.}

begin
   V← CD          {árvore ASM começa com conjunto de vértices = CD}
   E ← ∅          {árvore ASM começa com conjunto de arestas vazio}
   C ← ∅          {Conjuntos formados por diversas ASM}
   for all di ∈ V do
     begin
       ci ← ci ∪ {di}     {começa com cada di sendo considerado
                           uma componente conexa}
       C ← C ∪ {ci}
     end
   Ei ← Ordena_Aresta(Ei)    {ordena as arestas em ordem crescente de pesos, cada
                              aresta é formada por di ∈ ci e dj ∈ cj e (di,dj) ∈ Ei}
   while |E| ≠ n – 1 do
    begin
      (u, k) ← Retira_Min(Ei)  {retira uma aresta (u, k são os vértices da aresta)
                                na ordem crescente de pesos}
      if cu ≠ ck then
        begin
          E ← E ∪ (u, k)
          C ← C – {cu}
          C ← C – {ck}
          C ← C ∪ {(cu, ck)}   {coloca cu e ck na mesma componente conexa}
        end
    end
end
```

Algoritmo 3.2 Descrição do Algoritmo de Kruskal.

Para uma implementação eficiente do algoritmo é preciso organizar as arestas em ordem crescente de pesos. Inicialmente o algoritmo mantém uma floresta com um vértice do grafo em cada árvore. Em seguida, ordena as arestas, formadas por dois vértices pertencentes a duas árvores distintas, em ordem crescente de seus pesos. Para cada aresta escolhida o algoritmo verifica se esta pode ser adicionada à árvore em construção de forma a não formar ciclos. Para isso a seguinte estratégia é considerada: se os vértices-extremidade da aresta pertencem à mesma árvore então há a formação de ciclo, caso contrário, a aresta deve ser inserida na árvore e ocorre a união das árvores determinadas pelos vértices-extremidade da aresta escolhida. O algoritmo termina quando n − 1 arestas foram escolhidas.

3.3 AABG Baseado em ASME (Árvore *Spanning* Minimal Euclidiana)

3.3.1 Considerações Iniciais

O Algoritmo 3.3, denominado ASME, é considerado simples e serve de modelo para outros procedimentos que podem ser derivados a partir dele, visando a melhoria de seus resultados. Este método cria um grafo completo a partir de um conjunto de dados (função Cria_Grafo_Completo() em Algoritmo 3.3) e, utiliza a medida de distância euclidiana entre os pares de vértices, para determinação dos valores dos pesos de suas arestas. A partir do grafo completo, uma ASM é construída (função Cria_ASM() em Algoritmo 3.3), utilizando o Algoritmo 3.1 ou Algoritmo 3.2. Toda aresta em uma ASM é uma ponte (Definição 11 do Anexo A); a remoção de qualquer aresta da ASM produz novas componentes conexas. As arestas então, são organizadas em ordem decrescente de seus pesos (função Ordena_Aresta() em Algoritmo 3.3) para a posterior remoção de k − 1 arestas evidenciando os k grupos previstos (função Seleciona_Aresta() e Remove_Aresta() e Identifica_Componentes_Conexas() em Algoritmo 3.3).

Para o correto funcionamento do Algoritmo 3.3 é necessário a informação do parâmetro k, correspondente ao número número de grupos do agrupamento. Outro algoritmo base denominado ASMEZ (ASM Euclidiana de Zahn) utiliza a técnica de medida de inconsistência definida por Zahn [Zahn 1971], dispensando a necessidade da informação

fornecida por k – tal algoritmo será apresentado na Seção 3.4 identificado como ASM-PI, uma vez que se baseia na detecção de pontes inconsistentes.

Com o objetivo de amenizar a imprecisão ou formação de grupos desnecessários utilizando o algoritmo ASME, em [Grygorash *et al.* 2006] foram propostos dois algoritmos. Um deles, denominado ASMEH, está descrito na Seção 3.3.2. O outro utiliza o método de análise de regressão para determinar o número de grupos que o algoritmo deverá formar. Ou seja, o algoritmo não faz o uso do parâmetro k, como em ASME e ASMEH. Este algoritmo não será abordado neste trabalho.

```
Procedure ASME(CD, k)

{entrada: CD: Conjunto de dados CD = {d1, d2,..., dn}, |CD| = n e
          k:  Número de grupos do agrupamento AG.
saída:    AG: Agrupamento AG = {G1, G2, ... Gk} – um conjunto contendo as
          componentes conexas do grafo construído pelo algoritmo - cada
          componente conexa é um grupo do agrupamento obtido.}

begin
   AG ← ∅
   Gi ← Cria_Grafo_Completo(CD)
   GASM = (V, E) ← Cria_ASM (Gi)     {ASM_Kruskal(Gi) ou ASM_Prim(Gi, r)}
   Eord ← Ordena_Aresta(GASM)        {em ordem decrescente de seus pesos}
   Eord ← Seleciona_Aresta(Eord, k)  {seleciona as primeiras k – 1arestas}
    for all ei ∈ Eord do
      GASM ← Remove_Aresta(GASM, ei)
   AG ← Identifica_Componentes_Conexas(GASM)
end
```

Algoritmo 3.3 Descrição do Algoritmo ASME.

3.3.2 O Algoritmo ASMEH (ASM Euclidiano e Hierárquico)

O ASMEH, descrito em Algoritmo 3.4, exige como entrada o parâmetro k, para a remoção de k – 1 arestas da ASM. Esse processo é realizado iterativamente formando uma hierarquia de grupos que são evidenciados em cada iteração. Na remoção de cada aresta, são avaliados a média m_e e o desvio padrão σ_e de todos os pesos w(e) das arestas da ASM e, se a condição estabelecida na Equação (3.1) for satisfeita, a aresta é retirada, promovendo assim a partição do grafo.

$$w(e_i) > m_e + \sigma_e \tag{3.1}$$

Considere q como o número de arestas removidas conforme a Equação (3.1) e k como o número de grupos que o agrupamento deve possuir. O algoritmo ASMEH deve remover k – 1 arestas para evidenciar k grupos. Para isso, o procedimento então verifica as duas condições:

- $q < k - 1$

  Neste caso o número de grupos formados com as q arestas removidas é menor que o número de grupos previstos, ou seja, k. Assim, serão removidas as próximas (k– 1) – q arestas com maiores pesos da ASM formando assim os k grupos previstos.

- $q > k - 1$

  Neste caso o número de grupos formados com as q arestas removidas é maior que o número de grupos previstos. O algoritmo então determina os centros de cada grupo (Definição 13 do Anexo A) do agrupamento formado pela remoção das q arestas. Após a determinação dos centros c_i, os i dados representativos de cada grupo G_i são definidos pela Equação (3.2). Um conjunto S é construído com os i dados representativos e o algoritmo é então executado novamente com o conjunto de dados representado por S. Ao término do algoritmo, os dados que não pertencem a S, devem ser acrescentados a seus respectivos grupos mais próximos.

$$\mathrm{dist}(\mathrm{d}, \mathrm{c_i}) = \min\left(\mathrm{dist}(\mathrm{d_j}, \mathrm{c_i})\right), d_j \in G_i \tag{3.2}$$

O Algoritmo 3.4 descreve os passos para execução do ASMEH. Os principais passos são:

- Criação do grafo completo a partir de um conjunto de dados CD (via função Cria_Grafo_Completo() em Algoritmo 3.4);
- Criação da ASM a partir do grafo G_i, utilizando o Algoritmo 3.1 ou Algoritmo 3.2 (via função Cria_ASM() em Algoritmo 3.4);
- Cálculo da médio e desvio padrão da ASM (via funções Calcula_Media() e Calcula_Desvio_Padrao() em Algoritmo 3.4);

- Remoção das arestas de acordo com a média e desvio padrão (via função Remove_Aresta() em Algoritmo 3.4);
- Verificação de cada um dos três casos possíveis, ou seja, se $q = k - 1$, o algoritmo termina. Se $q < k - 1$, mais arestas serão removidas (via função Remove_Maior_Aresta() em Algoritmo 3.4) e, finalmente, se $q > k - 1$, uma nova ASM com um conjunto de dados reduzido será construída (via funções Centros_Grupos() e Dados_Representativos() em Algoritmo 3.4). O algoritmo é executado novamente utilizando dessa vez, um conjunto de dados reduzido. Neste caso, ao término do algoritmo, os dados que não pertencem ao conjunto de dados reduzido, deverão ser incluídos aos seus grupos mais próximos (via função Grupo_Mais_Proximo() em Algoritmo 3.4).

```
Procedure ASMEH(CD, k)

{entrada: CD: Conjunto de dados CD = {d1, d2,..., dn}, |CD| = n e
          k: número de grupos do agrupamento AG.
saída:    AG: Agrupamento AG = {G1, G2, ... Gk} – um conjunto contendo as
componentes conexas do grafo construído pelo algoritmo - cada componente
conexa é um grupo do agrupamento obtido.}

begin
   AG ← ∅
   q ← 0
  while q ≠ k − 1 do
   begin
      Gi ← Cria_Grafo_Completo(CD)
      GASM = (V, E) ← Cria_ASM(Gi)   {ASM_Kruskal(Gi) ou ASM_Prim(Gi, r)}
      me ← Calcula_Media(GASM)
      σe ← Calcula_Desvio_Padrao(GASM)
      for all ei ∈ E then
        begin                                   {caso 1}
            GASM ← Remove_Aresta(GASM, me, σe)    {conforme Equação 3.1}
            q ← q + 1
        end
      if q < k − 1 then
        begin                                   {caso 2}
            while q ≠ k − 1 do
              begin
                 GASM ← Remove_Maior_Aresta(GASM)
                 q ← q + 1
              end
        end
      else if q > k − 1then
        begin                                   {caso 3}
           AG ← Identifica_Componentes_Conexas(GASM)
           C ← Centros_Grupos (AG)
           S ← Dados_Representativos(CD, C)       {conforme Equação 3.2}
           CDant ← CD
            ASMEH(S, k)
        end
    end
    AG ← Identifica_Componentes_Conexas(GASM)
    if |CDant| ≠ ∅ then AG ← Grupo_Mais_Proximo(AG, CD, CDant)
end
```

Algoritmo 3.4 Descrição do Algoritmo ASMEH.

No que segue são mostrados alguns exemplos de diferentes situações de uso do ASMEH. Os três casos que são descritos, são identificados em Algoritmo 3.4 por *caso 1*, *caso 2* e *caso 3*.

- *caso 1:* $q = k - 1$, com $k = 3$

  Para exemplificar o Algoritmo 3.4 no *caso 1,* será utilizada a Figura 3.1. Para o grafo da figura, o algoritmo calcula a média e o desvio padrão, dados respectivamente por: média $m_e = 7{,}67$ e desvio padrão $\sigma_e = 6{,}56$. Como $m_e + \sigma_e = 14{,}23$, as arestas e_3 e e_5 serão removidas e o algoritmo termina, pois $q = k - 1 = 2$. O resultado do algoritmo evidencia três grupos G_1, G_2 e G_3 mostrados na Figura 3.1.

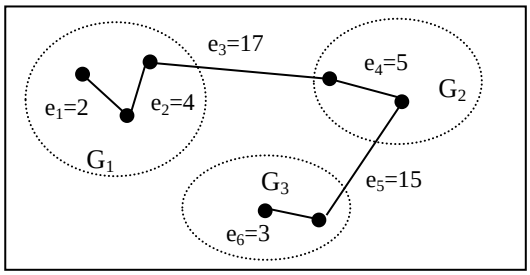

Figura 3.1 ASME utilizado como exemplo para o *caso 1*.

- *caso 2:* $q < k - 1$, com $k = 3$

  Para exemplificar o Algoritmo 3.4 *no caso 2* será utilizado o grafo mostrado na Figura 3.2, com média $m_e = 7$ e desvio padrão $\sigma_e = 5{,}83$. Como $m_e + \sigma_e = 12{,}83$, apenas a aresta e_3 será removida. O procedimento então deverá continuar com as subsequentes remoções, e as arestas com os maiores pesos deverão ser removidas até que $q = k - 1$. Como $q < k - 1$, o algoritmo continua, removendo a próxima aresta com maior peso, e_5. Assim, o algoritmo termina, pois $q = k - 1 = 2$. O resultado do algoritmo evidencia três grupos G_1, G_2 e G_3 mostrados na Figura 3.2.

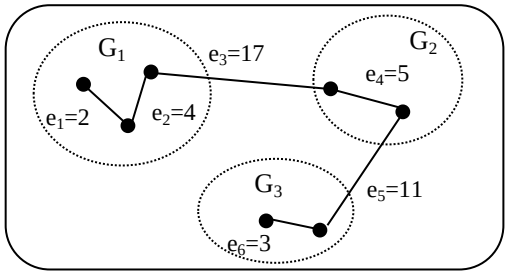

Figura 3.2 ASME utilizado como exemplo para o *caso 2*.

- *caso 3:* $q > k - 1$, com $k = 2$

Para exemplificar o Algoritmo 3.4 no *caso 3* será utilizada a Figura 3.3 (a) com média $m_e = 0{,}75$ e desvio padrão $\sigma_e = 2{,}17$. Como $m_e + \sigma_e = 2{,}92$, as arestas e_3 e e_5 serão removidas. Como $q > k - 1$, os centros dos três grupos G_1, G_2 e G_3 serão definidos utilizando a Tabela 3.1, com as coordenadas dos dados da Figura 3.3 (a) no espaço bi-dimensional. Os centros de G_1, G_2 e G_3 são determinados respectivamente como $c_1 = (2, 6)$, $c_2 = (6, 7)$ e $c_3 = (7, 1)$ (Definição 13 do Anexo A). Os dados representativos em relação aos centros c_1, c_2, e c_3 serão, respectivamente, d_2, d_4 e d_6, conforme a definição da Equação (3.2).

O ASMEH então repete os mesmos passos para o novo conjunto de dados S = {d_2, d_4 e d_6}. Após a criação do grafo completo, a ASM da Figura 3.3 (b) é construída, cuja média $m_e = 4{,}5$ e desvio padrão $\sigma_e = 0{,}71$. Como $m_e + \sigma_e = 5{,}21$ nenhuma aresta será removida e, portanto, $q < k - 1$, a ASM da Figura 3.3 (b) será tratada pelo *caso 2* já descrito anteriormente. Assim, a maior aresta e_1 será removida e, portanto, os grupos G_1 e G_2 serão formados conforme mostrado na Figura 3.3 (b). A Figura 3.3 (b) apresenta apenas a ASM reduzida considerado pelo procedimento com o objetivo de mostrar a remoção da aresta e_1. Portanto os dados d_1 e d_3 estão ocultos na Figura 3.3 (b), e pertencerão a G_1; d_5 e d_7 também ocultos na Figura 3.3 (b), pertencerão a G_2.

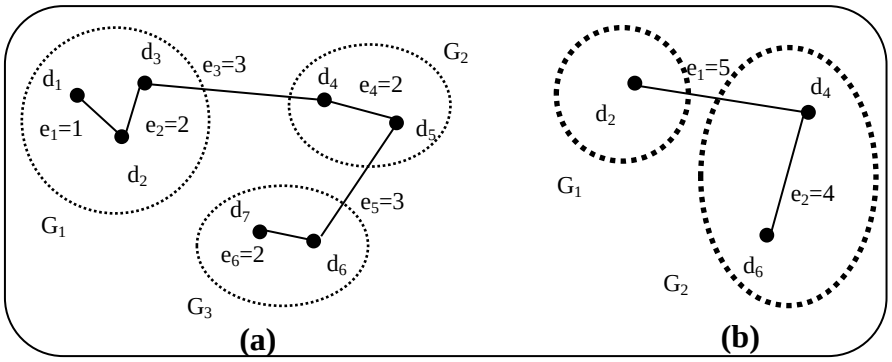

Figura 3.3 ASME utilizada como exemplo para o *caso 3*.
(a).ASME com remoção de e_3 e e_5 evidenciando 3 grupos inesperados.
(b) ASME reduzida gerada a partir de **(a)**.

Tabela 3.1 Coordenadas dos dados da Figura 3.3 (a).

	$\mathbf{d_1}$	$\mathbf{d_2}$	$\mathbf{d_3}$	$\mathbf{d_4}$	$\mathbf{d_5}$	$\mathbf{d_6}$	$\mathbf{d_7}$
coordenadas	(1,7)	(2,4)	(3,7)	(5,7)	(7,7)	(6,1)	(8,1)

3.4 AABG Baseado em ASM e na Identificação de Pontes Inconsistentes

3.4.1 Considerações Iniciais

O método de agrupamento a partir da construção da ASM considerando um conjunto de dados (representados por vértices da ASM) CD, discutido a seguir, é baseado naquele proposto em [Zahn 1971] e reescrito em [Theodoridis & Koutroumbas 1998].

Cada ponte (Definição 11 do Anexo A) da ASM definida por G_{ASM}= (V, E), é a ponte com menor peso que conecta dois vértices. Intuitivamente, dados em diferentes grupos deveriam ser conectados por pontes com maior peso na ASM. Essas pontes são chamadas de pontes inconsistentes e podem ser abordadas como conectando dois blocos de uma partição de V: P e V – P. A remoção das pontes inconsistentes da ASM induz os potenciais grupos de um agrupamento dos dados de V; entretanto, a remoção que considera apenas a distância euclidiana da ponte não produz agrupamentos razoáveis. Para obter bons agrupamentos, na maioria dos casos, é necessário impor algumas restrições ao critério de identificação das pontes que conectam partições diferentes de uma dada ASM. No caso, as métricas de média aritmética e desvio padrão foram consideradas.

3.4.2 Descrição e Análise do ASM-PI (ASM Baseado em Remoção de Pontes Inconsistentes)

O Algoritmo 3.5 descreve os passos para a indução de grupos a partir de um conjunto de dados CD. Basicamente, o algoritmo tem três passos principais:

- construção de um grafo completo G_i a partir de um conjunto de dados CD, onde a distância euclidiana entre dois dados é utilizada como peso das arestas de G_i (função Cria_Grafo_Completo() em Algoritmo 3.5);

- criação de uma G_{ASM} = (V,E) a partir de G_i, com V = CD. Este passo é realizado utilizando o algoritmo de Prim ou Kruskal, ambos apresentados na Seção 3.2 (função Cria_ASM() em Algoritmo 3.5);
- criação de um subconjunto com as pontes que estão a k passos da ponte atual (representado pela variável CE em Algoritmo 3.5);
- criação de componentes conexas da G_{ASM}, por meio da remoção de pontes inconsistentes (funções Remove_Aresta() e Identifica_Componentes_Conexas() em Algoritmo 3.5), baseando-se na média (função Calcula_Media() em Algoritmo 3.5) e desvio padrão (função Calcula_Desvio_Padrao() em Algoritmo 3.5), calculados a partir do subconjunto de pontes formado no passo anterior.

O parâmetro k determina que duas pontes e_i e e_j estão a k passos longe uma da outra, ou seja, se o mínimo caminho que conecta vértices de e_i aos vértices de e_j possuir um tamanho igual a k − 1, ou seja, contém k − 1 pontes. O parâmetro q é utilizado na comparação para determinar se uma ponte é considerada inconsistente conforme mostra Equação (3.3), em que w_{e_i} é o peso da aresta e_i, m_{e_j} é a média aritmética e σ_{e_j} é o desvio padrão das pontes $e_j \in E$ que estão a k passos de uma ponte $e_j \neq e_i$.

$$(w_{e_i} - m_{e_j}) / \sigma_{e_j} > q \tag{3.3}$$

Inicialmente o grafo completo G_i é construído a partir do conjunto de dados CD. O principal procedimento do Algoritmo 3.5 é determinar a ASM a partir de G_i (produzido pelo algoritmo de Prim ou Kruskal) e, após isso remover as pontes cujos pesos sejam excessivamente maior que o peso de suas vizinhas, condição que as caracteriza como inconsistentes. Na comparação, considera-se para cada ponte e_i, todas as outras e_j que estão a k passos de e_i, e então, calcula-se a média m_{e_j} e o desvio padrão σ_{e_j} de seus pesos. A condição para a eliminação da ponte é o w_{e_i} ser maior que q (dado como entrada) desvios padrões distantes da média. A remoção ocorrerá caso essa condição seja satisfeita, pois a ponte possivelmente estará conectando dois grupos diferentes. É importante notar que, ao contrário dos algoritmos 3.3 e 3.4, não é possível informar e portanto controlar, quantos grupos serão formados pelo algoritmo. Após a remoção das arestas consideradas

inconsistentes, cada componente conexa ou grupo é identificada pelo algoritmo e armazenada em AG.

```
Procedure ASM-PI(CD, k,q)

{entrada: CD: Conjunto de dados CD = {d1, d2,..., dn}, |CD| = n,
          k: quantidade de arestas entre ei e ej e
          q: valor limite utilizado na remoção de uma ponte.
saída:    AG: Agrupamento AG = {G1, G2, ... Gm} – um conjunto contendo m
              componentes conexas do grafo construído pelo algoritmo - cada
              componente conexa é um grupo do agrupamento obtido.}

begin
   AG ← ∅
   Gi ← Cria_Grafo_Completo(CD)
   GASM = (V, E) ← Cria_ASM (Gi)     {ASM_Kruskal(Gi) ou ASM_Prim(Gi, r)}
   for all ei ∈ E do
     begin
       CE ← ∅          {Conjunto com as arestas que estão a k passos de ei}
       for all ej ∈ E e Passos(ei, ej) <= k do
           CE = CE ∪ ej
       mej ← Calcula_Media(CE)
       σej ← Calcula_Desvio_Padrao(CE, mej)
       If (wei – mej)/σej > q then
            GASM ← Remove_Aresta(GASM, ei)
     end
    AG ← Identifica_Componentes_Conexas(GASM)
end
```

Algoritmo 3.5 Descrição do Algoritmo ASM-PI.

Considerando k = 2 e q = 3, no exemplo da Figura 3.4, as pontes que estão a dois passos da ponte e_0 são e_i, i = 1,2,...,11. Calcula-se o m_{e_j} = 2,45 e o σ_{e_j} = 1,03. Então e_0 está 7,33 desvios-padrão longe de m_{e_j} e, portanto, a ponte e_0 é inconsistente, já que 7,33 > q.

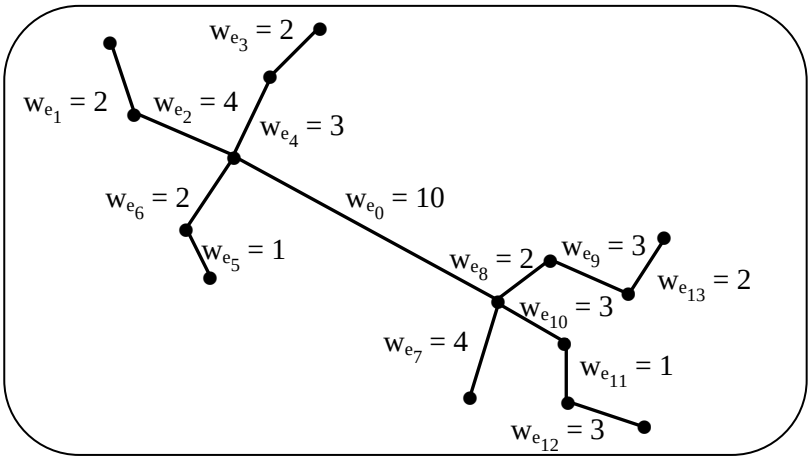

Figura 3.4 ASM com a ponte inconsistente e_0.

As diferentes maneiras de caracterizar inconsistência dão origem aos vários possíveis algoritmos que implementam o mecanismo geral da indução de grupos por meio da remoção das pontes inconsistentes. O Algoritmo 3.5 considera a média aritmética e o desvio padrão para a identificação das pontes da ASM que serão removidas. Outras métricas podem ser utilizadas sem a necessidade de alterações significativas na estrutura definida em Algoritmo 3.5.

3.5 AABG Baseado em ASM modificado

3.5.1 Considerações Iniciais

O algoritmo referenciado nesse texto como ASM-TXE, proposto em [Edla *et al.* 2012], foi utilizado em experimentos que envolvem dados biológicos disponibilizados no repositório do UCI Machine Learning [Lichman 2013]. Uma comparação com *k-means* [MacQueen 1967] e SFASM [Päivinen 2005] foi usada para validar a técnica proposta. O algoritmo ASM-TXE obteve resultados iguais e, em alguns casos, superiores aos dos outros dois algoritmos com os quais foi comparado (k-*means* e SFASM). Portanto, empiricamente foi mostrado que, além de ser simples e de fácil entendimento, o algoritmo é também eficaz. O ASM-TXE pode ser considerado similar ao Algoritmo 3.5, desde que algumas modificações, utilizadas no critério de remoção das pontes do ASM, sejam realizadas.

A principal dificuldade dos procedimentos envolvidos é decidir a quantidade de grupos k que leva a uma solução ótima. No procedimento descrito no Algoritmo 3.6 é

utilizada uma técnica baseada na comparação entre os pesos das arestas ordenadas para determinar o critério de parada das partições, e consequentemente, de k.

3.5.2 Análise e Descrição do ASM-TXE (ASM Baseado em Taxa de Erro)

O algoritmo ASM-TXE também utiliza uma ASM gerada a partir de um conjunto de dados, com suas arestas organizadas em ordem decrescente de seus pesos. Considere uma ASM formada por um conjunto com m arestas $E = \{e_1, e_2, ... e_m\}$ ordenadas decrescentemente conforme o valor de seus pesos. Dada duas arestas e_i e $e_{i+1} \in E$, tem-se que $w(e_i) > w(e_{i+1})$. Para determinar o valor de número de grupos k, o procedimento estabelece uma taxa de diferença entre os pesos das arestas e_{i+1} e e_i. Essa taxa, denominada taxa_erro, é a relação entre o peso de uma dada aresta e_{i+1} e sua antecessora e_i. A taxa_erro é definida pela Equação (3.4).

$$taxa\_erro = \frac{w(e_{i+1})}{w(e_i)} \times 100\% \tag{3.4}$$

Se a taxa entre os pesos de duas arestas em sequência não ultrapassar 50%, as partições subsequentes serão inibidas. O procedimento não verifica as relações de todos os pesos das arestas, pois o conjunto de arestas está ordenado, assim a averiguação pode ser interrompida logo que encontrar a primeira taxa menor que 50%.

O Algoritmo 3.6 descreve os passos necessários para o agrupamento de um conjunto de dados. Basicamente, o algoritmo constrói um grafo completo a partir de um conjunto de dados CD (via função Cria_Grafo_Completo() em Algoritmo 3.6). A ASM então é construída a partir do grafo completo e suas arestas são organizadas em ordem descrescente de seus pesos (via função Ordena_Arestas() em Algoritmo 3.6). O algoritmo então inicia um processo iterativo para seleção das arestas que serão removidas, baseando-se no cálculo da taxa de erro (função Calcula_Taxa() em Algoritmo 3.6). Ao término dessas iterações, as arestas selecionadas são removidas da ASM (via função Remove_Arestas() em Algoritmo 3.6). Cada grupo ou componente conexa é identificada através do procedimento Identifica_Componente_Conexas() no Algoritmo 3.6.

```
Procedure Agrupamento_ASM_TXE(CD)

{entrada: CD: Conjunto de dados CD = {d1, d2,..., dn}, |CD| = n.
saída:    AG: Agrupamento AG = {G1, G2, ... Gk} – um conjunto contendo as
            componentes conexas do grafo construído pelo algoritmo - cada
            componente conexa é um grupo do agrupamento obtido.}

begin
   Gi ← Cria_Grafo_Completo(CD)
   GASM = (V, E)← Cria_ASM (Gi)      {ASM_Kruskal(Gi) ou ASM_Prim(Gi, r)}
   Eord ← Ordena_Arestas(GASM)       {ordem decrescente de seus pesos}
   i ← 1
   termina ← false
   while i < |Eord| and not termina do
     begin
        taxa_erro ← Calcula_Taxa(ei, ei+1)
        If taxa_erro > 50% then
          begin
             i ← i + 1
             k ← i
          end
        else
             termina ← true
     end
  GASM ← Remove_Arestas(k, GASM, Eord)        {remove k arestas na ordem de Eord }
  AG ← Identifica_Componentes_Conexas(GASM)
end
```

Algoritmo 3.6 Descrição do Algoritmo ASM-TXE.

Para um melhor entendimento do Algoritmo 3.6, os passos de remoção de arestas da ASM da Figura 3.5 são exemplificados.

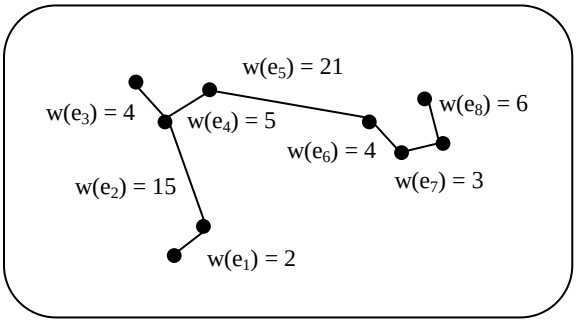

Figura 3.5 ASM para descrição de passos do ASM-TXE.

Os pesos das arestas na ASM da Figura 3.5 seguem a seguinte ordem descrescente: $w(e_5)$, $w(e_2)$, $w(e_8)$, $w(e_4)$, $w(e_3)$, $w(e_6)$, $w(e_7)$, $w(e_1)$. A Tabela 3.2 mostra os resultados das taxas de erros calculados a cada iteração. Note que na segunda iteração a taxa_erro < 50%

e, portanto, duas arestas serão removidas do conjunto de arestas organizado em ordem decrescente, ou seja $w(e_5)$ e $w(e_2)$ serão retiradas de ASM e três grupos serão evidenciados.

Tabela 3.2 Cálculo de taxa_erro para a ASM da Figura 3.5, usando o ASM-TXE.

iteração	**taxa_erro**
1	$\frac{15}{21} \times 100\% = 71\%$
2	$\frac{6}{15} \times 100\% = 40\%$

3.5 AABG Baseado em ASM com Estrutura de Escala Livre

3.5.1 Estrutura de Escala Livre – Considerações Iniciais

A topologia da estrutura de escala livre é caracterizada por focalizar vértices que possuem um alto grau de conectividade (chamados de *hubs*) *i.e.*, vértices que estejam ligados por meio de um número elevado de arestas a vértices com baixo grau de conectividade. Essa estrutura, devido à sua propriedade particular de representação do grafo, é utilizada para modelar redes em diversas áreas, tais como a Internet, redes sociais e de redes de computadores.

O modelo de redes de escala livre mais conhecido é o modelo de Barabási-Albert [Barabási & Albert 1999] que utiliza o conceito de crescimento incremental e preferencial. O crescimento incremental permite que um grafo aleatório com n vértices possua uma topologia de rede aberta em que, continuamente, um novo vértice v é acrescentado ao grafo. A adição de v, entretanto, é feita por meio de uma conexão preferencial, ligando v a um vértice específico do grafo. No modelo matemático de construção da rede de escala livre, proposto por Barabási-Albert, a probabilidade de um novo vértice ser conectado a um vértice v é dada pela razão entre o número atual de conexões de v e o número total de conexões da rede. Ou seja, a conexão é preferencialmente feita com os vértices de alto grau de conectividade. Este modelo tornou-se a base de estudo das propriedades das redes de escala livre.

3.5.2 O Algoritmo de Agrupamento para a Indução de Grafo com Estrutura de Escala Livre

O método de agrupamento que induz um grafo com estrutura de escala livre (*Clustering with Minimum Spanning Tree of Scale-free Structure- SFASM*), proposto em [Päivinen 2005], é baseado no conceito de árvore *spanning* mínima. O grafo induzido pelo algoritmo possui uma única componente conexa que deve ser interpretada da seguinte maneira: vértices com alto grau de conectividade representam centros de grupos que contém todos os vértices conectados a este *hub*. Na representação gráfica do grafo, as arestas representadas por longos segmentos de reta, são interpretadas como ligando vértices que não pertencem a nenhum dos grupos induzidos pelo algoritmo.

O algoritmo para a indução do grafo com estrutura de escala livre baseia-se no algoritmo de Prim (utilizado para a construção da ASM). Seus passos estão descritos no Algoritmo 3.7, no qual o grafo GSFASM = (S, P) é determinado pelo conjunto de vértices S e pelo conjunto de arestas P. Inicialmente o conjunto de aresta E é construído pela função Inicializa_E() com todas as possíveis arestas entre os vértices pertencentes ao conjunto de dados CD. A matriz D é calculada por meio do cálculo da distância euclidiana entre todos os pares formados a partir dos n dados ou vértices (função Inicializa_Distancia() em Algoritmo 3.7). A matriz de distância reversa W é então inicializada (função Inicializa_Distancia_Reversa() em Algoritmo 3.7) conforme mostra Equação (3.5).

$$W[d_i][d_j] = \lceil \max(D) \rceil - D[d_i][d_j],\ i = 1..n \text{ e } j = 1 .. n \qquad (3.5)$$

O vértice d_i com maior peso em relação à d_j corresponde a menor distância euclidiana entre d_i e d_j. O algoritmo sempre escolhe o vértice com maior peso através da função Seleciona_Maior_Peso(E, S), mantendo a conectividade do GSFASM e evitando a formação de ciclos.

As modificações no algoritmo de Prim, responsáveis pela estrutura diferenciada da ASM, são relativas à utilização de uma matriz com os valores das distâncias reversas entre os vértices W e o procedimento de atualização dos pesos das arestas. A atualização dos pesos permite o crescimento preferencial do grafo e é controlada por um parâmetro de entrada na, fornecido pelo usuário. O parâmetro na é utilizado para verificar se um vértice possui um número suficiente de arestas para que seu peso seja atualizado. A atualização do

peso de um vértice d_i em relação a todos os possíveis vértices d_j ao qual d_i pode se conectar, é dada pela Equação (3.6). A equação mostra que o ganho no peso (representado pela expressão $na \times c^{na}$) de um vértice d é levemente aumentado de acordo com o número de arestas na de d, e é diminuído quando na de d for suficiente. O parâmetro c é dado como entrada pelo usuário e preferencialmente tem o seu valor entre 0,5 e 1

$$W[d_i][d_j] = W[d_i][d_j] + na \times c^{na} \tag{3.6}$$

O procedimento Atualiza_Peso() do Algoritmo 3.7 é necessário para a formação da topologia de escala livre pois toda conexão tem um efeito no peso de um determinado vértice d, fazendo com que outros vértices próximos a v tenham preferência pela conexão com d.

O valor ideal para o parâmetro na, determinado empiricamente, é 3, indicando que um vértice do GSFASM precisa ter pelo menos 3 arestas para que haja atualização de seu peso. Os valores do parâmetro c podem ser analisados da seguinte maneira:

- c = 1, a estrutura do grafo tenderá a representar um único hub pois o ganho do peso sempre crescerá, conforme o aumento do número de arestas de um vértice.
- c = 0,1, a estrutura do grafo tenderá a representar uma ASM já que os pesos não tem um significativo ganho no peso.
- c = 0,5, é um valor ideal, pois o ganho do peso tende a diminuir quando o vértice possuir 10 arestas.

O Algoritmo 3.7 não constrói a usual rede de escala livre uma vez que o procedimento não é aleatório; cada vértice é escolhido pelo seu maior peso e pela sua conectividade com o grafo, de forma a manter a estrutura de árvore.

```
Procedure Agrupamento _SFASM(CD, na, c)

{ entrada: CD: Conjunto de dados CD = {d1, d2,..., dn}, |CD| = n,
           na:valor de limite inferior de arestas e
           c:valor utilizado para cálculo do ganho de peso c.
saída: GSFASM: grafo com estrutura escala livre GSFASM = (S, P), S é o
               conjunto de vértices e P é o conjunto de arestas
               do GSFASM}

begin
   S = ∅
   P = ∅
   E = Inicializa_E(CD)
   D = Inicializa_Distancia(E)
   W = Inicializa_Distância_Reversa(D)
   (u, v) = Seleciona_Maior_Peso_Inicial(E)
   S = S ∪ {u, v}
   P = P ∪ {(u, v)}
   E = E – {(u, v)}
   while |S| ≠|V| do
     begin
        (u, v) = Seleciona_Maior_Peso(E, S)   {verifica se u ∈ S e v ∉S}
        S = S ∪ {v}
        P = P ∪ {(u, v)}
        E = E – {(u, v)}
        Atualiza_Peso(W)
     end
end
```

Algoritmo 3.7 Descrição do Algoritmo SFASM.

3.5.3 Exemplos

No que segue um conjunto de dados do UCI Machine Learning [Lichman 2013] foi utilizado e teve a projeção de seus dados realizada através do software Pajek [Pajek & Vladimir 2008] e a biblioteca OpenGL. A Tabela 3.3 descreve os dados utilizados e suas principais características.

As figuras 3.8, 3.9 e 3.10 representam os grupos formados a partir do conjunto de 150 dados 4-dimensionais da planta íris, sem valor ausente de atributo. A classe da íris-setosa representada pela cor vermelha é linearmente separável das outras duas classes: íris-versicolor e íris-virgínica representadas pelas cores verde e azul, respectivamente.

Tabela 3.3 Descrição dos dados utilizados.

Dataset	#Instâncias	#Classes	#Instâncias/Classe
Iris	150	3	50/setosa 50/virginica 50/versicolor

A Figura 3.6 mostra o grafo de estrutura de escala livre para c = 0.5 e na = 3. Nota-se a formação de único grupo para a classe da íris-setosa e quatro grupos com dados das outras duas classes. A Figura 3.7 mostra o grafo de estrutura de escala livre com os valores de parâmetros c = 0.1 e na = 3, como esperado, o GSFASM induzido é similar à ASM. A Figura 3.8 mostra o GSFASM do conjunto de dados da planta íris com os parâmetros c = 1 e na = 3. Como esperado o grafo induzido possui um único hub.

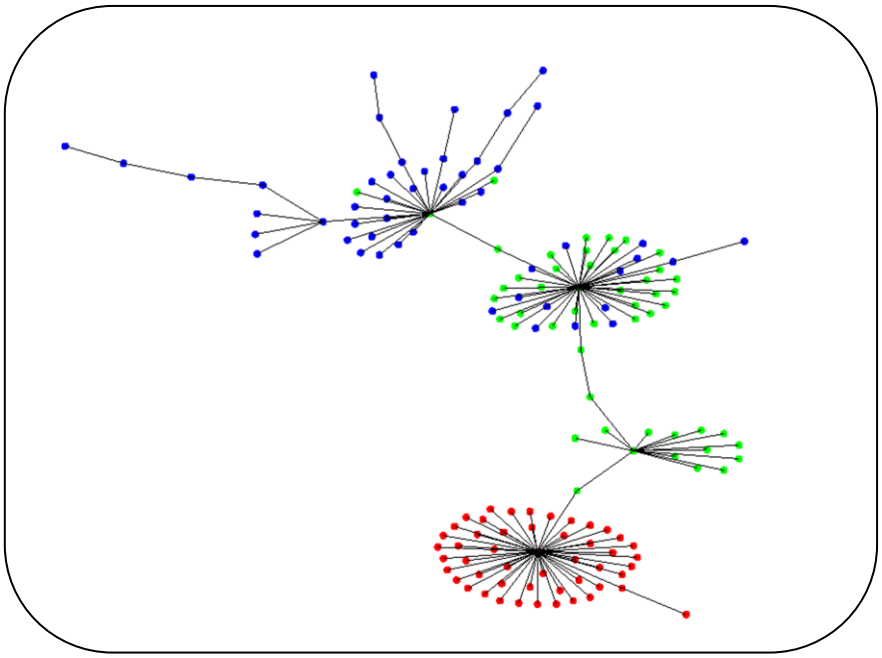

Figura 3.6 GSFASM para o conjunto de dados da planta íris com c = 0.5 e na = 3.

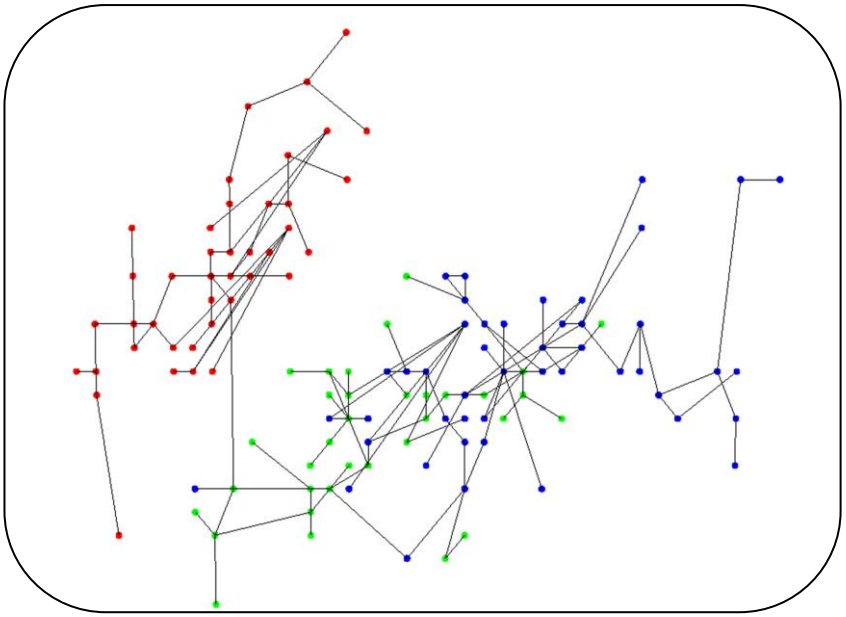

Figura 3.7 GSFASM para o conjunto de dados da planta íris com c = 0.1 e na = 3.

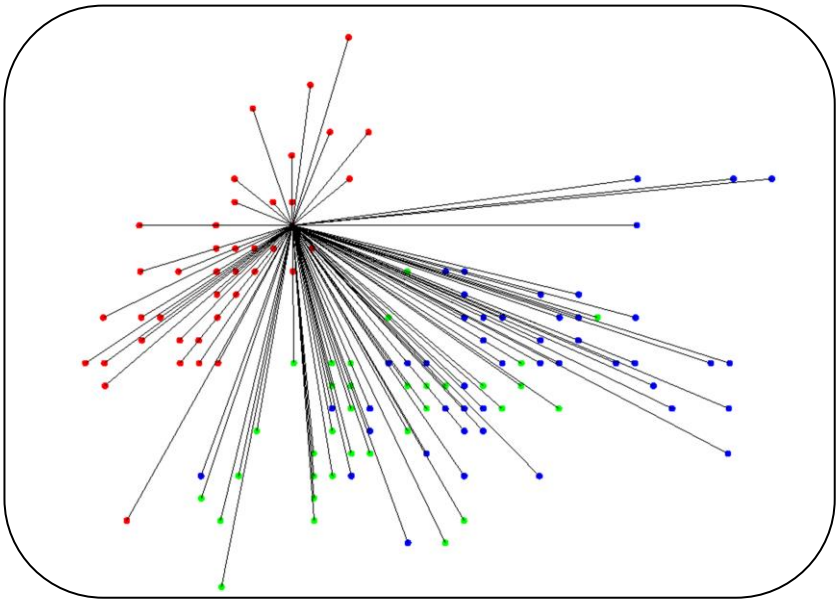

Figura 3.8 GSFASM para o conjunto de dados da planta íris com c = 1 e na = 3.

Capítulo 4

AABGs Baseados em Região de Influência

4.1 Considerações Iniciais

O Algoritmo 3.5 visto no Capítulo 3 pode ser estendido por meio da exploração do conceito de *região de vizinhança* (ou região de influência) [Toussaint 1980, Urquhart 1982, Asano 1988]. Nessa nova abordagem o conceito de ASM não é mais usado. O algoritmo que usa o conceito de região de vizinhança para determinar se existe (ou não existe) uma aresta entre quaisquer dois pontos ou dados induz um único grafo de vizinhança. Seu passo principal consiste em encontrar todos os pares de dados cujas respectivas regiões de vizinhança satisfazem uma dada propriedade P.

No que segue são apresentados o algoritmo para a indução do grafo de vizinhança bem como são revistas seis diferentes maneiras de definir região de vizinhança entre dois pontos (de dados) [Kasahara & Nicoletti 2009]. A maneira como a região de vizinhança é definida tem impacto direto no resultado de um algoritmo que faz uso desse conceito quando da inferência do grafo associado a um conjunto de dados.

4.2 Método de Agrupamento Baseado em Região de Influência

O algoritmo descrito em Algoritmo 4.1 é um algoritmo geral para a indução de um grafo de vizinhança, a partir de um conjunto de dados do R^2. O principal passo do Algoritmo 4.1 é a determinação da região de vizinhança entre dois vértices quaisquer de um grafo e a subsequente verificação da região, na satisfação de uma dada propriedade P. A propriedade P descrita no Algoritmo 4.1 consiste em inibir (ou não) a criação de uma aresta entre dois vértices d_i e d_j se houver (ou não) um terceiro vértice na região de vizinhança (determinada por d_i e d_j). Dependendo do formalismo matemático implementado na função *regiao_vizinhanca*(d_i, d_j) e da determinação da propriedade P, uma aresta conectando d_i a d_j pode (ou não) ser incluída no grafo – obviamente isso tem influência no número de

componentes conexas do grafo final, isto é, no número de grupos que o algoritmo irá produzir. Portanto dependendo da definição da região de vizinhança. bem como da propriedade P, diferentes grafos podem ser construídos.

```
Procedure Grafo_Vizinhança(CD)

{entrada: CD:conjunto de dados CD = {d1, d2, …, dn}. |CD| = n.
 saída:    AG: Agrupamentos AG = {G1, G2, ... Gm} – um conjunto contendo as
componentes conexas do grafo construído pelo algoritmo}
{O procedimento cria um conjunto de arestas E de um grafo inicial nulo (E=∅) G=(V,E)}

begin
    k ← 1 {índice de controle das arestas que são incluidas}
    E ← ∅
    for i ← 1 to n do
       for j ← i + 1 to n do
          begin
             U_di,dj ← Regiao_Vizinhanca(di,dj)
             if U_di,dj ∩ (CD – {di,dj}) = ∅  then {propriedade P da definição do grafo de
vizinhança}
                begin
                   ek ← (di,dj)
                   E ← E ∪ {ek}
                   k ← k + 1
                end
          end
end

Procedure Encontrar_Grupos(CD)
  begin
     G(V, E) ← Grafo_Vizinhanca(CD)
     AG ← Identifica_Componentes_Conexas(G)
  end
```

Algoritmo 4.1 Descrição do Algoritmo baseado na região de vizinhança.

4.3 As Diferentes Definições de Região de Vizinhança

Essa seção descreve as 6 diferentes maneiras de construção de região de vizinhança, identificadas como: Gabriel, Gabriel Modificado [Urquhart 1982], Vizinhança Relativa e Vizinhança Relativa Modificada [Urquhart 1982], Gabriel Elíptico [Park & Shin 2006] e β-Skeleton [Kirkpatrick 1985].

A seguir, os principais tipos de região de influência estudados e implementados são descritos para o espaço R^2. As notações utilizadas para descrever as equações matemáticas

são: $\delta(d_i,d_j)$ é a distância entre dois pontos d_i e d_j e $B(d_i,r)$ é um círculo aberto centrado em d_i e com raio r, i.e., $B(d_i,r) = \{d_j \mid \delta(d_i,d_j) < r\}$.

4.3.1 Região de Vizinhança de Gabriel

O grafo de Gabriel pode ser caracterizado como um grafo de vizinhança cuja região U_{d_i,d_j} induzida por dois pontos d_i e $d_j \in GV$ é um círculo com o diâmetro definido pela segmento de linha (d_i,d_j) (veja Figura 4.1(a)), como descrita na Equação (4.1) com a aresta $(d_i,d_j) \in E$ se e somente se $U_{d_i,d_j} \cap CD = \varnothing$. Conectando todos os pares vizinhos de determinados pela região de Gabriel, obtém-se o grafo de Gabriel.

$$U_{d_i,d_j} = B((d_i + d_j)/2,\delta(d_i,d_j)/2) \qquad (4.1)$$

Com a introdução do parâmetro σ definido pelo usuário, denominado ponte de consistência relativa, o círculo básico definido pela Equação 4.1 pode ser modificado conforme mostra a Equação (4.2) [Jaromczyk & Toussaint 1992]. O grafo de vizinhança correspondente é conhecido como grafo de Vizinhança de Gabriel Modificado (Figura 4.1(b)).

$$U_{d_i,d_j}(\sigma) = B((d_i + d_j)/2,\delta(d_i,d_j)/2) \cup \{x \mid \sigma \times \min\{\delta(d_i,x), \delta(d_j,x)\} < \delta(d_i,d_j)\} \qquad (4.2)$$

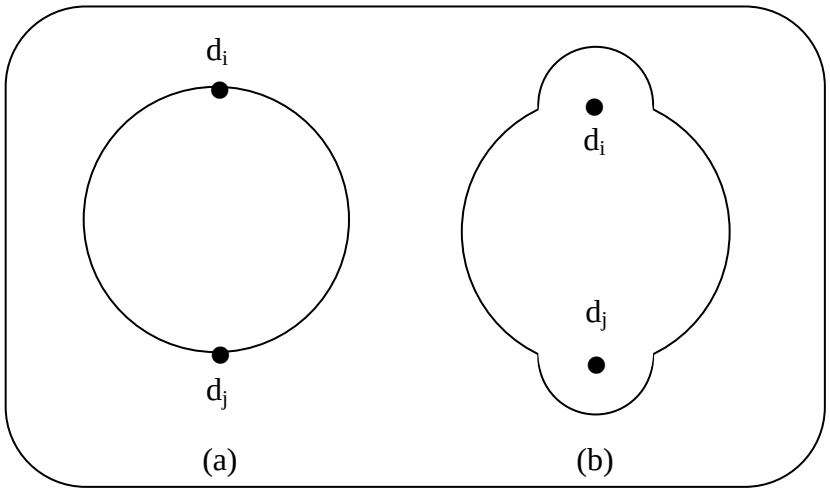

Figura 4.1 (a) Região de Vizinhança de Gabriel
(b) Região de Vizinhança de Gabriel Modificado.

Com a atribuição de diferentes valores ao fator (σ), a região de vizinhança pode ser modificada. Assim, diferentes agrupamentos podem ser formados, considerando que diferentes grupos (componentes conexos) podem ser formados, por meio da modificação da área de abrangência da região em volta dos pontos d_i e d_j.

4.3.2 Região Relativa de Vizinhança

O grafo Relativo de Vizinhança proposto em [Jaromczyk & Toussaint 1992], tem como região de vizinhança induzida por dois pontos d_i e $d_j \in CD$, U_{d_i,d_j} uma *luna*, como definida na Equação (4.3). Similarmente à região de vizinhança de Gabriel Modificado, uma aresta $(d_i,d_j) \in E$ se e somente se $U_{d_i,d_j} \cap CD = \varnothing$. O grafo relativo de vizinhança (Figura 4.2(a)) também possui uma versão modificada com a introdução do parâmetro σ, descrita na Equação (4.4), responsável pelo crescimento da região de vizinhança conforme mostra Figura 4.2(b).

$$U_{d_i,d_j} = B(d_i, \delta(d_i,d_j)) \cap B(d_j, \delta(d_i,d_j)) \tag{4.3}$$

$$U_{d_i,d_j}(\sigma)=B(d_i, \delta(d_i,d_j)) \cap B(d_j, \delta(d_i,d_j)) \cup \{x \mid \sigma \times \min\{\delta(d_i,x), \delta(d_j,x)\}< \delta(d_i,d_j)\} \tag{4.4}$$

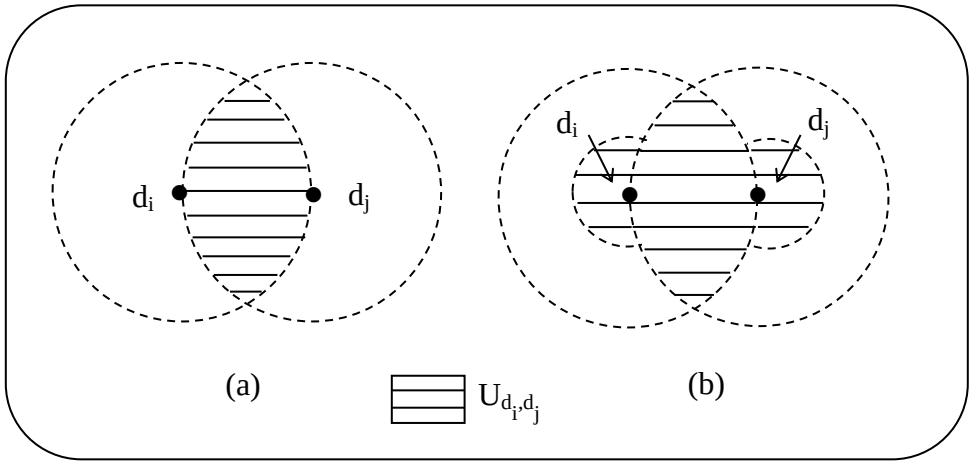

Figura 4.2 (a) Região Relativa de Vizinhança **(b)** Região Relativa de Vizinhança Modificada.

Conforme observado na região de vizinhança de Gabriel Modificado pelo valor do fator σ, esta região também possuirá a característica de particionar o conjunto de dados em mais componentes conexos já que houve um aumento no formato de sua região.

4.3.3 Região de Vizinhança de Gabriel Elíptico

O grafo de Gabriel Elíptico proposto em [Shin & Park 2006] tem como objetivo o aumento da quantidade de arestas no grafo de vizinhança induzido pela região elíptica. Para isso, um modelo matemático (veja Figura 4.3) foi construído para observar a influência de um ponto v em relação à conexão entre dois pontos, d_i e d_j. Intuitivamente nota-se que quanto mais próximo v estiver do ponto médio M entre d_i e d_j, mais provável e, portanto melhor, é a formação das pontes (v, d_i) e (v, d_j) do que (d_i, d_j). Portanto a influência de v na conexão de (p,q) está relacionada à distância entre v e M. Essa distância pode ser decomposta no eixo horizontal x e eixo vertical y, e ambos os eixos têm contribuições diferentes na influência de v na conexão entre d_i e d_j. Baseando-se no fato da influência de um ponto ser determinada pelo eixo y na elipse, o grafo de vizinhança de Gabriel Elíptico foi proposto.

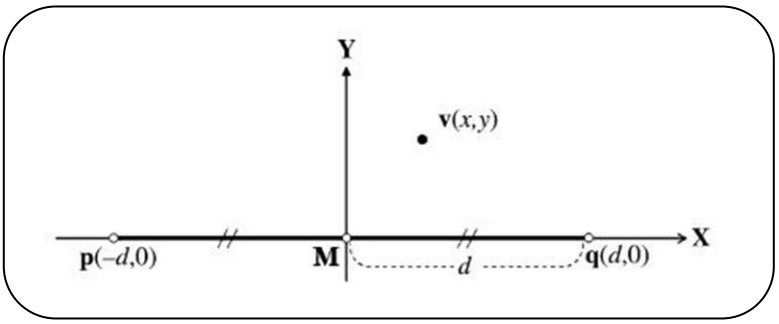

Figura 4.3 Modelo matemático da influência de v.

O grafo de Gabriel Elíptico é uma família parametrizada de grafos conforme mostram as equações (4.5), (4.6) e (4.7) nas quais o parâmetro α permite a elongação do formato das elipses ao longo do eixo y com o aumento de seu valor. Dependendo do valor de α, três situações diferentes podem ocorrer:

- Se α = 1 o resultado da figura geométrica é um círculo,
- Se α < 1 o resultado da figura geométrica é uma elipse com seu eixo x maior que seu eixo y e
- Se α > 1 o resultado da figura geométrica é uma elipse com seu eixo y maior que seu eixo x.

Considere dois pontos d_i e $d_j \in CD$ e suas coordenadas $d_i = (x_{d_i}, y_{d_i})$ e $d_j = (x_{d_j}, y_{d_j})$ e que $x_{d_j} > x_{d_i}$ e $y_{d_j} > y_{d_i}$. Dado o centro (x_c, y_c) do segmento de linha que liga estes dois pontos serem: $x_c = (x_{d_j} - x_{d_i})/2 + x_{d_i}$ e $y_c = (y_{d_j} - y_{d_i})/2 + y_{d_i}$. As três possíveis fórmulas que definem três regiões de vizinhanças elípticas são descritas nas equações (4.5), (4.6) e (4.7).

$\alpha=1$ $$U_{d_i,d_j}(\alpha) = \{<x,y> \mid (x - x_c)^2 + (y - y_c)^2 < (\delta(d_i,d_j)/2)^2\} \quad (4.5)$$

$\alpha < 1$ $$U_{d_i,d_j}(\alpha) = \{<x,y> \mid (x - x_c)^2 + ((y - y_c)/\alpha)^2 < (\delta(d_i,d_j)/2)^2\} \quad (4.6)$$

$\alpha > 1$ $$U_{d_i,d_j}(\alpha) = \{<x,y> \mid ((x - x_c)/\alpha) + (y - y_c)^2 < (\delta(d_i,d_j)/2)^2\} \quad (4.7)$$

Dado dois pontos d_i e d_j e o mesmo critério anteriormente descrito para estabelecer uma aresta entre d_i e d_j é adotado. Ou seja, (d_i e d_j) $\in$ E se e somente se $U_{d_i,d_j} \cap CD = \varnothing$. Analisando os valores de α conclui-se que:

- se $\alpha = 1$ o grafo de vizinhança de Gabriel Elíptico é idêntico ao grafo de vizinhança de Gabriel. Isso ocorre pois quando $\alpha = 1$, temos a equação do círculo definida tanto pela equação 5 quanto pela equação 1.
- se $\alpha = 0$ o grafo de vizinhança de Gabriel Elíptico é um grafo completo, pois com $\alpha = 0$, não será formada a região elíptica e, portanto, não haverá o crescimento da região no eixo y. Com isso todas as arestas serão conectadas já que $U_{d_i,d_j} \cap CD = \varnothing$ para $\forall d_i$ e $\forall d_j$.
- quanto maior o α, menos arestas entre d_i e d_j serão formadas, pois haverá um crescimento da região elíptica e portanto será mais provável que $U_{d_i,d_j} \cap CD \neq \varnothing$.

Figura 4.4 mostra um diagrama geral para a família parametrizada da região elíptica de Gabriel, destacando a influência do parâmetro α no formato da região de vizinhança.

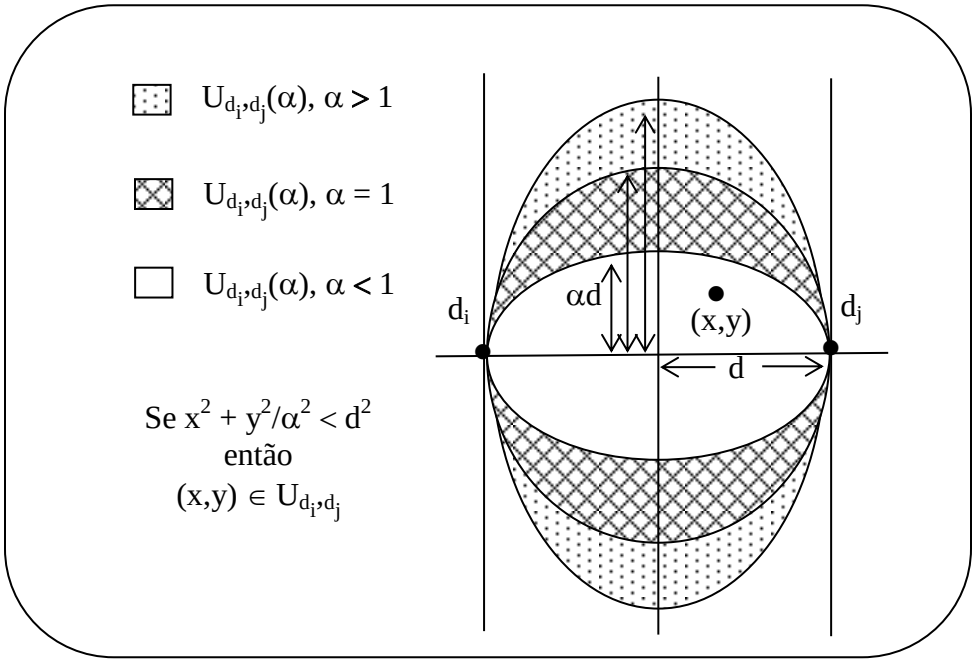

Figura 4.4 Família Parametrizada da região de Vizinhança Elíptica.

4.3.4 Região de Vizinhança de β–Skeleton

A família de grafos de vizinhança do β–Skeleton definida em [Kirkpatrick & Radke 1985] pode ser considerada como um caso geral dos grafos definidos anteriormente. Três casos podem ser considerados, dependendo do valor do parâmetro β.

- Se β = 1 o resultado da figura geométrica é um círculo
- Se 0 < β < 1 o resultado da figura geométrica é a intersecção de dois círculos
- Se 1< β < ∝ o resultado da figura geométrica é a união de dois círculos.

As regiões de vizinhanças do β–Skeleton são definidas pelas equações 4.8, 4.9 e 4.10.

$$\beta=1 \qquad U_{d_i,d_j}(\beta) = B((1-\beta/2)d_i + \beta/2\, d_j,\ \beta/2 \times \delta(d_i,d_j)) \cap B((1-\beta/2)d_j + \beta/2\, d_i,\ \beta/2 \times \delta(d_i,d_j)) \tag{4.8}$$

$$0<\beta<1 \qquad U_{d_i,d_j}(\beta) = B((1-\beta/2)d_i + \beta/2\, d_j,\ \delta(d_i,d_j)/(2\beta)) \cap B((1-\beta/2)d_j + \beta/2\, d_i,\ \delta(d_i,d_j)/(2\beta)) \tag{4.9}$$

$$1<\beta<\infty \qquad U_{d_i,d_j}(\beta) = B((1-\beta/2)d_i + \beta/2\, d_j,\ \beta/2 \times \delta(d_i,d_j)) \cap B((1-\beta/2)d_j + \beta/2\, d_i,\ \beta/2 \times \delta(d_i,d_j)) \tag{4.10}$$

A Figura 4.5 mostra um diagrama geral da família parametrizada do β–Skeleton enfatizando a influência do valor do parâmetro β no formato da região de vizinhança.

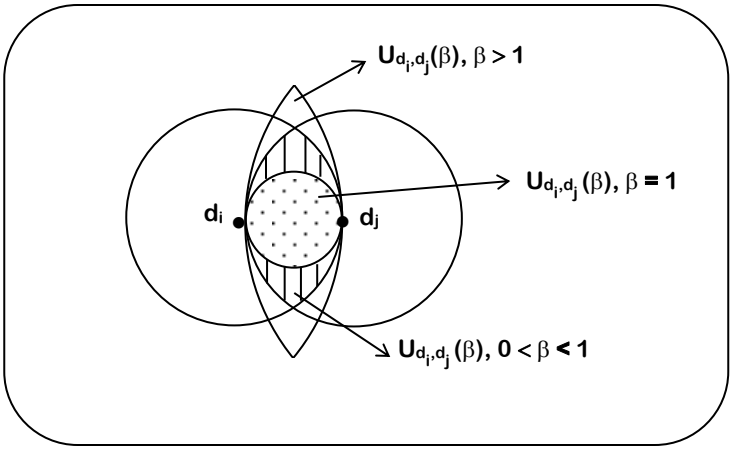

Figura 4.5 Família parametrizada da região de Vizinhança de β-Skeleton.

Capítulo 5

AABG Espectral – O Algoritmo de *Ncut*

5.1 Introdução

O algoritmo *Ncut* foi proposto em [Shi & Malik 2000] especificamente para o problema de segmentação de imagens. Via de regra é utilizado para particionar uma imagem digital em múltiplos segmentos, com o objetivo de simplificar ou mudar a representação da imagem para uma outra representação, mais significativa e de fácil análise. O critério de similaridade entre partes de uma imagem pode, por exemplo, ser estabelecido com base dos diversos tons de cinza associados aos pixels ou, então, à proximidade entre pixels, na imagem.

Considere CD como o conjunto de n dados e um grafo G = (V, E), construído a partir de CD de tal forma que V = CD = $\{d_1, d_2, ..., d_n\}$. O processo de agrupamento do grafo G visa particionar o conjunto de dados CD em k grupos ou subconjuntos disjuntos, $G_1, G_2, ..., G_k$ ($k < n$) de tal maneira que a medida de similaridade entre os dados pertencentes a cada um dos grupos G_i, $1 \leq i \leq k$ seja alta e, entre os dados pertencentes a G_i e a Gj, com $i \neq j$ e i, j =1, ...,k, seja baixa. O objetivo do algoritmo *Ncut* é encontrar o corte (Definição 20 do Anexo A) mínimo de um grafo de uma forma eficiente e sem a presença de grupos que contém apenas um vértice.

Considere um grafo G = (V, E) particionado em dois grupos distintos $G_1 = (V_1, E_1)$ e $G_2 = (V_2, E_2)$ tal que $V_1 \cup V_2 = V$ e $V_1 \cap V_2 = \varnothing$. No que segue são apresentadas algumas definições estabelecidas pelos autores, utilizadas para evitar a formação de G_1 ou G_2 com apenas um vértice.

Definição 5.1 (dissociação). O cálculo para a medida de dissociação entre grupos também chamada de *corte normalizado* (*Ncut*) é apresentada na Equação (5.1). O custo do

corte detalhado na Definição 20 do Anexo A, corresponde ao *cut*(G_1, G_2) utilizado na definição de *Ncut* na Equação (5.1).

$$\mathrm{Ncut}(G_1,G_2)=\frac{\mathrm{cut}(G_1,G_2)}{\mathrm{assoc}(G_1,G)}+\frac{\mathrm{cut}(G_1,G_2)}{\mathrm{assoc}(G_2,G)} \tag{5.1}$$

em que $\mathrm{assoc}(G_1,G)=\sum_{V_1\in G_1,V_2\in G} w(d_i,d_j)$, com é a soma dos pesos das arestas de todos os vértices d_i de G_1 para todos os vértices d_j de G. De forma similar é definido o $\mathrm{assoc}(G_2,G)=\sum_{V_1\in G_2,V_2\in G} w(d_i,d_j)$ como a soma dos pesos das arestas de todos os vértices d_i de G_2 para todos os vértices d_j de G.

Definição 5.2 (associação). Da mesma forma, determina-se a medida de associação dentro dos grupos conforme mostra a Equação (5.2).

$$\mathrm{Nassoc}(G_1,G_2)=\frac{\mathrm{assoc}(G_1,G_1)}{\mathrm{assoc}(G_1,G)}+\frac{\mathrm{assoc}(G_2,G_2)}{\mathrm{assoc}(G_2,G)} \tag{5.2}$$

em que $\mathrm{assoc}(G_1,G_1)=\sum_{V_1\in G_1,V_2\in G_1} w(d_i,d_j)$, isto é, $\mathrm{assoc}(G_1,G_1)$ é a soma de todos os pesos das arestas pertencentes a G_1. De forma similar é definido o $\mathrm{assoc}(G_2,G_2)=\sum_{V_1\in G_2,V_2\in G_2} w(d_i,d_j)$ como a soma de todos os pesos das arestas pertencentes a G_2.

O principal problema deste processo está em encontrar o corte mínimo do grafo, de forma que a partição seja considerada ótima, ou seja, o corte deverá maximizar a medida de associação e minimizar a medida de dissociação.

Um exemplo de corte mínimo normalizado de um grafo que evita a geração de vértices isolados é mostrado na Figura 5.1. Considere na Figura 5.1 as duas diferentes bipartições do grafo G em $G_1=(V_1, E_1)$ e $G_2=(V_2, E_2)$ e o *cut*(G_1, G_2) = 1. Todas as arestas possuem a mesma medida de similaridade igual a 1. Na Figura 5.1(a) observa-se que $V_1=\{d_1, d_2, d_3, d_4, d_5\}$ e $V_2=\{d_6, d_7, d_8\}$ com assoc(G_1, G) = 5 e assoc(G_2, G) = 4. Portanto o *Ncut*(G_1,G_2) = 1/5+1/4=0,45. Na Figura 5.1(b) observa-se que $V_1=\{d_1\}$ e $V_2 = \{d_2, d_3, d_4,$

d_5, d_6, d_7, d_8} com assoc(G_1, G) = 1 e assoc(G_2, G) = 7. Portanto o *Ncut*(G_1,G_2) = 1/1+1/7=1,14. Como 0,45 < 1,14 o melhor corte mínimo é o mostrado em Figura 5.1(a).

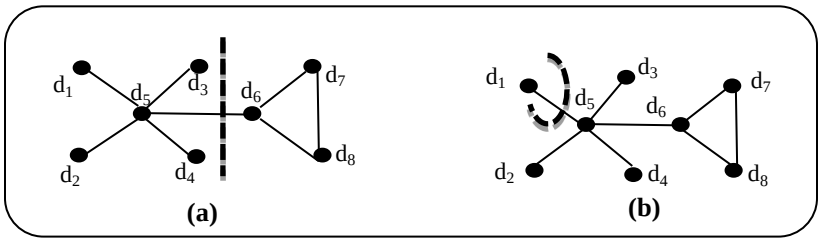

Figura 5.1 (a) bipartição de G em dois subgrafos cujos conjuntos de vértices são, respectivamente, V_1={d_1, d_2, d_3, d_4, d_5} e V_2={d_6, d_7, d_8}
(b) bipartição de G em dois subgrafos cujos conjuntos de vértices são, respectivamente, V_1={d_1} e V_2={d_2, d_3, d_4, d_5, d_6, d_7, d_8}.

O custo para encontrar o corte mínimo de um grafo é considerado exponencial devido a enorme quantidade de partições possíveis. Devido a isso, essa questão pode ser simplificada utilizando a solução para o problema dos autovalores e autovetores da Álgebra Linear (Definição 15 do Anexo A) para a matriz de adjacência que representa o grafo que será particionado.

A solução do problema dos autovetores pode ser utilizado para a escolha de uma partição eficiente de um grafo G=(V,E). Ou seja, para particionar V em dois grupos G_1 e G_2 considere z, um autovetor indicador, tal que, se $z_i = 1$ então o vértice i pertencerá a G_1, caso contrário, $z_i = -1$ então i pertencerá a G_2. Para modelar matematicamente o problema de encontrar um corte mínimo ótimo, considere:

- uma matriz A como uma matriz ponderada, ou seja, A(i,j) = w_{ij} onde w_{ij} é o custo ou peso de similaridade da aresta que conecta os vértices i e j.

- uma matriz D como uma matriz que contém a soma dos custos ou pesos associados ao vértice i, ou seja, D(i,i) = $\sum_j w(i,j)$ e D(i,j) = 0.

A solução eficiente para o critério de corte normalizado para a partição de um grafo pode ser resolvida pelo problema dos autovalores conforme mostra a Equação (5.3).

$$D^{-\frac{1}{2}}(D-A)D^{-\frac{1}{2}}z = \lambda z \tag{5.3}$$

em que z é o segundo menor autovetor associado ao autovalor λ. Não há a necessidade de apurar a precisão do valor de z, pois a partição do grafo considera apenas se seu valor é positivo ou negativo.

5.2 Algoritmo *Ncut*

Em Algoritmo 5.1 é apresentado o pseudocódigo para o procedimento de *Ncut*. Inicialmente, é construído o grafo a partir do conjunto de dados CD (função Cria_Grafo() em Algoritmo 5.1), com o peso das arestas definidos por uma medida de similaridade. Ou seja, quanto maior o peso da aresta entre os dados d_i e d_j, maior é a similaridade entre d_i e d_j. Este conceito é o oposto ao utilizado pela distância euclidiana pois esta representa uma medida de dissimilaridade. Nos experimentos descritos na Seção 9, é utilizado o valor do inverso da distância euclidiana dist, i.e., 1/dist. Também é utilizado, neste passo, o grafo completo. Após a construção do grafo G, a matriz laplaciana pode ser definida (função Cria_Matriz_Laplaciana() em Algoritmo 5.1). A partir disso, os autovetores podem ser calculados e seu segundo autovetor (função Segundo_Autovetor() em Algoritmo 5.1) é utilizado para biparticionar G (função Particiona_Grafo() em Algoritmo 5.1).

```
Procedure Ncut(CD, i)
{entrada:  CD: Conjunto de dados CD = {d1, d2,..., dn}, n = |CD| e
           i: número de iterações.
saída: AG: Agrupamento AG = {G1, G2, ... Gm} – um conjunto contendo as
componentes conexas do grafo construído pelo algoritmo - cada componente
conexa é um grupo do agrupamento obtido.}

begin
   k ← 0
   while k < i do
      begin
        AG ← ∅
        G = (V, E) ← Cria_Grafo(CD)
        L ← Cria_Matriz_Laplaciana(G)          {Definição 18 Anexo A}
        z ← Segundo_Autovetor(L)
       G ← Particiona_Grafo(G, z)
       k ← k + 1
     end
  AG ← Identifica_Componentes_Conexas(G)
end
```

Algoritmo 5.1 Pseudocódigo alto nível do algoritmo *Ncut*.

Com o objetivo de facilitar a determinação do término do Algoritmo 5.1, um parâmetro i, que determina o número de bipartições, é utilizado como critério de parada do algoritmo. Este parâmetro não é especificado em [Shi & Malik 2000].

Para exemplificar os passos descritos em Algoritmo 5.1, o grafo apresentado na Figura 5.2 será utilizado para o detalhamento dos passos que levam a sua bipartição. Os valores da matriz de adjacência representados pelos pesos das arestas são mostrados na Tabela 5.1. O peso de uma dada aresta e_{ij} é definida pelo inverso da distância euclidiana entre os vértices i e j.

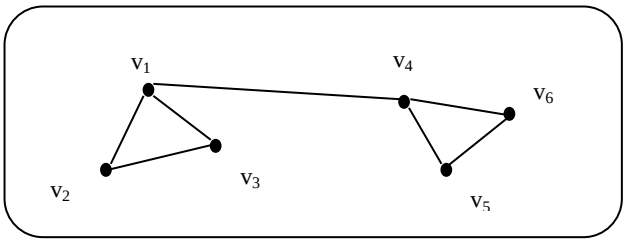

Figura 5.2 Grafo com dois grupos evidentes.

Tabela 5.1 Pesos das arestas do grafo da Figura 5.2.

$w(e_{12})$	$w(e_{13})$	$w(e_{14})$	$w(e_{23})$	$w(e_{45})$	$w(e_{46})$	$w(e_{56})$
10	15	2	12	12	14	15

Na construção da matriz laplaciana é utilizada a Definição 18 do Anexo A, ou seja, será definida uma matriz laplaciana não-normalizada com os seguintes valores:

$$L = \begin{pmatrix} 37 & -10 & -15 & -12 & 0 & 0 \\ -15 & 27 & -12 & 0 & 0 & 0 \\ -15 & -12 & 27 & 0 & 0 & 0 \\ -2 & 0 & 0 & 26 & -12 & -14 \\ 0 & 0 & 0 & -12 & 27 & -15 \\ 0 & 0 & 0 & -14 & -15 & 29 \end{pmatrix}$$

em que linha i e a coluna j correspondem aos vértices v_i e v_j da Figura 5.2. Os autovalores de L são $\begin{bmatrix} -0{,}60720 \\ 4{,}097434 \\ 37{,}48149 \\ 38{,}99999 \\ 43{,}44411 \\ 49{,}58415 \end{bmatrix}$ e o segundo autovetor tem seus valores iguais a $\begin{bmatrix} -5{,}54415 \\ -7{,}62777 \\ -7{,}62776 \\ 0{,}68978 \\ 1{,}01637 \\ 1 \end{bmatrix}$

Os vértices devem ser categorizados em dois grupos G_1 e G_2 conforme o sinal de cada valor do segundo autovetor, conforme mostra a Tabela 5.2. Note que se os sinais dos valores do segundo autovetor forem iguais, então não haverá bipartição do grafo.

Tabela 5.2 Vértices da Figura 5.2 e seus correspondentes grupos.

vértice	valor	Grupo
$\mathbf{v_1}$	-5,54415	G_1
$\mathbf{v_2}$	-7,62777	G_1
$\mathbf{v_3}$	-7,62776	G_1
$\mathbf{v_4}$	0,68978	G_2
$\mathbf{v_5}$	1,01637	G_2
$\mathbf{v_6}$	1	G_2

Capítulo 6

Avaliação dos Agrupamentos induzidos por AABGs

6.1 Introdução aos Índices Relativos e Internos

A apresentação que segue pode ser considerada para qualquer algoritmo de agrupamento; o foco do trabalho, no entanto, são os AABGs. Os índices de validação de agrupamentos servem para dois propósitos: determinação de um número plausível de grupos e designação do melhor agrupamento formado pelos algoritmos [Pakhira 2005]. A qualidade de um AABG depende da estrutura do grafo, construído a partir do conjunto de dados, e da técnica utilizada pelo procedimento. Portanto é necessário uma avaliação apropriada de um índice com o objetivo de analisar a estrutura do grafo e verificar a eficácia do AABG.

Basicamente os índices de avaliação de agrupamentos são definidos através da combinação de duas propriedades [Rendón et al. 2011]:

- Compacidade

  Relacionada à medida de proximidade dos dados pertencentes aos grupos. Essa medida é denominada variância. Quanto menor a variância entre os dados de um grupo, mais próximo os dados estarão um do outro, e portanto melhor será o resultado do agrupamento.
- Separabilidade

  Medida baseada na distância entre os dados representativos dos grupos com o objetivo de determinar o quão afastados os grupos estão um do outro. Para grupos distintos, a medida de separabilidade será maior, e portanto melhor será o resultado do agrupamento.

Diversos índices para a avaliação dos grupos produzidos pelos algoritmos de agrupamento foram propostos, em uma tentativa de contornar o problema de ter que informar, ao algoritmo, o número de grupos que o agrupamento a ser construído deverá possuir. Como esse parâmetro é muitas vezes desconhecido, a estratégia para obter o melhor agrupamento é executar diversas vezes o algoritmo, variando o valor do parâmetro de número de grupos. O índice de validação deve ser calculado em todas as execuções e o número de grupos que contribuiu para a obtenção do melhor valor de índice, é o que deve ser mantido [Gunter & Bunke 2003].

Também, como pode ser evidenciado na literatura, uma das principais dificuldades associadas a algoritmos de agrupamento em geral, diz respeito à avaliação dos resultados produzidos por tais algoritmos. Vários deles, inclusive os mais populares, não têm bom desempenho em muitos tipos de dados (ver, por exemplo [Zaïane et al. 2002]). Além disso, algoritmos de agrupamento, via de regra, dependem de um conjunto de parâmetros; apenas com o estabelecimento de valores adequados a esses parâmetros é que esses algoritmos tendem a produzir bons resultados. Para determiná-los, entretanto, é imprescindível que a qualidade do resultado produzido pelo algoritmo possa ser avaliada e, para isso, podem ser encontrados na literatura inúmeros índices de avaliação.

Vários estudos que investigam tais índices os organizam em três categorias distintas [Theodoridis & Koutroubas 2001] [Kovacs et al. 2001]: (1) índices internos, (2) índices externos e (3) índices relativos. Como comentado em [Kovacs et al. 2001], tanto os índices internos quanto os externos são baseados em métodos estatísticos e são, geralmente, computacionalmente custosos. Índices externos avaliam o agrupamento com base em algum critério (geralmente intuitivo) fornecido pelo usuário enquanto que índices internos fazem uso de métricas aplicadas tanto ao conjunto de dados quanto ao método de agrupamento usado. Índices relativos são direcionados pela comparação entre diferentes esquemas de agrupamento. Um ou mais algoritmos de agrupamentos são executados múltiplas vezes, com diferentes valores de parâmetros de entrada, mas tendo sempre como entrada o mesmo conjunto de dados. O objetivo do índice relativo é escolher o melhor esquema de agrupamento com base nos diferentes resultados obtidos. A base de comparação é o índice de avaliação. A chamada *validação interna* de agrupamentos, utilizando índices, é realizada por meio de um procedimento que avalia o resultado

utilizando uma função quantitativa e objetiva. Via de regra índices internos são fortemente baseados em medidas estatísticas e consideram as propriedades ou a estrutura do conjunto de dados.

Nas seções seguintes são descritos e detalhados os índices que serão usados neste trabalho. Esses índices podem ser utilizados para avaliar o melhor resultado de um algoritmo de agrupamento para um mesmo conjunto de dados, através da comparação dos resultados obtidos por diferentes algoritmos, ou pelo mesmo algoritmo, com a diversificação dos valores dos parâmetros exigidos pelo procedimento.

6.2 Índice de *Davies-Bouldin – DB* (1979)

Esse índice foi introduzido por David L. Davies e Donald W. Bouldin em 1979 [Davies & Bouldin 1979] e é representado pela Equação (6.1).

$$DB = \frac{1}{m}\sum_{i=1}^{m} \max(dist_{ij})_{j=1...m;\ j\neq i},\ dist_{ij} = \frac{\sigma_i + \sigma_j}{dist(c_i, c_j)} \tag{6.1}$$

Na Equação (6.1) m representa o número de grupos, σ_i é a média da distância entre todos os dados pertencentes ao grupo i e o seu centro, o mesmo ocorre para σ_j, e, dist(c_i, c_j) é a distância entre os centros dos grupos c_i e c_j. A distância pode ser medida utilizando a distância euclidiana. Considere o exemplo mostrado na Figura 6.1 com $d_1 \in G_1$, d_2, $d_3 \in G_2$ e d_4, d_5, $d_6 \in G_3$. Os cálculos realizados são apresentados nas Tabelas 6.1, 6.2, 6.3, 6.4 e 6.5.

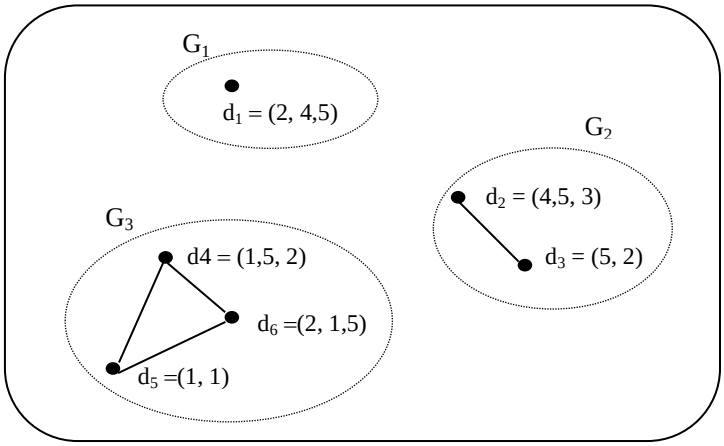

Figura 6.1 Exemplo de dados para cálculo dos índices.

Tabela 6.1 Centro e σ do grupo 1 da Figura 6.1.

G_1	
Centro	$c_1 = \frac{(2,4,5)}{1} = (2,4,5)$
σ	$\sigma_1 = \frac{0}{1} = 0$

Tabela 6.2 Centro e σ do grupo 2 da Figura 6.1.

G_2	
Centro	$c_2 = \frac{(4,5+5,\ 3+2)}{2} = (4,25,\ 2,5)$
σ	$\sigma_2 = \frac{\sqrt{(4,5-4,25)^2+(3-2,5)^2}+\sqrt{(5-4,25)^2+(2-2,5)^2}}{2} = 0,73$

Tabela 6.3 Centro e $\boldsymbol{\sigma}$ do grupo 3 da Figura 6.1.

G_3	
Centro	$c_3 = \frac{(1,5+2+1,\ 2+1,5+1)}{3} = (1,5,\ 1,5)$
σ	$c_3 = \frac{\sqrt{(1,5-1,5)^2+(2-1,5)^2}+\sqrt{(2-1,5)^2+(1,5-1,5)^2}+\sqrt{(1-1,5)^2+(1-1,5)^2}}{3} = 0,57$

Com as medidas dos centros c e σ, pode-se calcular o $dist_{ij}$ conforme mostrado na Tabela 6.4.

Tabela 6.4 Distâncias entre os centros dos grupos da Figura 6.1.

Distância entre os centros dos grupos	
$\mathbf{d_{12}}$	$dist_{12} = \frac{0 + 0,73}{\sqrt{(2-4,25)^2 + (4,5-2,5)^2}} = 0,24$
$\mathbf{d_{13}}$	$dist_{13} = \frac{0 + 0,57}{\sqrt{(2-1,5)^2 + (4,5-1,5)^2}} = 0,1875$
$\mathbf{d_{23}}$	$dist_{23} = \frac{0,73 + 0,57}{\sqrt{(4,25-1,5)^2 + (2,5-1,5)^2}} = 0,44$

Tabela 6.5 Valor máximo das distâncias entre os centros dos grupos da Figura 6.1.

$\mathbf{max(d_{12}, d_{13})}$	max(0,24, 0,1875) = 0,24
$\mathbf{max(d_{21}, d_{23})}$	max(0,24, 0,44) = 0,44
$\mathbf{max(d_{31}, d_{32})}$	max(0,1875, 0,44) = 0,44

O índice então pode ser calculado DB = 1/3 (0,24 + 0,44 + 0,44) = 0,37. Observa-se que a distância entre os centros dos grupos $dist_{ij}$ tem seu valor minimizado se os grupos i e j são compactos e seus centros estão relativamente distantes um do outro, portanto quanto menor o valor de DB, melhor será o agrupamento.

6.3 Índice de *Dunn* (1974)

Esse índice foi introduzido por J.C. Dunn em 1974 [Dunn 1973] e é baseado em medidas geométricas dos grupos e, assim como no DB, considera as características de densidade e separação. O índice é definido pela Equação (6.2).

$$D=\frac{dist_{min}}{dist_{max}},\ dist_{min}=min_{i,j\in\{1,\ldots,m\},i\neq j}dist_{d_i,d_j}\ \text{e}\ dist_{max}=max_{i,j\in\{1,\ldots,m\},i=j}dist_{d_i,d_j} \quad (6.2)$$

Na Equação (6.2) m representa o número de grupos e d_i e d_j dados pertencentes a diferentes grupos G_1 e G_2, respectivamente; $dist_{min}$ é o valor mínimo da distância entre dois dados pertencentes grupos diferentes e $dist_{max}$ é o valor máximo da distância entre dois dados pertencentes ao mesmo grupo. Grupos considerados compactos possuem um maior valor de $dist_{min}$ e um menor valor para $dist_{max}$. Portanto, maximixar a função D significa encontrar o melhor resultado. Outras variantes do índice *Dunn* são mostradas e detalhadas em [Vendramin *et al.* 2010]. Na Figura 6.1 o $dist_{min} = dist(d_1, d_4) = \sqrt{(2-1{,}5)^2 + (4{,}5-2)^2} = 2{,}55$ e o $dist_{max} = d(d_4, d_6) = d(d_5, d_6) = d(d_2, d_3) = \sqrt{0{,}5^2 + 1^2} = 1{,}12$. Portanto $D = \frac{2{,}55}{1{,}12} = 2{,}28$.

6.4 Índice de *C* (1976)

Esse índice foi criado por Hubert e Schultz em 1976 [Hubert & Schultz 1976] e também é baseado em distâncias internas e externas entre os grupos do agrupamento. É representado pela Equação (6.3).

$$C=\frac{S - S_{min}}{S_{max}-S_{min}} \quad (6.3)$$

onde S é a soma das distâncias entre todos os n pares de dados pertencentes ao mesmo grupo. A variável S_{min} é a soma das n menores distâncias considerando todos os pares de dados do agrupamento e S_{max} corresponde a soma das n maiores distâncias considerando todos os pares de dados do agrupamento. Essa função deve ser minimizada para um melhor agrupamento e seus valores pertencem ao intervalo do conjunto [0, 1]. Quanto menor o valor de *C* for, melhor o agrupamento pois o numerador da Equação (6.3) indica que os dados mais próximos pertencem ao mesmo grupo.

A Tabela 6.6 mostra os valores em ordem crescente das distâncias entre todos os pares de dados mostrados na Figura 6.1.

Tabela 6.6 Valores das distâncias em orderm crescente entre os dados da Figura 6.1.

$\mathrm{dist}_{d_3 d_5} = \mathrm{dist}_{d_5 d_3}$	$\sqrt{(5-1)^2 + (2-1)^2} = 4{,}12$
$\mathrm{dist}_{d_2 d_5} = \mathrm{dist}_{d_5 d_2}$	$\sqrt{(4{,}5-1)^2 + (3-1)^2} = 4{,}03$
$\mathrm{dist}_{d_1 d_3} = \mathrm{dist}_{d_3 d_1}$	$\sqrt{(2-5)^2 + (4{,}5-2)^2} = 3{,}90$
$\mathrm{dist}_{d_1 d_5} = \mathrm{dist}_{d_5 d_1}$	$\sqrt{(2-1)^2 + (4{,}5-1)^2} = 3{,}64$
$\mathrm{dist}_{d_3 d_4} = \mathrm{dist}_{d_4 d_3}$	$\sqrt{(5-1{,}5)^2 + (2-2)^2} = 3{,}5$
$\mathrm{dist}_{d_2 d_4} = \mathrm{dist}_{d_4 d_2}$	$\sqrt{(4{,}5-1{,}5)^2 + (3-2)^2} = 3{,}16$
$\mathrm{dist}_{d_3 d_6} = \mathrm{dist}_{d_6 d_3}$	$\sqrt{(5-2)^2 + (2-1{,}5)^2} = 3{,}04$
$\mathrm{dist}_{d_1 d_6} = \mathrm{dist}_{d_6 d_1}$	$\sqrt{(2-2)^2 + (4{,}5-1{,}5)^2} = 3$
$\mathrm{dist}_{d_1 d_2} = \mathrm{dist}_{d_2 d_1}$	$\sqrt{(2-4{,}5)^2 + (4{,}5-3)^2} = 2{,}91$
$\mathrm{dist}_{d_2 d_6} = \mathrm{dist}_{d_6 d_2}$	$\sqrt{(4{,}5-2)^2 + (3-1{,}5)^2} = 2{,}91$
$\mathrm{dist}_{d_1 d_4} = \mathrm{dist}_{d_4 d_1}$	$\sqrt{(2-1{,}5)^2 + (4{,}5-2)^2} = 2{,}55$
$\mathrm{dist}_{d_2 d_3} = \mathrm{dist}_{d_3 d_2}$	$\sqrt{(4{,}5-5)^2 + (3-2)^2} = 1{,}12$
$dist_{d_4 d_5} = dist_{d_5 d_4}$	$\sqrt{(1{,}5-1)^2 + (2-1)^2} = 1{,}12$
$\mathrm{dist}_{d_5 d_6} = \mathrm{dist}_{d_6 d_5}$	$\sqrt{(1-2)^2 + (1-1{,}5)^2} = 1{,}12$
$\mathrm{dist}_{d_4 d_6} = \mathrm{dist}_{d_6 d_4}$	$\sqrt{(1{,}5-2)^2 + (2-1{,}5)^2} = 0{,}71$

De acordo com a Tabela 6.6, S é determinado pela soma das distâncias dentro de cada grupo, i.e., em G_1, $S_1 = 0$, em G_2, $S_2 = \text{dist}(d_4, d_5) + \text{dist}(d_4, d_6) + \text{dist}(d_5, d_6) = 2{,}98$ e, em G_3, $S_3 = \text{dist}(d_2, d_3) = 1{,}12$. Então $S = S_1 + S_2 + S_3 = 4{,}1$. O número de pares a ser considerado é $n = 4$. Então $S_{min} = 4{,}07$ e $S_{max} = 15{,}69$. Assim o índice de $C = (4{,}1 - 4{,}07) / (15{,}69 - 4{,}07) = 0{,}02582$.

Capítulo 7

MiRNAs, Dados Utilizados e Pré-Processamento dos Dados

7.1 Considerações Iniciais

Este capítulo apresenta informações básicas sobre o domínio de aplicação *i.e.*, aquele de MicroRNAs, no qual os testes com os algoritmos de agrupamento baseados em grafos, estudados e implementados neste trabalho, foram usados, com o objetivo de agrupar miRNAs pertencentes à uma mesma família.

As seções 7.2 e 7.3 introduzem algumas definições básicas associadas à Biologia Molecular, enfatizando, particularmente, aquelas relacionadas à miRNAs. Na Seção 7.4 são considerados três conjuntos de dados, associados a três diferentes famílias de miRNAs utilizadas no trabalho. Finalmente, a Seção 7.5 aborda o processo de pré-processamento adotado neste trabalho, com o propósito de 'traduzir' cadeias de nucleotídeos, que caracterizam miRNAs, em informações passíveis de serem usadas como atributos, para numericamente descreverem miRNAs, com vista ao seu uso pelos algoritmos de agrupamentos investigados

7.2 Composição do DNA e RNA

Informações genéticas de organismos vivos estão armazenadas em suas células, em macromoléculas chamadas *ácidos nucleicos*, que se apresentam em duas formas distintas: *ácido desoxirribonucleico* (DNA) e *ácido ribonucleico* (RNA). Tanto o DNA quanto o RNA são sintetizados (*i.e.*, produzidos) nas células pelas enzimas DNA polimerase e RNA polimerase, respectivamente.

As moléculas de DNA e RNA são compostas por cadeias de unidades básicas, chamadas *nucleotídeos*, ligados entre si; tais cadeias podem ter centenas, milhares e até mesmo milhões de nucleotídeos. Nucleotídeos podem ser encontrados nas células ou como moléculas individuais ou como parte de ácidos nucleicos. Cada nucleotídeos, por sua vez, é

uma molécula complexa, formada por três componentes distintos: (1) uma molécula chamada *base nitrogenada*; (2) uma molécula de *açúcar* e (3) um molécula de ácido fosfórico.

Nucleotídeos contém um dos dois tipos de base nitrogenadas: purina ou pirimidina. As bases nitrogenadas encontradas em moléculas de DNA são (a) bases purinas com duplo anel: *adenina* (A) e *guanina* (G) (b) bases pirimidina com único anel: *citosina* (C) e *timina* (T). As bases de nucleotídeos de moléculas de RNA são as mesmas que aquelas do DNA, exceto que a timina (T) é substituída pela uracila (U).

Quase sempre moléculas de DNA contém dois polinucleotídeos enrolados um em torno do outro, formando a estrutura de espiral-dupla, como proposta por Watson & Crick em 1953. Os dois polinucleotídeos participantes da espiral-dupla são dispostos de tal maneira que as respectivas fitas se posicionam com os respectivos açúcar-fosfato para fora da espiral e as suas respectivas bases para dentro. Os dois polinucleotídeos são anti-paralelos, significando com isso que eles se orientam em direções opostas; um é orientado na direção 5' $\rightarrow$ 3' e o outro na direção 3' $\rightarrow$ 5; o antiparalelismo promove a estabilidade da espiral dupla.

Um aspecto importantíssimo de todos nucleotídeos é que têm duas finalizações distintas: a finalização 5' e a finalização 3', que referem-se aos carbonos 5' e 3' do açúcar que compõe o nucleotídeo. Tanto para o DNA quanto para o RNA, a finalização 5' termina com um grupo fosfato e a finalização 3' com um grupo hidroxil.

As bases na parte interna da dupla espiral estão organizadas de maneira que uma adenina em um polinucleotídeo está sempre adjacente à uma timina do outro polinucleotídeo e, similarmente, uma guanina é sempre adjacente à uma citosina, formando o que é caracterizado como *pareamento de bases*, em um processo que envolve a formação de ligações de hidrogênio entre as bases participantes. O pareamento entre adenina e timina envolve duas ligações de hidrogênio e entre a citosina e guanina, três ligações de hidrogênio. Como consequência do pareamento de bases, as sequências dos dois polinucleotídeos nas espirais são complementares – a sequência em uma espiral determina a sequência na outra espiral.

Enquanto que o DNA é composto por duas espirais entrelaçadas de polinucleotídeos, uma molécula de RNA, embora também sendo uma cadeia de nucleotídeos bem mais curta

(que as que formam o DNA), é frequentemente encontrada na natureza como uma única fita e não como duas fitas pareadas, como é encontrada na molécula de DNA. Em muitos dos papéis biológicos desempenhados pelo RNA, ele participa como uma molécula constituída de uma única fita de polinucleotídeos. Tal fita pode, entretanto, se dobrar, formando um pareamento de pares de base complementares e se apresentar como duas espirais constituídas por partes da mesma fita (*hairpin*), como mostra a Figura 7.1.

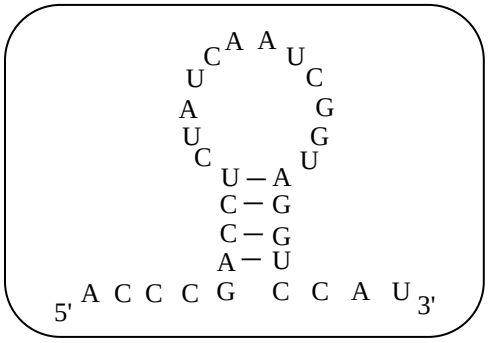

Figura 7.1 *Hairpin* formado em uma fita RNA.

Organismos celulares usam o *RNA mensageiro* (mRNA) para levar informação genética, que direciona a síntese de determinadas proteínas, em ribossomas. O processo usa moléculas de *RNA transportador* (tRNA) para entregar aminoácidos ao ribossomo, onde então o *RNA ribossômico* (rRNA) agrupa aminoácidos para formar proteínas.

O mecanismo que rege a síntese de proteínas é conhecido como *Dogma Central da Biologia Celular* e estabelece que o fluxo de informação genética é "DNA para RNA para proteína" ou, de uma forma mais simplista, "DNA faz RNA, que faz proteínas as quais, por sua vez, facilitam os dois passos prévios, bem com a replicação do DNA". A Figura 7.2 apresenta um diagrama sumarizado do Dogma, que implementa dois processos fundamentais: *transcrição* e *tradução*.

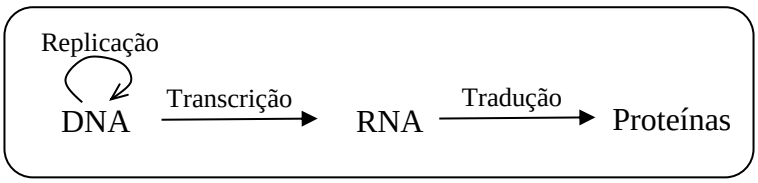

Figura 7.2 Diagrama do Dogma Central da Biologia Celular.

A codificação do RNA mensageiro (mRNA) determina a sequência de aminoácidos, na proteína, que deve ser produzida nos ribossomos. O mRNA é codificado de tal maneira

que cada três nucleotídeos (um *códon*) corresponde a um aminoácido. Aminoácidos são, desta forma, as unidades básicas constituintes de uma proteína.

Em células eucarióticas um mRNA precursor (pré-mRNA), que foi transcrito do DNA, é processado em *mRNA maduro*, por meio da remoção de seus introns (seções não codificadoras que estão presentes no pré-RNA). O mRNA maduro é então exportado do núcleo para o citoplasma da célula, onde ele se liga a ribossomos e, então, é traduzido em sua correspondente forma proteica, com a ajuda do tRNA. Em células procarióticas, que não têm núcleo, o mRNA pode se ligar aos ribossomos enquanto está sendo transcrito a partir do DNA. Após um certo tempo a mensagem se degrada em seus nucleotídeos componentes, com a assistência de ribonucleases.

O RNA transportador (tRNA) é uma cadeia RNA pequena. com cerca de 80 nucleotídeos, que armazena a chave para que os códons do mRNA sejam traduzidos em aminoácidos transfere um específico aminoácido à uma cadeia de polipeptídeos em crescimento, no local da síntese proteica, no ribossomo, durante o processo de tradução.

O RNA ribossômico (rRNA) é o componente catalítico dos ribossomos. Células eucarióticas contem quatro moléculas diferentes de rRNA: rRNA 18S, 5.8S, 28S e 5S. No citoplasma, rRNA e proteínas se combinam para formar uma nucleoproteína chamada ribossomo. O ribossomo se liga ao mRNA e realiza a síntese da proteína. Vários ribossomos podem se ligar a um único mRNA e praticamente todo RNA encontrado em uma típica célula eucariótica é rRNA.

Muitos RNAs, entretanto, não codificam para proteína – cerca de 97% da saída transcrita não é codificação para proteína. Esses RNAs são chamados RNAs não codificadores (ncRNA). Os exemplos típicos de ncRNA são os RNA transportador (tRNA) e RNA ribossômico (rRNA), ambos envolvidos no processo de tradução.

7.3 Uma Breve Descrição de MicroRNAs (MiRNA)

Como comentado em [Williams 2008], microRNAs (miRNAs) são uma família de RNAs (ácido ribonucleico) não codificadores, que regulam a expressão genética pós-transcrição, por meio da inibição da tradução feita pelo mRNA (RNA mensageiro) ou, então, pela degradação do mRNA. As bases nitrogenadas que comparecem em miRNA são, pois, adenina (A), guanina (G), citosina (C) e uracila (U).

7.3.1 MiRNAs e Algumas de suas Funcionalidades

Cada gene miRNA codifica um miRNA maduro, com um tamanho aproximado de 22 nucleotideos [Williams 2008]. Presume-se que miRNAs regulem a expressão de centenas de moléculas de RNA; entretanto, a função da maioria dos miRNAs é desconhecida, embora muitos sejam filogeneticamente conservados.

O primeiro miRNA foi descoberto em 1993 [Ambros *et al.* 1993], quando do estudo do gene *lin-4*, responsável por controlar o tempo de desenvolvimento do nematóide *Caenorhabditis elegans* (*C. elegans*), por meio da supressão do gene *lin-14*. Quando o gene *lin-4* foi isolado, descobriu-se que, ao invés de produzir um mRNA codificando uma proteína, ele produzia um par de RNAs não-codificadores curtos. Um deles era um RNA com aproximadamente 22 nucleotídeos e o outro, com aproximadamente 61 nucleotídeos. Notou-se então que esses RNAs associados ao gene *lin-4* tinham complementaridade *antisense* a múltiplas regiões do 3' UTR do gene *lin-14*. Um outro miRNA relevante no desenvolvimento do *C. elegans*, o *let-7*, foi mais tarde identificado em várias espécies animais [Pasquinelli *et al*, 2000] contendo sequências parcialmente complementares a múltiplas sequências no 3 UTR do mRNA *lin-14*. Essa complementaridade tinha como objetivo inibir a tradução do mRNA *lin-14* para a proteína LIN-14.

Na ocasião conjecturou-se que o pequeno RNA *lin-14* era uma idiossincrasia do nematóide. Apenas em 2000 foi caracterizado um segundo pequeno RNA, identificado como *let-7*, que suprime o *lin-14* para promover um desenvolvimento mais tardio do estágio larval, em *C. elegans.* Descobriu-se rapidamente que o RNA *let-7* participava de muitas espécies, levando à sugestão que o RNA *let-7* e "pequenos RNAs temporais" poderiam regular o tempo de desenvolvimento de diversos animais, inclusive de humanos.

Um ano mais tarde descobriu-se que os RNAs *lin-4* e o let-7 são parte de uma ampla classe de pequenos RNAs presentes em *C. elegans*, *Drosophila* e células humanas. Um grande número de novos RNAs descobertos deste tipo se mostraram similares ao *lin-4* e *let-7*, exceto que seus padrões de expressão eram usualmente inconsistentes com o papel de reguladores do tempo de desenvolvimento, o que sugeriu que a maioria deles pode atuar como reguladores de outros *pathways*. Nesse ponto da pesquisa, pesquisadores começaram a usar o termo microRNA para referirem-se à classe de pequenos RNAs reguladores.

Genes que codificam miRNAs tipicamente codificam transcritos primários longos (*pri-miRNAs)* que têm uma 5' capa e uma cauda *poly-A.* Transcritos são submetidos a inúmeros processamentos para que, finalmente, o miRNA maduro possa ser gerado. Pri-miRNAs podem ter mais de 1.000 nucleotídeos de comprimento e, invariavelmente, contém estrutura(s) *hairpin.* O *hairpin*, que tipicamente compreende de 60 a 120 nucleotídeos, é extraído do pri-RNA no núcleo da célula, pela ação de uma ribonuclease específica chamada Drosha.

O resultado do processo de extração, conhecido como pré-miRNA, é transportado ao citoplasma, via um processo que envolve a proteína Exportin-5. O pré-miRNA é, então, cortado, dessa vez pela enzima Dicer, para gerar um RNA curto, parcialmente com duas fitas (*double-stranded-ds*) no qual uma das fitas é um miRNA maduro. O miRNA maduro é recolhido por um complexo proteico que é similar, se não idêntico, ao RISC (*RNA Induced Silencing Complex*), para mediar o '*silenciamento*' pós-transcrição, de determinados mRNAs [Hurtvagner and Zamore 2002].

O reconhecimento de miRNA frequentemente envolve emparelhamento imperfeito de bases [Lee *et al.* 1993] o que, potencialmente, permite ao miRNA regular um grande número de genes que codificam proteínas. Múltiplos miRNAs podem ser produzidos a partir de um único pri-miRNA transcrito, e cada um deles tendo atuação independente. Muitos genes que codificam para miRNA são expressos em determinados tecidos e em determinados períodos de seu desenvolvimento. Para entender como esses padrões de expressão temporal e espacial são produzidos, é importante estudar e entender eventos de processamento e de transcrição que cooperam para produzir miRNAs específicos, no tempo e no local certos.

Em todos os modelos propostos associados a silenciamento de gene mediado por miRNA, miRNAs, quando maduros, são incorporados ao RISC, que media a degradação de determinados mRNAs e/ou, reprime sua tradução [Olive *et al.* 2010].

7.3.2 Regras de Nomenclatura

As regras descritas a seguir foram extraídas de https: //en.wikipedia.org/wiki/MicroRNA. Levando em conta um sistema de nomenclatura padrão, a atribuição de nomes àqueles miRNAs descobertos, que tenham sido

experimentalmente confirmados, é feita antes de serem tornados públicos. O prefixo 'miR' refere-se à forma madura do miRNA, enquanto que o prefixo 'mir' refere-se ao pre-miRNA e ao pri-miRNA e "MIR" refere-se ao gene que os codifica. O prefixo miR é seguido por um traço e um número; o número frequentemente indica a ordem de nomeação. Por exemplo, miR-124 foi descoberto e nomeado antes do miR-456.

Aqueles miRNAs com sequências praticamente idênticas, exceto por um ou dois nucleotídeos são notados por meio de uma letra minúscula adicional; o miR-124a é relacionado ao miR-124b.

Pre-miRNA, pri-miRNA e genes que codificam para miRNAs maduros 100% idênticos, mas que estão localizados em diferentes locais do genoma, são indicados com um sufixo adicional representado por um traço seguida de um número. Por exemplo, o pre-miRNA has-mir-194-1 e o pre-miRNA has-mir-194-2 levam a miRNA maduros idênticos (hsa-miR-194), mas são produzidos a partir de genes localizados em diferentes regiões do genoma. A espécie de origem é indicada com um prefixo de três letras, por exemplo, hsa-miR-124 é humano (*Homo sapiens*), oar-miR-124 é referente à espécie ovina (*Ovis aries*) e rno-miR-1 diz respeito ao camundongo comum (*Rattus norvegicus).*

Outros prefixos comuns incluem 'v', por viral (miRNA codificado por um genoma viral) e 'd' por *Drosophila* miRNA. Quando dois miRNA maduros se originam a partir de 'braços' opostos de um mesmo pre-miRNA e são encontrados em aproximadamente quantias semelhantes, eles são denotados com sufixos -3p ou -5p. Entretanto, o miRNA maduro encontrado em um 'braço' do *hairpin* é, usualmente, muito mais abundante que aquele encontrado a partir do outro braço e, neste caso, um asterisco seguindo o nome indica o encontrado em menor volume. Por exemplo, miRNA-124 e miRNA-124* compartilham o mesmo pre-miRNA *hairpin*, mas miRNA é encontrado com maior quantidade na célula. A estrutura *hairpin* do pré-miRNA precursor do lin-4 e o miRNA lin-4 maduro estão mostrados na Figura 7.3

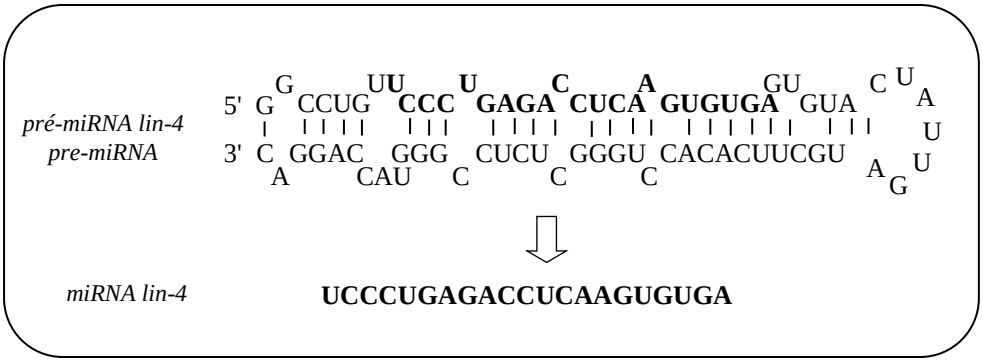

Figura 7.3 Estrutura do pré-miRNA precursor e o miRNA lin-4 maduro.

7.4 Sobre os Dados de MiRNA Utilizados nos Experimentos

O banco de dados conhecido como miRBase (http://www.mirbase.org/) [Griffiths-Jones *et al.* 2006] é, presentemente, considerado o principal banco de dados *online* sobre miRNAs. Os autores em [Yi & Guan 2012] informam que aproximadamente 75% os miRNAs registrados no miRBase foram agrupados em famílias; pressupõe-se que membros de uma mesma família podem ter funções biológicas semelhantes por possuírem uma sequência ou uma estrutura ancestral comum. Comentam também que, para a construção das diferentes famílias de miRNAs, métodos semi-automáticos não estão conseguindo acompanhar o volume de descobertas relacionadas à miRNA. Com o objetivo de agilizar/solucionar o processo, vários métodos baseados em aprendizado de máquina têm sido utilizados.

Na referência [Ding *et al.* 2011], por exemplo, um método automático de alinhamento, chamado *miRFam,* foi proposto para a classificação de miRNAs; é baseado na representação de miRNA em *n-grams* e, também, no uso de SVM (*Support Vector Machines*) [Cortes & Vapnik 1995]. Tal método se mostrou efetivo e eficiente; entretanto, considerando que é um método supervisionado, sequências miRNAs, previamente classificadas, são necessárias para o treinamento do classificador.

Um método não-supervisionado de aprendizado, para automaticamente agrupar grandes quantidades de miRNAs, foi proposto em [Yi & Guan 2012]. Tal método utiliza um refinamento do algoritmo *k-means* [Forgy 1965], chamado SKWIC [Fruigui & Nasraoui 2004]. O SKWIC, que dispõe de identificação simultânea de palavras-chave e agrupamento, foi proposto para uma aplicação em processamento de língua natural no contexto de agrupamento de documentos. Este método também utiliza *n-grams* combinado

com um mecanismo adaptativo de ponderação de atributos e realiza a redução do número de agrupamentos através do MSA (*Multiple Sequence Alignment*) [Sievers *et al.* 2011]. Outro método similar não-supervisionado, denominado *miRCluster*, foi proposto em [Wan *et al.* 2012]. O processo também faz o uso de *n-grams* para construção do vetor de atributos que representa as diversas sequências de miRNA. Associado a este processo, é realizado um procedimento para redução da dimensão do vetor para sua utilização em uma versão simples do algoritmo de agrupamento *k-means*.

Para os experimentos foram escolhidas três famílias de miRNA: *mir-9*, *mir-17* e *let-7*. Tal escolha foi subsidiada pela disponibilidade dos dados, via *download* e, também, por tais dados terem já sido utilizados em outros trabalhos de pesquisa na área (ver, por exemplo [Sievers *et al.* 2011]. e [Wan *et al.* 2012]), o que permitiria comparação dos resultados obtidos, com o objetivo de avaliar os algoritmos envolvidos no processo.

Os dados relativos às três famílias de miRNAs foram extraídos do banco de dados de miRNA versão 21 (http://www.mirbase.org/), referenciado como miRBase [Griffiths-Jones *et al.* 2006], como mostra a Tabela 7.1. Nesta versão existem 28.645 precursores *hairpin* que expressam 35.828 miRNA maduros presentes em 223 espécies.

Tabela 7.1 Características das três famílias de miRNA utilizadas nos experimentos.

Nome da família	Nro. sequências	Endereço Web
mir-9	227	http://www.mirbase.org/cgi-bin/mirna_summary.pl?fam=MIPF0000014
mir-17	336	http://www.mirbase.org/cgi-bin/mirna_summary.pl?fam=MIPF0000001
let-7	457	http://www.mirbase.org/cgi-bin/mirna_summary.pl?fam=MIPF0000002

7.5 Pré-Processamento de MiRNAs: Transformando MiRNAs em Vetores de Atributos Ponderados

As sequências primárias dos miRNAs alvos, disponibilizadas na miRBase, foram inicialmente pré-processadas com o objetivo de serem convertidas em vetores de atributos ponderados. Este processo envolve duas fases, extração e seleção de atributos. Basicamente, a fase de extração compreende a conversão das sequências de miRNA em um vetor numérico de atributos; a fase de seleção é complementar, e portanto, opcional, ao processo de extração, e envolve a redução da dimensão do vetor de atributos. Os experimentos descritos em [Yi & Guan 2012] não utiliza a seleção de atributos, já os

experimentos realizados em [Wan *et al.* 2012] faz o uso dessa fase. Portanto, os experimentos descritos na Seção 9 verificam o impacto da utilização ou não dessa fase nos resultados dos agrupamentos.

7.5.1 Extração de atributos

A extração de atributos visa associar cada sequência primária de miRNA disponibilizada, a um conjunto de *n-grams* (para n =1, 2, 3, 4, ..., N). Cada *n-gram* dá origem a um conjunto de atributos, quando da criação da representação vetorial de cada sequência primária.

Definido de uma maneira informal, um *n-gram* é uma subsequência contígua de n itens de uma dada sequência de texto (ou discurso). Essa sequência de texto é composta por um conjunto finito de símbolos pertencentes a um dado universo do alfabeto. No problema em questão, o texto é representado pela sequência de nucleotídeos que representa cada miRNA e, como comentado anteriormente, é uma sequência envolvendo apenas os caracteres A (Adenina), C (Citosina), G(Guanina) e U (Uracila). Portanto o conjunto do universo do alfabeto utilizado no caso das sequências de miRNA, é definido por {A, C, G, U}.

Dado um número inteiro N e uma sequência de miRNAs, o algoritmo consiste em decompor s em subsequências únicas (*n-grams*) com tamanhos definidos por n (n = 1, 2, 3, ..., N). Uma vez que essas sequências de miRNA contém apenas as quatro bases, existem apenas 4^1 = 4 únicos 1-grams, 4^2 = 16 únicos 2-grams, 4^3 = 64 únicos 3-grams, 4^4= 256 únicos 4-grams e assim por diante. Após a definição das subsequências únicas s_n de *n-grams* determinados a partir de s, o algoritmo realiza uma contagem para determinar a quantidade de vezes em que s_n aparece dentro de s. Por exemplo, um dos elementos da família MiRNA conhecida como *mir-9*, é representado pela sequência UAAAGCUAGAUUACCAAAGCAU, denominada *dgr-miR-79*. Considerando N = 4, a criação de *n-grams* associados à essa sequência, para n = 1, ..., N totaliza 340 termos (subsequências) possíveis. O número de termos ou *n-grams* únicos corresponde a dimensão do vetor de atributos. Portanto para N = 4, o vetor possuirá 340 dimensões que correspondem a 340 atributos. A Tabela 7.2 mostra apenas os 1-grams e 2-grams associados à sequência *dgr-miR-79*.

Tabela 7.2 *n-grams* associados à sequência UAAAGCUAGAUUACCAAAGCAU.

N	*n-grams*	Quantidade de *n-grams* na sequência
1	A	10
	C	4
	G	3
	U	5
2	AA	4
	AC	1
	AG	3
	AU	2
	CA	2
	CC	1
	CG	0
	CU	1
	GA	1
	GC	2
	GG	0
	GU	0
	UA	3
	UC	0
	UG	0
	UU	1

Após a contagem dos *n-grams* associados a cada sequência miRNA disponibilizada, um processo de ponderação, chamado *fator de concentração* [Ding *et al.* 2011], é usado para combinar esses diferentes valores em um vetor ponderado de atributos. Para um dado valor inteiro de N e denotando o número de *n-grams* únicos do tipo i como n_i, a *concentração de tipo i* é definida como a razão de n_i pelo número total de *n-grams* gerados, representada pela Eq. (7.1).

$$C_i = \frac{n_i}{\sum_{j=1}^{N} n_j}, \; i = 1, \ldots, N \qquad \text{Eq. (7.1)}$$

Considere N = 4 e o processo de geração de *n-grams*. Para o cálculo do fator de concentração para o tipo i = 3, ou seja, 3-grams (tri-grams), $n_3 = 4^3$. Uma vez que $\Sigma_{j=1}^{N} n_j = 4^1 + 4^2 + 4^3 + 4^4 = 340$, $C_3 = 4^3/340 = 0{,}188$.

Os elementos de um vetor de atributos de um determinado miRNA é construído de acordo com a Equação (7.2).

$$f_j = \frac{t_j}{T_i} \times C_i \text{ em que } j \in Z \text{ e } 1 \leq j \leq 4^N \qquad \text{Eq. (7.2)}$$

em que t_j é a frequência de ocorrência de um certo único *n-gram* de tipo i e T_i é a soma das frequências de ocorrência de cada um dos *n-grams* do tipo i. Considerando N = 4, um vetor de atributos conterá 340 posições, em que cada uma corresponde a um único *n-gram* de um certo tipo i, para i=1, 2, 3, 4. A soma de todas os valores de um vetor de atributos deve ser 1.

Por exemplo, para N = 4, as primeiras 20 posições do vetor de atributos de tamanho 340 (= $4^1 + 4^2 + 4^3 + 4^4$) correspondente à sequência *dgr-miR-79*, são mostradas na Tabela 7.3.

Tabela 7.3 Primeiras 20 posições (f_j, para j = 1, ..., 20) do vetor de atributos associado à sequência UAAAGCUAGAUUACCAAAGCAU.

n-gram	nro. *n-grams*	f_i	valor de f_i
A	10	f_1	10/(10+4+3+5)×(4/340) = 0,00534759
C	4	f_2	4/(10+4+3+5)×(4/340) =0,00213904
G	3	f_3	3/(10+4+3+5)×(4/340) = 0,00213904
U	5	f_4	5/(10+4+3+5)×(4/340) =0,00267380
AA	4	f_5	(4/21)×(16/340) = 0,00896358
AC	1	f_6	(1/21)×(16/340) = 0,00224090
AG	3	f_7	(3/21)×(16/340) = 0,00672269
AU	2	f_8	(2/21)×(16/340) = 0,00672269
CA	2	f_9	(2/21)×(16/340) = 0,00448179
CC	1	f_{10}	(1/21)×(16/340) = 0,00224090
CG	0	f_{11}	(0/21)×(16/340) = 0
CU	1	f_{12}	(1/21)×(16/340) = 0,00224090
GA	1	f_{13}	(1/21)×(16/340) = 0,00224090
GC	2	f_{14}	(2/21)×(16/340) = 0,00448179
GG	0	f_{15}	(0/21)×(16/340) = 0
GU	0	f_{16}	(0/21)×(16/340) = 0
UA	3	f_{17}	(3/21)×(16/340) = 0,00672269
UC	0	f_{18}	(0/21)×(16/340) = 0
UG	0	f_{19}	(0/21)×(16/340) = 0
UU	1	f_{20}	(1/21)×(16/340) = 0,00224090

Em [Yi & Guan 2012] foi utilizado dados referentes a miRBase versão 18 e em [Wan *et al.* 2012] foram utilizadas as versões 16 e 17 do miRBase. Em ambos, os experimentos tiveram um melhor resultado para N = 4.

7.5.2 Seleção de atributos

A seleção de atributos é um processo complementar a extração de atributos e tem como objetivo reduzir a dimensão do vetor de atributos. Nexte contexto, existem três métodos conhecidos que podem ser utilizados, *Latent Semantic Analysis* (LSA) [Deerwester *et al.* 1990] , *Locally Linear Embedding* (LLE) [Roweis & Saul 2000] and *Isometric Feature Mapping* (*Isomap*) [Tanenbaum *et al.* 2000]. O algoritmo de *Isomap* constrói o grafo de vizinhança utilizando o parâmetro k para especificar o número de vizinhos próximos a um determinado dado d. Ou seja, são utilizados no cálculo, os k dados próximos a d. Outro parâmetro de entrada é utilizado para especificar para qual dimensão o vetor deverá ser reduzido. O algoritmo de *Isomap* realiza a redução da dimensão através do cálculo dos menores caminhos encontrados no grafo, utilizando como medida de similaridade a distância geodésica entre os pares de dados.

Os experimentos descritos em [Wan *et al.* 2012] obtiveram os melhores resultados utilizando *Isomap* em um vetor de atributos de 340 dimensões (N = 4). Os valores ideais para os parâmetros utilizados pelo *Isomap* foram k = 10 e redução de dimensão para 150 atributos.

Capítulo 8

Metologia, Resultados e Análise dos Dados

8.1 Considerações Iniciais

Este capítulo apresenta a metodologia, resultados e análises de miRNA de um conjunto de experimentos realizados, com vistas a separar os dados de miRNA pertencentes a diferentes famílias. Na maioria dos experimentos descritos neste capítulo foram utilizados dados de três famílias, *mir-9*, *let-7* e *mir-17*, disponíveis na base de dados do miRBase [Griffiths-Jones *et al.* 2006], que atualmente está na versão 21. O objetivo dos experimentos descritos nos capítulos 8 e 9 foram a investigação e subsequente análise de desempenho dos algoritmos de agrupamento ASME, ASMEH, ASM-PI, ASM-TXE, SFASM, *Ncut* e Região de Influência, quando da criação de agrupamentos envolvendo dados das três famílias de miRNA, de maneira a separá-las em grupos distintos, participantes de um mesmo agrupamento. Devido à grande quantidade de parâmetros envolvidos para análise, diversos experimentos são apresentados e discutidos para o algoritmo simples ASME. Os experimentos que utilizam os demais algoritmos apresentados na Seção 3 (ASMEH, ASM-PI e ASMTXE), na Seção 4 (Região de Influência) e na Seção 5 (*Ncut*), são detalhados e discutidos no Capítulo 9.

8.2 Metodologia Empregada nos Experimentos

A Figura 8.1 mostra um fluxograma dos passos executados para realização dos experimentos. Nos passos 1.1, 1.3 e 2 são utilizados parâmetros cujos valores devem ser informados pelo usuário. Muitas vezes a determinação do valor otimal para esses parâmetros é um processo custoso e, por isso, tal determinação deve ser feita por meio de variações dos valores dentro de um intervalo pré-determinado. Esse valor otimal é

estabelecido com a ajuda de índices de avaliação ou análise visual do resultado do agrupamento.

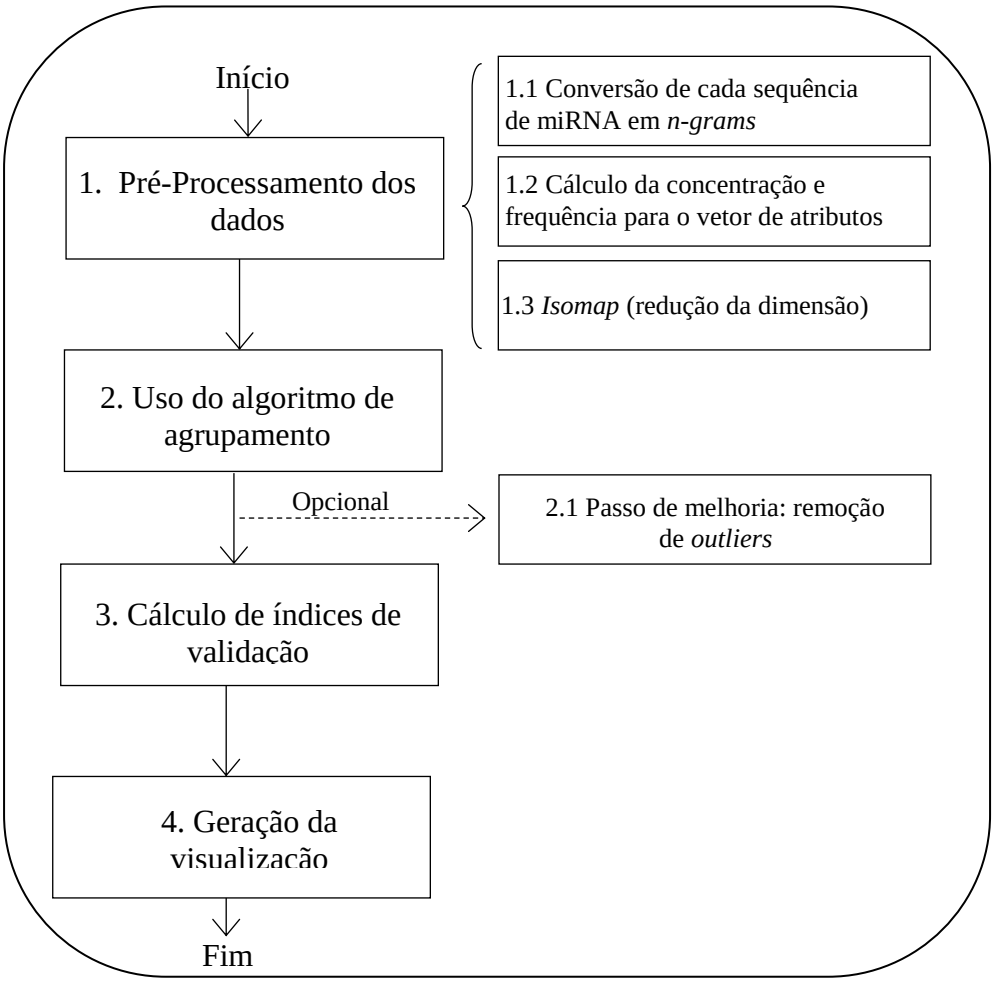

Figura 8.1 Fluxograma para execução dos experimentos.

No passo 1.1, relacionado ao uso de *n-grams* para a representação de cada sequência de miRNA, o parâmetro N foi utilizado com os valores 4 ou 5, ou seja, o cálculo de *n-grams* foi realizado por meio da combinação de *grams* com tamanho de 1 até 4 ou, então, de 1 até 5. Esse intervalo de valores foi escolhido contemplando N = 4 pois, com este valor, os melhores resultados foram obtidos nos experimentos descritos em [Yi & Guan 2012] e em [Wan *et al.* 2012]. No passo 1.2 são utilizadas as equações (7.1) e (7.2) descritas na Seção 7.5 para $N \in \{4, 5\}$.

O passo 1.3 é relacionado ao uso do algoritmo *Isomap* para a redução da dimensionalidade do conjunto de dados. Para os experimentos foi utilizado k = 10 (a escolha foi motivada pelo artigo [Wan *et al.* 2012]) e a dimensão de cada dado foi reduzida

de 340 (N = 4) ou 1364 (N = 5) para 150, em ambos os casos. Os valores dos parâmetros *i.e.*, N = 4, k = 10, e a redução de dimensão para 150 atributos, foram considerados ótimos para o cálculo da distância utilizando a distância *city-block* (Equação (2.2)), conforme os experimentos descritos em [Wan *et al.* 2012]. Várias implementações do algoritmo *Isomap* estão disponíveis em ferramentas como *RStudio* (https://www.rstudio.com/) e *Matlab* (http://www.mathworks.com/products/matlab/). Após vários testes foi selecionada a implementação utilizada no *Matlab*, uma vez que tal implementação disponibiliza, como parâmetro de entrada para o seu uso, a especificação da distância que será utilizada pelo algoritmo.

Todos os processos descritos nos passos 1.1, 1.2 e 1.3 foram implementados utilizando ambas as ferramentas *RStudio* e *Matlab*. Ao final do Passo 1 da Figura 8.1 é gerada uma planilha no formato estabelecido pela ferramenta Excel (https://products.office.com/pt-br/excel) com os vetores de atributos que representam cada sequência de miRNA que serão utilizadas pelo algoritmo de agrupamento (passo 2 da Figura 8.1).

Os algoritmos do passo 2 da Figura 8.1 utilizam os valores dos vetores de atributos (resultado do Passo 1 da Figura 8.1) para realizar o cálculo das distâncias, *city-block* ou euclidiana, entre todas as sequências de miRNA sendo consideradas. Essas distâncias são então utilizadas na construção do grafo, para representar os pesos das arestas entre as diversas sequências de miRNA. O passo 2, que consiste no uso de cada um dos algoritmos de agrupamento, é abordado individualmente para cada um dos sete algoritmos envolvidos, em uma seção própria, nos capítulos 8 e 9, juntamente com a descrição dos resultados e análises associadas. O parâmetro extra, identificado como *tipo da distância*, foi adicionado à execução de cada algoritmo, de maneira a permitir a investigação do papel da distância utilizada, nos resultados obtidos. Como comentado anteriormente, para os experimentos foram adotadas a distância euclidiana e a *city-block*.

O passo 2.1 implementa um processo de refinamento do agrupamento obtido no passo anterior, e tem por principal objetivo o tratamento daqueles grupos que possuem outliers. No contexto das sequências de miRNAs, *outliers* são os dados que não estão em conformidade com o resto dos padrões determinados pelas distâncias euclidiana ou *city-block* entre os dados pertencentes a mesma família. A presença de *outliers* é um indicativo,

quando do uso de um procedimento de criação de agrupamentos, da possível formação de grupos isolados com estes elementos, ou então, os *outliers* poderão estar mais próximos de dados não pertencentes à sua família. Em ambos os casos, o resultado do agrupamento será afetado negativamente, pois os *outliers* farão parte ou de grupos isolados ou de grupos aos quais não pertencem.

A remoção de grupos com *outliers* é realizada por meio da associação de cada um desses elementos a um outro grupo *i.e.*, àquele que lhe seja mais próximo. Formalmente, considere:

- agrupamento formado por k grupos, $AG = \{G_1, G_2, ..., G_k\}$.
- $|G_1 \cup G_2 \cup ... \cup G_k| = n$
- elemento identificado como *outlier* $d_i \in G_m$, $1 \leq m \leq k$.
- dist (d_i, d_j) representa a distância euclidiana ou *city-block* entre d_i e d_j, $d_j \notin G_m$.

O elemento d_i é associado ao grupo G_z, $1 \leq z \leq k$ e $z \neq m$, se $dist(d_i, d_j)$ for a menor distância entre d_i e qualquer outro dos $n - 1$ dados pertencentes a grupos de AG.

Na Figura 8.2 é mostrado o gráfico de distribuições das distâncias euclidianas dos dados dentro de cada família. O gráfico foi gerado a partir da construção da ASM com os dados de cada família. A ferramenta de análise estatística denominada *Tableau* (http://www.tableau.com/) foi utilizada para a construção do gráfico da Figura 8.2. A área destacada pelo retângulo tracejado corresponde a área com os maiores valores de distâncias euclidiana e é onde se encontram os principais *outliers.* É importante notar que a maioria dos *outliers* encontram-se presentes na família *let-7* enquanto que a família *mir-17* possui poucos dados considerados *outliers*.

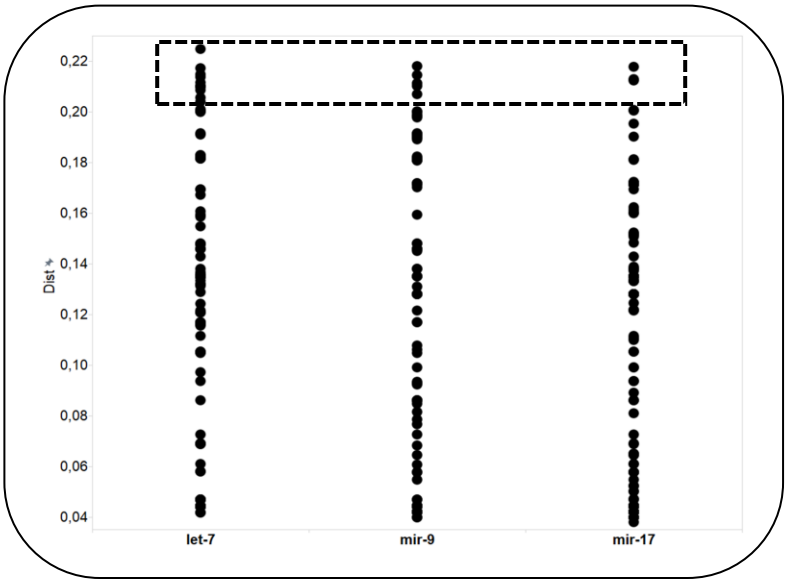

Figura 8.2 Distribuição da distância euclidiana dentro de cada família (*let-7*, *mir-9* e *mir-17*).

No passo 3 da Figura 8.1 foram utilizados os índices *Davies-Bouldin*, *C* e *Dunn* descritos no Capítulo 6. Os valores dos índices calculados nos resultados de cada experimento são detalhados e comparados localmente, quando tais resultados são mostrados.

No passo 4 da Figura 8.1 foi utilizada a ferramenta de banco de dados de grafos denominada *Neo4J* (http://neo4j.com/) para visualização dos resultados mostrados nas próximas seções. Por meio dessa ferramenta é possível visualizar os resultados dos agrupamentos, com o objetivo de detectar e explicar a estrutura formada pelas componentes conexas (i.e., grupos) obtidas. As cores utilizadas para identificar cada família são mostradas na Tabela 8.1.

Nos experimentos foram removidos dos conjuntos de dados originais as ocorrências repetidas de sequências de miRNA. A presença de sequências repetidas afeta o desempenho do algoritmo *Isomap*, uma vez que a distância entre elementos iguais é igual a zero, o que interfere no processo de escolha dos k dados mais próximos de um determinado dado, utilizado pelo algoritmo. Além disso, os algoritmos utilizados no passo 2 (Figura 8.1) executam com uma melhor performance já que a quantidade de dados no conjunto será reduzida. A Tabela 8.1 mostra o número de elementos por família e a quantidade reduzida de dados em cada família devido à remoção de sequências de miRNA repetidas.

Tabela 8.1 Número de sequências por família

Família	Cor	Conjunto original	Número de sequências do conjunto com remoção de *outliers* e de repetições
mir-9	vermelho	227	89
let-7	azul	336	121
mir-17	verde	457	129

8.3 Experimentos Usando o Algoritmo ASME (Árvore *Spanning* Minimal Euclidiana), Resultados e Análises

Nesta seção são descritos 13 experimentos realizados para avaliar o uso do algoritmo ASME para a geração de agrupamentos no domínio de dados das três principais famílias de miRNA *i.e.*, *mir-9*, *let-7*, *mir-17*, com objetivos diversos, especificados em cada subseção. Outras famílias de miRNA também foram acrescentadas aos experimentos *ASME-5families*, *ASME-versao17* e *miRCluster*, para análise de impacto de desempenho.

8.3.1 Influência de Grupos de *Outliers* no Uso da Distância euclidiana

Experimento ASME-euclidiana-1

O algoritmo ASME utiliza o parâmetro de entrada que determina o número de grupos formados no agrupamento. Esse parâmetro deveria ser determinado conforme o número de famílias *i.e.*, se existem 3 famílias, então 3 grupos devem ser formados e portanto 2 pontes serão removidas da ASM. Esse experimento tem por objetivo mostrar que o número de grupos deve ser maior que o número de famílias. Isso ocorre devido à presença de *outliers* no conjunto de dados, ou seja, os primeiros grupos formados pelo ASME são grupos isolados de *outliers*. Os parâmetros usados e seus respectivos valores são mostrados na Tabela 8.2.

Tabela 8.2 Parâmetros utilizados em *ASME-euclidiana-1*.

Algoritmo	Parâmetro	Valor
n-gram	N	5
Isomap	k	10
	Dimensão	150
ASME	Distância	euclidiana
	número de grupos	5

O resultado do algoritmo ASME tendo como dados de entrada o conjunto das sequências das três famílias (*mir-9* com 89 sequências, *let-7* com 121 e *mir-17* com 129), é exibido na Tabela 8.3.

Tabela 8.3 Resultado de *ASME-euclidiana-1* para os parâmetros da Tabela 8.2.

Grupo	*mir-9*	*let-7*	*mir-17*
G_1	51	119	128
G_2	38	0	0
G_3	0	1 (ssa-let-7i-3p)	0
G_4	0	1 (bbe-let-7a-2-3p)	0
G_5	0	0	1 (mdo-miR-20b-3p)

Os grupos G_3–G_5 evidenciam que a presença de *outliers* influenciou no resultado do algoritmo, uma vez que foram criados tais grupos, cada um deles contendo apenas 1 elemento. É interessante observar que os dados desses 3 grupos estão contidos no retângulo tracejado mostrado na Figura 8.2. De acordo com os resultados mostrados na Tabela 8.3, de um total de 339 sequências que representam as três famílias, 298 foram agrupadas no grupo G_1, 38 foram agrupadas no grupo G_2 e cada uma das 3 sequências restantes definiu um grupo unitário, totalizando assim os 5 grupos. Como o experimento não resultou na formação de grupos significativos, não será considerado o cálculo de índices.

Experimento ASME-euclidiana-2

Para esse experimento os parâmetros usados e seus respectivos valores são mostrados na Tabela 8.4. A única diferença entre os valores dos parâmetros usados no experimento *ASME-euclidiana-1* e os usados em *ASME-euclidiana-2*, está relacionada ao número de grupos formados no agrupado, que foi usado com valor 5 no *ASME-euclidiana-1* e que em *ASME-euclidiana-2* passou a ser 30. A mudança foi feita com o objetivo de fomentar um melhor resultado, tendo em conta a presença de *ouliers*.

Tabela 8.4 Parâmetros utilizados em *ASME-euclidiana-2*.

Algoritmo	Parâmetro	Valor
n-gram	N	5
Isomap	k	10
	Dimensão	150
ASME	Distância	*euclidiana*
	número de grupos	30

Após vários testes com a variação do valor do parâmetro número de grupos, no intervalo [5, 40], o valor 30 se mostrou uma escolha conveniente, em virtude dos resultados obtidos. Tal valor permitiu a formação de grupos significativos, o que não aconteceu no experimento *ASME-euclidiana-1*. O resultado do agrupamento é exibido na Tabela 8.5 e os resultados de três índices utilizados para avaliação do resultado do agrupamento induzido pelo ASME são mostrados na Tabela 8.6.

Os resultados da Tabela 8.5 podem ser interpretados como a família *mir-9* sendo constituída por dois grupos majoritários, G_1 e G_2 com 57,30% e 31,46% das 89 sequências da família, e evidenciam *outliers* nos grupos G_4–G_7; a família *let-7* é constituída pelos grupos majoritários, G_8 e G_9, com a 52,89% e 33,88% das 121 sequências da família e possui grupos formados por *outliers* em G_{10}–G_{24}; a família *mir-17* é constituída pelos grupos G_{19} e G_{25} com 12,4% e 37,21% das 129 sequências e evidenciam grupos *outliers* principalmente nos grupos G_{29} e G_{30}. Os *outliers* pertencem a região destacada na Figura 8.2. Também é interessante observar que a família *let-7* possui 1 *outlier* classificado incorretamente, no G_{19} (grupo formado pela familía *mir-17*) e a família *mir-17* possui 37 sequências de miRNAs classificadas incorretamente no G_8 (grupo formado pela família *let-7*) e 15 sequências de miRNAs classificadas incorretamente no G_9 (grupo formado pela família *let-7*).

Tabela 8.5 Resultado de *ASME-euclidiana-2* para os parâmetros da Tabela 8.4.

Grupo	*mir-9*	*let-7*	*mir-17*
G_1	51	0	0
G_2	28	0	0
G_3	6	0	0
G_4	1 (oha-miR-9-3-5p)	0	0
G_5	1 (tca-miR-9b-5p)	0	0
G_6	1 (dvi-miR-9b-3p)	0	0
G_7	1 (mdo-miR-9b-5p)	0	0
G_8	0	64	37
G_9	0	41	15
G_{10}	0	1 (mml-let-7e-3p)	0
G_{11}	0	2 (bfl-let-7a-3p e bbe-let-7a-3p)	0
G_{12}	0	1 (dme-let-7-3p)	0
G_{13}	0	1 (ssa-let-7i-3p)	0
G_{14}	0	1 (cqu-let-7-3p)	0
G_{15}	0	1 (bbe-let-7a-2-3p)	0
G_{16}	0	1 (sma-let-7-3p)	0
G_{17}	0	1 (rno-miR-3596b)	0
G_{18}	0	1 (cin-let-7d-3p)	0
G_{19}	0	1 (cin-let-7b-3p)	16
G_{20}	0	1 (cin-let-7a-1-3p)	0
G_{21}	0	1 (cel-let-7-3p)	0
G_{22}	0	1 (asu-let-7-3p)	0
G_{23}	0	1 (bta-miR-3596)	0
G_{24}	0	1 (rno-miR-3596d)	0
G_{25}	0	0	48
G_{26}	0	0	4
G_{27}	0	0	3
G_{28}	0	0	4
G_{29}	0	0	1 (pma-miR-17a-3p)
G_{30}	0	0	1 (mdo-miR-20b-3p)

Apesar do G_8 ser composto por um número significativo de sequências das famílias *let-7* e *mir-17*, esse grupo possui concentrações dos elementos de cada família que, visualmente, estão separadas. Existe apenas uma ponte significativa, destacada pelo retângulo, que interliga esses dois grupos, como mostra a Figura 8.3.

Tabela 8.6 Resultado dos índices para *ASME-euclidiana-2*.

Índice	Valor
Davies-Bouldin	0,63483665
Dunn	0,31226115
C	0,04974635

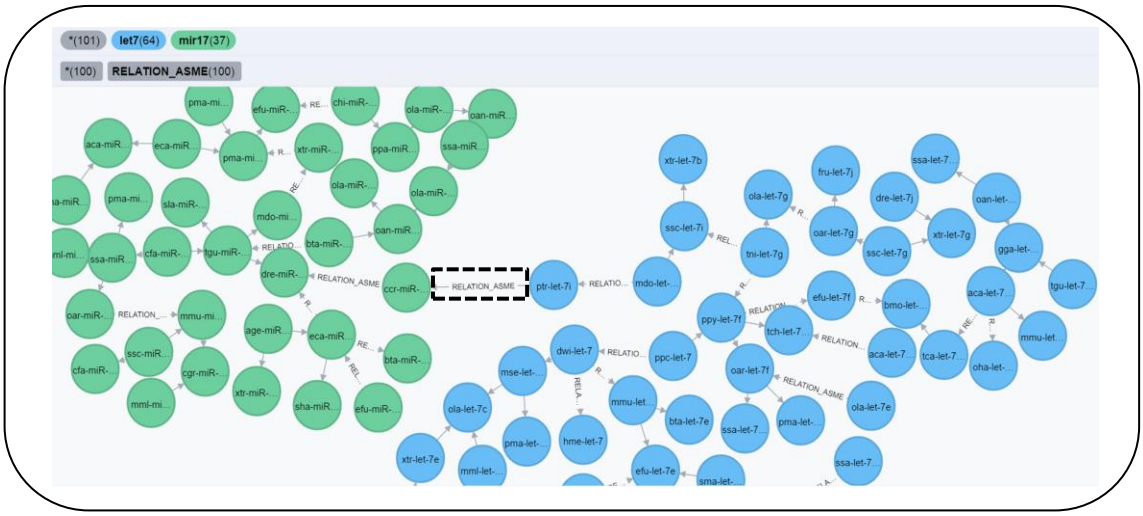

Figura 8.3 Ponte do G_8 entre ccr-miR-93 (*mir-17*) e ptr-let-7i (*let-7*).

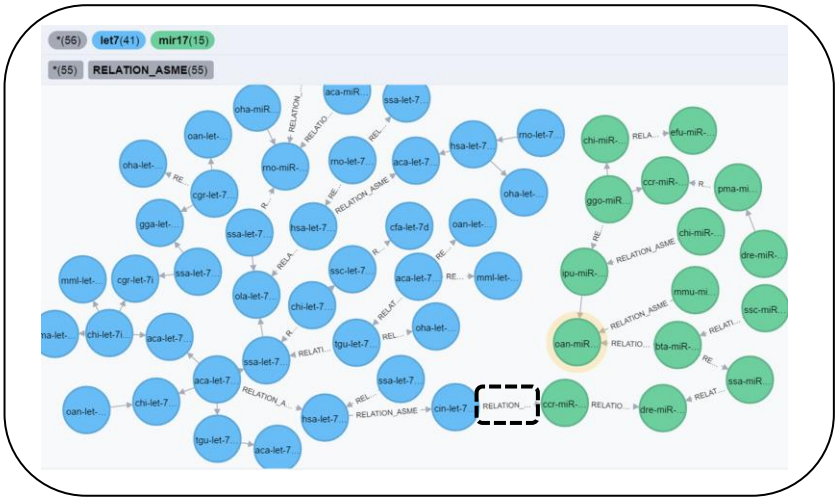

Figura 8.4 Ponte do G_9 entre ccr-miR-18c (*mir-17*) e cin-let-7a-2-3p (*let-7*).

Mesmo com o aumento do valor associado ao número de grupos, agora no intervalo [5, 40], a ponte destacada na Figura 8.3 não será descartada devido ao seu baixo valor de peso, no caso, a distância euclidiana entre os dados. Portanto outras pontes serão removidas antes desta, dando origem assim, a pequenos grupos isolados. A Figura 8.5(a) mostra a distribuição das distâncias euclidianas entre todos os pares de sequências presentes nas famílias *let-7* e *mir-17*. A Figura 8.5(b) destaca a ponte entre ccr-miR-93 (*mir-17*) e ptr-let-7i (*let-7*) dentre todas as distâncias euclidianas mostradas na Figura 8.5(a). Nota-se que o peso dessa ponte é de 843 (Figura 8.5(b)) e o máximo valor para a distância euclidiana entre todos os outros pares de sequências é por volta de 4.000 (conforme mostra o eixo y da

Figura 8.5(a)). Devido a isso outras pontes com valores maiores (*i.e.*, acima de 843) serão removidas.

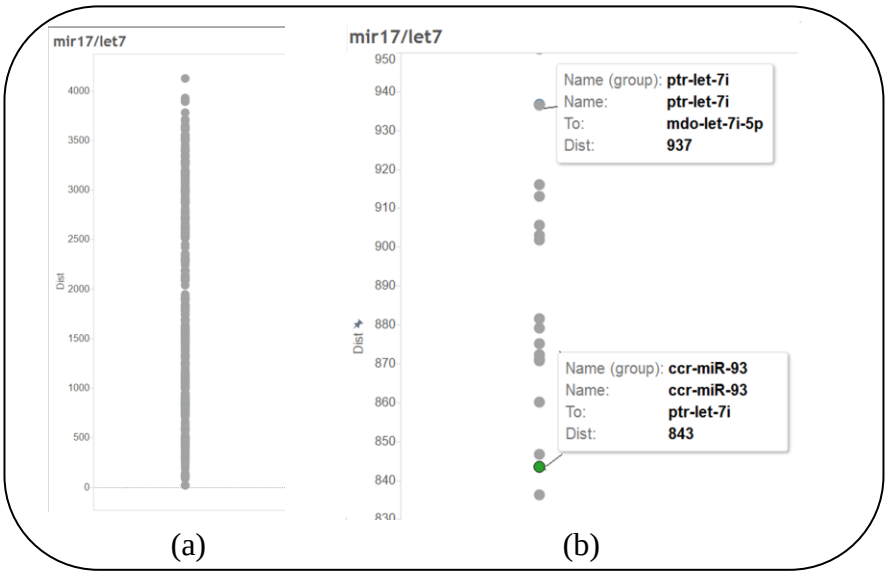

Figura 8.5 (a) Distribuição das distâncias euclidianas das famílias (*let-7* e *mir-17*) e **(b).** Destaque para a ponte entre ccr-miR-93 (*mir-17*) e ptr-let-7i (*let-7*).

8.3.2 Influência da Remoção de Grupos de *Outliers* no Uso da Distância Euclidiana

Experimento ASME-euclidiana-3

Os valores de parâmetros utilizados neste experimento estão descritos na Tabela 8.4 da Subseção 8.3.1. O objetivo do experimento é avaliar, empiricamente, se a implementação do passo 2.1 (Figura 8.1) efetivamente contribui para um refinamento dos resultados obtidos. Os resultados do experimento são exibidos na Tabela 8.7 e os cálculos dos três índices são mostrados na Tabela 8.8.

Tabela 8.7 Resultado de *ASME-euclidiana-3* para os parâmetros da Tabela 8.4.

Grupo	*mir-9*	*let-7*	*mir-17*
G_1	51	1 (cqu-let-7-3p)	0
G_2	30	0	0
G_3	8	0	0
G_4	0	66	37
G_5	0	51	15
G_6	0	1 (cin-let-7b-3p)	16
G_7	0	2 (cel-let-7-3p e asu-let-7-3p)	61

Tabela 8.8 Resultado dos índices para *ASME-euclidiana-3*.

Índice	Valor
Davies-Bouldin	1,09532919
Dunn	0,30991494
C	0,05847372

Os resultados do *ASME-euclidiana-3* mostram que a junção dos grupos de *outliers* com o seu grupo considerado mais próximo, melhora ligeiramente o resultado final do agrupamento. Os grupos unitários G_4–G_7 da Tabela 8.5 foram corretamente incorporados aos grupos G_2 e G_3 da Tabela 8.7, e os grupos G_{10}–G_{13}, G_{15}–G_{18}, G_{20}, G_{23} e G_{24} da Tabela 8.5 (todos unitários, exceto pelo G_{11}) foram corretamente fundidos aos grupos G_8 e G_9 da mesma tabela, resultando nos grupos G_4 e G_5 da Tabela 8.7. As mesmas pontes destacadas nas figuras 8.3 e 8.4 permaneceram nos grupos G_4 e G_5.

Na maioria dos casos os *outliers* foram corretamente agrupados, com exceção do cqu-let-7-3p da família *let-7*, que permaneceu como parte de G_1 (grupo que pertence a família *mir-9*) da Tabela 8.7, e os três grupos unitários G_{19}, G_{21} e G_{22} (*let-7*) da Tabela 8.5, que passaram a integrar os grupos G_6 e G_7 (*mir-17*) na Tabela 8.7.

Os valores dos índices mostrados na Tabela 8.8 evidenciam que não houve uma melhora no resultado do agrupamento, quando comparado aos valores dos mesmos índices, mostrados na Tabela 8.5. Apesar disso, pode ser notada uma melhoria entre os grupos G_2, G_3, G_8, G_9 e G_{25} (Tabela 8.5) e os grupos G_2, G_3, G_4, G_5 e G_7 (Tabela 8.7). O índice *Davies-Bouldin* mostra uma piora no resultado do agrupamento, pois houve um aumento de seu valor (entre os valores das Tabelas 8.6 e 8.8), devido à diminuição do número de grupos. O valor do índice de *Dunn* manteve-se praticamente inalterado. Os grupos G_1–G_3 pertencem a família *mir-9* e juntos correspondem a 100% do total de 89 sequências de miRNAs; os grupos G_4 e G_5 pertencem a família *let-7* e juntos correspondem a 96,69% do total de 121 sequências de miRNAs; os grupos G_6 e G_7 pertencem a família *mir-17* e juntos correspondem a 59,69% do total de 129 sequências de miRNAs.

8.3.3 Influência no Uso do *Isomap*

Experimento ASME-euclidiana-4

O objetivo desse experimento é verificar o impacto da utilização do passo 1.3 (Figura 8.1), que corresponde a redução de dimensão do vetor de atributos. Através da comparação de resultados com o experimento *ASME-euclidiana-3* é possível verificar se houve uma diminuiçao de desempenho do algoritmo devido a uma provável perda de informação causada pela redução de dimensão do vetor de atributos. Os valores dos parâmetros utilizados são mostrados na Tabela 8.9. O parâmetro número de grupos foi determinado no intervalo [20, 45], onde o valor mais conveniente foi escolhido, conforme o resultado do agrupamento.

O resultado do agrupamento é exibido na Tabela 8.10 e os valores calculados para os índices são mostrados na Tabela 8.11. O passo 2.1 (Figura 8.1) também foi utilizado para melhorar o resultado do algoritmo ASME.

A Tabela 8.10 mostra um dos melhores resultados do algoritmo ASME considerando todos os experimentos que utilizam as três famílias e a distância euclidiana. Houve um aumento nos valores do índice *Dunn* apresentados nas tabelas 8.8 e 8.11, indicando um melhoria no resultado do agrupamento. Os valores dos índices *Davies-Bouldin* e *C* mostrados nas tabelas 8.8 e 8.11 mostram que não houve uma melhora no resultado do agrupamento. A mesma ponte mostrada na Figura 8.3, responsável pela junção das famílias *let-7* e *mir-17* do G_4 (Tabela 8.7), permanece no grupo G_3 (Tabela 8.11). Já a ponte mostrada na Figura 8.4, responsável pela junção das famílias *let-7* e *mir-17* do G_5 (Tabela 8.7) foi removida separando as famílias *let-7* e *mir-17* nos grupos G_4 e G_6 (Tabela 8.10), respectivamente.

Tabela 8.9 Parâmetros utilizados em *ASME-euclidiana-4*.

Algoritmo	Parâmetro	Valor
n-gram	N	5
ASME	Distância	euclidiana
	número de grupos	41

Tabela 8.10 Resultado de *ASME-euclidiana-4* para os parâmetros da Tabela 8.9.

Grupo	*mir-9*	*let-7*	*mir-17*
G_1	52	0	0
G_2	37	0	0
G_3	0	55	37
G_4	0	66	0
G_5	0	0	16
G_6	0	0	15
G_7	0	0	61

Tabela 8.11 Resultado dos índices para *ASME-euclidiana-4*.

Índice	Valor
Davies-Bouldin	2,12420422
Dunn	0,62421916
C	0,18207388

Os grupos G_1 e G_2 pertencem à família *mir-9* e, juntos, correspondem a 100% do total de 89 sequências de miRNAs; os grupos G_3 e G_4 pertencem à família *let-7* e, juntos, correspondem a 100% do total de 121 sequências de miRNAs; os grupos G_5–G_7 pertencem à família *mir-17* e, juntos, correspondem a 71,32% do total de 129 sequências de miRNAs.

Experimento ASME-euclidiana-5

O objetivo desse experimento é verificar se é possível separar as famílias *let-7* e *mir-17* do G_3 da Tabela 8.10, por meio de um grande aumento no valor do parâmetro número de grupos e, também, como base na comparação de resultados com aqueles obtidos pelo experimento *ASME-euclidiana-4*, que obteve melhor desempenho. Os valores dos parâmetros utilizados são mostrados na Tabela 8.12. Apesar do alto valor atribuído ao parâmetro número de grupo a remover, o passo 2.1 (Figura 8.1) foi utilizado para reduzir o número de grupos formados.

O resultado do agrupamento é exibido na Tabela 8.13 e os valores calculados para os índices são mostrados na Tabela 8.14.

Tabela 8.12 Parâmetros utilizados em *ASME-euclidiana-5*.

Algoritmo	Parâmetro	Valor
n-gram	N	5
ASME	Distância	euclidiana
	número de grupos	51

Tabela 8.13 Resultado de *ASME-euclidiana-5* para os parâmetros da Tabela 8.12.

Grupo	*mir-9*	*let-7*	*mir-17*
G_1	52	0	0
G_2	37	0	0
G_3	0	55	0
G_4	0	66	0
G_5	0	0	16
G_6	0	0	15
G_7	0	0	37
G_8	0	0	61

Tabela 8.14 Resultado dos índices para *ASME-euclidiana-5*.

Índice	Valor
Davies-Bouldin	1,97227739
Dunn	0,59628940
C	0,18101275

A Tabela 8.13 mostra que não há nenhuma sequência de miRNA agrupada incorretamente, ou seja, o G_3 da Tabela 8.10 (*ASME-euclidiana-4*) corresponde, neste experimento, aos grupos G_3 e G_7 da Tabela 8.13. Apesar dessa melhoria, houve um aumento no número de grupos formados, 7 grupos em *ASME-euclidiana-4* e 8 grupos neste experimento. Devido a esse aumento, o valor do índice *Davies-Bouldin* da Tabela 8.14 mostra que houve uma melhoria no resultado do agrupamento quando comparado ao valor do mesmo índice na Tabela 8.11. O valor do índice de *Dunn* da Tabela 8.14 mostra que não houve uma melhoria no resultado do agrupamento quando comparado ao valor do mesmo índice mostrado na Tabela 8.11. Já os valores do índice de *C* nas tabelas 8.11 e 8.14 mostram que houve uma sutil melhoria no resultado do algorimo.

Este experimento e o *ASME-euclidiana-4* podem ser considerados os melhores resultados obtidos pelo algoritmo ASME utilizando distância euclidiana, e servem de comparação para os próximos experimentos.

8.3.4 Influência de *Outliers* no Uso da Distância *city-block*

Experimento ASME-city-block-1

Nesse experimento foram utilizados os valores de parâmetros mostrados na Tabela 8.15. À semelhança com o experimento *ASME-euclidiana-2*, decidiu-se utilizar um valor maior que o número de famílias, associado ao parâmetro número de grupos, como uma tentativa para melhorar o resultado do agrupamento com relação à presença de *outliers*.

Após vários testes com a variação do valor desse parâmetro no intervalo [3, 30], foi decidido adotar o valor 25 pois, com esse valor, foram formados grupos significativos. O resultado do agrupamento é exibido na Tabela 8.16 e os valores dos índices calculados são mostrados na Tabela 8.17.

Tabela 8.15 Parâmetros utilizados em *ASME-city-block-1*.

Algoritmo	Parâmetro	Valor
n-gram	N	5
Isomap	K	10
	Dimensão	150
ASME	Distância	*city-block*
	número de grupos	25

Os resultados da Tabela 8.16 podem ser interpretados como a família *mir-9* sendo constituída por dois grupos majoritários, G_1 e G_2 com 56,18% e 31,46% das 89 sequências da família, e evidenciam *outliers*, nos grupos G_3–G_{11}; a família *let-7* é constituída pelo grupo majoritário G_{12} com 88,43% das 121 sequências da família e possui grupos *outliers* em G_1 e G_{13}–G_{24}; a família *mir-17* é constituída pelos grupos majoritários G_{20} e G_{25} com 44,19% e 0,03% das 129 sequências da família. Também é interessante observar que a família *let-7* possui 2 *outliers* classificados incorretamente, um no G_1 (grupo formado pela familía *mir-9*), e outro no G_{20} (grupo formado pela família *mir-17*). Já a família *mir-17* possui 68 sequências de miRNAs agrupadas incorretamente no grupo G_{12} (grupo formado pela família *let-7*).

Apesar do G_{12} ser composto por um número significativo de sequências das famílias *let-7* e *mir-17*, esse grupo possui concentrações dos elementos de cada família que, visualmente, estão separadas. Existem apenas quatro pontes significativas, destacadas por retângulos, que interligam esses dois grupos, conforme mostra a Figura 8.6. A Figura 8.7 mostra uma ponte, também do G_{12}, que une as famílias *let-7* e *mir-17*. Porém é importante observar que essa ponte une apenas 16 elementos pertencentes a família *mir-17* a família *let-7*.

Tabela 8.16 Resultado de *ASME-city-block-1* usando os parâmetros da Tabela 8.15.

Grupo	*mir-9*	*let-7*	*mir-17*
G_1	50	1 (cqu-let-7-3p)	0
G_2	28	0	0
G_3	3 (cel-miR-79-5p, tca-miR-9d-5p e crm-miR-79-5p)	0	0
G_4	1(sme-miR-79-5p)	0	0
G_5	1 (oha-miR-9-3-5p)	0	0
G_6	1 (bmo-miR-79-5p)	0	0
G_7	1 (sme-miR-9a-3p)	0	0
G_8	1 (tca-miR-9b-5p)	0	0
G_9	1 (dvi-miR-9b-3p)	0	0
G_{10}	1 (mdo-miR-9b-5p)	0	0
G_{11}	1 (pma-miR-9b)	0	0
G_{12}	0	107	68
G_{13}	0	1 (oan-let-7b-3p)	0
G_{14}	0	2 (bfl-let-7a-3p e bbe-let-7a-3p)	0
G_{15}	0	1 (dme-let-7-3p)	0
G_{16}	0	1 (ssa-let-7i-3p)	0
G_{17}	0	1 (bbe-let-7a-2-3p)	0
G_{18}	0	1 (sma-let-7-3p)	0
G_{19}	0	1 (rno-miR-3596b)	0
G_{20}	0	1 (cel-let-7-3p)	57
G_{21}	0	1 (asu-let-7-3p)	0
G_{22}	0	1 (oha-let-7a-3-3p)	0
G_{23}	0	1 (bta-miR-3596)	0
G_{24}	0	1 (rno-miR-3596d)	0
G_{25}	0	0	4

Tabela 8.17 Resultado dos índices nos resultados do *ASME-city-block-1*.

Índice	Valor
Davies-Bouldin	0,79231302
Dunn	0,50140001
C	0,13544366

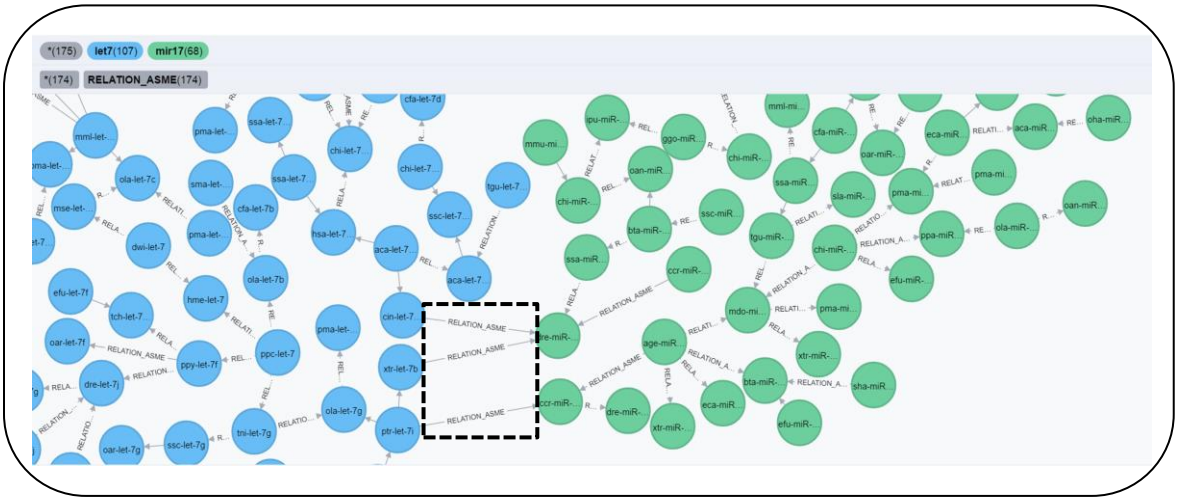

Figura 8.6 Duas pontes entre dre-miR-18c (*mir-17*) e cin-let-7a-2-3p / xtr-let-7b (*let-7*) e uma ponte entre ccr-miR-93 (*mir-17*) e ptr-let-7i (*let-7*) do G_{12}.

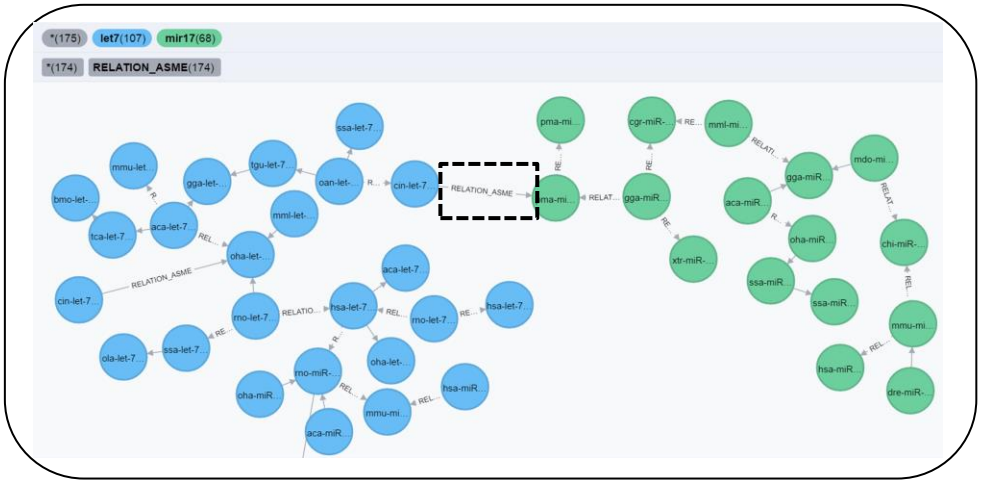

Figura 8.7 Ponte entre pma-miR-18b-3p (*mir-17*) e cin-let-7b-3p (*let-7*) do G_{12}.

As Figuras 8.8(a) e 8.8(b) mostram a distribuição dos valores das distâncias *city-block* entre os dados das famílias *let-7*, representado pela cor azul, e *mir-17*, representado pela cor verde, do agrupamento. Nelas são destacados os dados referentes às pontes mostradas na Figura 8.6. É possível notar, nas figuras 8.8(a) e 8.8(b), que as distâncias entre os elementos de famílias diferentes são menores que as distâncias entre elementos da mesma família. Ou seja, na Figura 8.8(a) a ponte entre dre-miR18c (*mir-17*) e cin-let-7a-2-3p (*let-7*) tem o valor da distância de 9.431 e entre dre-miR18c (*mir-17*) e xtr-let-7b (*let-7*) tem o valor da distância de 10.475. Essas distâncias são menores que a distância dos elementos cin-let-7a-2-3p (*let-7*) e aca-let-7a-2-3p (*let-7*) que possui o valor de 14.982. Isso

implica que um aumento no valor do parâmetro número de grupos, utilizado pelo algoritmo ASME, provocará a remoção de pontes entre elementos de uma mesma família antes da remoção de pontes que unem elementos pertencentes a famílias diferentes, levando à formação incorreta de grupos. Uma análise semelhante pode ser feita com relação à Figura 8.8(b).

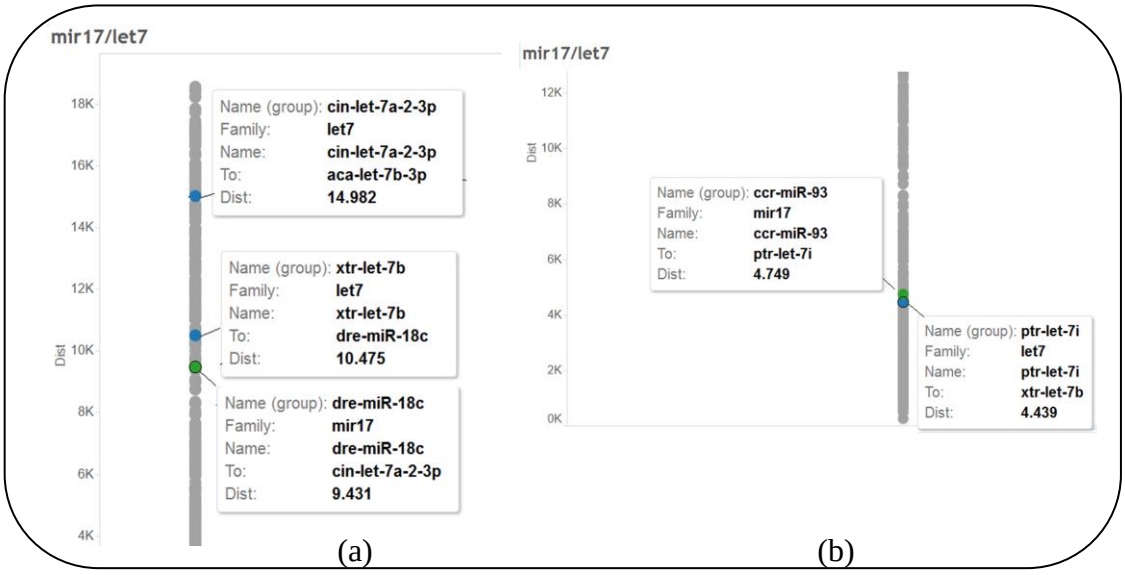

Figura 8.8 (a) Distâncias entre as duas pontes mostradas na Figura 8.6 e **(b)** Distância entre as sequências que definem a ponte mostrada na Figura 8.7.

A Figura 8.9 mostra o resultado do G_1 da Tabela 8.16.

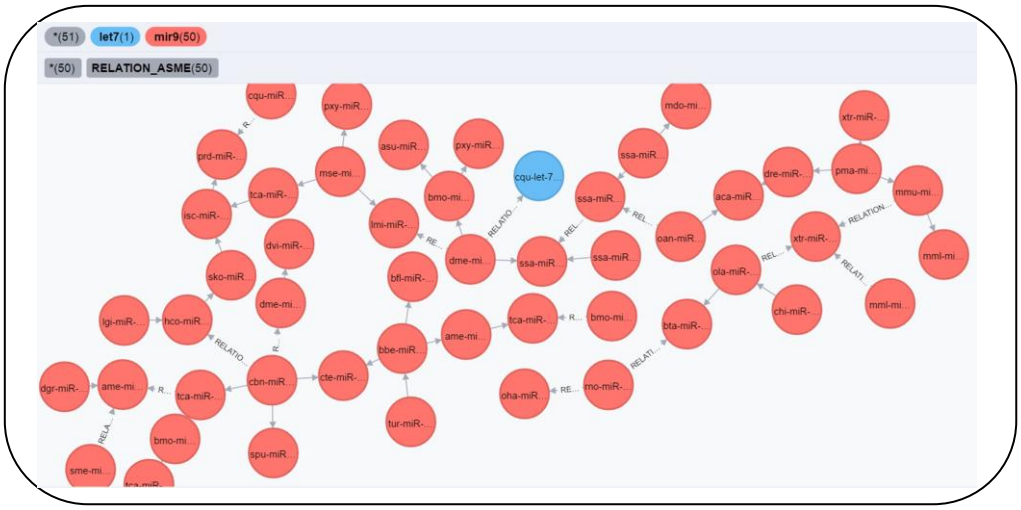

Figura 8.9 Grupo G_1 da Tabela 8.16.

8.3.5 Influência da Remoção de Grupos de Outliers no Uso da Distância city-block

Experimento ASME-city-block-2

Neste experimento foram utilizados os mesmos valores de parâmetros mostrados na Tabela 8.15. Esse experimento é similar ao experimento realizado na Subseção 8.3.2 com o objetivo de validar o passo 2.1 (Figura 8.1). O resultado do agrupamento é exibido na Tabela 8.18 e os valores dos três índices calculados são mostrados na Tabela 8.19.

Tabela 8.18 Resultado de *ASME-city-block-2* usando os parâmetros da Tabela 8.15.

Grupo	*mir-9*	*let-7*	*mir-17*
G_1	52	1 (cqu-let-7-3p)	0
G_2	37	0	0
G_3	0	118	68
G_4	0	1 (cel-let-7-3p)	57
G_5	0	1 (asu-let-7-3p)	4

Tabela 8.19 Resultado dos índices para *ASME-city-block-2*.

Índice	Valor
Davies-Bouldin	2,23151121
Dunn	0,50959865
C	0,15383636

O experimento mostra que a junção dos grupos de *outliers* com o grupo considerado o seu mais próximo melhora ligeiramente o resultado do agrupamento, o que também aconteceu no experimento *ASME-euclidiana-3*, descrito na Subseção 8.3.2. Os grupos G_4–G_{11} da Tabela 8.16 foram incorporados aos grupos G_1 ou G_2 da Tabela 8.18 e os grupos G_{13}–G_{19} e G_{22}–G_{24} da Tabela 8.16 foram associados ao G_{12} da mesma tabela, resultando no grupo G_3 da Tabela 8.18. As quatro pontes destacadas nas figuras 8.6 e 8.7 do grupo G_{12} da Tabela 8.16 permaneceram no grupo G_3 na Tabela 8.18 mantendo assim, as famílias *let-7* e *mir-17* unidas.

Na maioria dos casos os *outliers* foram corretamente agrupados, com exceção do G_{21} da Tabela 8.16 que foi incorretamente agrupado em G_{25} (*mir-17*) da Tabela 8.16, resultando no G_5 da Tabela 8.18. Pode ser percebido um aumento nos valores do índice *C* e *Davies-Bouldin* nas tabelas 8.17 e 8.19 o que indica uma piora no resultado do agrupamento. Os valores do índice *Dunn* nas tabelas 8.17 e 8.19 mostram que houve uma ligeira melhoria no

resultado do agrupamento. Os grupos G_1 e G_2 são considerados como pertencentes à família *mir-9* e, juntos, correspondem a 100% do total de 89 sequências de miRNAs. O grupo G_3 pertence à família *let-7* e corresponde a 97,52% do total de 121 sequências de miRNAs. Já os grupos G_4 e G_5 pertencem à família *mir-17* e, juntos, correspondem a 47,28% do total de 129 sequências de miRNAs.

Experimento ASME-city-block-3

Os valores de parâmetros utilizados neste experimento estão descritos na Tabela 8.20. O objetivo desse experimento é avaliar a utilização dos mesmos parâmetros utilizados em [Wan *et al.* 2012], em que foi obtido o melhor resultado do algoritmo de agrupamento. Os resultados do experimento são exibidos na Tabela 8.21 e os cálculos dos valores dos três índices são mostrados na Tabela 8.22. O passo 2.1 (Figura 8.1) foi utilizado para obter um melhor resultado do algoritmo ASME e o parâmetro referente ao número de grupos foi determinado por meio da variância de seus valores pertencentes ao intervalo [20,40]. É importante observar que com o mesmo valor de número de grupos igual a 41, conforme utilizado no *ASME-euclidiana-4*, foram formados 25 grupos, ou seja, um número muito elevado quando comparado ao número de famílias.

Tabela 8.20 Parâmetros utilizados em *ASME-city-block-3.*

Algoritmo	Parâmetro	Valor
n-gram	N	4
Isomap	k	10
	Dimensão	150
ASME	Distância	*city-block*
	número de grupos	36

Tabela 8.21 Resultado dos índices para *ASME-city-block-3* para os parâmetros da Tabela 8.20.

Grupo	*mir-9*	*let-7*	*mir-17*
G_1	52	0	0
G_2	37	0	0
G_3	0	55	63
G_4	0	9	9
G_5	0	55	55
G_6	0	2 (cqu-let-7-3p e cel-let-7-3p)	2 (chi-miR-17-3p e hsa-miR-106a-3p)

Tabela 8.22 Resultado dos índices para *ASME-city-block-3*.

Índice	Valor
Davies-Bouldin	2,24394249
Dunn	0,50473580
C	0,11317495

Os valores dos índices de *Davies-Bouldin* e *C* mostrados na Tabela 8.22 evidenciam que houve uma piora no resultado do agrupamento, quando comparado com os valores dos mesmos índices mostrados na Tabela 8.19. Os valores do índice de *Dunn* nas tabelas 8.19 e 8.22 mostram que houve uma sutil melhoria no resultado do agrupamento. Apesar disso, uma análise do resultado do agrupamento mostrado na Tabela 8.21 mostra que os grupos G_3–G_6 pertencem as famílias *let-7* e *mir-17*, e são os principais responsáveis pela queda de desempenho do algoritmo. Não é possível determinar a predominância de quantidade de sequências miRNAs nos grupos G_4–G_6 e, portanto, não é possível determinar à qual família pertence.

8.3.6 Influência no Uso do *Isomap*

Experimento ASME-city-block-4

Esse experimento tem por objetivo verificar o impacto da utilização de um processo de redução de dimensões (passo 1.3 da Figura 8.1) por meio da comparação do resultado desse experimento com o resultado do experimento *ASME-city-block-2*. Desse modo é possível verificar se houve alguma perda de informação relevante ao utilizar o algoritmo *Isomap*. Também é interessante comparar o resultado do agrupamento com aquele produzido no experimento *ASME-euclidiana-4*, para verificar o impacto da utilização da distância *city-block* ao invés da distância euclidiana.

Os valores dos parâmetros estão descritos na Tabela 8.23. Após vários testes com o valor do parâmetro número de grupos no intervalo [24, 40], o valor 30 foi escolhido, pois com o seu uso, obteve-se grupos significativos.

O resultado do agrupamento é exibido na Tabela 8.24 e os valores calculados para os índices são mostrados na Tabela 8.25. O passo 2.1 (Figura 8.1) foi utilizado para melhoria do resultado do algoritmo.

Tabela 8.23 Parâmetros utilizados em *ASME-city-block-4.*

Algoritmo	Parâmetro	Valor
n-gram	N	5
ASME	Distância número de grupos	*city-block* 30

Tabela 8.24 Resultado de *ASME-city-block-4* para os parâmetros da Tabela 8.23.

Grupo	*mir-9*	*let-7*	*mir-17*
G_1	52	0	0
G_2	37	0	0
G_3	0	55	37
G_4	0	66	0
G_5	0	0	16
G_6	0	0	15
G_7	0	0	61

Tabela 8.25 Resultado dos índices para *ASME-city-block-4*.

Índice	Valor
Davies-Bouldin	1,48496709
Dunn	0,61930063
C	0,14253081

O resultado da Tabela 8.24 é igual ao resultado da Tabela 8.10 referente ao experimento *ASME-euclidiana-4*, ou seja, o resultado desse experimento é considerado um dos melhores resultados, entre todos os experimentos que utilizam ASME como algoritmo de agrupamento e distância *city-block*. Quando comparado ao *ASME-city-block-2*, é possível observar que houve uma diminuição nos valores dos índices *Davies-Bouldin* e *C* bem como um aumento no valor do índice de *Dunn* quando comparados os valores das tabelas 8.25 e 8.19, ou seja, um indicativo de melhoria no resultado do agrupamento. Além disso três das quatro pontes responsáveis pela união das famílias *mir-17* e *let-7* (figuras 8.6 e 8.7) foram removidas pelo algoritmo ASME. Como consequência dessas remoções o grupo G_3 da Tabela 8.18 foi dividido nos grupos G_3–G_7 da Tabela 8.24, melhorando o resultado do agrupamento. A única ponte pertencente ao G_3 da Tabela 8.24, responsável por manter 37 sequências *mir-17* no mesmo grupo que as 55 sequências *let-7*, está identificada por meio do retângulo tracejado, na Figura 8.10.

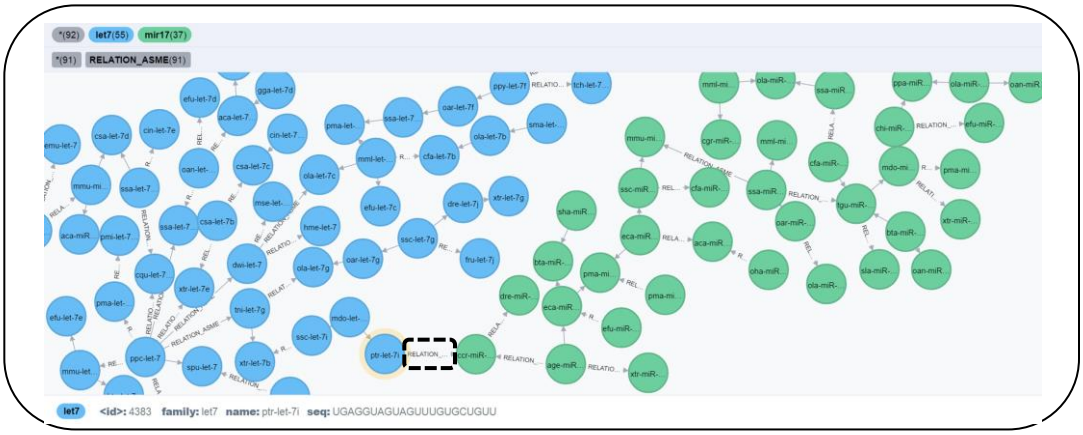

Figura 8.10 Ponte entre ccr-miR-93 (*mir-17*) e ptr-let-7i (*let-7*) do G_3.

Experimento ASME-city-block-5

O objetivo desse experimento é verificar se é possível separar as famílias *let-7* e *mir-17* do G_3 da Tabela 8.24, por meio de um grande aumento no valor do parâmetro número de grupos e, também, por meio da comparação de resultados com o experimento *ASME-city-block-4* que obteve melhor desempenho. Os valores dos parâmetros utilizados são mostrados na Tabela 8.26. Apesar do alto valor atribuído ao parâmetro número de grupos, o passo 2.1 (Figura 8.1) foi utilizado para reduzir o número de grupos formados.

O resultado do agrupamento é exibido na Tabela 8.27 e os valores calculados para os índices são mostrados na Tabela 8.28.

Tabela 8.26 Parâmetros utilizados em *ASME-city-block-5*.

Algoritmo	Parâmetro	Valor
n-gram	N	5
ASME	Distância	*city-block*
	número de grupos	56

Tabela 8.27 Resultado de *ASME-city-block-5* para os parâmetros da Tabela 8.26.

Grupo	*mir-9*	*let-7*	*mir-17*
G_1	32	0	0
G_2	37	0	0
G_3	20	0	0
G_4	0	55	0
G_5	0	59	0
G_6	0	7	0
G_7	0	0	16
G_8	0	0	15
G_9	0	0	37
G_{10}	0	0	61

Tabela 8.28 Resultado dos índices para *ASME-city-block-5*.

Índice	Valor
Davies-Bouldin	1,35717285
Dunn	0,45351576
C	0,14086077

A Tabela 8.27 mostra que não há nenhuma sequência de miRNA agrupada incorretamente, ou seja, o G_3 da Tabela 8.24 (*ASME-city-block-4*) corresponde, neste experimento, aos grupos G_4 e G_9 da Tabela 8.27. Apesar dessa melhoria, houve um aumento no número de grupos formados, 7 grupos em *ASME-city-block-4* e 10 grupos neste experimento, devido a formação do G_3 (*mir-9*) G_6 (*let-7*) e G_9 (*mir-17*) da Tabela 8.27. Devido a esse aumento, o valor do índice *Davies-Bouldin* da Tabela 8.28 evidencia que houve uma melhoria no resultado do agrupamento quando comparado ao valor do mesmo índice, mostrado na Tabela 8.25. Os valores do índice *C* das tabelas 8.25 e 8.28 também evidenciam que houve uma sutil melhoria no resultado. Os valores do índice *Dunn* da Tabela 8.28 não evidenciam que houve uma melhoria no resultado do agrupamento quando comparado aos valores do mesmo índice mostrados na Tabela 8.25.

É importante observar que este experimento é similar ao *ASME-euclidiana-5* porém, com um número maior de grupos formados, 8 grupos em *ASME-euclidiana-5* e 10 grupos neste experimento. Os valores do índice de *Dunn* das tabelas 8.14 e 8.28 mostram que o *ASME-euclidiana-5* obteve um melhor resultado enquanto que os valores dos índices de *Davies-Bouldin* e *C* das tabelas 8.14 e 8.28 mostram que este experimento obteve um melhor resultado. Este experimento e o *ASME-city-block-4* representam os melhores resultados obtidos pelo algoritmo ASME com distância *city-block*, e servem de comparação para os próximos experimentos.

8.3.7 Impacto do Aumento do Número de Famílias miRNA

Experimento ASME-5families

Esse experimento tem por objetivo analisar o impacto do aumento do número de famílias de miRNA consideradas. Além das três famílias sendo consideradas, duas novas famílias, escolhidas aleatoriamente, foram incorporadas, com o objetivo de avaliar um possível impacto no resultado obtido pelo algoritmo de agrupamento. A Tabela 8.29 lista as famílias participantes do experimento, o número de sequências participantes e o número de sequências sem repetições.

Tabela 8.29 Famílias de miRNA utilizadas em *ASME-5families*.

Família	Cor	Nro. original de sequências/Nro. de sequências sem repetição
mir-9	vermelho	227/89
let-7	azul	336/121
mir-17	verde	457/129
mir-139	lilás	44/20
mir-182	amarelo	54/22

A Figura 8.11 mostra a distribuição de distância *euclidiana* de todas as famílias apresentadas na Tabela 8.29, com destaque para área que contém os *outliers*.

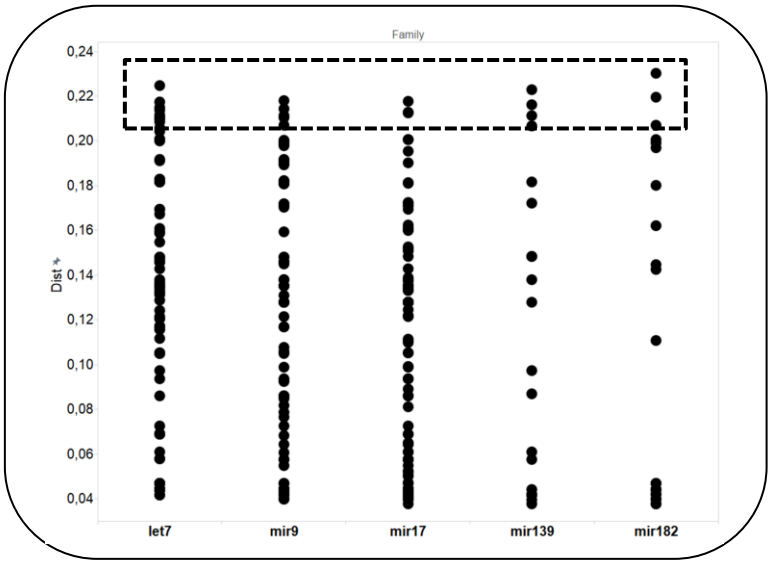

Figura 8.11 Distribuição da distância *city-block* dentro de cada família (*let-7, mir-9, mir-17, mir-139 e mir-182*).

Esse experimento utiliza os valores dos parâmetros que obtiveram o melhor resultado dentre todos os experimentos anteriores, ou seja, parâmetros determinados pelas tabelas 8.9

e 8.12 (*ASME-euclidiana-4* e *ASME-euclidiana-5*), ou tabelas 8.23 e 8.26 (*ASME-city-block-4* e *ASME-city-block-5*). Os testes foram realizados com os tipos de distâncias euclidiana e *city-block*, a distância *city-block* foi escolhida pois com o seu uso, o algoritmo obteve um melhor resultado. O passo 1.3 (Figura 8.1), relacionado a redução de dimensão, não será aplicado. Os valores dos parâmetros utilizados em *ASME-5families* estão descritos na Tabela 8.30.

O resultado do agrupamento é exibido na Tabela 8.31 e os valores calculados para os índices são mostrados na Tabela 8.32. Nesse experimento foi utilizado o passo 2.1 (Figura 8.1) para melhorar o resultado do agrupamento formado.

Tabela 8.30 Parâmetros utilizados em *ASME-5families*.

Algoritmo	Parâmetro	Valor
n-gram	N	5
ASME	Distância	*city-block*
	número de grupos	35

Tabela 8.31 Resultado do *ASME-5families* para os parâmetros da Tabela 8.30.

Grupo	*mir-9*	*let-7*	*mir-17*	*mir-139*	*mir-182*
G_1	52	0	0	0	0
G_2	37	0	0	0	5
G_3	0	55	37	0	0
G_4	0	66	0	0	0
G_5	0	0	16	0	0
G_6	0	0	15	0	0
G_7	0	0	61	1 (ssa-miR-139-2-3p)	0
G_8	0	0	0	9	0
G_9	0	0	0	10	0
G_{10}	0	0	0	0	13
G_{11}	0	0	0	0	4

Tabela 8.32 Resultado dos índices para *ASME-5families*.

Índice	Valor
Davies-Bouldin	1,30096987
Dunn	0,61808222
C	0,14898405

Os resultados mostram que o algoritmo consegue manter, na maioria dos casos, o melhor resultado obtido em *ASME-city-block-4* mesmo com o aumento no número de famílias. Uma das exceções é o G_7 (*mir-17*) da Tabela 8.31 que foi unido ao *outlier* pertencente a família *mir-139* conforme mostra o G_7 da Tabela 8.24. Esse *outlier* pertence a

área destacada na Figura 8.11. Outra exceção é o G_2 (*mir-9*) da Tabela 8.24 que foi unido as cinco sequências de *mir-182* conforme mostra o G_2 da Tabela 8.31. Essa união foi incorretamente realizada pelo passo 2.1 da Figura 8.1. As figuras 8.12 e 8.13 mostram os grupos G_2 e G_7 da Tabela 8.31, respectivamente.

Os grupos G_1–G_7 das tabelas 8.31 e 8.24 não sofreram grandes mudanças significativas. Os grupos G_8 e G_9 da Tabela 8.31 representam 95% do total de 20 sequências da família *mir-139* e os grupos G_{10} e G_{11} representam 77,27% do total de 22 sequências da familia *mir-182*. Não houve mudanças nas porcentagens das famílias *mir-9*, *let-7* e *mir-17* discutidas em *ASME-city-block-4*.

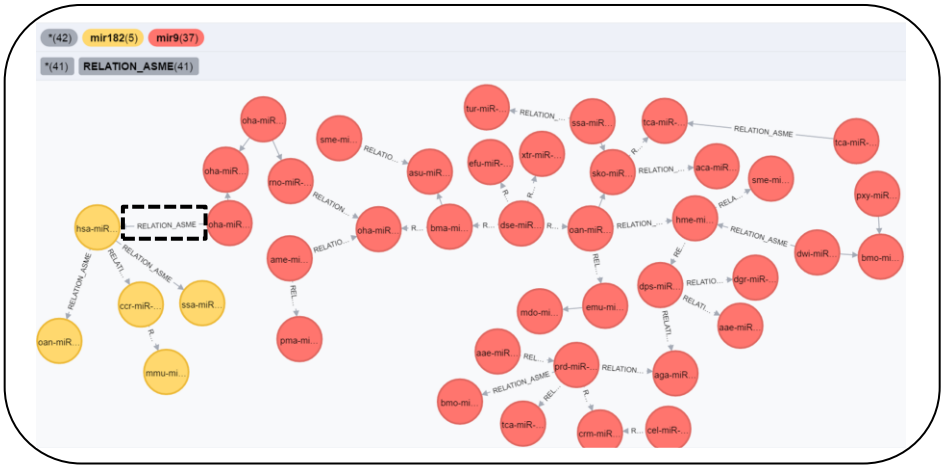

Figura 8.12 Ponte entre hsa-miR-182-3p (*mir-139*) e oha-miR-9-3-5p (*mir-9*) do G_2.

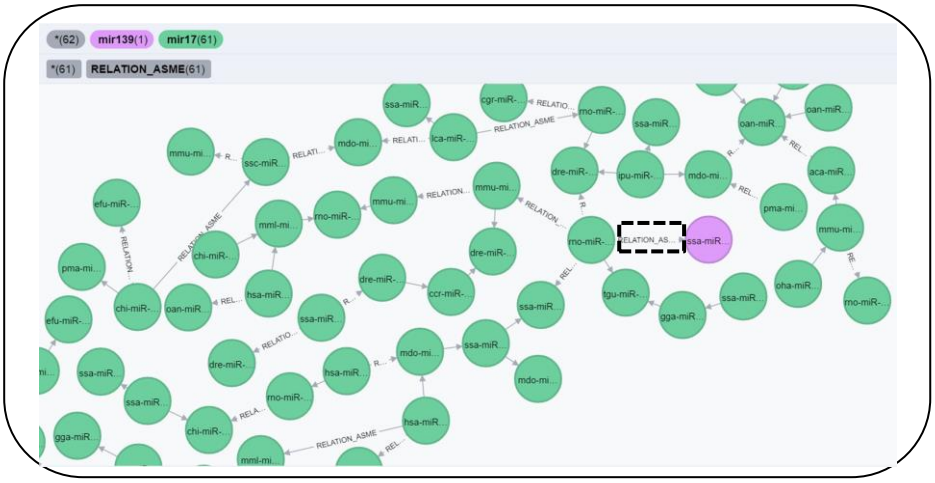

Figura 8.13 Ponte entre ssa-miR-139-2-3p (*mir-182*) e rno-miR-20b-3p (*mir-17*) do G_7.

8.3.8 Influência da Versão da Base de Dados miRBase

Experimento ASME-versao17

Os experimentos descritos nas seções anteriores usaram dados da última versão disponibilizada da base de dados miRBase *i.e.*, versão 21. Nos experimentos descritos em [Wan *et al.* 2012] foi usada a versão 17 dessa base de dados. Por isso, os experimentos descritos nessa seção têm por objetivo comparar o resultado do agrupamento utilizando a mesma versão da base de dados utilizada em [Wan *et al.* 2012]. Foram selecionadas 11 famílias, mostradas na Tabela 8.33, que possuem os maiores números de sequências, da versão 17 do miRBase. Para um melhor desempenho do algoritmo também houve a remoção das sequências repetidas encontradas em cada família.

Tabela 8.33 Nro de sequências sem repetição usadas em *ASME-versao17*.

Família	Cor	Nro. original de sequências/Nro. de sequências sem repetição
mir-154	lilás	237/70
let-7	azul	207/43
mir-17	verde	187/41
mir-156	cinza	162/26
mir-166	–	159/22
mir-515	laranja	146/70
mir-395	–	156/27
mir-9	vermelho	145/39
mir-2	rosa	140/46
mir-25	amarelo	133/46
mir-171_1	–	127/31

A Figura 8.14 mostra a distribuição de distância euclidiana das 11 famílias mostradas na Tabela 8.33, com destaque para área que contém os *outliers*. É importante observar que as famílias *mir-2* e *mir-154* são as famílias que possuem uma maior quantidade de *outliers*.

A Tabela 8.34 mostra os parâmetros utilizados no experimento. Apesar do alto valor associado ao parâmetro número de grupos, foi aplicado o passo 2.1 (Figura 8.1) com o objetivo de evitar a formação de grupos com *outliers*. Os testes foram realizados com o valor do número de grupos no intervalo [40, 120] e com distância *city-block* ou euclidiana. A distância euclidiana foi adotada pois, com o seu uso, obteve-se melhores resultados. O valor N de *n-grams* foi determinado conforme o melhor resultado descrito em [Wan *et al.* 2012]. Devido ao experimentos *ASME-euclidiana-4*, *ASME-euclidiana-5*, *ASME-city-*

block-4 e *ASME-city-block-5* mostrarem que o melhor resultado é obtido sem o uso da redução de dimensão, o passo 1.3 (Figura 8.1) não será utilizado neste experimento. Nestes mesmos experimentos também é possível observar que o valor adequado para N de *n-grams* é 5.

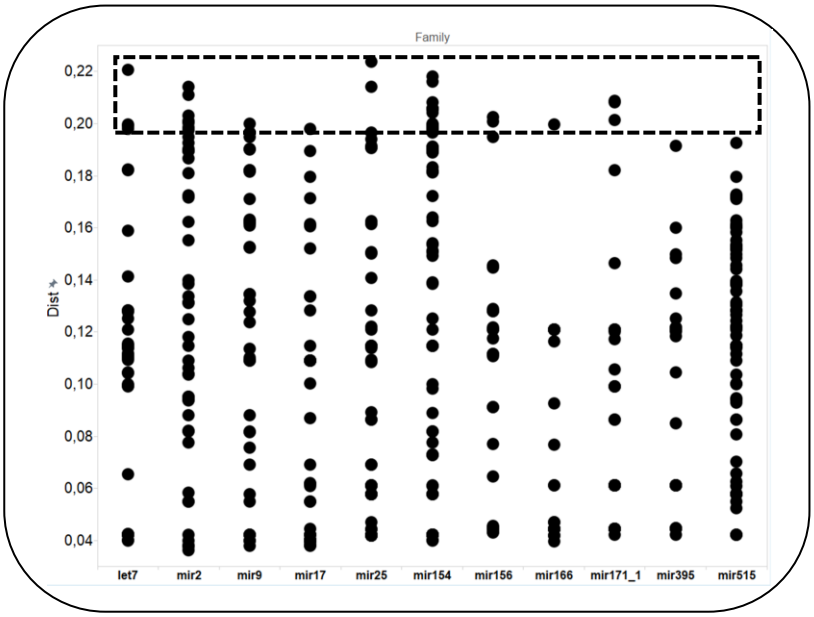

Figura 8.14 Distribuição da distância euclidiana dentro de cada família (*let-7, mir-2, mir-9, mir-17, mir-25, mir-154, mir-156, mir-166, mir-171_1, mir-395 e mir-515*).

O resultado do agrupamento é detalhado na Tabela 8.35 e os valores dos três índices são mostrados na Tabela 8.36.

Tabela 8.34 Parâmetros utilizados em *ASME-versao17*.

Algoritmo	Parâmetro	Valor
n-gram	N	4
ASME	Distância	euclidiana
	número de grupos	104

Tabela 8.35 Tabela 8.35 Resultado de *ASME-versao17* para os parâmetros da Tabela 8.34.

Grupo	*mir-154*	*let-7*	*mir-17*	*mir-156*	*mir-166*	*mir-515*	*mir-395*	*mir-9*	*mir-2*	*mir-25*	*mir-171_1*
G_1	9	0	0	0	0	0	0	0	0	0	0
G_2	7	0	0	2	0	70	0	0	1	0	0
G_3	7	3	0	0	0	0	0	0	0	0	0
G_4	6	0	0	0	0	0	0	0	0	0	0
G_5	8	1	0	0	0	0	0	0	0	0	0
G_6	13	0	0	0	0	0	0	0	0	0	0
G_7	1	1	0	0	0	0	0	0	0	42	0
G_8	4	0	0	0	0	0	0	0	0	0	0
G_9	3	0	0	0	0	0	0	0	0	1	28
G_{10}	3	0	0	0	0	0	0	19	0	0	0
G_{11}	3	0	0	0	0	0	0	0	0	0	0
G_{12}	3	0	0	0	0	0	0	0	0	0	0
G_{13}	1	0	0	24	0	0	0	0	0	0	0
G_{14}	1	0	35	0	0	0	0	0	0	3	0
G_{15}	1	0	0	0	0	0	0	0	31	0	1
G_{16}	0	38	0	0	0	0	0	0	1	0	0
G_{17}	0	0	6	0	0	0	0	0	0	0	0
G_{18}	0	0	0	0	22	0	0	0	0	0	0
G_{19}	0	0	0	0	0	0	27	0	0	0	0
G_{20}	0	0	0	0	0	0	0	20	0	0	0
G_{21}	0	0	0	0	0	0	0	0	13	0	2

Tabela 8.36 Resultado dos índices para *ASME-versao17*.

Índice	Valor
Davies-Bouldin	1,76332245
Dunn	0,59201417
C	0,16059290

A Tabela 8.37 mostra um resumo das porcentagens de sequências pertencentes em cada grupo mostrado na Tabela 8.35. É importante observar que a família *mir-154* é a principal responsável pela queda de desempenho do algoritmo, pois suas sequências ficaram distribuídas em diversos grupos.

Tabela 8.37 Porcentagem de sequências dos grupos por família.

Família miRNA	Grupos que pertencem a família	Total de Sequência miRNA da família nos grupos	Porcentagem do total de sequências
mir-154	G_1, G_3, G_4,G_5, G_6, G_8, G_{11} e G_{12}	53	75,71%
let-7	G_{16}	38	88,37%
mir-17	G_{14} e G_{17}	41	100%
mir-156	G_{13}	24	92,31%
mir-166	G_{18}	22	100%
mir-515	G_2	70	100%
mir-395	G_{19}	27	100%
mir-9	G_{10} e G_{20}	39	100%
mir-2	G_{15} e G_{21}	44	95,65%
mir-25	G_7	42	91,30%
mir-171_1	G_9	28	90,32%

A Figura 8.15 mostra o grupo G_{14} da Tabela 8.35 e a Figura 8.16 mostra o grupo G_2 da Tabela 8.35. Ambas as figuras mostram a presença de *outliers* associados a uma família dominante do grupo.

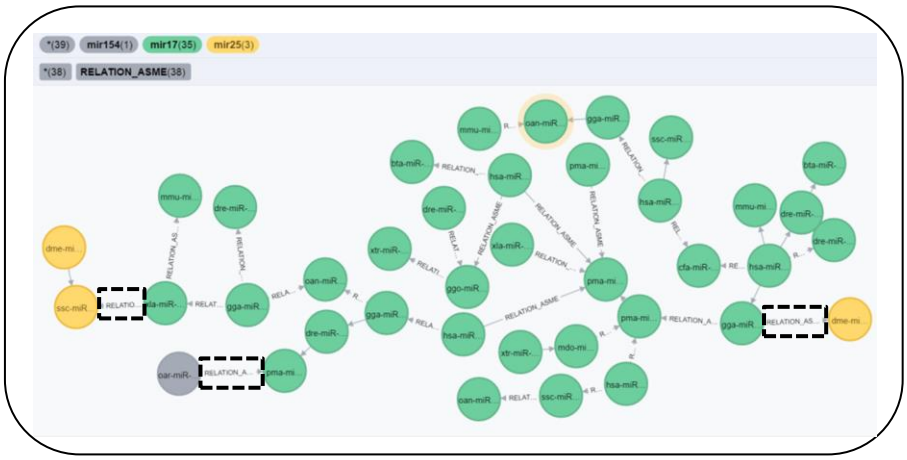

Figura 8.15 Pontes entre ssc-miR-92b-5p (*mir-25*) e xla-miR-18 (*mir-17*), entre dme-miR-92b-5p (*mir-25*) e gga-miR-17-5p (*mir-17*) e entre oar-miR-154b-5p (*mir-154*) e pma-miR-18b (*mir-17*) do G_{14}.

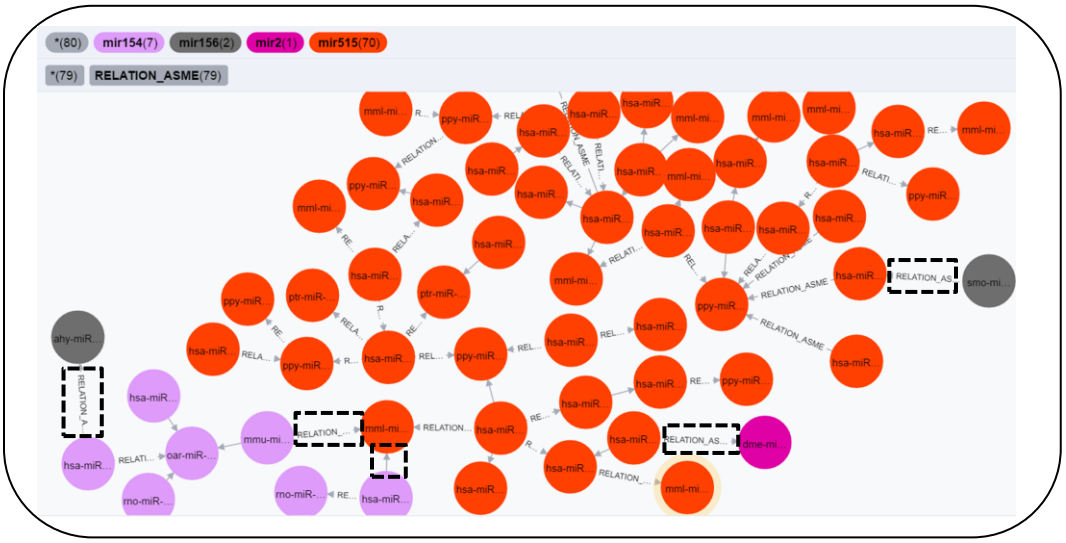

Figura 8.16 Pontes entre ahy-miR156b-3p (*mir-156*) e hsa-miR-381 (*mir-154*), entre mmu-miR-300 / hsa-miR-377 (*mir-154*) e mml-miR-524 (*mir-515*), entre hsa-miR-515-5p (*mir-515*) e dme-miR-2c-5p (*mir-2*) e entre hsa-miR-515-3p (*mir-515*) e smo-miR1088-3p (*mir-156*) do G_2.

Experimento miRCluster

A implementação do algoritmo descrito em [Wan *et al.* 2012] foi disponibilizada em uma página Web para *download* (http://admis.fudan.edu.cn/projects/miRCluster.html) e foi usada para o experimento descrito nessa seção. O algoritmo *Isomap* foi implementado em *Matlab* e o algoritmo *k-means*, bem como o pré-processamento dos dados, foram implementados utilizando a linguagem *Python*, durante o trabalho de pesquisa realizado, descrito nesta dissertação.

No experimento foi utilizado a versão 17 da base de dados miRbase [Griffiths-Jones *et al.* 2006] com 419 diferentes famílias, totalizando 10.680 sequências de miRNA. Durante a fase de pré-processamento desses dados com o objetivo de eliminação de sequências repetidas, o número de sequências foi reduzido de 10.680 a 3.453. A Tabela 8.38 mostra os resultados obtidos pelo miRCluster das 11 das 419 maiores famílias selecionadas. No total foram induzidos 834 grupos utilizando os parâmetros descritos na Tabela 8.39. A segunda coluna mostra os grupos em que a família é predominante, ou seja, a família possui a maior quantidade quando comparada com o total de sequências miRNAs no grupo. Essa coluna segue o seguinte padrão: x/y, em que x é a quantidade da família no grupo, e y é a quantidade total de sequências no grupo. A terceira coluna mostra a porcentagem de sequências miRNAs nos grupos da família.

Tabela 8.38 Grupos formados pelo miRCluster.

Família miRNA	Grupos que pertencem a família	% do total de sequências
mir-154	G_{11} (17/17),G_{100} (11/11),G_{106} (20/20),G_{117} (8/8),G_{144} (2/4),G_{151} (10/12),G_{157} (11/14),G_{233} (9/11),G_{258} (2/5),G_{313} (5/10),G_{332} (9/9),G_{344} (5/7),G_{373} (13/13),G_{447} (11/11),G_{470} (9/9),G_{525} (3/5),G_{561} (7/7),G_{570} (21/21),G_{627} (11/11),G_{678} (14/14),G_{720} (2/5),G_{727} (4/4),G_{736} (3/3)	87,34%
let-7	G_{168} (12/14),G_{206} (36/38),G_{333} (3/3),G_{596} (26/27),G_{648} (126/126)	98,06%
mir-17	G_{93} (44/46),G_{102} (19/19),G_{182} (19/27),G_{445} (15/21),G_{538} (45/45),G_{639} (44/44)	99,47%
mir-156	G_{29} (145/152),G_{249} (15/38)	98,76%
mir-166	G_{268} (152/152),G_{401} (7/10)	100%
mir-515	G_{85} (9/18),G_{253} (13/13),G_{261} (31/31),G_{290} (3/5),G_{291} (29/29),G_{327} (8/18),G_{394} (8/8),G_{457} (5/7),G_{531} (3/3),G_{600} (18/20),G_{643} (12/13)	95,20%
mir-395	G_{94} (2/3),G_{363} (139/143),G_{611} (12/14)	98,08%
mir-9	G_{161} (67/67),G_{336} (14/17),G_{348} (4/5),G_{351} (2/3),G_{711} (13/13),G_{742} (39/40)	95,86%
mir-2	G_{21} (2/3),G_{41} (2/3),G_{155} (4/4),G_{177} (105/105),G_{404} (2/3),G_{607} (8/12)	87,86%
mir-25	G_{35} (20/24),G_{91} (3/6),G_{98} (2/4),G_{218} (18/18),G_{468} (2/5),G_{588} (16/16),G_{749} (64/76)	93,98%
mir-171_1	G_{103} (91/94),G_{172} (26/38)	92,12%

Tabela 8.39 Parâmetros utilizados em *miRCluster*.

Algoritmo	Parâmetro	Valor
n-gram	N	4
Isomap	k	10
	Dimensão	150
k-means	Distância	*city-block*
	número de grupos	834

É importante observar que, similarmente ao experimento *ASME-versao17*, os elementos da família *mir-154* foram distribuídos entre diversos grupos, no caso 23 grupos. Já as famílias *mir-166* e *mir-395* apresentaram os melhores desempenhos em ambos os experimentos, *ASME-versao17* e *miRCluster*; formando poucos grupos com uma quantidade alta de sequências pertencentes às suas respectivas famílias de origem. O valor do número de grupos utilizado (ver parâmetro número de grupos na Tabela 8.39) é maior que as 419 famílias utilizadas no experimento. Esse alto valor no número de grupos também ocorre nos experimentos descritos nos capítulos 8 e 9.

Capítulo 9

Experimentos Complementares

9.1 Considerações Iniciais

Este capítulo apresenta 8 experimentos completares aos experimentos mostrados no Capítulo 8, ou seja, são experimentos que utilizam os algoritmos apresentados na Seção 3 (ASMEH, ASM-PI e ASMTXE), na Seção 4 (Região de Influência) e na Seção 5 (*Ncut*) e que possuem os melhores resultados. As três famílias de miRNAs (*mir-9, let-7* e *mir-17*) mostradas na Tabela 8.1 e as cinco famílias de miRNAs (*mir-9, let-7, mir-17, mir-139* e *mir-182*) mostradas na Tabela 8.29 foram utilizadas nos experimentos.

9.2 Experimentos Usando o Algoritmo ASM-PI (ASM baseada em remoção de Pontes Inconsistentes), Resultados e Análises

Experimento ASM-PI-euclidiana

Nesta seção é apresentado o melhor resultado obtido pelo algoritmo ASM-PI utilizando as três famílias, *mir-9, let-7* e *mir-17* descritas na Tabela 8.1. Os experimentos intermediários, realizados com variação dos valores dos parâmetros, não serão apresentados. O algoritmo ASM-PI utiliza dois parâmetros, o número de passos entre duas arestas, denominado k, e o valor limite que determina se uma aresta será removida ou não, denominado q. Nos experimentos $k \in [3, 10]$ e $q \in [0{,}3, 0{,}7]$. Como os experimentos *ASME-euclidiana-4*, *ASME-euclidiana-5*, *ASME-city-block-4* e *ASME-city-block-5* evidenciaram que a utilização do *Isomap* afeta negativamente o desempenho do algoritmo, o passo 1.3 (Figura 8.1) não foi utilizado. Nestes mesmos experimentos também é possível observar que o valor adequado para o N de *n-grams* é 5.

O melhor resultado obtido foi com a utilização de k = 3, q = 0,5 e distância euclidiana. A Tabela 9.1 mostra o resultado do agrupamento e a Tabela 9.2 mostra os valores calculados dos três índices.

Tabela 9.1 Melhor resultado obtido pelo ASM-PI.

Grupo	*mir-9*	*let-7*	*mir-17*
G_1	52	0	0
G_2	37	0	0
G_3	0	55	0
G_4	0	66	0
G_5	0	0	77
G_6	0	0	15
G_7	0	0	37

Tabela 9.2 Resultado dos índices para os grupos da Tabela 9.2.

Índice	Valor
Davies-Bouldin	2,06287877
Dunn	0,59628940
C	0,18898126

O resultado mostrado na Tabela 9.1 evidencia um desempenho superior aos resultados mostrados na Tabela 8.10 (*ASME-euclidiana-4*) e Tabela 8.24 (*ASME-city-block-4*), pois nenhuma sequência foi incorretamente agrupada pelo algoritmo ASM-PI. Quando comparado com os experimentos *ASME-euclidiana-5* e *ASME-city-block-5*, em que não há sequências de miRNAs agrupadas incorretamente, também nota-se um melhor desempenho, pois a quantidade de grupos formados (Tabela 9.1) é menor que a quantidade de grupos formados nos dois experimentos citados acima (ver Tabela 8.13 e Tabela 8.27). Ou seja, o ideal é o número de grupos formados esteja próximo do número de famílias utilizadas no experimento. Os valores dos índices da Tabela 9.2, quando comparados com os valores mostrados em Tabela 8.11, Tabela 8.14, Tabela 8.25 e Tabela 8.28 evidenciam que, em geral, não houve uma melhoria no resultado do agrupamento. A exceção foi o aumento do valor do índice de *Dunn* (comparar Tabela 8.28 e Tabela 9.2), implicando melhoria no resultado do agrupamento.

Experimento ASM-PI-5families

O experimento *ASM-PI-euclidiana* obteve um dos melhores resultados dentre todos os experimentos que utilizaram as famílias *mir-9*, *let-7* e *mir-17* da miRBase, versão 21. Isso deve-se ao fato de que não houve sequência de miRNA agrupada incorretamente e o número de grupos formados foi o mais próximo do número de famílias utilizadas no experimento. Para verificar se o aumento do número de famílias afeta o desempenho do

algoritmo, foi realizado o experimento *ASM-PI-5families*, descrito na sequência. As famílias utilizadas nos experimentos estão especificadas na Tabela 8.29 (*mir9, let-7, mir-17, mir-139* e *mir-182).*

Nesse experimento a variação dos valores dos parâmetros do algoritmo ASM-PI foi k $\in$ [3, 5] e q $\in$ [0,01, 1] para as famílias descritas na Tabela 8.29. A Tabela 9.3 mostra o resultado do algoritmo ASM-PI com distância euclidiana, utilizando os parâmetros q = 3 e k = 0,9. Com a utilização desses valores o algoritmo obteve um melhor resultado dentre todos os testes realizados com variação dos valores de k e q. Como os experimentos *ASME-euclidiana-4, ASME-euclidiana-5, ASME-city-block-4* e *ASME-city-block-5* evidenciaram que a utilização do *Isomap* afeta negativamente o desempenho do algoritmo, o passo 1.3 (Figura 8.1) não foi utilizado. Nestes mesmos experimentos também é possível observar que o valor adequado para N de *n-grams* é 5. A Tabela 9.4 mostra o resultado do cálculo dos três índices utilizados para avaliação do agrupamento formado.

Tabela 9.3 Resultado do ASM-PI para *ASM-PI-5families*.

Grupo	*mir-9*	*let-7*	*mir-17*	*mir-139*	*mir-182*
G_1	33	0	0	0	0
G_2	37	0	0	0	5
G_3	19	0	0	0	0
G_4	0	44	0	0	0
G_5	0	19	0	0	0
G_6	0	11	0	0	0
G_7	0	47	0	0	0
G_8	0	0	16	0	0
G_9	0	0	15	0	0
G_{10}	0	0	37	0	0
G_{11}	0	0	61	1 (ssa-miR-139-2-3p)	3
G_{12}	0	0	0	9	0
G_{13}	0	0	0	10	0
G_{14}	0	0	0	0	14

Tabela 9.4 Resultado dos índices para os grupos da Tabela 9.1.

Índice	Valor
Davies-Bouldin	1,84653748
Dunn	0,44170606
C	0,19235520

No experimento similar a este, o *ASME-5families* da Subseção 8.3.7, foram formados 11 grupos com 43 sequências miRNAs agrupadas incorretamente (37 do G_3 (*mir-17*), 1 do

G_7 (*mir-139*) e 5 do G_2 (*mir-182*)), (ver Tabela 8.31). A Tabela 9.3 mostra os 14 grupos formados com 9 sequências de miRNAs agrupadas incorretamente (5 do G_2 (*mir-182*), 1 do G_{11} (*mir-139*) e 3 do G_{11} (*mir-182*)). Apesar dos índices *Davies-Bouldin*, *Dunn* e *C* da Tabela 8.32 evidenciarem que o *ASME-5families* obteve um melhor agrupamento, quando comparado ao agrupamento obtido cujos valores dos mesmos índices estão na Tabela 9.4, essa melhoria não pode ser confirmada devido à sua alta quantidade de sequências agrupadas incorretamente no agrupamento.

A Tabela 9.3 evidencia que os grupos G_1–G_3 (juntos correspondem a 100% do total de sequências na família) pertencem à família *mir-9*, G_4–G_7 (juntos correspondem a 100% do total de sequências na família) pertencem à família *let-7* e G_8–G_{11} (juntos correspondem a 100% do total de sequências na família) pertencem à família *mir-17*. Os grupos G_{12} e G_{13} pertencem à família *mir-139* e juntos, correspondem a 95% do total de sequências da família. Por fim, o grupo G_{14} pertence à família *mir-182* e corresponde a 63,63% do total de sequências na família.

9.3 Experimentos Usando o Algoritmo ASM-TXE (ASM baseado em Taxa de Erro), Resultados e Análises

Experimento ASM-TXE-euclidiana

Nesta seção é apresentado o melhor resultado obtido pelo algoritmo ASM-TXE, utilizando as três famílias, *mir-9*, *let-7* e *mir-17*, descritas na Tabela 8.1. O algoritmo ASM-TXE remove pontes com base no cálculo de taxa de erro, que não deve ser superior a 50%. Com o valor de 50%, o algoritmo forma uma única componente conexa pois as distâncias entre as sequências de miRNAs são muito próximas, ou seja, ao relacionar os pesos das arestas para o cálculo da taxa de erro, seus valores ficam acima de 96%. O algoritmo ASM-TXE foi então alterado para remover as pontes caso a taxa de erro seja menor que 99,989%. Ou seja, uma ASM com os pesos de suas arestas organizadas em ordem decrescente, deverá ter suas arestas removidas enquanto a proporção entre o peso de uma aresta e_{i+1} e e_i não for superior a 99,989%. Isso implica que vértices cujas arestas possuem valores de pesos muito próximos, acima de 99,989%, pertencem ao mesmo grupo.

Como os experimentos *ASME-euclidiana-4*, *ASME-euclidiana-5*, *ASME-city-block-4* e *ASME-city-block-5* evidenciaram que a utilização do *Isomap* afeta negativamente o

desempenho do algoritmo, o passo 1.3 (Figura 8.1) não foi utilizado. Nestes mesmos experimentos também é possível observar que o valor adequado para N de *n-grams* é 5. A Tabela 9.5 mostra o resultado do agrupamento e a Tabela 9.6 mostra os valores calculados dos três índices.

Tabela 9.5 Melhor resultado obtido pelo ASM-TXE.

Grupo	*mir-9*	*let-7*	*mir-17*
G_1	52	0	0
G_2	37	0	0
G_3	0	55	0
G_4	0	59	0
G_5	0	7	0
G_6	0	0	16
G_7	0	0	15
G_8	0	0	37
G_9	0	0	61

Tabela 9.6 Resultado dos índices para os grupos da Tabela 9.5.

Índice	Valor
Davies-Bouldin	1,87538727
Dunn	0,59628940
C	0,17994838

A Tabela 9.5 mostra que o algoritmo ASM-TXE obteve um resultado similar aos experimentos *ASME-euclidiana-5* e *ASME-city-block-5*, com a diferença no número de grupos formados, 9 grupos na Tabela 9.5, 8 grupos na Tabela 8.13 e 10 grupos na Tabela 8.27, respectivamente. As tabelas 9.5, 8.13 e 8.27 mostram que nestes experimentos não há sequência de miRNA agrupada incorretamente. Os valores do índice de *Dunn* mantiveram-se iguais nas tabelas 8.14 e 9.6 e esses valores são maiores que o valor do mesmo índice na Tabela 8.28, evidenciando que os resultados neste experimento e em *ASME-euclidiana-5*, são melhores que o resultado do experimento *ASME-city-block-5*. Em geral, os valores dos índices de *Davies-Bouldin* e *C* (Tabela 8.11 e Tabela 8.14) mostram que houve uma melhoria no resultado, quando comparados com os valores dos mesmos índices contidos na Tabela 9.6; considerando agora a Tabela 8.25 e a Tabela 8.28, os valores desses índices indicam que houve uma piora no resultado do agrupamento quando comparados com os valores dos mesmos índices mostrados na Tabela 9.6.

9.4 Experimentos Usando o Algoritmo ASMEH(ASM euclidiana e Hierárquica), Resultados e Análises

Os resultados do algoritmo ASMEH foram idênticos aos resultados do algoritmo ASME. Ou seja, foram utilizados os mesmo valores de parâmetros descritos em *ASME-euclidiana-4*, *ASME-euclidiana-5*, *ASME-city-block-4* e *ASME-city-block-5* e obteve-se os mesmos resultados. Portanto não serão exibidos os resultados nesta seção.

9.5 Experimentos Usando o Algoritmo *Ncut* (*Normalized Cut*), Resultados e Análises

Experimento Ncut-euclidiana

O algoritmo de *Ncut* utiliza apenas um parâmetro para controlar a quantidade de cortes que serão realizadas na ASM. Esse parâmetro é definido pelo número de iterações, onde em cada iteração o algoritmo poderá efetuar uma bipartição do grafo. Os testes utilizaram valores para o número de iterações no intervalo [2, 30]. O melhor valor utilizado para este parâmetro foi 10, ou seja, com 10 iterações o algoritmo obteve seu melhor resultado. Como os experimentos *ASME-euclidiana-4*, *ASME-euclidiana-5*, *ASME-city-block-4* e *ASME-city-block5* evidenciaram que a utilização do *Isomap* afeta negativamente o desempenho do algoritmo, o passo 1.3 (Figura 8.1) não foi utilizado. Nestes mesmos experimentos também é possível observar que o valor adequado para N de *n-grams* é 5.

A Tabela 9.7 mostra o resultado do *Ncut* e a Tabela 9.8 mostra os valores calculados dos três índices utilizados para avaliação do resultado do agrupamento. Apesar do algoritmo evitar a formação de grupos com apenas 1 elemento, o passo 2.1 (Figura 8.1) foi utilizado com objetivo de melhorar seu resultado, pois este passo realiza a junção de grupos com mais de 1 elemento.

Os resultados da Tabela 9.7 mostram que nenhuma sequência de miRNA foi incorretamente agrupada pelo algoritmo. Esse resultado é similar aos resultados obtidos nos experimentos *ASME-euclidiana-5*, *ASME-city-block-5* e nos experimentos com os algoritmos ASM-PI e ASM-TXE. Em todos esses experimentos não há sequências de miRNAs agrupadas incorretamente. A diferença entre eles está no número de grupos formados em cada experimento, 7 grupos em *ASM-PI-euclidiana* (ver Tabela 9.1), 8 grupos em *ASME-euclidiana-5* (ver Tabela 8.13), 9 grupos em *ASM-TXE-euclidiana* e *NCut-*

euclidiana (ver tabelas 9.6 e 9.8) e 10 grupos em *ASME-city-block-5* (ver Tabela 8.27). Ao comparar os valores da Tabelas 9.6 e Tabela 9.8 é possível identificar que os índices *Davies-Bouldin* e *Dunn* evidenciam que o experimento com o ASM-TXE produziu um resultado melhor. Já o valor do índice *C* apresentado nas mesmas duas tabelas evidencia que o resultado do agrupamento produzido pelo *Ncut* é melhor que o resultado do agrupamento utilizando o ASM-TXE.

Tabela 9.7 Melhor resultado obtido pelo *Ncut*.

Grupo	*mir-9*	*let-7*	*mir-17*
G_1	52	0	0
G_2	37	0	0
G_3	0	55	0
G_4	0	66	0
G_5	0	0	16
G_6	0	0	23
G_7	0	0	29
G_8	0	0	48
G_9	0	0	13

Tabela 9.8 Resultado dos índices para os grupos da Tabela 9.7.

Índice	Valor
Davies-Bouldin	2,03394980
Dunn	0,41864733
C	0,17523717

9.6 Experimentos Usando o Algoritmo SFASM (Algoritmo baseado em ASM com Estrutura de Escala Livre), Resultados e Análises

Nesta seção são descritos alguns experimentos realizados para avaliar o uso do algoritmo SFASM para a geração de agrupamentos no domínio de dados das seguintes famílias de miRNA *mir-9*, *let-7*, *mir-17*, *mir-139* e *mir-182*. O resultado desse algoritmo tem como particularidade a visualização dos dados em formato de *hubs* e longos ramos. Cada *hub* representa um conjunto de sequências que pertencem ao mesmo grupo com um elemento central, responsável pela concentração das sequências de miRNAs. Os longos ramos representam os dados que não pertencem a nenhum grupo ou *hub*. Essa visualização é feita por meio de uma única componente conexa, o que dificulta a contagem das sequências de miRNAs que pertencem ou não a um determinado grupo. Ou seja, não é possível especificar os limites de onde começa e onde termina um grupo ou um longo ramo.

O algoritmo SFASM utiliza dois parâmetros relacionados à quantidade mínima de *links* ou arestas e um fator para determinar o valor de atualização de peso das arestas. Os valores desses parâmetros foram variados no intervalo [2, 5] e [0,05, 2], respectivamente. Os resultados mais interessantes estão descritos nas tabelas 9.9 e 9.10. Foi utilizado N = 4 ou N = 5 pois em ambos os casos os resultados obtidos foram significativos. Neste experimento também foi realizado testes com e sem a utilização de *Isomap,* tanto para distância *city-block* quanto euclidiana. Os resultados que visualmente geraram grupos consistentes foram os que utilizaram distância euclidiana e, neste caso, tanto a utilização ou não do *Isomap* produziu grupos com características significativas. O passo 2.1 (Figura 8.1) assim como cálculo dos valores índices e contagem da quantidade de sequências nos grupos não serão utilizados pois não são formados várias componentes conexas como nos experimentos anteriores.

Experimento SFASM-1

No experimento a seguir foram utilizados os valores de parâmetros mostrados na Tabela 9.9.

Tabela 9.9 Parâmetros utilizados em *SFASM-1*.

Algoritmo	Parâmetro	Valor
n-gram	N	5
	Distância	euclidiana
SFASM	número de links	3
	fator	0,018

As figuras 9.1, 9.2 e 9.3 mostram os *hubs* formados pelo algoritmo SFASM. As figuras 9.1 e 9.3 mostram *hubs* consistentes, ou seja, formados unicamente pelas famílias *let-7* e *mir-17*. Os *hubs* mostrados na Figura 9.2 são formados pela maioria das sequências pertencentes à família *mir-9*, porém possuem algumas sequências miRNAs pertencentes às famílias *let-7* e *mir-17*.

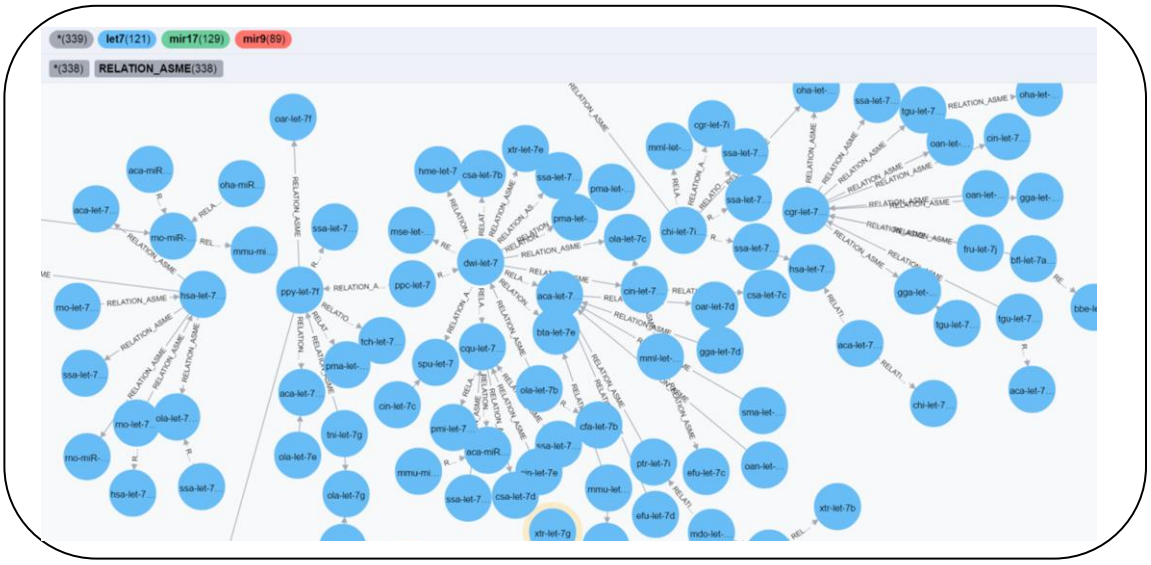

Figura 9.1 *Hubs* pertencentes à família *let-7*.

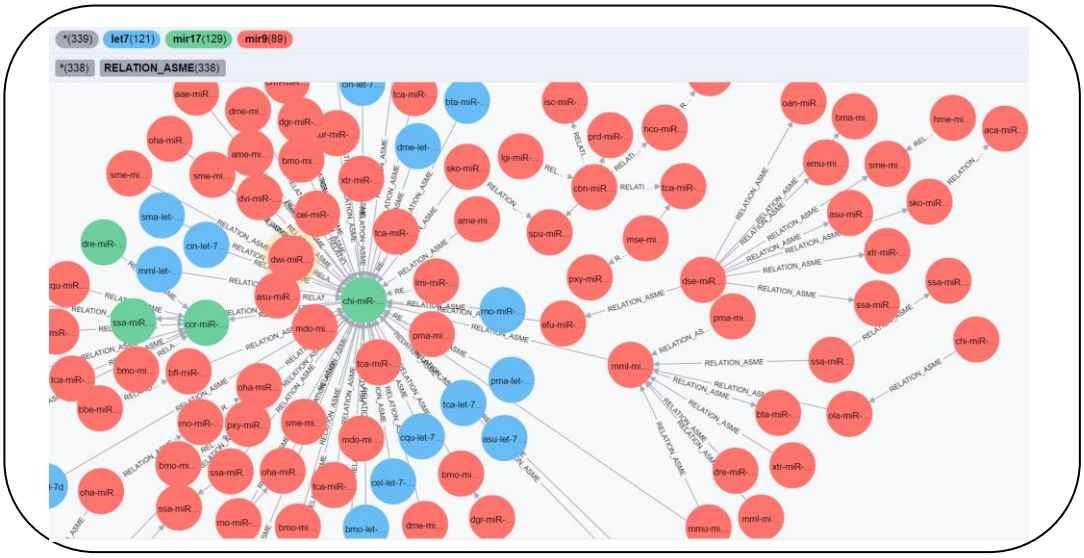

Figura 9.2 *Hubs* que contém a maioria das sequências pertencentes à família *mir-9*.

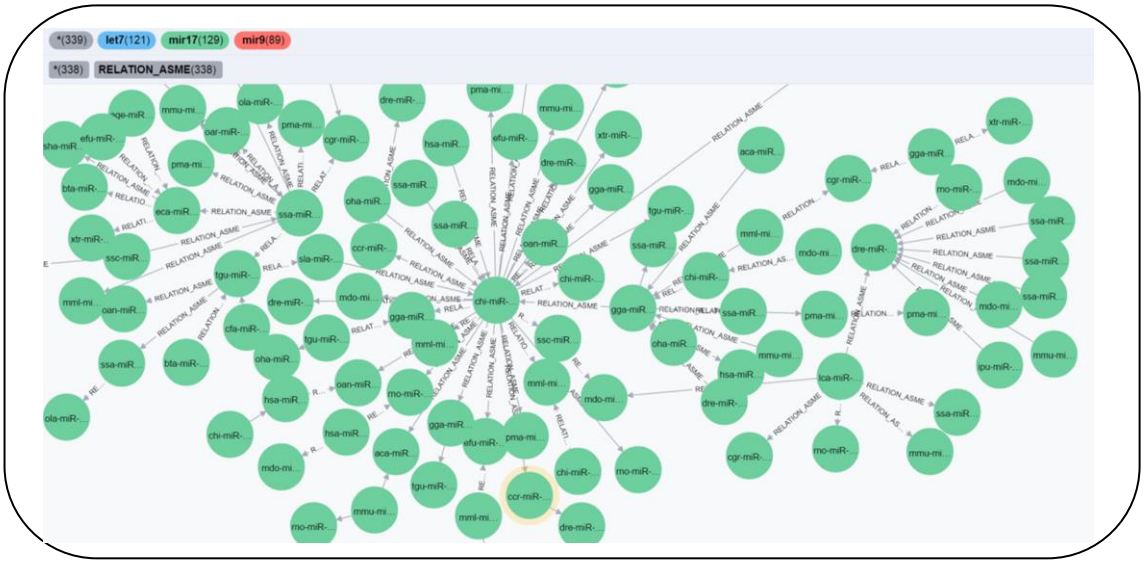

Figura 9.3 *Hubs* pertencentes à família *mir-17*.

Experimento SFASM-2

Neste experimento foram utilizados os valores de parâmetros mostrados na Tabela 9.10.

Tabela 9.10 Parâmetros utilizados em *SFASM-2*.

Algoritmo	Parâmetro	Valor
n-gram	N	5
Isomap	k	10
	Dimensão	150
	Distância	euclidiana
SFASM	número de links	3
	fator	0,95

As figuras 9.4, 9.5, 9.6 e 9.7 mostram o resultado do algoritmo SFASM. Não houve formação de *hubs* compostos por uma única família como no experimento *SFASM-1*. É importante observar também que os *hubs* formados neste experimento são maiores e, portanto, foram criados em menor número, quando comparados ao número de *hubs* criados em *SFASM-1*.

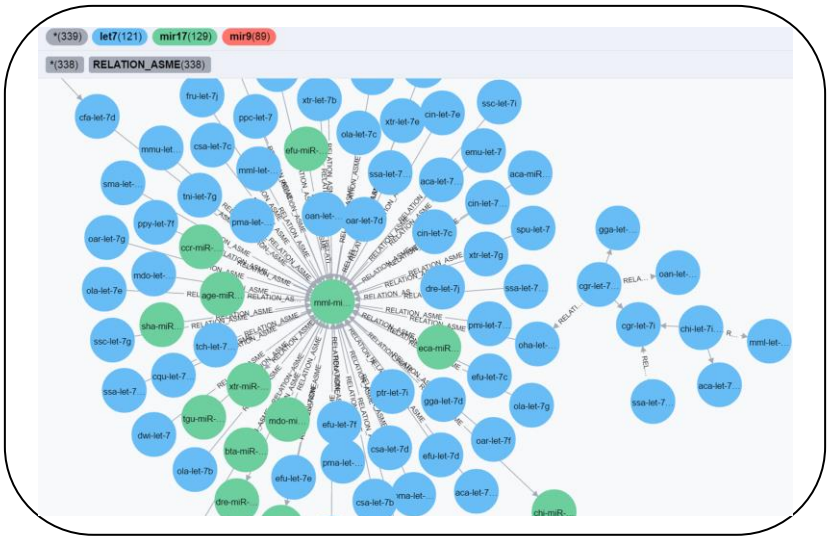

Figura 9.4 *Hubs* que contém a maioria das sequências pertencentes à família *let-7*.

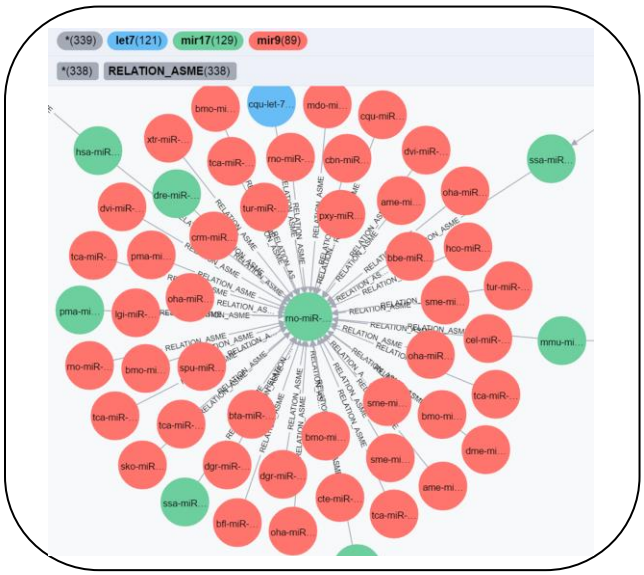

Figura 9.5 *Hub* que contém a maioria das sequências pertencentes à família *mir-9*.

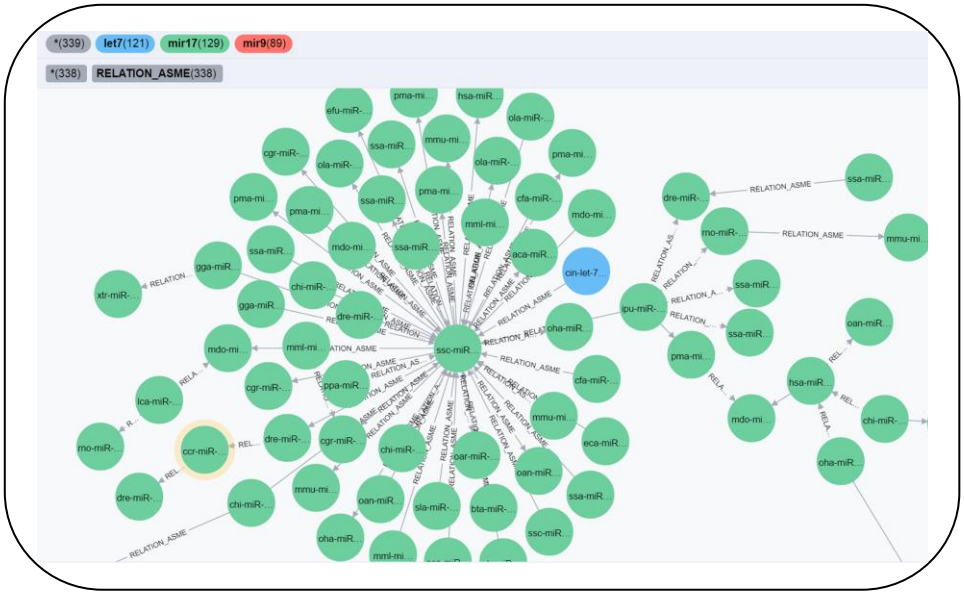

Figura 9.6 *Hubs* que contém a maioria das sequências pertencentes à família *mir-17*.

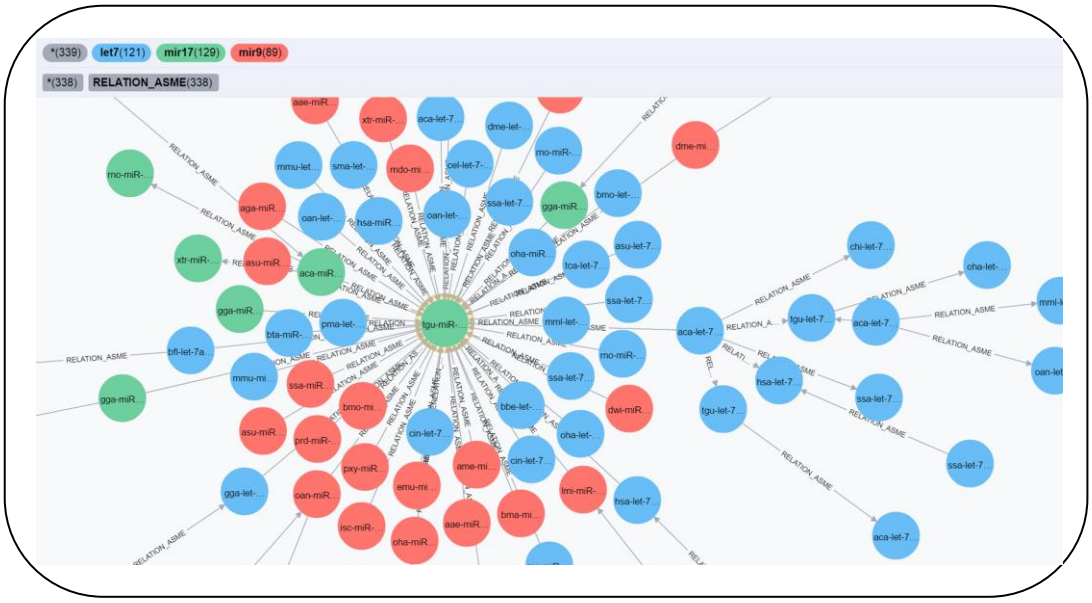

Figura 9.7 *Hubs* que contém sequências pertencentes às famílias *let-7*, *mir-17* e *mir-9*.

Experimento SFASM-5families

Os experimentos *SFASM-1* e *SFASM-2* mostraram resultados visuais interessantes e, devido a isso, o experimento *SFASM-5families* foi idealizado com o objetivo de verificar a dinâmica de formação de *hubs* com o aumento de sequências de miRNAs. A Tabela 9.11 mostra os valores dos parâmetros utilizados, em que o valor de fator foi determinado

pertencer ao intervalo [0,018, 0,3]. O algoritmo *Isomap* foi utilizado pois permitiu a formação de *hubs* mais concentrados. As figuras 9.8, 9.9, 9.10 e 9.11 mostram os *hubs* formados pelo algoritmo.

Tabela 9.11 Parâmetros utilizados em *SFASM-5families*.

Algoritmo	Parâmetro	Valor
n-gram	N	5
Isomap	k	10
	Dimensão	150
	Distância	euclidiana
SFASM	número de links	3
	fator	0,8

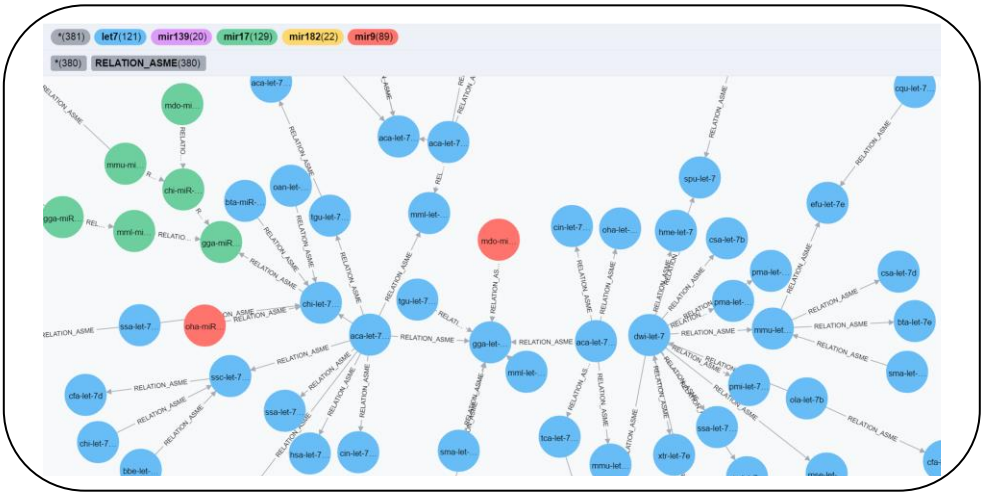

Figura 9.8 *Hubs* que contém a maioria das sequências pertencentes à família *let-7*.

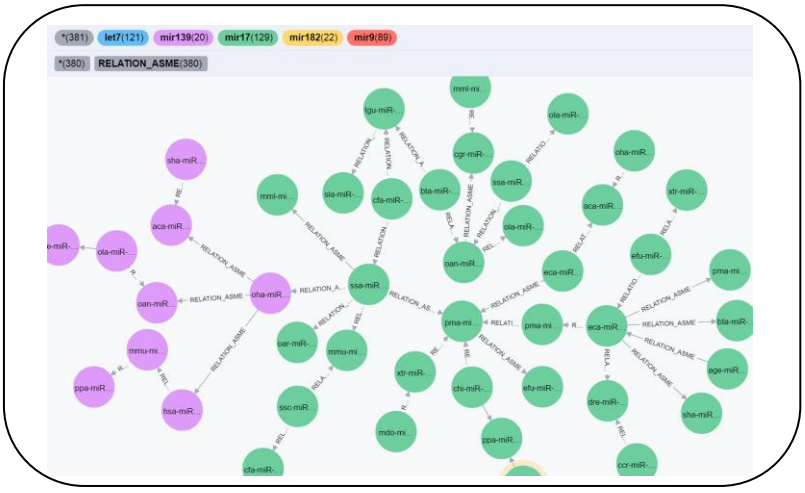

Figura 9.9 *Hubs* pertencentes às famílias *mir-17* e *mir-139*.

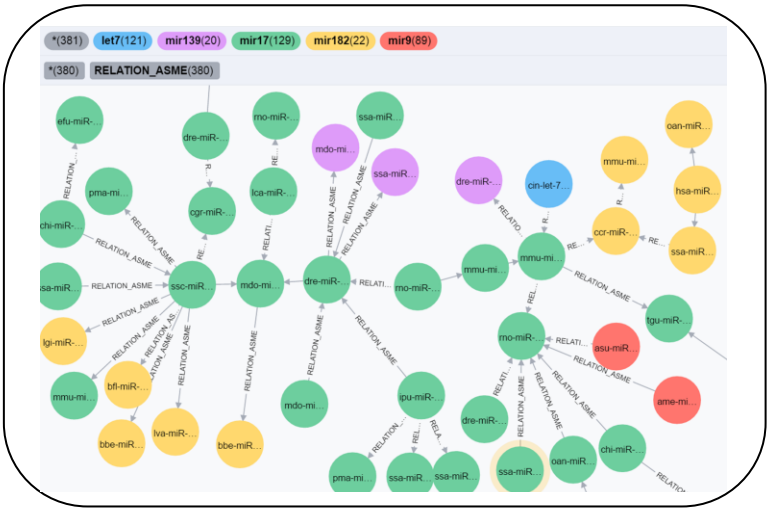

Figura 9.10 *Hubs* que contém a maioria das sequências pertencentes às famílias *mir-17* e *mir-182*.

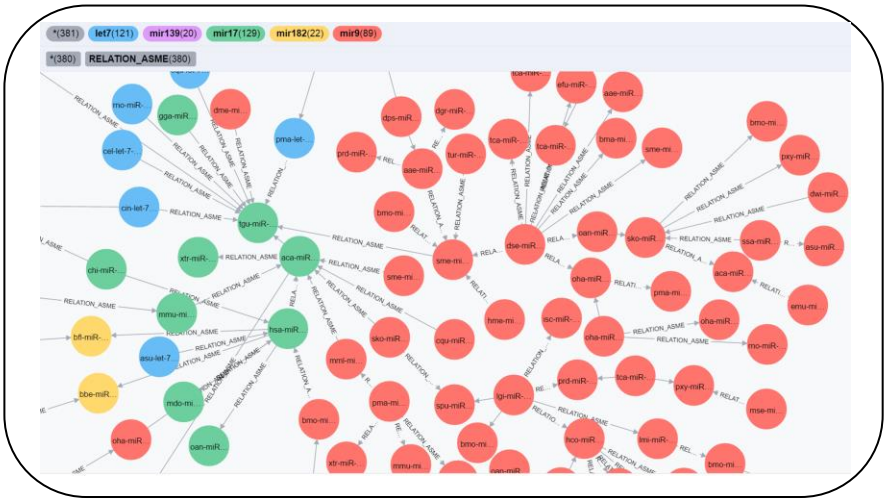

Figura 9.11 *Hubs* que contém a maioria das sequências pertencentes à família *mir-9*.

9.7 Experimentos Usando o Algoritmo Região de Influência, Resultados e Análises

Experimento RegiaoInfluencia-euclidiana

Neste experimento foi utilizado o algoritmo de Região de Influência para verificar e comparar com os outros experimentos realizados, o desempenho do algoritmo. Foram realizados diversos testes com o valor fator $\in$ [0,5, 2,5], com os seis tipos de região de influência descritos no Capítulo 4. Também foram realizados testes com distância euclidiana e distância *city-block*. Os valores dos parâmetros que obtiveram um melhor

resultado de agrupamento são mostrados na Tabela 9.12. Conforme descrito nos experimentos *ASME-euclidiana-4*, *ASME-euclidiana-5*, *ASME-city-block-4* e *ASME-city-block-5*, a redução de dimensão induz a uma perda de informação relevante e, portanto, não foi utilizada neste experimento. Nestes mesmos experimentos também é possível observar que o valor adequado para N de *n-grams* é 5.

O resultado do agrupamento é mostrado na Tabela 9.13 e os valores calculados dos três índices estão na Tabela 9.14. O passo 2.1 foi utilizado para melhorar o desempenho do algoritmo.

Tabela 9.12 Parâmetros utilizados para obter o melhor resultado para Região de Influência.

Algoritmo	Parâmetro	Valor
n-gram	N	5
Região de Influência	Distância região fator	euclidiana *GG-Mod* 1,3

Tabela 9.13 Melhor resultado obtido pela Região de Influência.

Grupo	*mir-9*	*let-7*	*mir-17*
G_1	43	0	0
G_2	16	0	0
G_3	30	21	0
G_4	0	43	0
G_5	0	46	0
G_6	0	11	0
G_7	0	0	16
G_8	0	0	15
G_9	0	0	16
G_{10}	0	0	21
G_{11}	0	0	17
G_{12}	0	0	44

Tabela 9.14 Resultado dos índices para os grupos da Tabela 9.13.

Índice	Valor
Davies-Bouldin	2,07698586
Dunn	0,44222133
C	0,30895528

A Tabela 8.49 mostra que o resultado do algoritmo evidencia sequências de miRNAs agrupadas incorretamente no G_3, e há um número elevado de grupos quando comparado ao número de famílias utilizadas no experimento. Todos os valores dos índices da Tabela 9.14

mostram que houve uma piora no resultado do agrupamento quando comparado com os valores dos mesmos índices nas tabelas 8.11, 8.14, 8.25, 8.28, 9.5 e 9.7. A Figura 9.12 mostra uma parte do G_3 da Tabela 9.13. É possível observar as diversas arestas pertencentes a uma única sequência e que se relacionam com várias outras sequências de miRNAs. O algoritmo forma uma aresta entre quaisquer duas sequências, se não houver outra sequência pertencente a região de influência, ao contrário do conceito utilizado na formação da ASM, onde há apenas uma única aresta entre uma sequência de miRNA e todas as outras sequências de miRNA. A Figura 9.13 mostra o G_2 e as arestas que mostram as relações entre as sequências *mir-9* da Tabela 9.13.

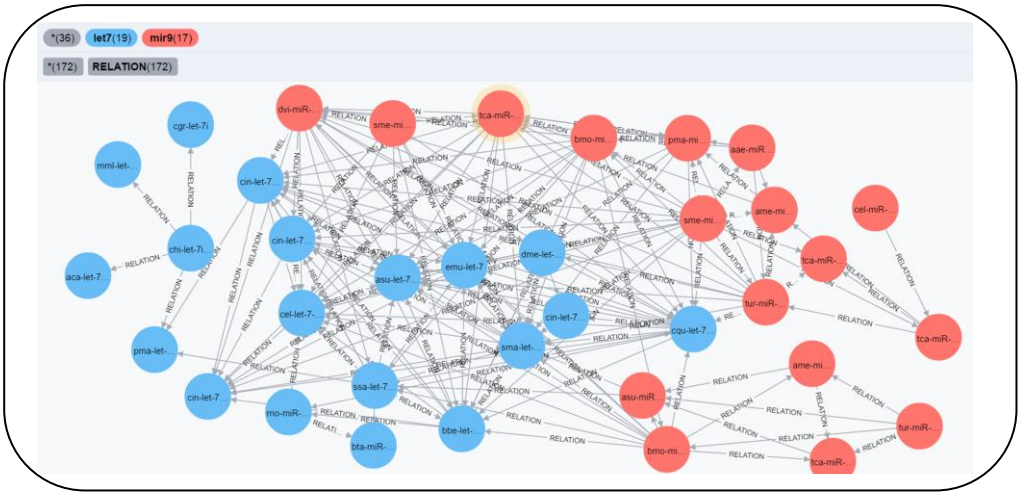

Figura 9.12 Grupo G_3 formado pelas famílias *let-7* e *mir-9*.

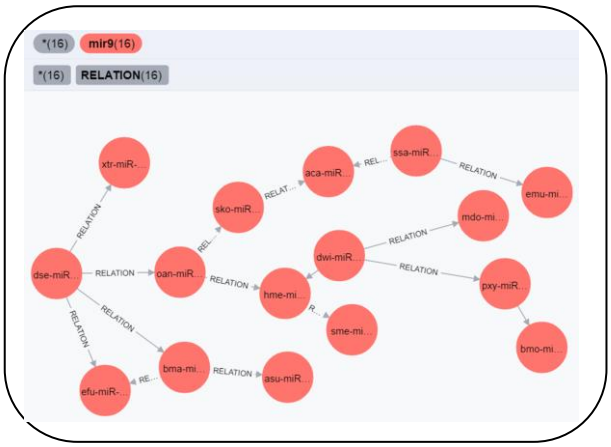

Figura 9.13 Grupo G_2 formado pela família *mir-9*.

Capítulo 10

Conclusão

Os experimentos realizados com diversos métodos mostraram que os algoritmos baseados em grafos são capazes de resolver problemas relativamente complexos, tal como aquele tratado nesta pesquisa *i.e.*, o agrupamento de miRNAs. Os principais objetivos dos experimentos eram obter resultados de agrupamentos cujos grupos não possuíssem sequência de miRNA agrupada incorretamente e, com o número de grupos mais próximo ou igual ao número de famílias utilizadas. Uma das principais dificuldades para atingir esses objetivos é relacionada à alta variabilidade de parâmetros utilizados, não só pelos algoritmos de agrupamentos mas, também, pelos algoritmos utilizados para preparação dos dados, tal como o *Isomap*. A utilização de parâmetros dentro de um intervalo pré-estabelecido também pode afetar negativamente os resultados, pois um valor fora desse intervalo pode conduzir à formação correta dos grupos. Os resultados também podem ser prejudicados com a possível combinação de valores entre os diversos parâmetros utilizados em um único experimento. Mesmo com a utilização de índices para validação dos resultados dos diferentes agrupamentos formados, a determinação dos valores parâmetros, para obtenção de grupos consistentes, não é um processo trivial.

O índice de *Davies-Bouldin* não foi eficiente para a decisão do melhor resultado do agrupamento, pois mostrou-se sensível ao número de grupos formados, ou seja, o valor do índice conduz à interpretação de que quanto maior for o número de grupos formados, mais compacto e portanto, melhor seria o resultado. Os índices *Dunn* e *C*, em alguns casos, mostraram-se eficientes na decisão do melhor resultado do agrupamento. A interpretação dos índices, também, pode ser prejudicada ao comparar experimentos que utilizam diferentes tipos de distâncias. Como há uma diferença entre os valores de distância euclidiana e *city-block*, suas comparações, por meio de valores de índices, pode beneficiar um resultado que não é necessariamente o melhor resultado. Isso pode ser confirmado nos experimentos *ASME-euclidiana-4* e *ASME-city-block-4* que possuem resultados iguais,

mostrados na Tabelas 8.10 e Tabela 8.24, respectivamente, mas valores de índices diferentes conforme, mostrados na Tabela 8.11 e Tabela 8.25.

Não é trivial decidir qual o melhor algoritmo de agrupamento, assim como quais devem ser os valores de seus parâmetros para atingir o melhor resultado. Por exemplo, na comparação entre os experimentos *ASME-Euclidiana-4* e *ASME-Euclidiana-5* ou *ASME-city-block-4* ou *ASME-city-block-5*, não é possível afirmar qual o melhor resultado. Entre todos os experimentos, o algoritmo ASM-PI mostrou ser o mais eficiente pois formou um agrupamento com o menor número de grupos e sem qualquer sequência agrupada incorretamente. Na maioria dos casos, com a utilização da distância euclidiana os algoritmos de agrupamentos obtiveram melhores resultados com exceção do experimento *ASME-5families*.

A visualização dos grupos por meio de grafos ajuda na compreensão das relações entre os dados permitindo uma melhor interpretação dos resultados, como no caso do grupo G_3 da Tabela 8.10 do experimento *ASME-Euclidiana-4*. Neste caso, apesar das famílias *let-7* e *mir-17* permanecerem no mesmo grupo, visualmente é possível verificar que há somente uma ponte que as mantém unidas. Também, os experimentos *SFASM-1*, *SFASM-2* e *SFASM-5families* mostraram uma visualização interessante dos dados em cada *hub* formado. Apesar do algoritmo de *Isomap* agravar os resultados nos outros experimentos, no caso particular do SFASM, o seu uso permitiu a formação de *hubs* mais concentrados, ao invés da formação de estrutura em árvore ou *hubs* dispersos. Esses diferentes tipos de visualizações dos dados podem ser explorados para uma melhor compreensão dos relacionamentos bem como, da natureza dos dados. Apesar do algoritmo de Região de Influência não apresentar um melhor resultado com relação ao número de grupos formados, sua visualização também pode ser interessante, já que mostra a relação de uma sequência de miRNA com todas as outras sequências e não somente com apenas uma outra sequência, como no caso da ASM.

O passo 2.1 apresentado na Figura 8.1 pode ser refinado para que o número de grupos formados seja igual ao número de famílias utilizado no experimento. Muitos algoritmos, tais como os de Região de Influência, *Ncut* e ASM-TXE podem ter seus resultados melhorados com essa modificação. Em [Yi & Guan 2012] é utilizado um método eficiente para a redução do número de grupos formados denominado *Multiple Sequence Alignment*.

Além disso, novas famílias devem ser acrescentadas ao experimentos, para verificar o impacto no resultado do algoritmo e, também, para uma melhor definição dos valores dos parâmetros e do algoritmo de agrupamento a serem utilizados. Para avaliação dos resultados dos agrupamentos é necessário utilizar, além dos índices de validação, uma interpretação visual e, também, estatística, como por exemplo a contagem da quantidade de dados agrupados corretamente e incorretamente em cada grupo.

Anexo A – Conceitos Básicos e Nomenclatura Padronizada

A.1 Considerações Iniciais

A proposta de trabalho de pesquisa em nível de mestrado descrita neste documento focaliza essencialmente algoritmos de agrupamento baseados em grafo. Com o objetivo de desenvolver um texto autocontido e compreensível, esse capítulo apresenta a conceituação básica e relevante relativa a Teoria dos Grafos, no que tange aos assuntos específicos cobertos por essa pesquisa. No que segue é apresentado o formalismo algébrico que substancia algoritmos de agrupamento baseados em grafo bem como a conceituação necessária ao entendimento dos algoritmos discutidos. Como comentado anteriormente, várias definições, particulares a determinados algoritmos que serão abordados em capítulos seguintes, são introduzidas e discutidas no respectivo capítulo/seção que trata de cada um deles. As definições apresentadas a seguir que envolvem conceitos relacionados à Teoria dos Grafos foram extraídas de [Nicoletti & Hrushka 2008] e as demais, extraídas de [Khan & Ahmad 2004] e [Luxburg 2007].

A.2 Definições Relevantes

Definição 1 (Grafo) Um grafo, notado por G=(V,E) consiste de dois conjuntos finitos:

V – que é um conjunto não vazio de elementos chamados vértices.

E – que é um conjunto (que pode ser vazio) de elementos chamados arestas.

A cada aresta $e \in E$ é atribuído um par não ordenado de vértices (u,v), chamados vértices-extremidade de e.

Definição 2 (Subgrafo) Sejam dois grafos, $G_1=(V_1,E_1)$ e $G_2=(V_2,E_2)$. Diz-se que G_2 é subgrafo de G_1 se $V_2 \subseteq V_1$ e $E_2 \subseteq E_1$ e para toda aresta $e \in E_2$, se e for incidente a v_1 e v_2, então $v_1, v_2 \in V_2$.

Definição 3 (Subgrafo *Spanning*) $G_1=(V_1,E_1)$ é um subgrafo *spanning* de $G=(V,E)$ se G_1 for um subgrafo de G tal que $V_1 = V$, ou seja, G_1 e G têm exatamente o mesmo conjunto de vértices.

Definição 4 (Grafo acíclico, Árvore e Floresta) Seja $G=(V,E)$ um grafo.

- G é *acíclico* se não contém ciclos;
- G é uma *árvore* se for um grafo acíclico conexo;
- G é uma *floresta* se for acíclico, independentemente de ser conexo ou não.

Definição 5 (Grafo Ponderado) Seja $G=(V,E)$ um grafo com $|E| = m$. G é chamado *grafo ponderado* se cada aresta $e \in E$ tiver um número real associado a ela, $w(e)$ chamado peso de e. A soma dos pesos de todas as arestas do grafo, $w(e_1) + w(e_2) + ... + w(e_m)$ é o peso de G, notado por $w(G)$.

Definição 6 (Árvore *Spanning*) Seja $G=(V,E)$ um grafo. Uma árvore *spanning* do grafo G é um subgrafo *spanning* de G, que é uma árvore.

Definição 7 (Árvore Spanning Minimal) Seja $G=(V,E)$ um grafo ponderado. Uma árvore *spanning* minimal do grafo G é um subgrafo *spanning* de G, que é uma árvore cujos pesos das arestas $e_i \in E$ são mínimos.

Definição 8 (Grafo Completo) Seja $G=(V,E)$ um grafo tal que $|V| = n$. G é um *grafo completo de ordem n*, notado por K_n, se G tiver exatamente uma aresta conectando cada um dos possíveis pares de vértices distintos.

Definição 9 (Componente Conexo) Seja $G=(V,E)$ um grafo. Dado qualquer vértice $u \in V$, seja C(u) o conjunto de todos os vértices de G que estão conectados a u. O subgrafo de G induzido por C(u) é chamado componente conexo contendo u ou, simplesmente, componente conexo.

Definição 10 (Caminho) Um *caminho* em um grafo G=(V,E) é uma sequência de vértices $v_{1,}...v_n$ tal que (v_i, v_{i+1}) é um elemento de E, para $1 \leq i < n - 1$. Essa sequência é dita um caminho de v_1 a v_n, onde nenhum vértice pode aparecer mais de uma vez, com a possível exceção de que v_1 e v_n podem ser o mesmo vértice.

Definição 11 (Ponte) Seja G = (V,E) um grafo. Uma aresta e de G é chamada *ponte* (ou *aresta de corte*) se o subgrafo G – e tiver mais componentes conexos que G.

Seja CD um conjunto de dados. O *grafo de vizinhança* (GV = (V,E)) de CD é um grafo cujos vértices são os dados de CD e a existência de uma aresta entre dois vértices indica uma relação de vizinhança entre os dados que tais vértices representam (ver Definição 12 para a definição formal do conceito). A construção de GV a partir de CD é feita adicionando arestas entre dois vértices d_i e $d_j \in V$ caso não haja nenhum outro vértice de GV dentro da região de influência definida pelo par de vértices $\{d_i ,d_j\}$. Via de regra a definição de uma *região de vizinhança* é feita por meio do estabelecimento de uma equação matemática que deve ser satisfeita pelo par de vértices ao qual ela é aplicada. A região de vizinhança determinada por um par de vértices, dependendo da equação matemática, representa uma figura geométrica.

Definição 12 (Grafo de Vizinhança) Seja V um conjunto de vértices que representam pontos de dados em R^2. Cada par não ordenado de vértices, $(d_i, d_j) \in V \times V$, $d_i \neq d_j$, é associado com a região vizinhança $U_{d_i,d_j} \subseteq R^2$. Seja P a propriedade definida em $U = \{U_{d_i,d_j} \mid (d_i,d_j) \in V \times V\}$. Um *grafo de vizinhança* $G_{U,P} = (V,E)$, definido pela propriedade P, é um grafo com o conjunto de vértices V e com o conjunto de arestas E tal que $(d_i,d_j) \in E$ se e somente se U_{d_i,d_j} possui a propriedade P.

Definição 13 (Centro do grupo) Considere um agrupamento $AG = \{G_1, G_2, ..., G_k\}$, composto por k grupos G; cada grupo G é formado por z dados d_i. O centro de um dado grupo G é definido por $(\Sigma_{i=1...z} d_i) / z$, ou seja, o centro de um grupo é a média aritmética das coordenadas de seus dados localizados no espaço n-dimensional.

Como exemplo, considere um grupo G formado pelos dados $d_1 = (4, 9)$ e $d_2 = (2, 1)$ no espaço bi-dimensional. O centro c do grupo G é $c = (4 + 2)/2, (9 + 1)/2 = (3, 5)$.

Definição 14 (Transformação linear) Uma transformação linear de dois espaços vetoriais V e W é um mapeamento $T: V \rightarrow W$ que segue duas propriedades:

- $T(v_1 + v_2) = T(v_1) + T(v_2)$, com v_1 e $v_2 \in V$.
- $T(\alpha v) = \alpha T(v)$, para qualquer valor escalar α e $v \in V$.

Definição 15 (Autovalor e Autovetor) Considere uma matriz A com dimensões $n \times n$ e a seguinte transformação linear $T(v) = \lambda(v)$, $\lambda \in R$ (Definição A.14). Todo vetor não-nulo v que satisfaz essa função definida pela transformação linear é denominado de autovetor de T correspondente ao autovalor λ. Ou seja, λ é um autovalor ou valor característico da matriz A, onde para um dado valor de λ temos um $v \neq 0$ como uma solução do sistema. As correspondentes soluções $v \neq 0$ são os autovetores ou vetores característicos associados ao autovalor λ.

Considere os seguintes elementos $a_{ij} \in A$.

$$\begin{bmatrix} a_{11}v_1 + & \cdots & +a_{1n}v_n = \lambda v_1 \\ a_{21}v_1 + & \ldots & +a_{2n}v_n = \lambda v_2 \\ \vdots & \ddots & \vdots \\ a_{n1}v_1 + & \cdots & +a_{nn}v_n = \lambda v_n \end{bmatrix}$$ equivalente a

$$\begin{bmatrix} (a_{11} - \lambda)v_1 + a_{12}v_2 + & \cdots & +a_{1n}v_n = 0 \\ a_{21}v_1 + (a_{22} - \lambda)v_2 + & \ldots & +a_{2n}v_n = 0 \\ \vdots & \ddots & \vdots \\ a_{n1}v_1 + a_{n2}v_2 + & \cdots & +(a_{nn} - \lambda)v_n = 0 \end{bmatrix}$$

Em notação matricial $(A - \lambda I)v = 0$, onde I é a matriz identidade ou elemento neutro do produto das matrizes.

Os problemas que envolvem a determinação de autovalores de uma matriz A são chamados de problemas de autovalores. Ou seja, a procura por um vetor solução v tal que $v_i \neq 0$ para pelo menos algum i (solução não-trivial). Para que isso seja possível, o determinante da matriz de coeficientes é zero. A notação matricial é apresentada na Equação (A.1).

$$DC(\lambda) = \det(A - \lambda I) = 0 = \begin{bmatrix} a_{11} - \lambda & a_{12} & \cdots & a_{1n} \\ a_{11} & a_{22} - \lambda & \cdots & a_{2n} \\ a_{n1} & a_{n2} & \cdots & a_{nn} - \lambda \end{bmatrix} = 0 \quad \text{(A.1)}$$

DC(λ) é denominado *determinante característico* e a Equação (A.2) é a equação característica da matriz A. A solução desta equação polinomial de grau n em função de λ é o polinômio característico de A.

$$(a_{11} - \lambda) \times (a_{22} - \lambda) \times \ldots (a_{nn} - \lambda) - (a_{1n} \times \ldots (a_{22} - \lambda) \times a_{n1}) = 0 \quad \text{(A.2)}$$

Segue um exemplo para a determinação de autovalores e autovetores de uma matriz. Considere a matriz $A = \begin{bmatrix} -5 & 2 \\ 2 & -2 \end{bmatrix}$. Como DC(λ) = det(A – λI) = 0 então tem-se que $\begin{bmatrix} -5 & 2 \\ 2 & -2 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = 0$ e, portanto, $\begin{bmatrix} -5 - \lambda & 2 \\ 2 & -2 - \lambda \end{bmatrix} = 0$. O polinômio característico da matriz e sua solução é calculada da seguinte maneira:

$$(-5 - \lambda)(-2 - \lambda) - 4 = 0$$
$$\lambda^2 + 7\lambda + 6 = 0$$
$$\lambda_1 = -1 \text{ e } \lambda_2 = -6$$

λ_1e λ_2 são os autovalores da matriz A. Para a determinação dos autovetores associados a λ_1e λ_2, o conceito de autoespaço deve ser considerado, como apresentado na Definição 16.

Definição 16 (Autoespaço) Para todo autovalor λ de uma matriz A, existe um autoespaço associado a λ. O autoespaço é o conjunto de todos os vetores obtidos pela combinação linear dos autovetores v associados a λ, ou seja, é o espaço nulo da matriz resultante de A – λI. O conjunto que define o autoespaço é definido pela Equação (A.3).

$$E_\lambda = \{v \in V | Av = \lambda v\} \quad \text{(A.3)}$$

Dada a equação (A – λI)v = 0, para determinar os autovetores associados aos autovetores λ_1e λ_2 determina-se:

a) Para $\lambda_1 = -1$:

$$\begin{bmatrix} -5-(-1) & 2 \\ 2 & -2-(-1) \end{bmatrix} \begin{bmatrix} v_i \\ v_j \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} -4 & 2 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} v_i \\ v_j \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} -2 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_i \\ v_j \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

tem-se que $v_j = 2v_i$, portanto $E_{\lambda=-1} = \left\{ \begin{bmatrix} v_i \\ v_j \end{bmatrix} = t \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \forall\, t \in R \right\}$.

b) Para $\lambda_2 = -6$:

$$\begin{bmatrix} -5-(-6) & 2 \\ 2 & -2-(-6) \end{bmatrix} \begin{bmatrix} v_i \\ v_j \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} v_i \\ v_j \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_i \\ v_j \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

tem-se que $v_j = -0{,}5v_i$, portanto o $E_{\lambda=-6} = \left\{ \begin{bmatrix} v_i \\ v_j \end{bmatrix} = t \begin{bmatrix} 2 \\ -1 \end{bmatrix}, \forall\, t \in R \right\}$.

Definição 17 (Espectro de um Grafo) Considere um grafo G = (V, E) e sua matriz de adjacência A. O *espectro* de um grafo G é o espectro de sua matriz de adjacência A. Ou seja, o espectro de A é definido pelo conjunto de autovalores (Definição 15) determinados por A.

Definição 18 (Matriz Laplaciana Não-Normalizada) Considere um grafo G = (V, E), sua matriz de adjacência A com os graus dos vértices ou pesos das arestas, e, uma matriz M cuja diagonal representa a soma dos pesos das arestas de G. A matriz laplaciana não-normalizada L é definida pela Equação (A.4) em que cada elemento da matriz de L, L_{ij} é definido pela Equação (A.5).

$$L = M - A \qquad \text{(A.4)}$$

$$L_{ij} = \begin{cases} -A_{ij}, \text{se } i \neq j \\ M_{ij}, \text{se } i = j \end{cases} \quad (A.5)$$

Definição 19 (Matriz Laplaciana Simétrica Normalizada) Seja um grafo G = (V, E), sua matriz de adjacência A com os graus dos vértices ou pesos das arestas, e, uma matriz M cuja diagonal representa a soma dos pesos das arestas de G. A definição de uma matriz laplaciana simétrica normalizada é dada pela Equação (A.6), em que I é a matriz de identidade. A matriz L_{sym} é simétrica em relação a diagonal principal e a antidiagonal.

$$L_{sym} = M^{-\frac{1}{2}} L M^{-\frac{1}{2}} = I - M^{-\frac{1}{2}} A M^{-\frac{1}{2}} \quad (A.6)$$

Definição 20 (Corte) Considere um grafo G = (V, E) com matriz de adjacência A que contém os pesos das arestas e $\in$ E. Dados dois vértices $v_1 \in V$ e $v_2 \in V$, o elemento $w(v_1, v_2)$ de A é a medida de similaridade entre os vértices v_1 e v_2. O grafo G pode ser particionado, por meio da remoção de suas arestas, em dois conjuntos distintos $G_1 = (V_1, E_1)$ e $G_2 = (V_2, E_2)$ tal que $V_1 \cup V_2 = V$ e $V_1 \cap V_2 = \varnothing$. O custo do corte (*cut*) da remoção das arestas, ou grau de dissimilaridade, é o custo total dos pesos das arestas removidas, apresentada na Equação (A.7).

$$cut(G_1, G_2) = \sum_{v_1 \in V_1, v_2 \in V_2} w(v_1, v_2) \quad (A.7)$$

Bibliografia e Referências

[Ambros 1989] Ambros, V. (1989) A hierarchy of regulatory genes controls a larva-to-adult developmental switch in *C. elegans*, *Cell*, v. 57, pp. 49–57.

[Ambros & Horvitz 1987] Ambros, V.; Horvitz, H. R. (1987) The lin-14 locus of *Caenorhabditis elegans* controls the time of expression of specific postembryonic developmental events, *Genes Dev.*, v. 1, pp. 398–414.

[Andrew *et al.* 2002] Andrew, Y. N.; Jordan, I. M.; Weiss, Y. (2001) On spectral clustering: analysis and algorithm, In: Proc. of 14th Advances in Neural Information Processing Systems. pp. 849-856.

[Asano *et a.l* 1988] Asano, T.; Bhattacharia, B.; Keil, M.; Yao, F. (1998) Clustering algorithms based on minimum and maximum spanning trees, In: Proc. of the Fourth Annual Symposium on Computational Geometry, Urbana-Champaign, Illinois, pp. 252-257.

[Lichman 2013] Lichman, M. (2013) UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[Barabási & Albert 1999] Barabási, A. L.; Albert, R. (1999) Emergence of scaling in random networks, Science, v. 286 , pp. 509-512.

[Cheung 2008] Cheung, L. M. (2008) SimAffling - um ambiente computacional para suporte e simulação do processo de DNA *shuffling*, Tese de doutorado, Programa de Pós-Gradução em Biotecnologia, UFSCar, S. Carlos.

[Cortes & Vapnik 1995] Cortes, C.*;* Vapnik, V. (1995) Support-vector networks, *Machine Learning*, v.20, no. 3, pp. 273-297.

[Davies & Bouldin 1979] Davies, D. L.; Bouldin, D. W. (1979) A cluster separation measure, In: IEEE Trans. Pattern Analysis and Machine Intelligence, v. 1, pp. 224-227.

[Deerwester *et al.* 1990] Deerwester, S.; Dumais, T. S.; Funas, W. G.; Landauer, K. T.; Harshman, R. (1990) Indexing by latent semantic analysis, Journal of the American Soc. For Information Science, v. 41, pp. 391-407.

[Ding *et al.* 2011] Ding, J.; Zhou, S.; Guan, J. (2011) Mirfam: an effective automatic MiRNA classification method based on n-grams and a multiclass SVM, *BMC Bioinformatics*, v. 12, pp. 216.

[Dunn 1973] Dunn, J. C. (1973) A fuzzy relative of the isodata process and its use in detecting compact well separated clusters, In: Journal of Cybernetics, v. 3, pp. 32-57.

[Edla *et al.* 2012] Edla, D. R.; Machavarapu, S.; Jana, P. K. (2012) An improved MST-based clustering for biological data, In: International Conference on Data Science & Engineering, pp. 42-47.

[Eddy 2001] Eddy, S. R. (2001) Non-coding RNA genes and the modern RNA world, *Nat. Rev. Genet.*, v. 2, no. 12, pp. 919-929.

[Esther *et al.* 1996] Esther, M.; Kriegel, H.; Xu, X. (1996) A density-based algorithm for discovering clusters in a large spatial database with noise, In: Proc. of the Second International Conference on Knowledge Discovery and Data Mining, pp. 226-231.

[Fiduccia & Mattheyses 1982] Fiduccia, C. M.; Mattheyses, R. M. (1982). A linear-time heuristic for improving network partitions, In: Proc. Design Automation Conference, pp. 175-181.

[Flake *et al.* 2004] Flake, G. W.; Tarjan, R. E.; Tsioutsiouliklis, K. (2004) Graph clustering and minimum cut trees, Internet Mathematics, v. 1, no. 4, pp. 385-408.

[Forgy 1965] Forgy, W. (1965) Cluster analysis of multivariate data: efficiency versus interpretability of classifications, *Biometrics*, v. 21, pp. 768-769.

[Fruigui & Nasraoui 2004] Fruigui, H.; Nasraoui, O. (2004) Simultaneous clustering and dynamic keyword weighting for text documents, in Survey of Text Mining, M. Berry (Ed.), Heidelberg: Springer-Verlag, 00. 45-70.

[Gowda & Diday 1992] Gowda, K. C.; Diday, E. (1992) Symbolic Clustering using a new similarity measure, IEEE Transactions on Systems, Man, Cybernetics, v. 22, no. 2, pp. 368-378.

[Grygorash *et al.* 2006] Grygorash, O.; Zhou, Y.; Jorgensen, Z. (2006) Minimum spanning tree based algorithms, In: Proc. IEEE International Conference on Tools with Artificial Intelligence, pp. 73-81.

[Griffiths-Jones *et al.* 2006] Griffiths-Jones, S.; Grocock, R. J.; van Dongen, S.; Bateman, A.; Enright, A. J. (2006) miRBase: microRNA sequences, targets and gene nomenclature, *Nucleic Acids Research*, v. 44, pp. D140-D144.

[Grira *et al.* 2005] Grira, N.; Crucianu, M.; Boujemaa, N. (2005) Unsupervised and semi-supervised clustering: a brief survey, In: Proc. of 7th ACM SIGMM International Workshop on Multimedia Information Retrieval, pp. 9-16.

[Gunter & Bunke 2003] Gunter, S.; Bunke, H. (2003) Validation indices for graph clustering, Pattern Recognition, v. 24, pp. 1107-1113.

[Hartuv & Shamir 1999] Hartuv, E.; Shamir, R. (1999) A clustering algorithm based on graph connectivity, Information Processing Letters, v. 76, no. 4-6, pp 175-181.

[Halkidi *et al.* 2002] Halkidi, M.; Batistakis Y.; Vazirgiannis, M. (2002) Clustering validity checking methods – Part II, ACM Sigmod Record, v. 31, pp. 19-27.

[Hubert & Schultz 1976] Hubert, L; Schultz, J. (1976) Quadratic assignment as a general data analysis strategy, British Journal of Math. and Statist. Psychol., v. 29, pp. 190-241.

[Hurtvagner and Zamore 2002] Hurtvagner, G.; Zamore, P. D. (2002) A microRNA in a multiple-turnover RNAi enzyme complex, *Science*, v. 297, no. 5589, pp. 2056-2060.

[Jain 2010] Jain, A. K. (2010) Data clustering 50 years beyond k-means, Pattern Recognition, v. 31, pp. 651-666.

[Jain *et al.* 1999] Jain, A. K.; Murty, M. N.; Flynn, P. J. (1999) Data clustering: a review. ACM Computing Surveys, vol. 31, no.3, pp. 264-323.

[Kasahara & Nicoletti 2009] Kasahara, V. A.; Nicoletti, M. C. (2009) Investigating neighborhood graphs for inducing density based clusters, Part I, Chapter 3 of Foundations of Computational Intelligence, A. Abraham, A.-E. Hassanien, V. Snásel (Eds.), Spring-Verlag, pp. 57-78.

[Kawaji 2001] Kawaji, H.; Yamaguchi, Y.; Matsuda, H.; Hashimoto, A. (2001) A graph-based clustering method for a large set of sequences using a graph partitioning algorithm, Genome Informatics, pp. 93-102.

[Khan & Ahmad 2004] Khan, S. S.; Ahmad, A. (2004) Cluster center initialization algorithm for k-means clustering (2004), Pattern Recognition, v. 25, pp. 1293- 1302.

[Kim 2005] Kim, V. N. (2005) MicroRNA biogenesis: coordinated cropping and dicing, Nature Review Molecular Cell Biology, v. 6, pp. 376-385.

[King 1967] King, B. (1967) Step-wise clustering procedures, Journal of the American Statistical Association, v. 62, no. 317, pp. 86-101.

[Kirkpatrick 1985] Kirkpatrick, D. G.; Radke, J. D. (1985) A framework for computational morphology, In: Computational Geometry, Toussaint G. (ed.), North-Holland, pp. 217-248.

[Kleinberg 2002] Kleinberg, J. (2002) An impossibility theorem for clustering, MIT Press, Cambridge, USA.

[Kruskal 1965] Kruskal, J. B. (1965) On the shortest spanning subtree of a graph and the traveling salesman problem, In: Proc. American Math. Soc., v. 7, pp. 48-50.

[Lee *et al.* 1993] Lee, R. C.; Feinbaum, R. L.; Ambros, V. (1993) The *C. elegans* heterochronic gene lin-4 encodes small RNAs with antisense complementarity to lin-14, *Cell*, v. 75, pp. 843-854.

[Luxburg 2007] Luxburg, U. (2007) A tutorial on spectral clustering, Journal Statistics and Computing, v. 17, pp. 395-416.

[MacQueen 1967] MacQueen, J. (1967) Some methods for classification and analysis of multivariate observations, In: Proc. of the 5th Berkeley Symp. on Math. Statist. and Prob., v. 1, pp. 281-297.

[Mattick & Makunin 2006] Mattick, J. S.; Makunin, I. V. (2006) Non-coding RNA, *Human Molecular Genetics*, v. 15, R 17-29.

[Meila 2002] Meila, M. (2002) Comparing clusterings – an axiomatic view, In: Proc. 22$^{nd.}$ International Conference on Machine Learning, Bonn, Germany, pp. 577-584.

[Nicoletti & Hruschka 2005] Nicoletti, M.C.; Hruschka, E.R. (2005) Fundamentos da teoria dos grafos para a computação, EdUFSCar.

[Ostergard 2002] Ostergard, P. R. J. (2002) A fast algorithm for the maximum clique problem, Discrete Applied Mathematics, v. 120, pp. 197-207.

[Päivinen 2005] Päivinen, N. (2005) Clustering with minimum spanning tree of scale-free structure, Pattern Recognition, v. 26, pp. 921-930.

[Pajek & Vladimir 2008] Pajek, A. M.; Vladimir, B. (2008) Program for large network analysis [http://vlado.fmf.uni-lj.si/pub/networks/pajek/].

[Pakhira 2005] Pakhira, M. K.; Bandyopadhyay, S.; Maulik U. (2005) A study of some fuzzy cluster validity indices, genetic clustering and application to pixel classification, Journal Fuzzy Sets and Systems, v. 155 pp. 191-214.

[Pasquinelli *et al.* 2000] Pasquinelli, A. E.; Reinhart, B. J.; Slack, F.; Martindale, M. Q.; Kuroda, M. I. (2000) Conservation of the sequence and temporal expression of let-7 heterochronic regulatory RNA, *Nature*, v. 408, pp. 86–89.

[Prim 1957] Prim, R. C. (1957) Shortest connection networks and some generalizations. Bell System Tech, J., v. 36, pp. 1389-1401.

[Rendón *et al.* 2011] Rendón, E.; Abundez, I.; Arizmendi, A.; Quiroz, E. M. (2011) Internal versus external cluster validation indexes, International Journal of Computers and Communications, v. 5, pp. 27-34.

[Roweis & Saul 2000] Roweis, T. S.; Saul, K. L. (2000) Nonlinear dimensionality reduction by locally linear embedding, Science, v. 290, pp. 2323-2326.

[Schaeffer 2007] Schaeffer, S. E. (2007) Graph clustering, Computer Science Review, v. 1, pp. 27-64.

[Sharan & Shamir 2000] Sharan, R.; Shamir, R. (2000) CLICK: a clustering algorithm with applications to gene expression analysis, In: Proc. of 8th International Conference on Intelligent Systems for Molecular Biology (ISMB), AAAI Press, pp. 307-316.

[Shi & Malik 2000] Shi, J.; Malik J. (2000) Normalized cuts and image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 22, no. 8, pp. 888–905.

[Shin & Park 2006] Park, J. C.; Shin, H.; Choi, B. K. (2006) Elliptic Gabriel graph for finding neighbors in a point set and its application to normal vector estimation, Computer-Aided Design 38, pp. 619-626.

[Sievers *et al.* 2011] Sievers, F.; Wilm, A.; Dineen, D.; Gibson, T.; Karplus, K.; Li, W.; Lopez, R.; McWilliam, H.; Remmert, M.; Soding, J.; Thompson, J.; Higgins, D. (2011) Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega, Molecular Systems Biology, vol. 7, no 539.

[Sneath & Sokal 1973] Sneath, P. H.; Sokal, R. R. (1973) Numerical Taxonomy, W H Freeman & Co.

[Stoer & Wagner 1997] Stoer, M.; Wagner, F. (1997) A simple min-cut algorithm, Journal of the ACM, v. 44, no. 4, pp. 585-591.

[Tanenbaum *et al.* 2000] Tanenbaum, B. J.; Silva, V.; Langford, C. J. (2000) A global geometric framework for nonlinear dimensionality reduction, v. 290, pp. 2319-2323.

[Theodoridis & Koutroumbas 1998] Theodoridis, S.; Koutroumbas, K. (1998) Pattern Recognition, Academic Press.

[Toussaint 1980] Toussaint, G. T. (1980) The relative neighborhood graph of a finite planar set, Pattern Recognition, v. 12, pp. 261-268.

[Toussaint 1988] Toussaint, G. T. (1988) A graph-theoretical primal sketch. In: Computational Morphology, G. T. Toussaint (Ed.), North-Holland, pp. 229-260.

[Urquhart 1982] Urquhart, R. (1982) Graph theoretical clustering based on limited neighborhood sets, Pattern Recognition, v. 15, no. 3, pp. 173-187.

[Vendramin et al. 2010] Vendramin, L.; Campello, R. J. G. B.; Hruschka, E. R. (2010). Relative clustering validity criteria: A comparative overview. Statistical Analysis and Data Mining, v. 3, pp. 209-235.

[Wallace 1989] Wallace, R. S. (1989) Finding natural clusters through entropy minimization, Ph. D. Thesis, CMU-CS-89-183, 144 pgs.

[Wan *et al.* 2012] Wan, L.; Ding, J.; Jin, T.; Guan, J.; Zhou, S. (2012) Automatically clustering large-scale miRNA sequences: methods and experiments, In: Journal BMC Genomics, v. 13.

[Weskamp *et al.* 2004] Weskamp, N.; Kuhn, D., Hullermeier, E.; Klebe, G. (2004) Efficient similarity search in protein structure databases by k-clique hashing, Bioinformatics, v. 20, no. 10, pp. 1522-1526.

[Williams 2008] Williams, A. E. (2008) Functional aspects of animal mircroRNAs, *Cellular and Molecular Life Sciences*, v. 65, pp. 545-562.

[Yi & Guan 2012] Yi, Y.; Guan, J. (2012) Effective clustering of microRNA sequences by n-grams and feature weighting, In: Proc. of the 6th IEEE International Conference on Systems Biology, pp. 203-210.

[Zahn 1971] Zahn, C. T. (1971) Graph-theoretical methods for detecting and describing gestalt clusters, IEEE Transactions on Computers, v. 20, no. 1, pp. 68-86.

[Zahn 1996] Zahn, T.; Ramakrishnan, R.; Livny, M. (1996) BIRCH: An efficient data clustering method for very large databases, In: Proc. of the 1996 ACM SIGMOD International Conference on Management of Data, pp. 103-114.