MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21c/2019/08.14.01.42-TDI

REDE PROTÓTIPO: UM ALGORITMO PARA AGRUPAMENTO DE FLUXO DE DADOS BASEADO EM REDES COMPLEXAS

Sandy Moreira Porto

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelo Dr. Marcos Gonçalves Quiles, aprovada em 21 de agosto de 2019.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34R/3TR9CEL>

INPE
São José dos Campos
2019

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GBDIR)
Serviço de Informação e Documentação (SESID)
CEP 12.227-010
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/7348
E-mail: pubtc@inpe.br

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21c/2019/08.14.01.42-TDI

REDE PROTÓTIPO: UM ALGORITMO PARA AGRUPAMENTO DE FLUXO DE DADOS BASEADO EM REDES COMPLEXAS

Sandy Moreira Porto

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelo Dr. Marcos Gonçalves Quiles, aprovada em 21 de agosto de 2019.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34R/3TR9CEL>

INPE
São José dos Campos
2019

Dados Internacionais de Catalogação na Publicação (CIP)

Porto, Sandy Moreira.

P838r Rede Protótipo: um algoritmo para agrupamento de fluxo de dados baseado em redes complexas / Sandy Moreira Porto. – São José dos Campos : INPE, 2019.

xxii + 135 p. ; (sid.inpe.br/mtc-m21c/2019/08.14.01.42-TDI)

Tese (Doutorado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2019.

Orientador : Dr. Marcos Gonçalves Quiles.

1. Fluxo de dados. 2. Redes complexas. 3. Agrupamento. 4. MODIS. 5. Observação da Terra. I.Título.

CDU 004.272.33:519.254

Aluno (a): ***Sandy Moreira Porto***

Título: **"REDE PROTÓTIPO: UM ALGORITMO PARA AGRUPAMENTO DE FLUXO DE DADOS BASEADO EM REDES COMPLEXAS"**

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de ***Doutor(a)*** em

Computação Aplicada

Dr. **Nandamudi Lankalapalli Vijaykumar**

Presidente / INPE / SJCampos - SP

( ) Participação por Video - Conferência

(X) Aprovado ***( ) Reprovado***

Dr. **Marcos Gonçalves Quiles**

Orientador(a) / INPE / São José dos Campos - SP

( ) Participação por Video - Conferência

(X) Aprovado ***( ) Reprovado***

Dr. **Elbert Einstein Nehrer Macau**

Membro da Banca / INPE / São José dos Campos - SP

( ) Participação por Video - Conferência

(X) Aprovado ***( ) Reprovado***

Dra. **Ana Carolina Lorena**

Convidado(a) / ITA / São José dos Campos - SP

( ) Participação por Video - Conferência

(X) Aprovado ***( ) Reprovado***

Dr. **Márcio Porto Basgalupp**

Convidado(a) / UNIFESP / SJCampos - SP

( ) Participação por Video - Conferência

(X) Aprovado ***( ) Reprovado***

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

São José dos Campos, 21 de agosto de 2019

*A meus irmãos **Matheus** e **Igor**.*

AGRADECIMENTOS

Agradeço à minha família, especialmente à minha mãe Ana Lúcia, pelo suporte, apoio e amor incondicional.

Agradeço ao Dr. Marcos Gonçalves Quiles, pela orientação, pelas ideias, pela paciência e confiança durante todos esses anos de parceria.

Agradeço a meu companheiro Gabriel, pelo amor e paciência nos momentos difíceis.

Às grandes amizades e contatos profissionais conquistados durante toda a jornada do doutorado.

Ao Instituto Nacional de Pesquisas Espaciais, pela oportunidade de estudar e aprender com as grandes mentes que compõe seus corpos docente e discente.

À CAPES, pelo apoio financeiro.

Meu sincero muito obrigada!

RESUMO

Fluxo de dados são um novo conceito surgido a partir do grande avanço tecnológico das últimas décadas. *Data streams*, como são chamados em inglês, são sequências de objetos que são gerados em tempo real e, portanto, trazem desafios únicos aos algoritmos que pretendem processá-los. Este trabalho está concentrado na tarefa de analisar os objetos que chegam com intuito de agrupá-los em conjuntos similares. Como os fluxos tendem ao infinito e, geralmente, os dados chegam com rapidez, os algoritmos de agrupamento para este tipo de dado, diferentemente das técnicas tradicionais, tem que desempenhar seus papéis com restrições quanto ao espaço de armazenamento e tempo de processamento que limitam sua atuação. Além disso, o algoritmo a tratar esses dados deve estar preparado para lidar com mudanças e evoluções no conceito dos dados ao longo do tempo. A metodologia apresentada neste trabalho, nomeada Rede Protótipo, utiliza uma estrutura de dados baseada em Redes Complexas para armazenar um sumário inteligente dos dados do fluxo, inteligente porque ao mesmo tempo que agrupa os dados em conjuntos similares, consegue acompanhar os movimentos de conceito sem maiores interferências do usuário. A proposta deste trabalho tem como maior vantagem a dependência de apenas dois parâmetros, $MAXV$ e H, sendo que o primeiro define a quantidade máxima de vértices da rede e o segundo a quantidade de dados recentes do fluxo a serem considerados. Os experimentos relatados nesta tese avaliam o desempenho da Rede Protótipo contra algoritmos clássicos na tarefa de agrupamento de fluxos como *CluStream* e *DenStream*, mas também contra outros algoritmos também baseados em Redes Complexas. Os algoritmos são testados com dados sintéticos que simulam mudanças e evoluções de conceito, além de dados provenientes de imagens de Observação da Terra, que se mostraram ainda mais desafiadores para os algoritmos de agrupamento de fluxo de dados.

Palavras-chave: Fluxo de Dados. Redes Complexas. Agrupamento. MODIS. Observação da Terra.

PROTOTYPE NETWORK: AN ALGORITHM FOR DATA STREAM CLUSTERING BASED ON COMPLEX NETWORKS

ABSTRACT

Data streams are a new concept that emerged from the significant technological advances of the last decades. Those data are sequences of objects that are generated in real-time and therefore pose unique challenges to the algorithms that intend to process them. This work is focused on the task of analyzing the objects that arrive with the intent of group them into similar sets. Since data streams tend to infinity and data usually arrive quickly, clustering algorithms for this type of data, unlike traditional techniques, have to play their role with storage space and processing time constraints that limit their performance. In addition, the algorithm handling this data must be prepared to deal with changes and developments in the concept of data over time. The methodology presented in this thesis, called Prototype Network, uses a data structure based on Complex Networks to store an intelligent data stream summary, intelligent because while grouping the data into similar sets, it can follow the concept movements without major user interference. The purpose of this work has the most significant advantage of relying on only two parameters, $MAXV$ and H, the first one defining the maximum amount of network vertices and the second the amount of recent stream data to be considered. The experiments reported in this thesis evaluate the performance of the Prototype Network against classical algorithms in the clustering task such as *CluStream* and *DenStream*, but also against other algorithms also based on Complex Networks. The algorithms are tested with synthetic data that simulate concept changes and evolution, as well as data from Earth Observation images, which have proven to be even more challenging for data streams clustering algorithms.

Keywords: Data Streams. Complex Network. Clustering. MODIS. Earth Observing.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE SÍMBOLOS

$\vec{x}$	–	vetor de atributos
$\vec{x}^i$	–	vetor de atributos com índice temporal i
$\vec{x}^i_j$	–	valor de um atributo j
F	–	fluxo de dados
N	–	tamanho do fluxo de dados
d	–	dimensão do vetor de atributos
C^x_i	–	*cluster i*
CF	–	*clustering feature*
PA	–	*prototype array*
H	–	tamanho da janela deslizante
t	–	tempo
Δt	–	intervalo de tempo
P^t	–	partição em *clusters* no tempo t
k	–	quantidade de *clusters*
n_C	–	quantidade de classes
Y_i	–	classe i
mc	–	*microcluster*
λ	–	parâmetro de peso da janela deslizante
V	–	conjunto dos vértices
E	–	conjunto das arestas
W	–	conjunto dos pesos das arestas
v_i	–	vértice v_i
(i, j)	–	aresta entre os vértices v_i e v_j
w_{ij}	–	peso da aresta (i, j)
k_i	–	grau do vértice i
$k_{\min}$	–	grau mínimo
$k_{\max}$	–	grau máximo
$\langle k \rangle$	–	grau médio
$d(.,.)$	–	distância entre dois objetos
M_{sim}	–	matriz de similaridade
ε	–	limiar
C^v_i	–	comunidade i de vértices
$MAXV$	–	parâmetro da rede protótipo
$centro_i$	–	centróide do vértice v_i
$filhos_i$	–	objetos representados pelo vértice v_i
$sim(.,.)$	–	similaridade entre dois objetos
μ	–	média
ρ	–	desvio-padrão

SUMÁRIO

Pág.

1 INTRODUÇÃO

Fluxo de dados são dados gerados e/ou transmitidos em altíssima velocidade e quantidade através de algum tipo de fonte, normalmente em tempo real (PUSCHMANN et al., 2017; AGGARWAL, 2007; GUHA et al., 2000; HAIDAR; GABER, 2018). Esse tipo de dado, ou mesmo de distribuição de dado, é uma novidade desse século para o público em geral, possível graças aos avanços tecnológicos obtidos nas últimas duas décadas, onde *hardware* e *software* deram um salto exponencial em potência de processamento e armazenamento (GABER et al., 2007; CHENG et al., 2008; NGUYEN et al., 2015).

Apesar de existir estudos e aplicações para esse conceito de dado anteriores ao século XXI, estes eram restritos a grandes empresas, instituições e universidades que eram capazes de financiar o alto custo desse processamento. Bancos, governos e centros de pesquisas são alguns exemplos (GAMA, 2012; ALMEIDA et al., 2018).

Bancos utilizam essa tecnologia na avaliação de crédito para seus clientes ou para decisões de investimentos. Instituições governamentais manipulam uma alta quantidade de dados para o censo, controle de epidemias, previsão do tempo e de desastres naturais, fiscalização ambiental e de fronteira, entre tantos outros. As universidades aliam, através de pesquisas multi-disciplinares, obtenção e processamento de diversos dados das mais diversas linhas de pesquisa.

Atualmente, dados são gerados em abundância e disponibilizados facilmente através de aplicações e aplicativos pela *internet*. Os exemplos estão em quase todas as áreas e não são mais restritos às grandes corporações. Monitoramento, vigilância, *internet* das coisas, redes sociais, GPS, reconhecimento facial, *logs*, transações comerciais e financeiras, clima, química, biologia, geologia, transporte, aviação, segurança pública, astronomia, observação da Terra são todos exemplos de áreas que geram dados em uma quantidade sem precedentes (IBRAHIM et al., 2018; VELOSO et al., 2003; GUPTA et al., 2014; MAO et al., 2018; AGGARWAL; YU, 2010; DUDA et al., 2017; HAIDAR; GABER, 2018; ABDULLATIF et al., 2018; PUSCHMANN et al., 2017; MESSNER et al., 2018).

Na literatura, muitas vezes se referem à esta época como a era do *Big Data*, este podendo ser definido como bases de dados que estão sempre aumentando. Mas, na era do *Big Data*, enormes montantes de dados não têm valor prático se estes não puderem ser analisados, manipulados, agrupados, classificados e transformados em informação que seja significativa para tomada de decisões e aprendizado (AMINI et al., 2014; CAO et al., 2006; SILVA et al., 2013; GUHA et al., 2000; AGGARWAL, 2007).

Quando uma enorme quantidade de dados precisa ser analisada, um dos aliados mais tradicionais para a tarefa na literatura é o *aprendizado de máquina*, visto que seria humanamente impossível analisar todos esses dados em tempo hábil. Logo, o estudo dos fluxos de dados anda lado a lado com o estudo do aprendizado de máquina, que é uma sub-área da *inteligência artificial* (AGGARWAL, 2007; JOSHI, 2017).

A Inteligência Artificial (IA) expandiu a capacidade de inovação ao oferecer novas formas de fazer todas as coisas que já eram feitas, mas agora com a ajuda dos computadores, que são máquinas muito eficientes em processamento sistemático de dados, além de muito úteis no armazenamento e organização de informações (GUHA et al., 2000; RUSSELL; NORVIG, 2009; AGGARWAL; REDDY, 2013; MOULTON et al., 2019). Além disso, são os computadores um dos grandes responsáveis pelos avanços tecnológicos já mencionados.

Um humano quando se encontra em uma situação desconhecida faz um rápido escaneamento do estado do ambiente e, após fazer comparações com experiências passadas, toma decisões e age. Essa associação da observação com a experiência é o que leva ao conhecimento, basicamente porque leva à identificação dos padrões, que são associações entre dados e resultados, ou ainda, entradas e saídas, causas e consequências (RUSSELL; NORVIG, 2009; JOSHI, 2017; MAINI; SABRI, 2017).

O aprendizado de máquina é um dos ramos mais populares da Inteligência Artificial (MITCHELL, 1997; MARSLAND, 2009; MAINI; SABRI, 2017; JOSHI, 2017). É o estudo sobre as possíveis maneiras que um computador pode aprender através da experiência, aumentando assim suas habilidades de planejar, decidir e agir (RUSSELL; NORVIG, 2009; KELLEHER et al., 2015; MAINI; SABRI, 2017). O objetivo é tornar as máquinas capazes de aprender por si, desenvolvendo *softwares* que podem examinar um conjunto de dados e partindo dessa observação conseguir fazer predições sobre novos dados desse mesmo conjunto (ZHU, 2005a; ZHU, 2005b).

O conceito de aprender dentro da IA difere bastante dependendo da sub-área de interesse, mas é o princípio fundamental desta área de estudo. As diferentes formas de aprendizado dependem quase sempre do contexto do problema que está sendo considerado e dos tipos de dados que são tratados (CONWAY; WHITE, 2012; ROSTEN; DRUMMOND, 2006; SEBASTIANI, 2002; LARRANAGA et al., 2006; FLACH, 2012). Dentro do aprendizado de máquina, existem diversas ramificações sobre diferentes tarefas para diferentes dados como o aprendizado supervisionado, o não-supervisionado, o semissupervisionado e por reforço (MITCHELL, 1997).

Classificação e *Agrupamento* são duas tarefas exaustivamente estudadas dentro do

aprendizado de máquina. A tarefa de classificação, que geralmente faz parte do aprendizado supervisionado, costuma treinar um modelo de decisão como uma *árvore bayesiana* ou uma *rede neural* com um grande conjunto de dados já rotulados em diferentes classes, para depois utilizá-la para classificar outros dados sem rótulo (GABER et al., 2007; BARBER, 2012; ABDULLATIF et al., 2018). A tarefa do agrupamento pode ser descrita como dividir um grupo de dados em grupos menores de forma que os grupos menores sejam formados por objetos mais similares entre si do que com os outros objetos externos ao grupo (RUSSELL; NORVIG, 2009; KELLEHER et al., 2015; JOSHI, 2017; MAINI; SABRI, 2017).

Tradicionalmente a tarefa do agrupamento é resolvida com a ajuda do aprendizado não-supervisionado, onde um modelo de decisão é formado para decidir qual o melhor particionamento dos dados com o objetivo de alcançar maior similaridade intra-grupo ao mesmo tempo que a similaridade inter-grupos é minimizada. Esta tarefa é difícil mesmo quando todos os dados estão disponíveis para consulta e quando se conhece *a priori* em *quantos* grupos os dados serão divididos (PHRIDVIRAJ; GURURAO, 2014; SCRUCCA et al., 2017; SUN et al., 2018).

O estudo de fluxos de dados tem estimulado a pesquisa no aprendizado de máquina, sobretudo no agrupamento, pois enquanto a classificação é uma tarefa que exige *softwares* mais complexos como redes neurais, o agrupamento pode ser realizado mantendo-se apenas abstrações de dados como um conjunto de centroides, o que atende melhor às restrições de memória e tempo características dos fluxos de dados (SILVA et al., 2013; WU et al., 2014; HAIDAR; GABER, 2018).

Neste contexto, esta tese apresenta uma nova metodologia para agrupamento de fluxo de dados utilizando estruturas de dados conhecidas como *redes complexas*. O objetivo é ajudar o processamento dos dados do fluxo oferecendo armazenamento eficiente para acesso fácil e rápida manipulação.

Uma rede complexa é essencialmente um grafo e, portanto, pode ser definida como sendo uma estrutura de dados com dois tipos de abstrações chamados *vértices* e *arestas*, onde o primeiro tipo representa um objeto do sistema e o segundo tipo representa uma relação entre dois objetos do mesmo sistema (COSTA et al., 2007; RODRIGUES, 2007; FORTUNATO, 2010; FORTUNATO, 2013).

Além da estrutura relacional do grafo, uma rede complexa pode armazenar *atributos* relacionados aos objetos e relações, e também executar *operações* relacionadas à inclusões ou eliminações de vértices e arestas. Essas possibilidades adicionais aumentam

ainda mais o poder de representação e organização dessas estruturas. Resumindo, estruturas do tipo rede complexa possibilitam a captura de características mais complexas das relações entre os objetos e dos objetos em si (CLAUSET et al., 2004; RAGHAVAN et al., 2007; PONS; LATAPY, 2006; ROSVALL et al., 2009).

As vantagens de uma rede complexa são exploradas na metodologia que é apresentada no Capítulo 4. A metodologia foi nomeada Rede Protótipo e utiliza uma rede complexa para armazenar um modelo de decisão que auxilia um algoritmo de agrupamento em fluxo de dados. Através de funções que atuam na estrutura, a metodologia Rede Protótipo mantêm um modelo representativo e dinâmico dos dados do fluxo com o objetivo de agrupamento.

No Capítulo 5, a metodologia é testada e avaliada ao comparar seu desempenho com algoritmos clássicos da literatura como o *CluStream* (AGGARWAL, 2003) e *DenStream* (CAO et al., 2006) e, além destes, a metodologia também terá seu desempenho comparado com algoritmos da literatura que também utilizam redes complexas em suas soluções, como o *SNCStream* (BARDDAL et al., 2015). Por fim, a metodologia também é testada com dados reais do contexto de observação da Terra.

Dentre as vantagens da metodologia proposta, destaca-se a necessidade de se definir apenas dois parâmetros, $MAXV$ e H. O primeiro determina o número máximo de vértices na rede e é definido baseado na quantidade disponível para armazenamento da estrutura, e o segundo determina a quantidade de dados recentes do fluxo que devem ser considerados no modelo de decisão. Os experimentos com dados sintéticos e dados reais de observação da Terra mostram que é possível tratar altas quantidades de dados com poucos vértices e que mesmo assim o algoritmo tem bom desempenho em comparação com a literatura.

1.1 Problemática

Na literatura, discute-se que dados para serem úteis precisam ser minerados para a obtenção de informações, e um aliado habitual dessa *mineração de dados* é o aprendizado de máquina. Para os fluxos de dados, das duas tarefas mais famosas no aprendizado de máquina, a classificação e o agrupamento, por enquanto, a segunda é mais viável que a primeira. Isso porque ainda é difícil, no cenário dos fluxos, ter acesso a uma fonte com uma quantidade suficiente de dados corretamente rotulados para a fase de treinamento sem recorrer a especialistas.

De qualquer forma, em alguns casos, a quantidade de dados é tão grande que mesmo com rótulos o bastante, o tempo de processamento e a quantidade de armazenamento que a classificação exige, requer *softwares* e *hardwares* mais robustos que nem todos

têm acesso. Quanto à tarefa de agrupamento, mesmo com recursos computacionais mais restritos, o algoritmo pode conseguir um desempenho satisfatório em alguns contextos, ainda mais se aliado à uma estrutura de dados robusta e flexível como é o caso das estruturas baseadas em Redes Complexas.

1.2 Motivação

Aplicação de algoritmos de mineração de dados e aprendizado de máquina em fluxos de dados é um tópico relativamente recente na literatura. Utilizar estruturas do tipo rede complexa na tarefa do agrupamento nesses dados também é uma abordagem recente. Além disso, até o momento não há trabalho publicado que faça agrupamento utilizando redes complexas em fluxo de dados formados por séries temporais.

1.3 Objetivos

O objetivo geral e os objetivos específicos desta tese estão listados a seguir.

1.3.1 Objetivo geral

O objetivo geral deste trabalho é apresentar uma metodologia de agrupamento em fluxo de dados auxiliada por uma estrutura do tipo rede complexa que obtenha desempenho eficiente frente a outros algoritmos da literatura.

1.3.2 Objetivos específicos

- Codificar na linguagem de programação R um conjunto de funções que operam uma estrutura do tipo rede complexa para auxiliar uma metodologia de agrupamento em fluxo de dados.

- Verificar se a metodologia consegue obter bom desempenho mesmo cumprindo restrições de tempo de processamento e memória de armazenamento.

- Testar a metodologia em diferentes cenários para verificar pontos fortes e pontos fracos.

- Comparar a metodologia proposta com outros algoritmos de agrupamento em fluxos de dados da literatura.

- Aplicar a metodologia em dados que são séries temporais de métricas relacionadas à observação da Terra.

1.4 Contribuição

A contribuição deste trabalho é um estudo mais amplo do impacto que o uso de uma estrutura do tipo rede complexa pode causar aos resultados de desempenho no agrupamento de dados provenientes de um fluxo. A metodologia apresentada mostrou-se eficiente em agrupar mesmo em cenários com presença de mudanças e evoluções de conceito. Além disso, a metodologia demonstra ser flexível pois com poucas adaptações foi possível utilizá-la no agrupamento de séries temporais.

2 AGRUPAMENTO EM FLUXOS DE DADOS

Neste capítulo, estão reunidas as revisões bibliográficas referidas à tarefa do agrupamento em fluxo de dados. Inicialmente, na Seção 2.1, são definidos os termos e propriedades necessários à introdução ao problema. Na Seção 2.2, são listadas as restrições e os requerimentos exigidos às soluções, assim como são discutidas os recursos comumente usadas na literatura, os movimentos que podem ocorrer nos *clusters* e como identificá-los. Na Seção 2.3, são apresentados algoritmos da literatura para a tarefa do agrupamento em fluxo de dados, dando ênfase aos algoritmos *CluStream* e *DenStream* que serão utilizados nos experimentos desta tese. Por fim, na Seção 2.4, são definidas as medidas para a avaliação e comparação dos resultados apresentados ao longo desta tese.

2.1 Definição do problema

A Inteligência Artificial (IA) tem como um dos ramos o aprendizado de máquina que, por sua vez, também se subdivide em ainda outros ramos de estudo (ZHU et al., 2003; ZHU, 2005b; ZHU, 2005a; AGGARWAL; REDDY, 2013). Considerando a literatura atual, as divisões são baseadas em quatro tipos diferentes de aprendizado (RUSSELL; NORVIG, 2009; GETZ et al., 2006; CHAPELLE et al., 2006; MITCHELL, 1997): *supervisionado*, *não-supervisionado*, *semissupervisionado* e *por reforço*. Dentre os quatro tipos de aprendizado há diferentes tarefas como: *classificação*, *regressão*, *agrupamento*, *aprendizado online* e *aprendizado ativo*, que podem agir separadamente ou em conjunto.

As diferenças entre as diversas tarefas do aprendizado de máquina estão relacionadas aos diferentes tipos de dados disponíveis para observação. Um algoritmo que deseja treinar uma máquina para aprender quais os padrões que associam os diferentes objetos dentro de um conjunto de dados tem como objetivos: (a) treinar o computador a identificar padrões ao observar o conjunto de dados; (b) construir um modelo que explique este conjunto a partir dessa observação; e (c) prever novas informações a partir desse modelo sem ter mais nenhuma regra pré-programada (ZYTKOW; RAUCH, 1999; BISHOP, 2006; FLACH, 2012).

Esta identificação de padrões ocorre ao comparar-se os dados observados, isto é, verificando como e quais características destes dados estão relacionadas e, baseando-se nos padrões encontrados, o algoritmo tenta prever características em dados desconhecidos (RUSSELL; NORVIG, 2009; MARSLAND, 2009; PHRIDVIRAJ; GURURAO, 2014; BIFET; GAVALDÀ, 2009; SETTLES, 2012).

Um algoritmo de aprendizado de máquina, ao identificar esses padrões, é capaz de formar heurísticas, representadas por modelos computacionais que automaticamente adquirem

mais conhecimento no domínio da aplicação ao observar novos exemplos do conjunto de dados que está sob análise (GABER et al., 2007; CHENG et al., 2008; NGUYEN et al., 2015).

Entretanto, a alta relação entre a capacidade de aprendizado da máquina e os dados leva a um problema: um modelo construído por um algoritmo de aprendizado de máquina só é tão poderoso quanto a qualidade dos dados disponíveis para observação, algo que nem sempre é fácil ou mesmo barato (WANG et al., 2018; GAMA, 2012; ALMEIDA et al., 2018; PUSCHMANN et al., 2017; MANSALIS et al., 2018; SILVA et al., 2017).

As tarefas do aprendizado supervisionado lidam com uma base inicial de dados que contém rótulos bem definidos e corretamente associados. Essa base inicial é dividida em dois lotes que são utilizados em maior parte para o treinamento do modelo e em menor parte para o teste de assertividade do modelo (AGGARWAL; REDDY, 2013; PHRIDVIRAJ; GURURAO, 2014; JOSHI, 2017).

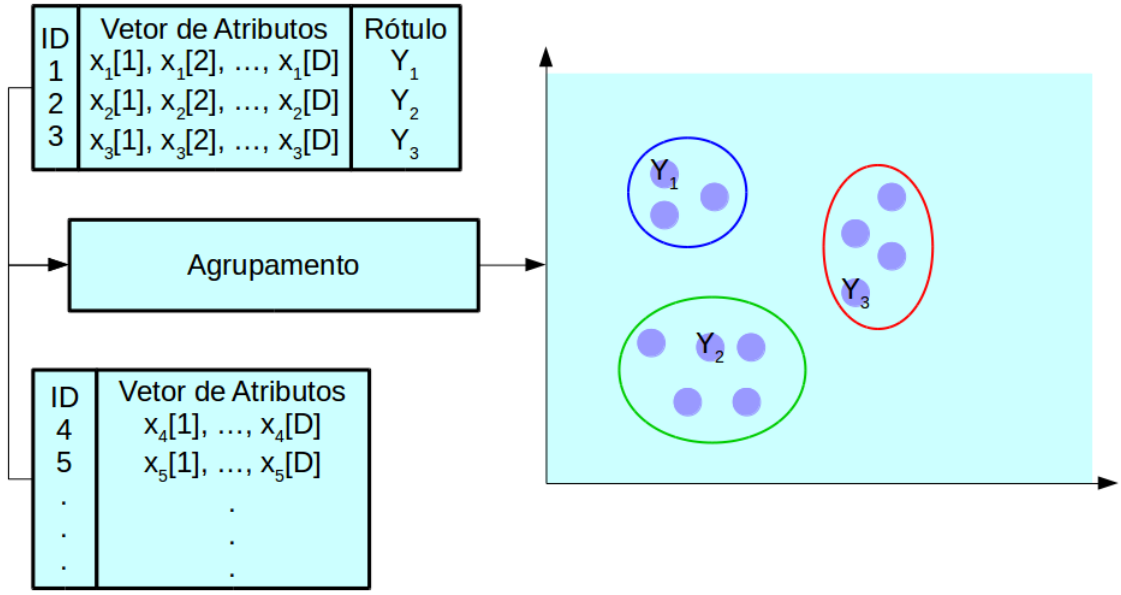
As tarefas de classificação e regressão geralmente fazem parte do aprendizado supervisionado, sendo que a diferença entre estas duas é que o rótulo almejado na classificação pertence a um conjunto finito e discreto de possibilidades, enquanto que na regressão, o rótulo é um valor contínuo (RUSSELL; NORVIG, 2009; ALMEIDA et al., 2018; NGUYEN et al., 2018b).

As tarefas do aprendizado não-supervisionado lidam com dados que não têm rótulos associados, mas que através da análise de seus atributos podem ser particionados em grupos que podem levar a *insigths* sobre os dados (GABER et al., 2007; GAMA, 2012). As tarefas do agrupamento e da redução de dimensionalidade geralmente fazem parte do aprendizado não-supervisionado (KELLEHER et al., 2015; HAIDAR; GABER, 2018; ABDULLATIF et al., 2018).

Outras tarefas que são também geralmente parte do aprendizado não-supervisionado, são relacionadas a detecção de *outliers*, ou ainda, *novelties*, que é a tentativa de identificar se um novo objeto do conjunto de dados é significadamente diferente do que já foi observado pelo modelo até então (AGGARWAL; YU, 2010; GAMA, 2012).

No aprendizado semissupervisionado há dados rotulados em menor quantidade e uma enorme quantidade de dados sem rótulo, dessa forma, algoritmos híbridos entre aprendizado supervisionado e não-supervisionado são utilizados (CHAPELLE et al., 2006; GETZ et al., 2006).

Figura 2.1 - Fluxo de atividades do aprendizado semissupervisionado.



O aprendizado semissupervisionado utiliza tanto dados rotulados como dados sem rótulo, mas o segundo tipo em bem mais quantidade que o primeiro. Agrupa-se as duas bases em conjunto analisando as distâncias entre os objetos no espaço de atributos e, se possível, faz a classificação através da propagação do rótulo. O aprendizado ativo pode ser um aliado nesse tipo de tarefa, ao sugerir a um especialista que rotule elementos que estão em posições estratégicas dentre os dados.

Fonte: Produção da Autora.

Baseando-se na regra chamada *cluster assumption* que diz que se dois objetos pertencem ao mesmo *cluster* muito provavelmente têm o mesmo rótulo, um algoritmo de classificação baseado em aprendizado semissupervisionado agrupa os dados e propaga os rótulos dos exemplos da base rotulada (VELOSO et al., 2003). Ou seja, o algoritmo de aprendizado semissupervisionado utiliza um conjunto de dados que contém uma porção relativamente menor de dados já rotulados e tenta classificar o restante dos dados sem rótulo sem depender tanto da fase de treinamento (CHAPELLE et al., 2006; ZHU, 2005b), na Figura 2.1 é dado um exemplo gráfico desse processo.

É possível também que o objetivo do algoritmo semissupervisionado seja somente o agrupamento, que nesse caso pode ser chamado *agrupamento com restrições*, ou ainda, *agrupamento com informações extras* (CHAPELLE et al., 2006; GETZ et al., 2006). O objetivo

continua sendo organizar os objetos em grupos similares, mas a base rotulada provê informações como, por exemplo, que dois objetos *devam* estar no mesmo *cluster* se tiverem o mesmo rótulo, ou ao contrário, que dois exemplos devam estar em grupos diferentes porque possuem rótulos diferentes. As informações extras, ou restrições, podem inclusive melhorar a forma como o algoritmo auto-avalia seus resultados (SETTLES, 2009; SETTLES, 2012).

O aprendizado semissupervisionado é uma categoria que tem muitas vantagens, pois pode ser considerado em várias tarefas do aprendizado de máquina. Além do que já foi exposto até aqui, ainda existe a possibilidade de aplicar o aprendizado ativo juntamente com o aprendizado semissupervisionado (BALCAN; URNER, 2016). Na literatura, quando o aprendizado semissupervisionado é utilizado há uma tendência para que o aprendizado ativo também o seja pois o tipo de dado disponível para o aprendizado semissupervisionado, muitos sem rótulo e poucos com rótulo, é também o tipo de dado que melhor se aplica ao aprendizado ativo (ZHU, 2005b).

O aprendizado ativo pode ser considerado um modelo de decisão que faz uma auto-análise para identificar potenciais dados que ao serem rotulados poderiam elevar a acurácia do modelo como um todo (SETTLES, 2009; SETTLES, 2012). É um sistema que iterativamente faz perguntas a um especialista para melhorar seu próprio desempenho. A hipótese é que se o modelo puder escolher quais exemplos expor aos especialistas baseados em suas posições no espaço de atributos, por exemplo, o desempenho da tarefa de aprendizado, normalmente de classificação, pode ficar bem menos custosa (BALCAN; URNER, 2016).

Contudo, apesar de os algoritmos semissupervisionados estarem sendo muito estudados atualmente, não há como garantir que os dados irão convergir corretamente quando se utiliza dados sem rótulo no modelo de classificação. Os dois tipos de dados (com e sem rótulo) utilizados nesse método influenciam fortemente o desempenho do algoritmo, ou seja, se os dados forem muito ruidosos, ou seja, contenha muitos dados que não necessariamente estejam de acordo com os padrões, o desempenho será baixo (SHAO et al., 2018; ZHOU et al., 2008).

Por fim, as tarefas de aprendizado por reforço agem por tentativa e erro até que regras (padrões) sejam identificadas entre os dados (MAINI; SABRI, 2017; JOSHI, 2017). O aprendizado por reforço geralmente é utilizado com a técnica do aprendizado *online* que é quando o algoritmo inicia com um modelo de decisão vazio que vai sendo incrementando conforme padrões vão sendo identificados (KELLEHER et al., 2015; JOSHI, 2017).

O aprendizado por reforço se refere ao processo de ensinar a um agente inteligente (de

certa forma, um modelo) a mapear o estado do ambiente à uma ação, ou seja, ao analisar uma situação o agente deve ser capaz de descobrir o que deve ser feito (RUSSELL; NORVIG, 2009; JOSHI, 2017).

Esse objetivo é alcançado através de uma recompensa que o agente tenta maximizar, sendo que a ação do agente pode levar ao aumento ou diminuição da recompensa acumulada até aquele momento. O agente inteligente, nesse caso, quase sempre inicia sem nenhuma conhecimento sobre o ambiente que vai explorar, ele procura através de tentativa e erro obter mais conhecimento sobre o espaço que está trabalhando e com isso executar ações que o levem a atingir o objetivo (ou mais recompensa) (AGGARWAL; REDDY, 2013; MAINI; SABRI, 2017).

Nesta tese, dentre as tantas tarefas do aprendizado de máquina, o agrupamento será o foco principal. Apesar dessa tarefa já ter sido exaustivamente abordada na literatura, só recentemente começou-se a estudar sua aplicação em fluxos de dados. Na próxima seção são especificados os desafios únicos que esses dados trazem à tarefa do agrupamento, o que acaba por exigir uma reconfiguração da forma como esta é feita.

2.2 Desafios relacionados aos fluxos de dados

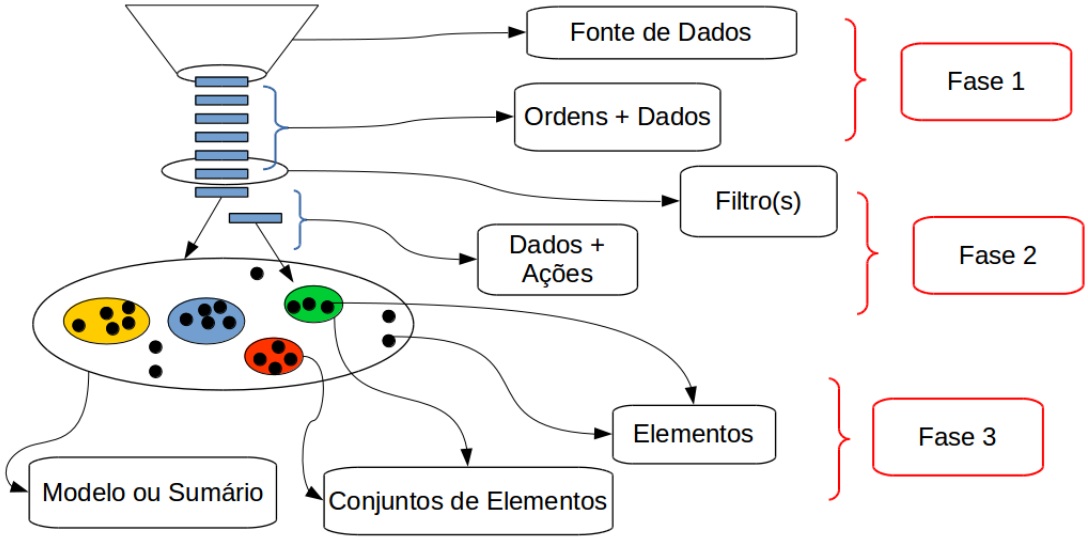
Um fluxo de dados é uma sequência de objetos que só se tornam disponíveis para análise em tempo real. Ou seja, diferentemente dos conjuntos de dados tradicionais, os dados não estão disponíveis para observação *a priori*. Esse novo conceito de conjunto de dados têm recebido bastante atenção nas últimas décadas pois exigem bastante pesquisa acadêmica para lidar com essa questão. Afinal, como fazer análises de dados que ainda estão por vir?

Além disso, apesar de um fluxo de dados ser uma sequência ilimitada de amostras, isto é, os objetos continuam a chegar infinitamente e em altas taxas de atualização, o modelo que pretende analisá-lo tem um conjunto finito e limitado de recursos computacionais para a tarefa, particularmente, tempo de processamento e memória de armazenamento (MANSALIS et al., 2018; PUSCHMANN et al., 2017; NGUYEN et al., 2015; GABER et al., 2007; GAMA, 2012).

As constatações descritas acima levam a algumas considerações. Em um primeiro momento, como a sequência de objetos cresce ilimitadamente, não é viável, nem prático, armazenar uma cópia completa de cada objeto em disco (GABER et al., 2007; NGUYEN et al., 2015; MANSALIS et al., 2018; BARDDAL et al., 2019). O ideal é que se mantenha um sumário dos dados que é atualizado esporadicamente. Com efeito, em um cenário ainda mais eficiente, espera-se que cada objeto seja analisado em sua chegada, resultando em

uma possível atualização no modelo, e seja descartado logo em seguida (GAMA, 2012; PUSCHMANN et al., 2017; MAO et al., 2018).

Figura 2.2 - Resumo geral das atividades relacionadas com a manipulação e análise de um fluxo de dados.



Atividades relacionadas à manipulação e análise de um fluxo de dados. As atividades podem ser divididas em três fases, cada fase representado tarefas de escopos diferentes. Na primeira estão tarefas relacionadas ao tratamento da fonte de dados, na segunda fase tarefas relacionadas a análise *on-line* dos dados e filtros de transformação e na terceira fase tarefas relacionadas ao armazenamento e análise *off-line* dos dados.

Fonte: Produção da Autora.

Em um segundo momento deve-se considerar que novos dados chegam em alta frequência no sistema que está a analisá-los e o processamento do último dado não deve exceder o tempo de chegada do novo dado que virá (SHAO et al., 2018; SUN et al., 2018).

Para além do que já foi exposto, fluxos de dados normalmente sofrem de fenômenos conhecidos na literatura como mudança de conceito (*concept drift*) e evolução de conceito (*concept evolution*) (GUHA et al., 2000; MESSNER et al., 2018; CARNEIN; TRAUTMANN, 2019), que é a alteração da probabilidade de distribuição dos dados no espaço de atributos. Isto é, em comparação com a análise tradicional de dados que supõe que a fonte geradora dos dados têm uma probabilidade de distribuição fixo, os fluxos de dados podem vir a ter essa probabilidade de distribuição alterando-se de tempos em tempos e,

portanto, ao criar-se um modelo de representação com os dados passados, este não será necessariamente um modelo assertivo para analisar todos os outros que virão.

Na análise de dados provenientes de um fluxo, o modelo deve ter mecanismos para identificar quando os padrões representados pelo modelo estão defasados e não representam mais o conceito atual do fluxo e automaticamente atualizar-se para a nova distribuição (NGUYEN et al., 2018b; HAIDAR; GABER, 2018).

Na Figura 2.2 pode ser observado um resumo geral das atividades relacionadas a um fluxo de dados. Essas atividades podem ser divididas em três fases. Na primeira fase são consideradas tarefas referentes ao tratamento da fonte de dados que incluem o recebimento do dado, mecanismos para lidar com concorrência de acesso, questões relacionadas à periodicidade da chegada dos dados ou ordens que devem ser aplicadas nos dados e ao suporte a fontes múltiplas e heterogêneas tanto das ordens como dos dados (IBRAHIM et al., 2018).

Além disso, considera-se ainda que a fonte dos dados pode emitir ordens além da geração de novos dados, como por exemplo a eliminação de um objeto ou ainda atualização dos valores dos atributos de um dado que já tenha sido observado anteriormente pelo modelo (AMINI et al., 2014; SILVA et al., 2017; KRAWCZYK et al., 2017).

Na segunda fase são consideradas as tarefas de transformações dos dados, análise *on-line* destes, tomadas de decisão e possíveis atualizações no modelo representativo (ou sumário) do fluxo (AGGARWAL; YU, 2010; GAMA, 2012). Por fim, na terceira fase são as tarefas relacionadas a *insights*, armazenamento otimizado, sumarização, análise *offline*, agrupamento, classificação ou mineração dos dados (WANG et al., 2018; AHMED, 2018).

Esta tese descreve uma metodologia que atua na terceira fase, onde assume-se que os dados chegam em sequências a intervalos fixos no tempo onde cada objeto $\vec{x}$ tem um *timestamp* i que o identifica unicamente.

2.2.1 Restrições e requerimentos

Do ponto de vista computacional, um fluxo de dados F pode ser representado como uma sequência de objetos $\vec{x}^1, \vec{x}^2, \ldots, \vec{x}^N$, tal que $F = \{\vec{x}^i\}_{i=1}^N$ e $N \to \infty$, porque essa sequência é potencialmente ilimitada. Cada objeto $\vec{x}^i$ tem intrínseco a si um índice temporal ou ainda um *timestamp* i que indica a ordem de chegada desse objeto no fluxo F e o objeto é, normalmente, um vetor de atributos de dimensão d, tal que $\vec{x}^i = [x_j^i]_{j=1}^d$ (AGGARWAL, 2003; CAO et al., 2006).

O objetivo do agrupamento, portanto, é separar esses objetos $\vec{x}^i$ em grupos (*clusters*) C_i^x tal que a similaridade entre objetos dentro de um mesmo grupo seja relativamente maior que a similaridade entre objetos de grupos diferentes (JOSHI, 2017; MAINI; SABRI, 2017).

Mas, no contexto dos fluxos, alguns requerimentos e restrições precisam ser levados em consideração. Por exemplo, os fluxos de dados tem um comportamento volátil e a quantidade de *clusters* pode mudar de tempos em tempos (MAO et al., 2018). Além disso, o algoritmo que for realizar o agrupamento terá restrições de tempo de processamento de cada objeto que surge no fluxo e de memória disponível para armazenamento dos dados que forem relevantes para a tarefa (GABER et al., 2007; GAMA, 2012).

Em alguns cenários, ainda é possível que o algoritmo tenha que lidar com *outliers*, que podem ser tanto *ruído*, como também o início de um novo *cluster* (CAO et al., 2006). Estão enumerados a seguir as restrições e requerimentos que são comumente citados na literatura para a tarefa do agrupamento em fluxos de dados (AGGARWAL, 2006; GABER et al., 2007; CHENG et al., 2008; GAMA, 2012; AMINI et al., 2014; MOULTON et al., 2019):

a) Restrições:

   - Os objetos continuam a chegar de forma *online* e o tamanho N total do fluxo é possivelmente infinito.
   - O algoritmo deve ser escalável para o crescimento contínuo do *stream*.
   - O modelo representativo deve ser compacto, mas também não deve crescer proporcionalmente ao valor de N, nem mesmo linearmente.
   - O processamento de um dado deve ser mais rápido que a chegada do novo dado.
   - Não faz parte do domínio do agrupamento a ordem em que os dados serão processados, o algoritmo segue a ordem de chegada destes. Ou seja, não é permitido acesso randômico aos dados, somente sequencial.
   - Objetos devem ser descartados após análise ou então mantêm-se apenas uma parte do objeto por um tempo para ser descartado depois.

b) Requerimentos:

   - O processo de geração dos dados é desconhecido e provavelmente não-estacionário, ou seja, a probabilidade de distribuição pode mudar ao longo do tempo. A adaptação às mudanças nos dados, sejam elas de movimentos na concentração no espaço de atributos, ou ainda, de aparição e dissipação de grupos deve ser feita o mais rápido possível.

- O algoritmo deve contar com mecanismos para identificar *outliers* e ruídos nos dados e agir neles de acordo com o contexto do problema.
- O algoritmo deve ser capaz de lidar com fontes heterogêneas de dados.

Lidar com fluxos dinâmicos e não-estacionários de dados implica que o modelo de decisão também é sujeito à evolução ao longo do tempo (PUSCHMANN et al., 2017). Desse modo, o desafio principal da tarefa pode ser resumido em: como manter uma representação eficiente em adquirir *insigths* quanto as partições em grupos de dados que têm como características serem *online*, *fluidos* e *voláteis*?

Algoritmos tradicionais de agrupamento de dados, em sua maioria, processam cada um dos objetos de um conjunto mais de uma vez para construir o modelo de decisão (FLACH, 2012). No entanto, essa estratégia não é viável para o contexto dos fluxos, visto que não se tem acesso aos dados em sua totalidade e novos dados surgem continuamente (BARDDAL et al., 2019). De qualquer forma, também não é interessante manter uma cópia original em disco de todos os objetos que já foram observados, o ideal é que os dados sejam descartados total ou parcialmente após um único processamento (SILVA et al., 2017).

Essa restrição é conhecida na literatura como *single-pass* e, para o contexto do agrupamento, os algoritmos são levados a considerar que é preciso utilizar *abstrações* eficientes de dados para armazenar somente o essencial de cada objeto (MANSALIS et al., 2018). Também leva a considerar que de tempos em tempos abstrações que representam objetos muito antigos do fluxo de dados sejam descartados, para que esses dados antigos não dominem os padrões do modelo de decisão (SHAO et al., 2018; MAO et al., 2018).

2.2.1.1 Abstrações e janela deslizante

Essas abstrações eficientes devem ser representações compactas de um conjunto de elementos e normalmente estes são organizados em uma estrutura maior que é o modelo representativo completo. Dentre as formas de abstrações, existem algumas que se destacam na literatura, como, por exemplo, *clustering features* (CF) ou *prototype arrays* (PA) (MANSALIS et al., 2018).

Enquanto o segundo é um centroide entre os objetos que estão sendo representados, o primeiro além de manter uma informação de centroide também armazena outras características dos dados (HAIDAR; GABER, 2018; CARNEIN; TRAUTMANN, 2019). Para o contexto do agrupamento, o importante é que essas microestruturas tenham propriedades que permitam que, de forma simples e fácil, cada um deles possa ser incrementado com um novo objeto e, também, que dois deles possam ser unidos em um único (BARDDAL et

al., 2019).

Um CF é formado por três valores (N, LS, SS), onde N é o número de dados que estão sendo representados pelo *cluster*, LS é a soma linear dos atributos dos dados e SS é a soma quadrática dos atributos. Para incrementar um novo objeto $\vec{x}^i$ a um CF basta adicionar $N = N+1$, $LS = LS+\vec{x}^i$ e $SS = SS+(\vec{x}^i)^2$. Para mesclar dois CF's, CF_1 e CF_2, basta somar suas partes $CF_{12} = (N_1+N_2, LS_1+LS_2, SS_1+SS_2)$ (AGGARWAL, 2003; AGGARWAL et al., 2003).

Para calcular um novo *prototype array* seja pela junção de dois PA's ou pela adição de um novo objeto basta fazer o cálculo do novo centroide, que calcula a média entre cada atributo dos *arrays*, sendo que essa média pode ser aritmética ou ponderada (MANSALIS et al., 2018). Essa é a estratégia utilizada na proposta desta tese, apesar de em um primeiro momento os PA's parecerem pouca coisa para se captar dos dados, a metodologia mantém essas abstrações organizadas por uma rede complexa, o que diminui a carga de dados que seria necessário manter nas abstrações, pois a rede em si carrega informações extras sobre os dados que são captados durante a construção e manutenção da estrutura.

O modelo de decisão também precisa ter uma estratégia para descartar essas abstrações de tempos em tempos, quando os objetos representados por esses já são muito antigos (GABER et al., 2007). Para esse fim, alguns algoritmos optam por usar uma técnica chamada *janela deslizante*, que é uma forma de manter no modelo representativo somente os dados mais recentes/representativos do conjunto, ou seja, a janela indica quando um objeto (abstração) do passado deve ser descartado (CAO et al., 2006; AGGARWAL, 2007; GAMA, 2012).

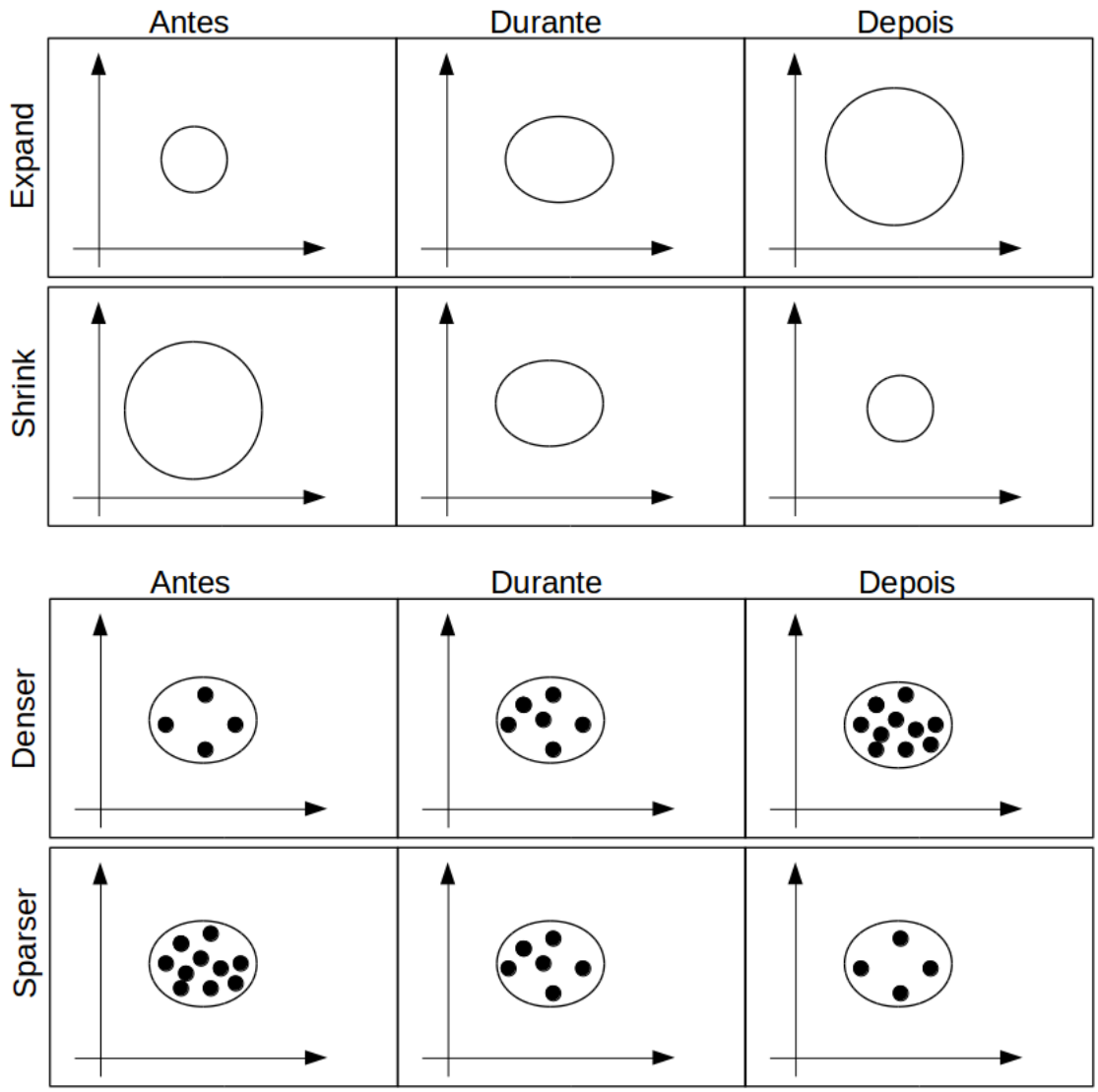
O mecanismo de janela deslizante tem dois tipos de abordagens, uma baseada no tempo e outra baseada no número total de objetos representados, na variação baseada no tempo, as abstrações que representam objetos com *timestamp* $i < t$ são descartados, sendo t um tempo limite no passado (MANSALIS et al., 2018). Na variação baseada no número total de objetos, o objetivo é manter a quantidade total de objetos representados abaixo de um limiar, ou seja, toda vez que um novo objeto chegar e esse limiar for ultrapassado, uma abstração do passado precisa ser eliminada (BARDDAL et al., 2019), esse limiar é normalmente representado por H.

2.2.1.2 Movimento dos *clusters*

No problema tradicional do agrupamento, é considerado que os dados foram gerados por uma fonte estacionária, ou seja, que tem probabilidade de distribuição dos dados

fixa (GAMA, 2012). No contexto dos fluxos, essa característica dos dados pode ser não-estacionária.

Figura 2.3 - Movimentos do tipo *Expand*, *Shrink*, *Denser* e *Sparser*.



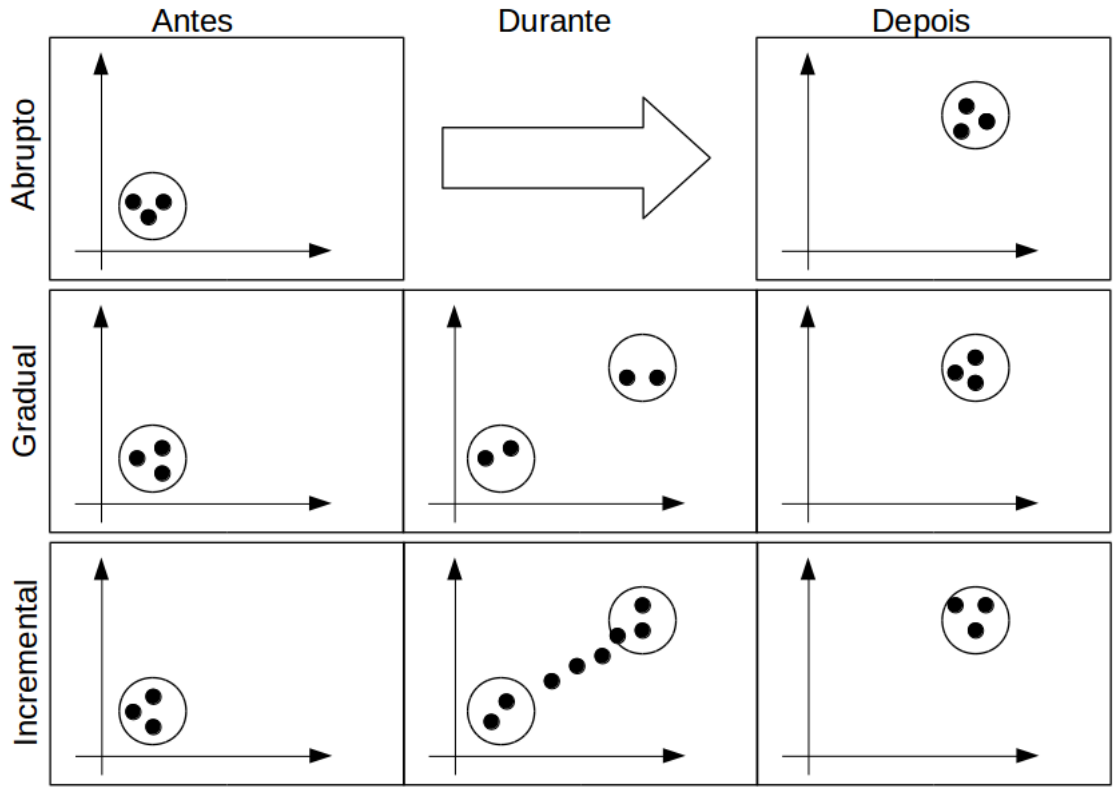
Nos movimentos de *expand* e *shrink*, o *cluster* aumenta ou diminui a ocupação no espaço de atributos, ou seja, o raio do *cluster* altera. Nos movimentos de *denser* e *sparser*, o volume de elementos aumenta ou diminui no *cluster*.

Fonte: Produção da Autora.

Essas alterações na probabilidade de distribuição podem ocorrer de duas formas, ou

intra-*cluster* que são mudanças no contexto interno de cada *cluster*, ou seja, número de objetos total, centro e raio, ou extra-*cluster*, que são movimentos como aparição de novos *clusters* ou dissipação de *clusters* antigos (MOULTON et al., 2019).

Figura 2.4 - Movimentos do tipo *Shift*: abrupto, gradual e incremental.



Nos movimentos de *shift*, o centro do *cluster* se move pelo espaço de atributos. No primeiro caso, no movimento abrupto, o centro está em uma posição no tempo t e no tempo $t + \Delta t$ já ocupa uma nova posição, ou seja, até o tempo t, elementos desse *cluster* só aparecem em volta da posição anterior do centro, a partir de $t + \Delta t$ os novos elementos só aparecem em volta da nova posição. Nos dois outros casos, gradual e incremental, há um período de transição entre os dois diferentes centros, só que no primeiro caso, no período de transição, os elementos ficam aparecendo em volta do antigo e do novo centro alternadamente. No caso do *shift* incremental, no período de transição, aparecem elementos que formam um caminho entre a antiga posição do centro e a nova posição.

Fonte: Produção da Autora.

É possível também que esses movimentos sejam sazonais, o que também é uma

característica interessante de se observar nos dados. As mudanças que envolvem movimentos intra-*cluster* são conhecidos na literatura como *concept drift* e movimentos extra-*cluster* são chamados *concept evolution* (IBRAHIM et al., 2018; CARNEIN; TRAUTMANN, 2019; MOULTON et al., 2019).

Formalmente os movimentos podem ser chamados: *Expand*, *Shrink*, *Denser*, *Sparser*, *Shift*, *Merge*, *Split*, *Disappear* e *New*. Os movimentos *Expand* e *Shrink* são relacionados ao raio do *cluster*, ou seja, o *cluster* pode crescer ou diminuir sua ocupação no espaço de atributos (MOULTON et al., 2019). *Denser* e *Sparser* estão relacionados com o volume de objetos dentro de um mesmo *cluster*, que pode aumentar ou diminuir. Esses quatro movimentos podem ser vistos na Figura 2.3.

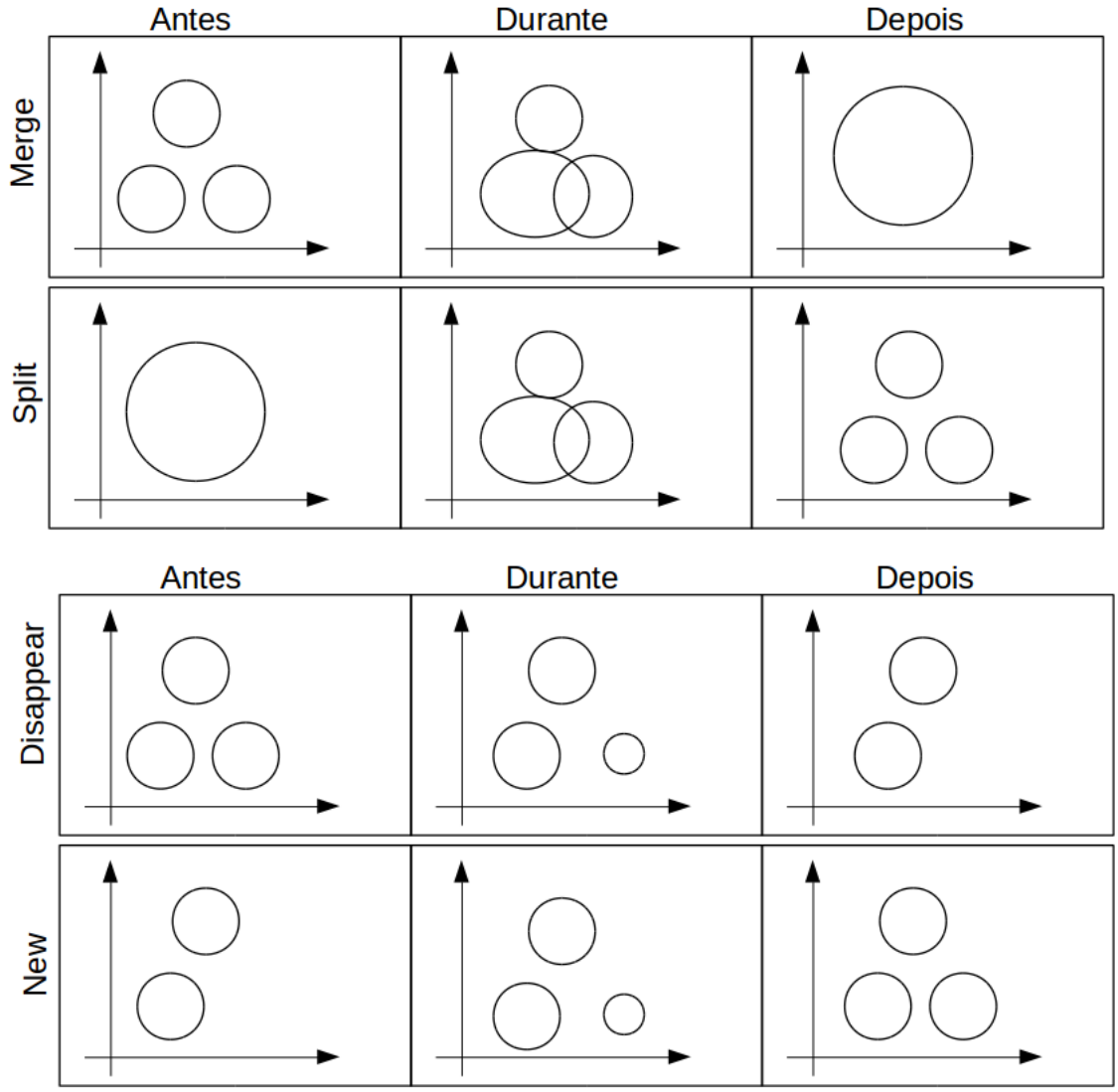
O movimento de *Shift* é quando o centro do *cluster* se move no espaço de atributos, esse *shift* pode ser abrupto, gradual ou incremental e estão representados graficamente na Figura 2.4. Esses últimos cinco movimentos são reunidos pelo nome de *concept drift* e não alteram a quantidade total de *clusters* (GAMA, 2012; RODRIGUES et al., 2018).

Já os movimentos que são do tipo *concept evolution*, que causam alteração na quantidade total de *clusters*, podem ser os movimentos de *Merge* e *Split*, que indica que dois ou mais *clusters* foram mesclados em um único, ou então que um *cluster* foi dividido em dois ou mais, além dos movimentos *Disappear* e *New*, que são quando um *cluster* "morre" ou então quando um novo *cluster* "nasce" (WANG et al., 2018; HAIDAR; GABER, 2018; NGUYEN et al., 2018a). Esses últimos quatro movimentos podem ser observados na Figura 2.5.

No mundo real esses 9 movimentos ocorrem de forma simultânea e independente em cada *cluster*/sistema e idealmente o algoritmo que for realizar agrupamento deve ter mecanismos para lidar com todos, além de lidar com as restrições e requerimentos já mencionados no início desta seção (SHAO et al., 2018; MANSALIS et al., 2018; HAIDAR; GABER, 2018; MOULTON et al., 2019).

Apesar de já terem aparecido diversos algoritmos e soluções para o problema relatado nesta seção, ainda há muito a se pesquisar na área. Na próxima seção, serão comentados algumas das soluções que foram propostas na literatura dando ênfase a dois algoritmos que são mais citados nos trabalhos científicos e que serão utilizados nos experimentos desta tese: *CluStream* (AGGARWAL, 2003; AGGARWAL et al., 2003) e *DenStream* (CAO et al., 2006).

Figura 2.5 - Movimentos do tipo *Merge*, *Split*, *Disappear* e *New*.



Nos movimentos de *merge* e *split*, o número total de *clusters* são alterados pois ou dois ou mais *cluters* se fundem em um único, ou então um único *cluster* se divide em dois ou mais. Nos movimentos de *disappear* e *new*, os elementos de um cluster vão desaparecendo ou aparecendo. No primeiro caso, chega um momento que não há mais nenhum elemento deste *cluster*, e no segundo caso, primeiramente um único elemento no novo *cluster* aparece seguido de outros elementos.

Fonte: Produção da Autora.

2.3 Algoritmos da literatura

Na seção anterior, discutiu-se que os dados provenientes de fluxos são um desafio maior para os algoritmos de aprendizado de máquina, sobretudo para a tarefa do agrupamento. Além de várias restrições e requerimentos quanto ao tempo, memória e acesso, há também que lidar-se com a volatilidade dos dados. A volatilidade implica que uma partição do conjunto de abstrações em k grupos no tempo t, $P^t = \{C_i^x\}_{i=1}^k$, pode ser diferente no tempo $t + \Delta t$, $P^t \neq P^{t+\Delta t}$. Como resultado, os padrões capturados pelo modelo de decisão em t podem não representar mais a realidade dos dados em $t + \Delta t$.

Várias soluções foram propostas para o problema, sendo que um dos primeiros foi o algoritmo Stream (GUHA et al., 2000), seguido pelo *CluStream* (AGGARWAL, 2003; AGGARWAL et al., 2003), *DenStream* (CAO et al., 2006), *D-Stream* (CHEN; TU, 2007), *FlockStream* (FORESTIERO et al., 2009), *ClusTree* (KRANEN et al., 2011) e vários outros que surgiram a partir desses, que estão reunidos e descritos em trabalhos como (AGGARWAL, 2006; GABER et al., 2007; GAMA, 2012; AMINI et al., 2014; CHEN; ZHANG, 2014; GUPTA et al., 2014; NGUYEN et al., 2015; QIN et al., 2016) e mais recentemente em (ABDULLATIF et al., 2018; HAHSLER et al., 2018; MANSALIS et al., 2018; MARCU et al., 2018; RODRIGUES et al., 2018; SHAO et al., 2018; HAIDAR; GABER, 2018; BARDDAL et al., 2019; CARNEIN; TRAUTMANN, 2019; MOULTON et al., 2019).

O algoritmo *Stream* aplica o *k-means* (MACQUEEN, 1967), uma solução tradicional, em um cenário com fluxo de dados de forma incremental, ou seja, de tempos em tempos o *k-means* é executado nos dados mais recentes indicados por um mecanismo de janela deslizante. Isto é, considerando uma partição $P^t = \{C_i^x\}_{i=1}^k$ e uma janela que tem tamanho H, os *clusters* são formados analisando-se os objetos $\vec{x}^j$, tal que o *timestamp* j esteja no intervalo $[t - H, t]$, sendo t o tempo atual.

As limitações do algoritmo *Stream* para o cenário que está sendo analisado nesta tese é que falta um componente *offline*, que somente é proposto posteriormente por Aggarwal *et al.* em (AGGARWAL, 2003; AGGARWAL et al., 2003) e também porque ao utilizar o algoritmo *k-means* para criar uma divisão em grupos, o algoritmo depende de um parâmetro k. Esse parâmetro k corresponde ao número final de *clusters* que terá no modelo ao final da execução do algoritmo, o ideal é que esse valor fosse inferido pelo próprio modelo.

A falta do componente *offline*, que pode ser interpretado como um conjunto de sucessivos *snapshots*[1], impede o algoritmo de fazer análise da *evolução* do modelo. Isto é, impede de verificar se ocorreu algum movimento nos *clusters* em um passado recente que levou

[1] *Snapshots* são como "fotografias" do estado atual de um sistema.

ao estado do que está sendo visto agora. Verificar se ocorreram movimentos pode indicar *tendências* que podem ser utilizadas pelo próprio modelo para *posicioná-lo* de melhor maneira, seja para incorporar nova informação ou descartar informação defasada.

Seguintes ao algoritmo *Stream*, foram propostos o *CluStream* (AGGARWAL, 2003), *Denstream* (CAO et al., 2006) e muitos outros. Cada um deles trouxe soluções inovadoras para o cenário, por exemplo, o *CluStream* utilizou um conceito chamado *microclusters* (*mc*) e propôs a manutenção de sucessivos *snapshots* desses *mc* em uma estrutura *offline* que possibilita uma análise de evolução no tempo.

O *DenStream* trouxe soluções para lidar com dinamicidade, incorporando conceitos para separar os *microclusters* em três tipos: core-*mc*, potencial-*mc* e outlier-*mc*, dessa forma, podendo lidar com dados que fossem *ruídos*.

O *ClusTree* (KRANEN et al., 2011) é um modelo que se auto-adapta ao *stream* e que utiliza uma estrutura da família das Árvores R (*R-tree*) que eficientemente organiza os dados em um sumário. O *FlockStream* (FORESTIERO et al., 2009) é um algoritmo que utiliza inspiração biológica para criar um modelo adaptativo.

O *D-stream* (CHEN; TU, 2007) segue a estratégia do *CluStream* mas adota um modelo baseado em *grids*, onde o espaço de atributos é dividido com granularidades diferentes dependendo da densidade de objetos na posição.

Para efeito de averiguação de desempenho e acurácia da metodologia proposta nesta tese, ela será comparada com os algoritmos *CluStream* e *DenStream*, por serem pioneiros e dois dos mais citados algoritmos da literatura. A seguir explica-se esses dois algoritmos em mais detalhes e na seção seguinte as métricas de avaliação que serão usadas na comparação.

2.3.1 *CluStream*

O *CluStream* (AGGARWAL, 2003; AGGARWAL et al., 2003; AGGARWAL, 2006) mantém q *microclusters* que são extensões de *clustering features* e de tempos em tempos armazena um *snapshot* do estado atual dos *microclusters*. Caso seja solicitado o agrupamento dos dados, o algoritmo aplica o *k-means* no estado atual dos *microclusters*, sendo que há *snapshots* armazenados por um período H, sendo possível, então, comparar o estado atual dos *clusters* com esse período no passado e verificar sua evolução. Os *snapshots* são armazenados em uma estrutura piramidal que ajuda a economizar espaço, pois consegue detectar e excluir redundâncias.

Os *microclusters* são as abstrações mantidas on-line, que armazenam cinco valores $(N, LS, SS, T1, T2)$ que além dos valores N, LS e SS que já foram explicados anteriormente neste capítulo, captura também duas componentes temporais que são a soma $T1$ dos *timestamps* i dos objetos $\vec{x}^i$ e a soma quadrática $T2$, onde $T1 = \sum_i^N i$ e $T2 = \sum_i^N i^2$.

Em um primeiro momento, os q *microclusters* são formados por um conjunto inicial do fluxo que são submetidos a uma execução do *k-means* tradicional. Isto é, após lido alguns objetos e um conjunto inicial de objetos ser formado, o algoritmo *k-means* é utilizado para formar os q primeiros *microclusters*.

Feito isso, a cada novo objeto $\vec{x}^i$ observado, o modelo precisa decidir se irá incorporá-lo em algum *microcluster* já existente ou se irá formar um novo *microcluster* composto por apenas esse objeto. Essa decisão é tomada avaliando-se a distância entre o novo objeto e os centroides dos *microclusters* atuais. Se houver um ou mais *microcluster* que tenha distância menor que um certo limite, aquele com a menor distância incorpora o novo objeto. Esse limite é definido pelos valores do próprio *microcluster*.

Se o novo objeto não for incorporado, então, ou ele é ruído, ou ele é o início de um novo *cluster*, ou ainda, o início de um novo movimento nos *clusters*. Independente disso, é preciso um tempo até o modelo poder decidir o que fazer com esse novo objeto, logo, um novo *microcluster* é criado que contém inicialmente os dados do novo objeto.

Caso ocorra de um novo objeto precisar ser colocado em um novo *microcluster*, verifica-se se já há q *microclusters* atualmente, se for o caso, o algoritmo precisa decidir se irá abrir espaço para o novo *microcluster* através da junção de dois outros ou então pela eliminação de um deles.

Para decidir se pode eliminar algum dos q *microclusters*, o algoritmo verifica o *revelance stamp* de cada *microclusters*, que são calculados a partir dos componentes temporais $T1$ e $T2$. Quando esse valor for menor que um parâmetro δ, então o *microcluster* pode ser eliminado.

Caso nenhum *microcluster* possa ser eliminado então dois *microclusters* terão que ser mesclados, os escolhidos são aqueles que tiverem a menor distância entre si. A mesclagem é feita somando-se as partes dos *microclusters*, assim como feito nos CF.

De tempos em tempos um *snapshot* do estado atual dos q *microclusters* é armazenado em memória. Com esses *snapshots* é possível visualizar a evolução de um *cluster* no tempo, pois analisando-os em sequência, os *microclusters* podem ser separados em grupos dos que foram adicionados, dos que foram eliminados e dos que foram mantidos entre os dois

tempos.

Dessa forma, com a sequencia de *snapshots*, o componente *offline*, que faz a formação dos *clusters*, precisa lidar apenas com os dados armazenados nos *microclusters*. O parâmetro H e o parâmetro k são as entradas do componente *offline*, o primeiro determina o tamanho da janela e o segundo a quantidade de *clusters* que devem ser formados.

2.3.2 *DenStream*

O *DenStream* (CAO et al., 2006; AMINI et al., 2014; RODRIGUES et al., 2018) é baseado em densidade e introduz soluções para o tratamento de ruídos. Ele utiliza três tipos de *microclusters*: core-*mc*, potencial-*mc* e *outlier-mc*. Os core-*mc* são formados quando o agrupamento é requisitado, tendo como entrada o estado atual dos potencial-*mc* e outlier-*mc*. O agrupamento é feito utilizando o algoritmo DBSCAN que também é baseado em densidade para formar os k core-*mc* (*clusters*).

O algoritmo usa um mecanismo de janela que dá maior peso aos valores dos objetos mais recentes, o peso de cada objeto decai segundo função $f(t) = 2^{-\lambda t}$, onde $\lambda > 0$ é um parâmetro definido pelo usuário. O *DenStream* mantêm uma estrutura *online* formada pelos potencial-*mc* e um *buffer* a parte para armazenar os *outlier-mc*.

Um potencial-*mc* é um conjunto de objetos do fluxo que tem um peso, um centro e um raio. O peso é utilizado para verificar se o potencial-*mc* está ou não defasado segundo o mecanismo de janela deslizante, o centro e o raio são utilizados para verificações de incorporação ou não de um novo objeto. O peso é calculado como $w = \sum_{x^j \in p-mc} f(t-j)$ e o potencial-*mc* é mantido enquanto esse peso for maior que um valor $\beta\mu$, sendo $0 < \beta \leq 1$ e μ parâmetros do algoritmo.

O que mantém o peso de um potencial-*mc* maior que o limite $\beta\mu$ é se objetos são continuadamente incorporados nele. Para cada novo objeto, o algoritmo verifica a distância com todos os potencial-*mc* através de seus centros, aqueles que estiverem mais perto são testados para verificar se ao incorporar o novo objeto o raio ultrapassa um parâmetro ε. Caso não, o novo objeto é incorporado ao potencial-*mc*.

Se nenhum potencial-*mc* se mostrar capaz de incorporar o novo objeto, então tenta-se incorporar o novo objeto em algum *outlier-mc*. Caso algum *outlier-mc* incorpore o novo objeto, verifica-se se o peso dele é agora maior que $\beta\mu$, se sim, então o *outlier-mc* é promovido a potencial-*mc*, ou seja, é retirado do *buffer* e colocado na estrutura *online*.Finalmente, caso nenhum *outlier-mc* incorpore o novo objeto então cria-se um novo *outlier-mc* que contém o novo objeto.

De tempos em tempos, a estrutura completa formada pela estrutura *online* e o *buffer* é escaneada para verificar o peso dos componentes, caso o mecanismo de janela deslizante verifique que os componentes, potencial-*mc* ou outlier-*mc*, estejam defasados através da medição do seu peso, então o respectivo é eliminado. Caso o agrupamento tenha sido requisitado, o algoritmo aplica o *DBSCAN* nos potencial-*mc* para formar os core-*mc* que é a partição dos dados representados em *clusters*.

Apesar do *DenStream* ser um dos algoritmos mais rápido e preciso dos que existem na literatura, há algumas limitações. Inicialmente o *DenStream*, assim como o *CluStream*, precisa observar um conjunto inicial de dados de fluxo antes de montar a primeira estrutura *online*, além disso, há muitos parâmetros a serem definidos e o número total de potencial-*mc* e outlier-*mc* é variável durante a execução e sensível ao parâmetro μ.

2.4 Medidas de avaliação

Para comparar o desempenho da metodologia proposta nesta tese com outros algoritmo da literatura serão utilizadas três medidas: *silhouette*, *purity* e *accuracy* (PUSCHMANN et al., 2017; MANSALIS et al., 2018; MOULTON et al., 2019; SHAO et al., 2018; CARNEIN; TRAUTMANN, 2019). Essas três medidas são bastante utilizadas em trabalhos recentes relacionados ao agrupamento em fluxo de dados. A primeira medida utiliza critérios internos para avaliar a consistências dos *clusters* formados, as duas últimas utilizam critérios externos, já que será possível o acesso aos verdadeiros rótulos dos objetos que estão sendo avaliados.

Silhouette (ROUSSEEUW, 1987) é um método que avalia uma partição em *clusters* com critérios internos. Esta medida é calculada para cada um dos objetos do conjunto de dados e obtêm-se um valor geral de avaliação fazendo-se uma média aritmética entre esses valores. A *silhouette* mede quão similar e coeso um objeto está em comparação com o seu próprio *cluster* e quão distante ele está do *cluster* mais próximo que não seja o seu próprio. O valor da medida varia entre -1 e $+1$, sendo que quanto maior, melhor.

Inicialmente, calcula-se a distância média a_i de um objeto $\vec{x}^i$ dos outros objetos pertencentes ao mesmo *cluster* C_i^x, conforme Equação 2.1, sendo $d(.,.)$ qualquer medida de distância, como, por exemplo, a euclidiana.

$$a_i = \frac{1}{|C_i^x| - 1} \sum_{\vec{x}^j \in C_i^x, i \neq j} d(\vec{x}^i, \vec{x}^j) \tag{2.1}$$

Ainda considerando o objeto $\vec{x}^i$, após o cálculo da distância média a_i, calcula-se qual a

menor distância média b_i do objeto para com os outros *clusters* da partição sem ser o seu próprio *cluster*. Este cálculo é feito segundo Equação 2.2.

$$b_i = \min_{k \neq i} \frac{1}{|C_k^x|} \sum_{\vec{x}^k \in C_k^x} d(\vec{x}^i, \vec{x}^k) \tag{2.2}$$

Por fim, na Equação 2.3 pode ser visto como calcular o valor final de *silhouette* s_i do objeto $\vec{x}^i$.

$$s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}} \tag{2.3}$$

Outra medida que será utilizada na avaliação do desempenho e que é bastante citada em outros trabalhos é a *purity* que calcula qual a porcentagem dos objetos em cada *cluster* pertencente ao rótulo dominante. O rótulo dominante C_i^d de um *cluster* C_i^x é aquele ao qual a maioria dos elementos do *cluster* pertence de fato. Assim, o melhor resultado de pureza é 100% quando todos os elementos de todos os *clusters* pertencem aos respectivos rótulos dominantes e o número de *clusters* é igual ao número de rótulos. A Equação 2.4 contém o cálculo da pureza, em que n_C é a quantidade de *clusters* e k é a quantidade de rótulos.

$$pureza = \frac{\sum_{i=1}^{n_C} \frac{|C_i^d|}{|C_i^x|}}{k} \times 100\% \tag{2.4}$$

A *purity* é uma ótima medida para verificar o desempenho de agrupamento, mas quando o número de *clusters* formados é diferente do número de rótulos de fato dos dados que estão sendo analisados, o valor pode acabar distorcido. Por exemplo, caso o número de *clusters* n_C seja maior, mas dois ou mais *clusters* sejam *puros* do ponto de vista da medida, ou seja, tenham somente objetos que pertençam ao rótulo dominante, a pureza terá um valor maior que 100%. Logo, para poder fazer uma medição melhor da acurácia dos *clusters* sem que o número de *clusters* formados interfira no resultado será utilizada a medida de *accuracy* que é calculada segundo Equação 2.5. Essa medida captura quão corretos estão os *clusters* mas ignora caso haja mais *clusters* que rótulos.

$$accuracy = \frac{\sum_{i=1}^{n_C} |C_i^d|}{\sum_{i=1}^{n_C} |C_i^x|} \times 100\% \tag{2.5}$$

A utilização dessas três medidas tem como objetivo analisar três aspectos dos *clusters* formados. O valor de *silhouette* analisa quão coesos são os objetos dentro dos *clusters*. A pureza mede se os *clusters* formados contém objetos que são homogêneos quanto ao rótulo, mas esse valor pode ser distorcido caso haja mais *clusters* do que rótulos, dessa forma, usa-se a medida *accuracy* para observar se, apesar de haver mais *clusters* do que rótulos, se os *clusters* são homogêneos.

3 REDES COMPLEXAS

Neste capítulo, são definidas algumas propriedades matemáticas das Redes Complexas úteis para os experimentos apresentados. Além disso, são apresentadas algumas considerações sobre construção e manutenção de uma Rede Complexa, detecção de comunidades e outras medidas. Ao fim, são apresentados algoritmos de agrupamento em fluxo de dados que são baseados em redes complexas da literatura.

3.1 Propriedades matemáticas

Os grafos ponderados e também as redes complexas são representados matematicamente por três conjuntos V, E e W que são o conjunto dos vértices, o conjunto das arestas e o conjuntos dos pesos, respectivamente. O conjunto dos vértices V reúne as representações dos objetos de um sistema, sendo $V = \{v_1, v_2, v_3, \ldots, v_i, \ldots, v_n\}$, onde cada vértice v_i possui um índice i único (TRUDEAU, 1993).

O conjunto das arestas E reúne as relações existentes entre os vértices e, por consequência, entre os objetos representados pelo vértice. Quando existe uma relação entre v_i e v_j é dito que eles são vértices *adjacentes*, portanto, o conjunto das arestas é representado por $E = \{(i,j)|v_i \text{ e } v_j \text{ são adjacentes}\}$ (BOLLOBÁS, 1998; FEOFILOFF et al., 2011).

No conjunto, se $(i,j) \in E$ não implicar em $(j,i) \in E$, então é dito que se trata de uma rede *direcionada*, ou seja, implica que apesar de haver uma relação de v_i em v_j, não necessariamente essa relação é simétrica, caso contrário, sempre que existir a relação (i,j) então a relação (j,i) também existe e as duas são consideradas como uma aresta só, uma relação recíproca (DIESTEL, 2005; ASUR et al., 2009).

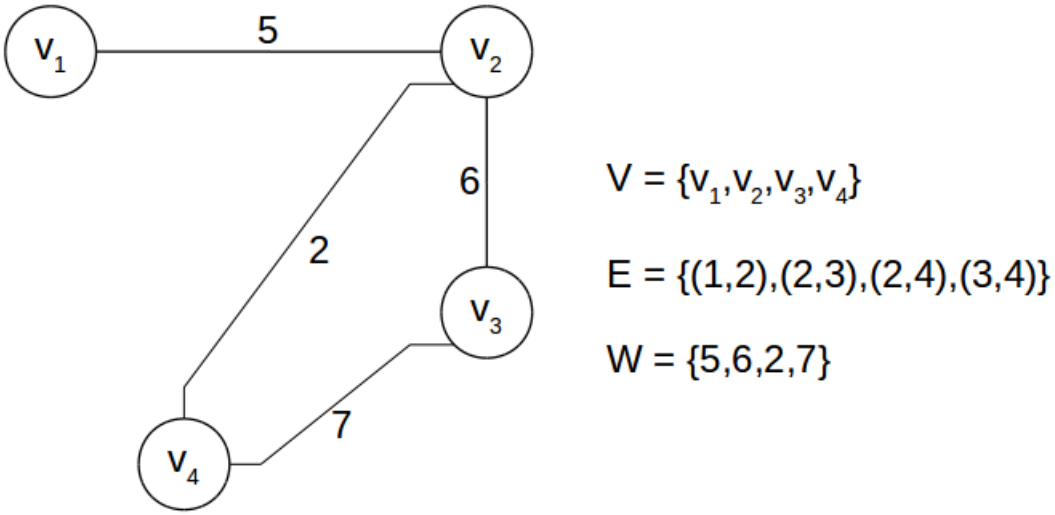
O conjunto dos pesos W armazena quão "forte" é uma relação dentro de uma rede e contém valores associados com todas as arestas da rede, sendo representado por $W = \{w_{ij}|(i,j) \in E\}$ (BOLLOBÁS, 1998; FEOFILOFF et al., 2011). Na Tabela 3.1 são resumidos os termos importantes referentes à representação matemática de uma Rede Complexa e na Figura 3.1 é mostrada uma apresentação gráfica de uma rede complexa e dos conjuntos que a formam.

Dentre as propriedades matemáticas, há aquelas que são formadas pelas informações fornecidas pelos conjuntos como, por exemplo, o grau k_i de um vértice v_i, que é a quantidade de arestas que têm o índice i como membro da tupla. A partir do cálculo dos graus de todos os vértices é possível obter ainda o grau mínimo $k_{min} = \min_{i=1}^{n} k_i$, grau máximo $k_{max} = \max_{i=1}^{n} k_i$ e o grau médio $\langle k \rangle = \frac{1}{n} \sum_{i=1}^{n} k_i$ da rede (NEWMAN; GIRVAN, 2004; FORTUNATO, 2010; ORMAN et al., 2012).

Tabela 3.1 - Conjuntos e elementos de uma Rede Complexa.

Símbolo	Significado
V	conjunto dos vértices
v_i	vértice v_i que possui índice i
E	conjunto das arestas
(i, j)	aresta entre os vértices v_i e v_j
W	conjunto dos pesos
w_{ij}	peso da aresta (i, j)

Figura 3.1 - Representação gráfica de uma rede complexa e seus conjuntos.



Rede com quatro vértices v_1, v_2, v_3 e v_4, quatro arestas $(1,2)$, $(2,3)$, $(2,4)$ e $(3,4)$, com pesos 5,6,2 e 7, respectivamente. Os valores fazem parte dos conjuntos V, E e W.

Fonte: Produção da Autora.

Outra propriedade das redes são os *caminhos*, que são sequências alternadas de vértices e arestas, começando e terminando em um vértice. A alternância entre vértices e arestas é tal que o índice i de um vértice v_i de um caminho é sempre um dos membros da tupla (i, j) que representa a aresta seguinte e implica que o próximo vértice do caminho é o v_j (RODRIGUES, 2007).

Uma rede é dita *conexa* se existe caminhos entre todos os pares de vértices da rede, caso contrário, é uma rede *desconexa*. Um caminho que inicia e termina no mesmo vértice é chamado *ciclo*, uma rede desconexa que não contém ciclos é chamada *floresta*, caso a

rede seja conexa, então é chamada *árvore* (BOLLOBÁS, 1998; FAN et al., 2007).

3.2 Modelos de redes

Ao longo da segunda metade do século XX, a modelagem de sistemas reais em grafos, ou seja, redes complexas, foram introduzidas na literatura. O primeiro modelo foi sugerido por Erdos e Rényi em 1959 (ERDOS; RÉNYI, 1959; ERDOS; RÉNYI, 1960), e que ficou conhecido por *grafos aleatórios* ou modelo *ER*.

Nesse modelo existem dois parâmetros, o número total de vértices n e uma probabilidade p. Cada par de vértices é conectado de acordo com essa probabilidade p independente dos outros vértices(COSTA et al., 2007; FORTUNATO, 2010).

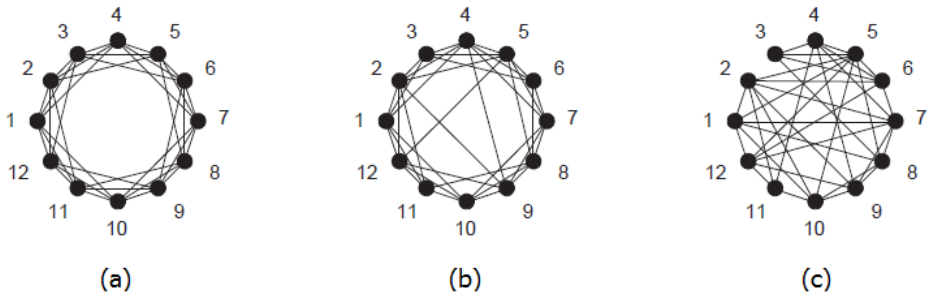
Uma rede pode ter como característica um *menor caminho médio* bem pequeno. Esse efeito é conhecido como *small-world*, introduzido em 1967 por Stanley Milgram (MILGRAM, 1967) e são extensões de *redes regulares*, que são um tipo especial de modelo onde cada vértice só se conecta com k vizinhos mais próximos. É um modelo muito usado na representação de sistemas geográficos, onde só existe ligação entre vértices que são vizinhos geograficamente.

Em 1998, Wattz e Strogatz observaram que a presença de ciclos de ordem três (triângulos) em sistemas reais eram muito maiores do que no modelo aleatório proposto por Erdos e Rényi (WATTS; STROGATZ, 1998). Esse fato mostrou que os sistemas reais, na verdade, não são completamente aleatórios e que existe algum tipo de lei de formação por trás da formação destes (RODRIGUES, 2007). Esse modelo ficou conhecido como *small-world de Watts-Strogatz*, ou simplesmente modelo *WS*.

Para obtê-lo deve-se: (1) começar com uma rede regular de tamanho N com k vizinhos; (2) cada aresta deve ser redirecionada para qualquer outro vértice na rede seguindo uma probabilidade p mas obedecendo duas restrições: nenhum vértice pode se ligar com ele mesmo, e não pode haver mais de uma conexão entre um par qualquer de vértices da rede (WATTS; STROGATZ, 1998; BARABÁSI; ALBERT, 1999).

Na Figura 3.2, são mostrados alguns exemplos do modelo. Quando $p = 0$ a rede se comporta como uma rede regular e o menor caminho médio cresce linearmente com N. Quando $p = 1$ o sistema se torna um grafo randômico e o menor caminho médio cresce proporcional ao logaritmo de N (modelo Erdos e Renyi).

Figura 3.2 - Modelo Wattz e Strogatz (WS).



Uma rede do tipo anel com $N = 12$ e $k = 6$, ou seja, contém 12 vértices e cada um deles tem 6 arestas, portanto, grau 6. Na Figura 3.2: (a) a rede tem o valor $p = 0$, então nenhuma aresta é redirecionada, logo se comporta como uma rede regular; (b) a rede tem valor $p = 0.15$, algumas arestas são redirecionadas e o caminho médio diminui; (c) o valor $p = 1$ faz a rede se comportar como uma rede do modelo ER.

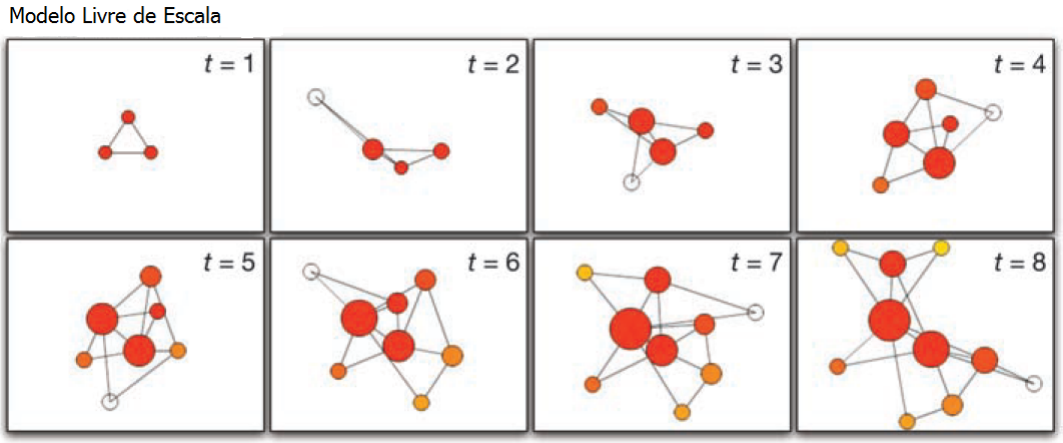
Fonte: Adaptada de Viana (2007).

Tanto no modelo proposto por Erdos e Renyi, como no proposto por Wattz e Strogatz, o número de vértices é fixo, além do que a probabilidade de quaisquer dois vértices se conectarem é aleatória e uniforme. Até 1999, os sistemas eram descritos utilizando um desses dois modelos, e na falta de dados ou mesmo *hardware* capaz de tal processamento, eles não puderam ser testados de fato no mundo real (BARABÁSI; ALBERT, 1999).

A partir de 1999, cada vez mais dados e capacidade de processamento foram incorporados nos estudos, e Barabási e Albert (BA) (BARABÁSI; ALBERT, 1999) demonstraram que a probabilidade $P(k)$ que um vértice na rede esteja conectado a outros k vértices segue uma *lei de potência*, descrita como $P(k) \sim k^{-\gamma}$.

Barabási e Albert argumentaram que existem dois aspectos genéricos nos sistemas que eles observaram que não estão incorporadas aos dois modelos apresentados anteriormente. Primeiro, a maioria dos sistemas são *abertos*, ou seja, nós podem ser adicionados (e também retirados), ao contrário, como já foi citado, dos modelos ER e WS, onde o número de vértices é fixo.

Figura 3.3 - Modelo Barabási e Albert (BA).



O nascimento de uma rede livre de escala. No tempo $t = 1$, os três nós iniciais estão conectados. No tempo $t = 2$ um quarto nó (branco) é adicionado e neste ponto esse nó tem igual probabilidade de se conectar com qualquer um dos três nós já presentes no sistema. No tempo $t = 3$ um quinto nó é adicionado mas dessa vez o nó tem probabilidade maior de se conectar com os vértices representados por círculos maiores, o tamanho do círculo no desenho é proporcional ao grau do nó. O processo continua até o tempo $t = 8$ quando atinge a configuração final deste exemplo.

Fonte: Adaptada de Barabási (2009).

Segundo, que as tais redes exibem um comportamento chamado *ligação preferencial*, que significa que um novo vértice ao ser adicionado a uma rede já existente tem maior probabilidade de ser ligado a um vértice que já tem um grande número de conexões. O modelo é baseado em dois passos (BARABÁSI et al., 2000):

a) *Crescimento*: Iniciando com um pequeno número N_0 de vértices, a cada passo adicionar um novo vértice com $m \leqslant N_0$ arestas, que serão conectadas aos vértices que já estão presentes no sistema.

b) *Ligação Preferencial*: Para decidir com quais vértices da rede o novo vértice irá se conectar, assume-se que a probabilidade $\Pi(k_i)$ de que o novo vértice se conecte ao vértice i depende da conectividade k_i deste vértice, tal que:

$$\Pi(k_i) = k_i / \sum_j k_j \tag{3.1}$$

Nas redes geradas pelo modelo BA, ou também chamados de modelos livre de escala, os vértices mais conectados, tendem a receber mais conexões. Após algum tempo, é

possível notar a existências de vértices centrais na rede, chamados *hubs*. Na Figura 3.3, são mostradas o surgimento desses *hubs* na evolução da rede. No tempo $t = 8$ percebe-se a existência de círculos mais bem relacionados que os restantes, representados na figura por círculos maiores pois o tamanho do círculo é proporcional ao grau do vértice.

Graças ao *crescimento* e à *ligação preferencial*, é observado um processo conhecido como *o rico fica mais rico* (em inglês, *rich-get-richer*). Nele, os vértices mais bem conectados adquirem ainda mais conexões do que aqueles que são pouco conectados, levando a concentração das conexões em apenas alguns nós, os *hubs* (BARABÁSI, 2009).

3.3 Construção e manutenção de estruturas do tipo rede complexa

Ao se construir uma rede para modelas uma base de dados, espera-se que a topologia desta rede seja uma representação dos dados dispostos no espaço $\mathbb{R}^n$. Nesta seção, cinco métodos de geração de redes serão descritos. O primeiro passo para todos eles é a criação de uma matriz M_{sim}. Tal matriz armazena as similaridades, ou o inverso das distâncias entre os objetos da base de dados original. Ela é por definição quadrada, simétrica e tem diagonal igual a zero. A matriz M_{sim} pode ser construída como na Equação 3.2:

$$M_{sim}[i,j] = \begin{cases} 1/d(i,j), & \text{se } (i,j) \in E \text{ e } d(i,j) \neq 0 \\ \infty, & \text{se } d(i,j) = 0 \\ 0, & \text{se } (i,j) \notin E \end{cases} \tag{3.2}$$

onde $d(i,j)$ é a distância entre os objetos. Assim, quanto menor a distância entre i e j, mais similares eles são.

Tabela 3.2 - Uma suposta M_{sim} para uma base de dados com cinco elementos.

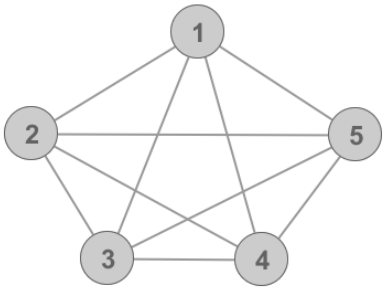
	x_1	x_2	x_3	x_4	x_5
x_1	0	0.2	0.45	0.95	0.8
x_2	0.2	0	0.7	0.3	0.4
x_3	0.45	0.7	0	0.55	0.15
x_4	0.95	0.3	0.55	0	0.6
x_5	0.8	0.4	0.15	0.6	0

O segundo passo em comum para os métodos é a criação de um vértice para representar cada objeto da tabela na rede. Após isso, cada método difere quanto à inserção de arestas.

Supondo uma situação simples, na qual a base de dados contém apenas cinco objetos, e a M_{sim} calculada, utilizando a distância euclidiana, seja a descrita na Tabela 3.2. A forma como os métodos geram uma rede a partir da dada M_{sim} e o resultado obtido são descritos na lista a seguir (ZHU, 2005b; ZHU, 2005a; JEBARA; SHCHOGOLEV, 2006):

- **Método 1 - Rede Totalmente Conectada**: Neste método é criado um grafo ponderado completo, ou seja, com uma aresta entre todo par de vértice, sendo que os pesos das arestas são um valor proporcional ao valor da similaridade contido na M_{sim}. Este método é depreciado já que exige um custo computacional elevado caso a base de dados original seja grande, pois a rede será muito densa. Na Figura 3.4 é ilustrada a rede gerada com este método.

Figura 3.4 - Rede gerada com o método 1 - Rede totalmente conectada.

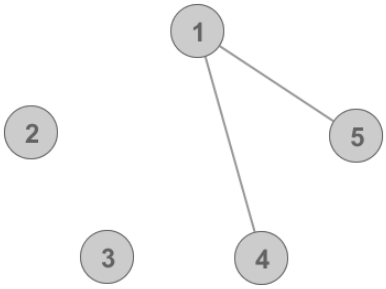


Rede gerada com o método *totalmente conectada*, onde todos os vértices estão ligados por uma aresta que contém peso proporcional ao valor de similaridade entre os objetos. O método não é muito utilizado na prática dada a quantidade de arestas que são incluídas que comprometem o armazenamento eficiente da estrutura.

Fonte: Produção da Autora.

- **Método 2 - Redes ε-*cut***: Neste método um limiar ε é definido e somente os pares que contenham similaridade igual ou superior a esse limiar terão uma aresta ligando-os. Este método tem como vantagem poder ser feito de forma eficiente, dependendo da linguagem de programação utilizada. No entanto, ele não garante a inexistência de vértices isolados, o que pode ser um problema em alguns contextos. Na Figura 3.5 é ilustrada a rede gerada com este método, considerando o caso não-ponderado.

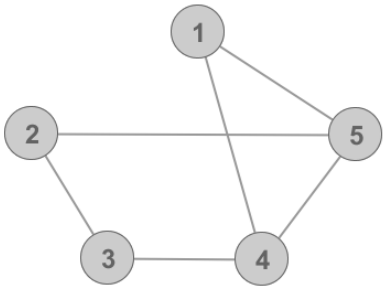
Figura 3.5 - Rede gerada com o método 2 - Redes ε-*cut*.



Rede gerada com o método ε-*cut* a partir de M_{sim} com limiar $\varepsilon = 0.75$. No exemplo, o método forma uma rede desconexa, o que nem sempre é interessante para a estrutura, mas a maior desvantagem do método é precisar definir um valor adequado para ε.

Fonte: Produção da Autora.

Figura 3.6 - Rede gerada com o método 3 - Redes kNN.



Rede gerada com o método kNN a partir de M_{sim} com $k = 2$. Esse método tem como maior vantagem a simplicidade de implementação, no entanto, é preciso escolher o valor de k adequadamente. Além disso, esse método não evita a formação de *hubs*.

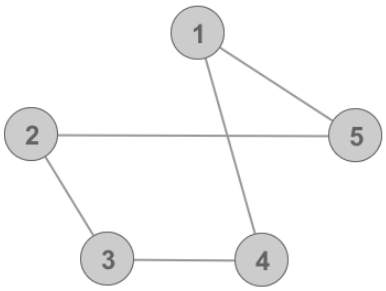
Fonte: Produção da Autora.

- **Método 3 - Redes kNN**: Tais redes são obtidas resolvendo-se o problema

dos k-vizinhos mais próximos para todo vértice na rede. Ou seja, para cada vértice, serão escolhidos os k vértices que representam os objetos que têm maior valor de similaridade na matriz M_{sim} e arestas serão adicionadas entre eles, ponderadas ou não. Este método garante que todo vértice terá pelo menos k-vizinhos, mas pode ser que alguma conexão, apesar de um valor alto de similaridade, seja o vizinho $k+1$ e acabe por não entrar na estrutura final da rede. Este método também pode formar *hubs* na rede, caso um dos vértices seja o vizinho mais próximo da maioria dos outros vértices, ele ficará com um grau muito maior que o resto. Na Figura 3.6 é ilustrada a rede gerada com este método.

- **Método 4 - Redes *b-matching***: Este método é parecido com o Método 3, mas ao invés de garantir que todos os vértices terão pelo menos grau k, ele garante que todo vértice terá exatamente b conexões, ponderadas ou não. Ele evita que *hubs* sejam criados na rede. *Hubs* são vértices que concentram muitas conexões, ou seja, que representam objetos que estão em uma posição central em uma nuvem de objetos no espaço de dados e que acabam transferindo esta característica para a rede. Dependendo do propósito da rede, os *hubs* podem ser necessários, até mesmo desejados, o que faria com que este método não fosse o ideal. Na Figura 3.7 é ilustrada a rede gerada com este método.

Figura 3.7 - Rede gerada com o método 4 - Redes *b-matching*.

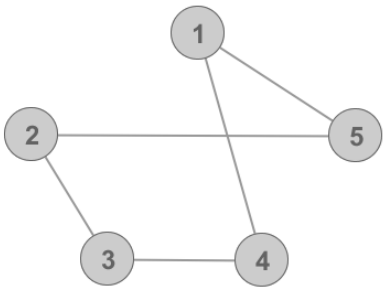


Rede gerada com o método *b-matching* a partir de M_{sim} com $b = 2$. O método evita os hubs, forma um grafo conexo, mas é preciso definir um b adequado.

Fonte: Produção da Autora.

- **Método 5 - Redes ε-*cut*+kNN**: Este método é a junção dos Métodos 2 e 3. Em um primeiro momento, o limiar ε faz o corte das arestas. Em um segundo momento, os vértices que ficarem isolados são ligados à rede utilizando o algoritmo kNN. Isto evita o problema do Método 2, pois nenhum vértice ficará isolado, e também evita o problema do Método 3, pois pares de vértices com similaridade alta terão arestas entre si, independentemente da quantidade de vizinhos que os vértices já tenham. A Figura 3.8 ilustra a rede gerada com este método.

Figura 3.8 - Rede gerada com o método 5 - Redes ε-*cut*+kNN.



Rede gerada com o método ε-*cut*+kNN a partir de M_{sim} com $\varepsilon = 0.75$ e $k = 1$. O método tem o mesmo resultado da rede do método anterior, mas depende da escolha de um ε e um k adequados.

Fonte: Produção da Autora.

Todos os métodos apresentados anteriormente têm em comum o fato que é necessário o cálculo da matriz M_{sim} previamente. Este é um procedimento que não é possível de ser feito no contexto dos fluxos de dados já que não é possível obter os valores de todos os objetos *a priori*. Além disso, nenhum desses métodos está preparado para tratar bases de dados dinâmicas, tendo que ser feitas adaptações.

Certamente alguns desses métodos podem se adaptar ao caso dinâmico, mas um problema permanece: o cálculo da matriz M_{sim} tem que ser refeita a todo novo elemento modificado, inserido ou removido da base de dados. No próximo capítulo será apresentado o método ε-*cut-ondemand*, que foi o método adaptado do Método 2 para o cenário de fluxos de dados.

3.4 Detecção de comunidades

Muitos sistemas complexos podem ser representados por redes onde as partes elementares do sistema são substituídas por vértices, e suas interações são representadas por arestas (NEWMAN; GIRVAN, 2004; FORTUNATO, 2010). Sistemas complexos são normalmente organizados em compartimentos que têm suas próprias regras ou funções.

Na representação por redes, esses compartimentos aparecem como grupos de vértices com uma alta densidade de conexão entre eles, enquanto conexões entre esses compartimentos são relativamente esparsas. Esses subgrafos são chamados comunidades, ou módulos, e ocorre na maioria dos sistemas (LANCICHINETTI; FORTUNATO, 2009).

A estrutura em comunidades é relativamente comum em vários sistemas, ou redes, e a habilidade de encontrar e analisar tais comunidades podem prover valorosa ajuda para entender e visualizar a estrutura global do sistema (NEWMAN; GIRVAN, 2004).

Detecção de comunidades também é importante na classificação dos vértices. Quando os módulos e suas fronteiras são identificados, vértices com uma posição central, ou seja, que possuem muitas arestas ligando-o à própria comunidade, têm papel fundamental no controle e estabilidade daquele grupo. Já se o vértice estiver numa posição de fronteira entre os módulos, ele representa um ponto de mediação entre dois grupos, e será o elemento que irá liderar as comunicações entre eles (FORTUNATO, 2010).

Uma medida que pode ser muito útil para a caracterização dos vértices na estrutura de comunidades é a *integração* (ou no termo em inglês *embeddedness*) que significa o quão bem o elemento está relacionado ou integrado à comunidade que pertence. Ou seja, quantos dos vizinhos de um elemento pertencem à mesma comunidade que ele.

$$e = k_{int}/k \tag{3.3}$$

O *grau interno* k_{int} de um nó é o número de conexões daquele nó que pertencem à mesma comunidade, e em oposição, o *grau externo* k_{ext} corresponde às conexões com vértices de outras comunidades (ORMAN et al., 2012). A *integração e* de um vértice pode ser definida como a razão entre seu grau interno e seu grau total, conforme Equação 3.3.

Quando um vértice tem valor de integração 1, isso significa que todos os seus vizinhos estão em sua própria comunidade. O valor da integração igual a 0 só é possível se o vértice em questão fosse uma comunidade isolada. Em redes do mundo real, a maioria dos vértices têm grau total baixo, e um valor de integração alto (ORMAN et al., 2012).

A *densidade* ρ de uma comunidade C^v é definida como a razão entre o número de arestas que existem dentro de uma comunidade, representada por l_C, e o número total de arestas que podem existir dentro da comunidade, ou seja, o número de arestas caso todos os vértices da comunidade fossem ligados entre si. Sendo n_C o número de vértices dentro da comunidade C^v, a equação da *densidade* é:

$$\rho = \frac{l_C}{\frac{n_C(n_C-1)}{2}} = \frac{2l_C}{n_C(n_C-1)} \tag{3.4}$$

Por definição, a densidade de uma comunidade deve ser maior que a densidade total do grafo. Outra medida que remete ao papel de um vértice dentro da rede é a *dominância*. Ela mede a existência de um *hub* dentro da comunidade. *Hubs* são nós conectados à grande maioria dos vértices da comunidade e a presença de um *hub* numa comunidade C^v pode ser avaliado usando a seguinte equação:

$$h(C^v) = \max_C(k_{int})/(n_C - 1) \tag{3.5}$$

O numerador é o grau interno mais alto encontrado dentro da comunidade C^v, e o denominador é o grau máximo interno possível, que seria o caso de um vértice que é conectado a todos os outros vértices da comunidade. O valor de $h(C^v) = 1$ significa que existe um *hub* conectado com todos os vértices existentes na comunidade.

Em geral encontrar a solução exata da divisão de uma rede em comunidades é considerado um problema *NP-completo* (FORTUNATO, 2010), pois pouco se sabe sobre a formação de uma estrutura de comunidades em um grafo. É incomum saber *a priori* em quantas comunidades uma rede pode ser dividida, ou qualquer outra indicação de qual vértice pertence a qual comunidade. A tarefa de detectar comunidades em um grafo não é trivial e pode ser abordada de diversas maneiras (LANCICHINETTI; FORTUNATO, 2009; CLAUSET et al., 2004).

O problema de detecção de comunidades envolve encontrar tais grupos mais conectados em uma dada rede e se tornou uma problema algorítmico popular nos últimos anos. O termo comunidade tem sido bastante usado na literatura em diferentes contextos e conotações, como por exemplo nas redes sociais. Ou seja, em redes complexas, as comunidades, também chamadas módulos ou *clusters*, são grupos de vértices que provavelmente compartilham propriedades em comum e/ou seguem regras similares dentro da rede (FORTUNATO, 2010).

Alguns dos algoritmos encontrados na literatura são o *Fast Greedy* (CLAUSET et al., 2004), *Infomap* (ROSVALL; BERGSTROM, 2008; ROSVALL et al., 2009), *Label Propagation* (RAGHAVAN et al., 2007) e *Walktrap* (PONS; LATAPY, 2006), para citar alguns exemplos. Cada um deles utiliza diferentes estratégias já conhecidas do aprendizado de máquina, mas utilizadas para um contexto bem diferente onde a análise é feita em cima de vértices e arestas e como eles se posicionam na rede e não somente dados reunidos em tabelas com várias informações como as bases de dados tradicionais (PALLA et al., 2007; LIN et al., 2008; FRANKE; GEYER-SCHULZ, 2009).

O *Fast Greedy* (CLAUSET et al., 2004) é baseado na medida de modularidade de uma partição em vértices de uma rede, a modularidade de uma partição mede quão boa é a divisão em grupos considerando a estrutura real da rede. O algoritmo então usa essa medida como uma função objetivo em um algoritmo de otimização, mas achar esse máximo global não é fácil já que o espaço de buscas é exponencial à quantidade de vértices na rede, dessa forma o algoritmo utiliza a heurística gulosa para alcançar o maior valor possível para a modularidade entre uma partição de vértice e uma estrutura em rede.

O algoritmo *Label Propagation* (RAGHAVAN et al., 2007) tem como ideia principal que dado um vértice v_j na rede que tem vizinhos v_{j1}, v_{j2}, ..., v_{jn} e que cada um desses vizinhos já tenha um rótulo de classificação, para decidir o rótulo de v_j bastaria verificar à qual classe pertence a maioria dos seus vizinhos. Sendo assim, o algoritmo inicia distribuindo um rótulo diferente entre os vértices e a cada passo escolhe uma ordem aleatória para visitar os vértices.

Em cada visita, o algoritmo verifica os rótulos dos vizinhos e se houver uma maioria, o vértice é também rotulado com o rótulo da maioria, caso ocorra um empate, o rótulo é escolhido aleatoriamente entre os rótulos que empataram. As visitas aos vértices se mantêm em ordem aleatória até que um equilíbrio na rede seja estabelecido.

Os vértices melhores conectados no algoritmo *Label Propagation* conseguem *disseminar* melhor seus rótulos, logo conseguem estabelecer quais são os outros vértices da rede que estão bem conectados entre si e quanto maior e mais denso o grupo, mais influência para agregar novos vértices. O algoritmo tem como desvantagem depender de muitos processos aleatórios e precisar ter que escolher quando parar de propagar os rótulos, ou seja, encontrar o equilíbrio.

O algoritmo *Walktrap* (PONS; LATAPY, 2006) é baseado na intuição de que quando um *random walker* "caminha" por uma rede, por seus vértices e arestas, esses tendem a ficar "presos" em comunidades, já que há uma alta densidade de conexões dentro delas.

Utilizando essa ideia, o algoritmo faz uma medição baseada nas estatísticas da ação de um *random walker* na rede. Ou seja, para cada par de vértices, o algoritmo calcula a média do tamanho de todos os caminhos percorridos pelo agente entre o par, e utiliza essa média como similaridade em um algoritmo de agrupamento hierárquico.

O algoritmo *Infomap* (ROSVALL; BERGSTROM, 2008; ROSVALL et al., 2009) também utiliza a ação de um *random walker* na rede para determinar sua estrutura em comunidades. Ele será utilizado na metodologia proposta nesta tese para determinar as comunidades de uma rede complexa e será explicado com mais detalhes a seguir.

3.4.1 *Infomap*

O *Infomap* (ROSVALL; BERGSTROM, 2008; ROSVALL et al., 2009) parte da atuação de um *random walker* na rede, e utilizando teoria dos códigos, especificamente o *código de Huffman*, propõe uma função objetivo a ser minimizada. O objetivo do algoritmo é encontrar uma partição para rede em que os vértices são agrupados em módulos, onde cada vértice pertence a um e somente um módulo. Essa partição deverá agrupar em um mesmo módulo vértices nos quais as informações, que são representadas pelo *random walker*, fluem rápida e facilmente (ROSVALL et al., 2009; ROSVALL; BERGSTROM, 2008).

O primeiro passo do algoritmo é calcular, a partir da atuação de um *random walker* q, uma variável aleatória P que é a distribuição da frequência de visitas que o elemento q faz aos vértices da rede num tempo infinito. Ou seja, $P = \{p_i\}, 1 \leq i \leq n$, sendo n o número de vértices da rede.

O código de *Huffman* tem por objetivo comprimir um código. Sendo esse código o caminho percorrido pelo elemento q dentro da rede, cada vértice deverá ter um código binário único associado, baseado na distribuição P. Vértices com maiores probabilidades de serem visitados deverão ter nomes menores, enquanto os poucos visitados terão nomes maiores.

Além disso, é levado em consideração que num caminho percorrido pelo elemento q, os vértices que deverão ser agrupados num mesmo módulo aparecerão em sequência e por longos períodos. Isso porque um *random walker* tende a ficar "preso" num módulo e raramente pular de um módulo para outro.

Mas para que essa sequência de *bits* que identificam o trajeto percorrido pelo *random walker* seja mínima, os módulos precisam ser apropriadamente identificados. Para isso, é introduzida uma função que calcula a quantidade média de *bits* necessários por passo para descrever o caminho do elemento q de acordo com uma partição M qualquer.

Essa função, chamada *map equation* e denotada por $L(M)$, calcula os *bits* necessários para codificar cada um dos módulos e seus correspondentes códigos de saída, assim como os *bits* necessários para codificar cada um dos vértices dentro dos módulos. Tendo as probabilidades calculadas inicialmente e representadas pela variável aleatória P, o teorema de *Shannon* pode ser invocado.

O teorema diz que quando usados x nomes para descrever x estados de uma variável aleatória X, o tamanho médio de compressão não pode ser menor que a entropia dessa variável: $H(X) = -\sum_1^x p_i log(p_i)$. Assim sendo, a função $L(M)$ é limitada por baixo pela entropia $H(P)$, conforme ilustra a Equação 3.6.

$$L(M) = q_{\curvearrowright} H(Q) + \sum_{i=1}^{m} p^i_{\circlearrowright} H(P^i) \tag{3.6}$$

A Equação 3.6 demonstra a *map equation*, e essa equação comprime dois termos: o primeiro é a entropia de movimento entre os módulos, e o segundo a entropia do movimento dentro dos módulos, incluindo o movimento de sair do módulo. Cada um dos módulos é associado a um peso que representa a frequência com que cada um desses movimentos ocorre de acordo com a partição M. Para detalhamento de cada um dos termos da função é recomendável a leitura do apêndice do artigo (ROSVALL; BERGSTROM, 2008).

A partir desse ponto o objetivo do algoritmo é minimizar a Equação 3.6. Para essa etapa, qualquer algoritmo de otimização da literatura poderia ser utilizado. No algoritmo *Infomap* a heurística escolhida foi a gulosa. No começo, cada vértice é associado a um módulo diferente, e dois a dois os módulos vão sendo fundidos de acordo com a junção que fará com que a $L(M)$ seja diminuída ao máximo, e esse processo se mantém até que o limite seja atingido. No *Infomap*, além do algoritmo guloso, o resultado é refinado pelo *simulated annealing* (KIRKPATRICK et al., 1983). Nessa segunda fase, alguns vértices podem ser trocados de módulos caso haja uma redução no valor da *map equation*.

3.5 Algoritmos baseados em redes complexas da literatura

O terreno em comum para os algoritmos baseados em estruturas do tipo rede complexa é que os *microclusters* são os vértices e que a estrutura é gerada e mantida segundo um conjunto de operações de funções que alteram sua topologia.

Os algoritmos *CNDenStream*, *SNCStream* e *SNCStream+* (BARDDAL et al., 2015; BARDDAL et al., 2019) são três algoritmos baseados no *DenStream* que utilizam uma estrutura

do tipo rede complexa para criar e rastrear a evolução dos *clusters* de um fluxo de dados. Os três algoritmos consideram que um *cluster* é formado quando um subgrafo desconexo é formado, ou seja, quando um grupo de vértices não tem arestas ligando-os a nenhum outro grupo.

O primeiro a ser proposto, o *CNDenStream* (*complex-network-DenStream*), é um algoritmo que emula o *DenStream* em uma rede complexa. Onde os vértices são os potencial-*mc* e os *outilier-mc* são mantidos em um *buffer* à parte. Assim como o *DenStream*, o *CNDenStream* precisa de um lote inicial de dados onde é executado o DBSCAN para formar os primeiros potencial-*mc* e *outlier-mc*.

Cada potencial-*mc* que deve ser incorporada à rede complexa é conectado com os ω potencial-*mc* mais similares (Método kNN). As arestas são inseridas com seus respectivos pesos (proporcional à distância), mas o uso do ωNN causa a formação de *hubs*, portanto, para evitar esse problema, o algoritmo faz um procedimento de *rewire*, ou seja, faz-se uma análise da distância entre o novo vértice com seus vizinhos de segundo grau[1]. Caso seja detectado que existem vértices mais próximos que os atuais vizinhos, então a aresta é redirecionada para o vértice mais próximo, de forma que o grau do novo vértice permaneça o mesmo.

O algoritmo considera que *clusters* são formados quando a rede se torna uma floresta, ou seja, um conjunto de redes que não tem conexão entre si.

O *SNCStream* (*social network clustering stream*) é uma versão aprimorada do *CNDenStream*, pois supera a fase inicial com o algoritmo DBSCAN ao aplicar o método ωNN desde o início da construção da rede. O *SNCStream+*, também uma evolução do mesmo algoritmo, muda a forma de fazer o *rewire*, que ao invés de usar a vizinhança de segundo grau completa, utiliza apenas aqueles que tenham tido suas arestas modificadas.

Nesta tese o algoritmo *SNCStream* será utilizado nos experimentos para ajudar a avaliar a metodologia que será apresentada no próximo capítulo. Além dos resultados de acurácia e desempenho que também serão comparados ao *CluStream* e o *DenStream*, o algoritmo SNCStream também tem métricas extras que são importantes avaliadores como o número total de arestas no modelo durante a execução, o número de comunidades e o número de vértices. A versão do SNCStream que será utilizada nos experimentos do Capítulo 5 foi incrementada pela autora para executar o *Infomap* na rede para obter os *clusters*.

[1] Vizinhos de Segundo Grau são todos os vértices que um vértice consegue alcançar usando um caminho de tamanho 2.

4 REDE PROTÓTIPO

A metodologia Rede Protótipo é apresentada neste capítulo. A metodologia é um conjunto de funções que operam uma estrutura do tipo rede complexa, que também é chamada Rede Protótipo, com o objetivo de agrupar objetos de um fluxo de dados. As funções reagem a eventos observados tanto na rede como no fluxo. A metodologia é apresentada função a função, na Seção 4.1, respeitando o fluxo de execução da mesma. Na Seção 4.2, são exemplificadas as operações das funções considerando um fluxo de dados simplificado. Na Seção 4.3, avaliam-se os efeitos no tempo de execução e na acurácia da metodologia considerando diferentes valores para os parâmetros. Ao fim, na Seção 4.4, são feitas algumas considerações sobre a metodologia e quais são seus pontos fortes e fracos comparando-a com outros métodos da literatura.

4.1 Descrição da metodologia

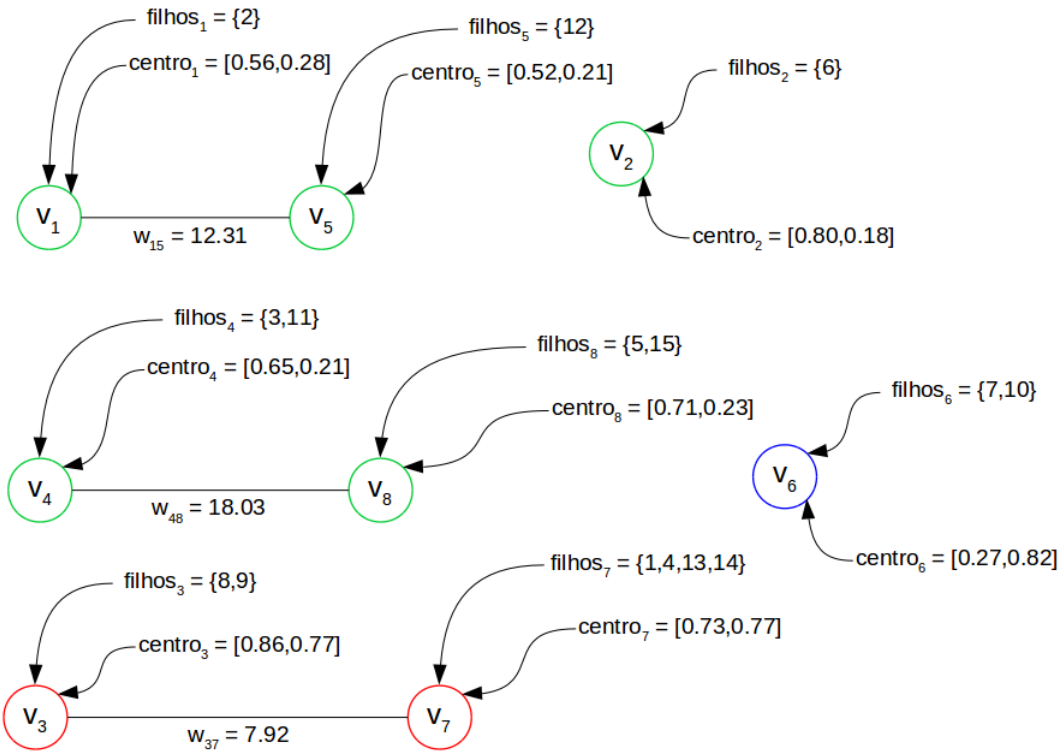
A metodologia Rede Protótipo (RP) age ao aplicar diversas funções em uma estrutura do tipo rede complexa, que será referida como estrutura RP ao longo deste capítulo. Isto significa que a estrutura RP é essencialmente um grafo com três conjuntos $RP = \{(V,E,W)\}$, onde o conjunto dos vértices é representado por $V = \{v_1, v_2, \ldots, v_{MAXV}\}$, o conjunto das arestas por $E = \{(i,j)|v_i \text{ e } v_j \text{ são adjacentes}\}$ e $W = \{w_{ij}|(v_i,v_j) \in E\}$ é o conjunto dos pesos das arestas.

O valor $MAXV$ é um parâmetro da metodologia que é definido de acordo com a memória disponível para armazenar a estrutura, e seu valor determina quantos vértices ao máximo devem ser mantidos ao mesmo tempo na rede. Cada vértice v_i tem um índice i único e mantém um conjunto de *timestamps* de objetos do fluxo $filhos_i$ e um vetor $centro_i$ que corresponde ao centroide dos atributos dos objetos em $filhos_i$. O centroide dos atributos é um vetor onde cada elemento é a média dos elementos dos vetores dos objetos na mesma posição. Cada aresta (i,j) tem um peso w_{ij} que é calculado conforme a similaridade entre os centros $centro_i$ e $centro_j$.

O cálculo da similaridade é estabelecido dependendo do contexto do problema ao qual a RP será aplicada, mas, de modo geral, ele é definido como a inversa da distância euclidiana, conforme a Equação 4.1. No caso particular de quando a distância entre os centroides é zero, ou seja, eles tem os mesmos valores de atributos no vetor, a similaridade é armazenada como uma constante Inf que corresponde a infinito.

$$w_{ij} = sim(v_i, v_j) = 1/dist(centro_i, centro_j) \tag{4.1}$$

Figura 4.1 - Exemplo gráfico de uma estrutura Rede Protótipo

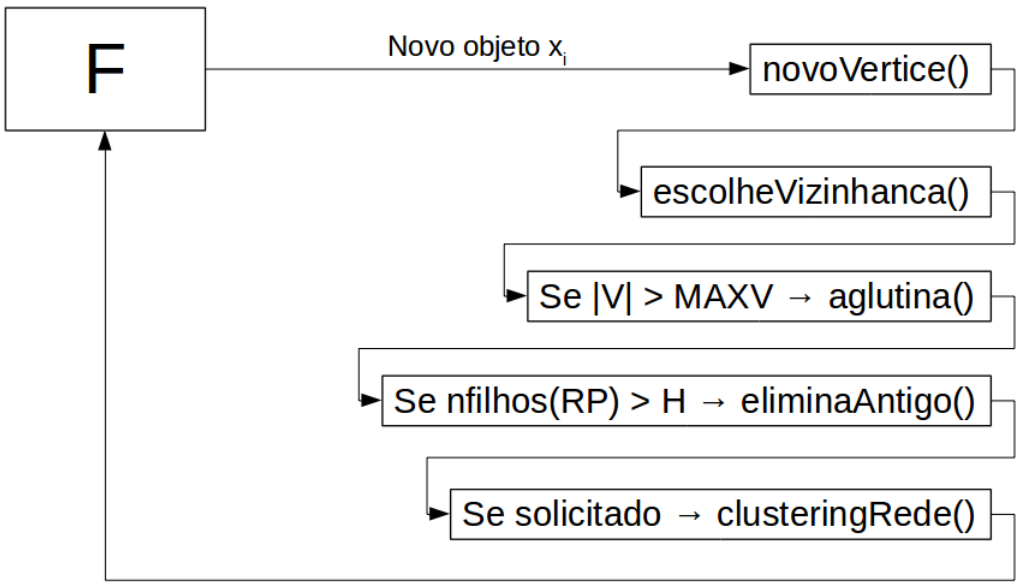


Exemplo de Rede Protótipo com 8 vértices e 15 objetos sendo representados. Cada objeto x^i contém um *timestamp* i e um vetor de atributos de dimensão 2 que podem ser observados na Tabela 4.1. Cada vértice v_i tem uma lista $filhos_i$ e um centróide $centro_i$ que armazenam informações pertinentes aos objetos representados. Além disso, cada aresta tem um peso w_{ij} que armazena a similaridade entre os vértices adjacentes, e cada vértice contém uma cor que representa as diferentes classes dos objetos, sendo azul a classe 1, vermelho a classe 2 e verde a classe 3.

Fonte: Produção da Autora.

Tanto a construção como a manutenção de uma estrutura RP seguem uma sucessão de funções que são desencadeadas com a observação de um novo objeto em um fluxo de dados F que está sob análise, conforme pode ser visto no fluxograma contido na Figura 4.2. Nas próximas seções, é detalhada a metodologia Rede Protótipo, ou ainda, o conjunto de funções para criar e manter uma rede complexa onde os vértices correspondem a *prototypes arrays*.

Figura 4.2 - Fluxograma das funções de construção e manutenção da estrutura Rede Protótipo.



Sequência de funções que descrevem o algoritmo Rede Protótipo, onde podem ser observadas suas cinco principais funções *novoVertice()*, *escolheVizinhanca()*, *aglutina()*, *eliminaAntigo()* e *clusteringRede()*, além de dois parâmetros *MAXV* e *H* que controlam mudanças na rede quando necessário.

Fonte: Produção da Autora.

4.1.1 Funções *novoVertice()* e *escolheVizinhanca()*

O ciclo da metodologia RP começa quando um novo objeto de um fluxo de dados F é observado. Inicialmente, cria-se um novo vértice v_j e adiciona-se a este vértice os atributos do objeto observado segundo as Equações 4.2 e 4.3.

$$filhos_j = \{i\} \tag{4.2}$$

$$centro_j = \vec{x}^i \tag{4.3}$$

Feita a adição do novo objeto $\vec{x}^i$ a um novo vértice v_j, este deve ser relacionado a outros vértices da rede, isto é, alguns vértices já contidos na rede devem ser escolhidos como vizinhos para v_j. Para tanto, compara-se o vetor $centro_j$, através do cálculo da similaridade, com os outros centroides da rede e, assim, uma matriz de similaridade será

criada. Nela aplica-se o algoritmo ε-*cut-ondemand* para a escolha dos vértices que serão adjacentes a v_j.

O algoritmo ε-*cut-ondemand* utiliza as similaridades calculadas para decidir o *threshold* ε_j segundo Equações 4.4, 4.5 e 4.6. Na Equação 4.4 é calculada a média μ_j entre todas as similaridade do novo vértice v_j e os vértices restantes no conjunto V da rede. Na Equação 4.5 é calculado o desvio-padrão ρ_j entre essas similaridades, para então na Equação 4.6 ser calculado o limite ε_j como a soma entre a média μ_j e o desvio-padrão ρ_j.

$$\mu_j = \frac{\sum_k^{k \in V} sim(v_j, v_k)}{(|V| - 1)}, \text{ tal que } j \neq k \tag{4.4}$$

$$\rho_j = \frac{\sqrt{\sum_k^{k \in V} (sim(v_j, v_k) - \mu)^2}}{(|V| - 1)}, \text{ tal que } j \neq k \tag{4.5}$$

$$\varepsilon_j = \mu_j + \rho_j \tag{4.6}$$

Após o cálculo do *threshold* ε_j, os vértices v_k que tiverem similaridade $sim(v_j, v_k) \geq \varepsilon_j$ são escolhidos como novos vizinhos do novo vértice v_j. Ou seja, analisa-se a similaridade entre o centroide $centro_j$ e o centroide $centro_k$, calculando-se a inversa da distância euclidiana entre eles, conforme Equação 4.1, e para cada vértice v_k, cuja similaridade com o novo vértice v_j seja maior que o limite ε_j, adiciona-se uma nova aresta (j,k).

Cada aresta (j,k) adicionada à estrutura RP deve ser associada a um peso w_{jk}. Este peso é definido como sendo a similaridade já calculada, tal que o peso w_{jk} recebe o valor da similaridade entre v_j e v_k, ou seja:

$$w_{jk} = sim(v_j, v_k).$$

Com esses valores e os parâmetros $MAXV$ (número máximo de vértices permitido na rede) e H (tamanho da janela deslizante), é verificado dois aspectos da rede: a) se o número de vértices $|V|$ está além do definido pelo parâmetro $MAXV$ e; b) se o número total de filhos $nfilhos(RP)$ é maior que o parâmetro H. No primeiro caso, se $|V| > MAXV$, aplica-se a função *aglutina()*, e, no segundo caso, se $nfilhos(RP) > H$, aplica-se a função *eliminaAntigo()*.

O parâmetro $MAXV$ é definido pelo usuário e indica qual o limite para a quantidade de vértices que podem ser mantidos ao mesmo tempo na rede. Já o parâmetro H faz parte do mecanismo de janela deslizante, que define que somente H objetos do fluxo de

dados F devem ser mantidos na estrutura RP ao mesmo tempo. Sendo assim, quando o número de vértices extrapola o valor do parâmetro $MAXV$, a função *aglutina()* mescla (ou aglutina, como o nome sugere) dois vértices em apenas um, dessa forma diminuindo em 1 a quantidade de vértices.

No entanto, esse procedimento de aglutinação não altera o número total de filhos que estão sendo representados, logo, caso esse valor esteja ultrapassando o parâmetro H, a função *eliminaAntigo()* elimina o vértice mais antigo da rede, eliminando junto os objetos que estão listados dentro de seus filhos.

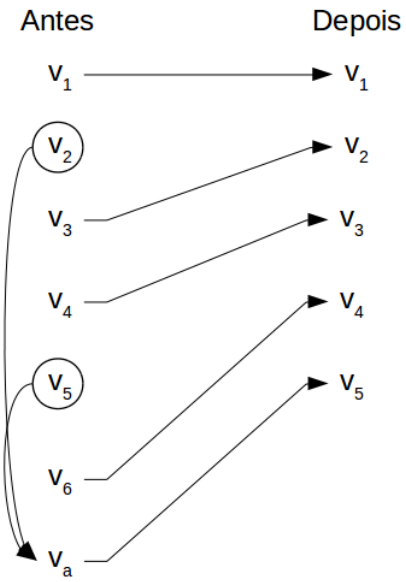
4.1.2 Função *aglutina()*

A aglutinação entre dois vértices v_i e v_j ocorre segundo os seguintes passos: os conjuntos $filhos_i$ e $filhos_j$ são unidos em um único, e os centros $centro_i$ e $centro_j$ passam pelo processo de cálculo de um novo centroide. Feito isso, os dois vértices são eliminados da rede junto com todas suas arestas e um novo vértice v_a portando o conjunto $filhos_a = filhos_i \cup filhos_j$ e o novo centróide $centro_a = centroide(centro_i, centro_j)$ é adicionado à rede seguindo os mesmos processos pelo qual passam novos objetos, inclusive escolhendo uma nova vizinhança.

A escolha de quais serão os vértices que passarão pela aglutinação é feita analisando os conjuntos E e W. A aresta $(i, j) \in E$, tal que $w_{ij} = \max(W)$, indica qual será o par de vértice que irá passar pela mescla. Essa escolha tem como objetivo aglutinar o par de vértices mais homogêneo possível quanto aos seus centros. Como o peso de cada aresta armazena o valor da similaridade entre seus vértices, a aresta com o maior peso indica o par de vértice mais próximo dentro da rede.

A eliminação de dois vértices e o acréscimo de um novo vértice afeta bastante a rede. Primeiro porque na eliminação, além dos vértices, as arestas que se ligavam a eles também são eliminadas e a inclusão do novo vértice aglutinado exige a inclusão de novas arestas. Geralmente o vértice v_a escolhe como vizinho os antigos vizinhos mais próximos dos antigos vértices v_i e v_j, que foram eliminados, já que o novo centroide $centro_a$ não é tão diferente dos centroides de v_i e v_j.

Figura 4.3 - Mudanças nos índices dos vértices causadas pela função *aglutina()*.



Nessa imagem supõe-se que os vértices v_2 e v_5 foram escolhidos para serem aglutinados. O vértice v_a é o resultado da aglutinação. Na primeira coluna são mostrados os índices antes da eliminação de v_2 e v_5, e a adição de v_a, e na segunda coluna são mostrado os índices após a execução dessas ações.

Fonte: Produção da Autora.

Outra mudança importante que a eliminação de dois vértices causa na rede é o ajuste dos índices dos vértices. Considerando como exemplo uma rede com 6 vértices $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$, caso o par (v_2, v_5) fosse ser aglutinado, os vértices v_i restantes, onde $i > 2$ ou $i > 5$, devem ser ajustados. Quando o índice i for maior que o valor 2 mas menor que o valor 5, será diminuído em 1, e quando o índice for maior ou igual a 5 será diminuído em 2. Finalmente, o novo vértice aglutinado v_a recebe o maior índice depois do ajuste. Na Figura 4.3 é apresentado esse exemplo para melhor entendimento. O ajuste dos índices dos vértices é útil e necessário principalmente porque dessa forma mantêm-se um controle da "idade" dos vértices na rede, que é uma informação necessária para a função *eliminaAntigo()*.

4.1.3 Função *eliminaAntigo()*

A função *eliminaAntigo()* considera os índices dos vértices para eliminar o mais antigo deles, ou seja, eliminar o vértice v_1. O objetivo é diminuir o número total de filhos $nfilhos(RP) = \sum_j^{j \in V} |filhos_j|$ para que a rede volte a respeitar o parâmetro H, para tanto, ao eliminar um vértice, elimina-se também a representatividade dos objetos que estavam em $filhos_1$, além de eliminar também todas as arestas do vértice v_1.

Toda essa mudança afeta a rede inicialmente ao eliminar o vértice v_1, que faz com que todos os vértices precisem ajustar seus índices, já que, nesse caso, todos os índices i são maiores que 1. Além disso, a ação afeta o conjunto das arestas E, o conjunto dos pesos W e os graus de alguns vértices. As arestas que são eliminadas fazem diminuir o tamanho dos conjuntos E e W, podendo alterar inclusive a aresta cujo peso é o maior do conjunto W. Outro efeito colateral da eliminação é que normalmente o vértice v_1, por ser o mais antigo da rede, tem exemplos de dados que ou são *hubs* defasados ou são objetos que isolaram-se em vértices com posições longínquas dos *hubs* e acabaram não participando de muitas aglutinações.

4.1.4 Função *clusteringRede()*

Por fim, seguindo o fluxograma na Figura 4.2, o último passo da metodologia Rede Protótipo é verificar se foi solicitado o agrupamento dos objetos representados pela estrutura, caso não, a rede fica a espera de uma nova observação no fluxo de dados.

Esse agrupamento é feito através da função *clusteringRede()*, onde inicialmente aplica-se à rede um algoritmo de detecção de comunidades que através das conectividades e densidades dos vértices e arestas divide os vértices em grupos conhecidos como comunidades. Definida as comunidades, a função *clusteringRede()* utiliza esse resultado para definir o agrupamento dos objetos na rede.

Uma comunidade é um grupo de vértices que possui maior densidade de arestas entre si do que entre os outros vértices restantes da rede, na metodologia RP, usa-se o algoritmo *Infomap* para fazer essa divisão. O algoritmo *Infomap* (ROSVALL; BERGSTROM, 2008; ROSVALL et al., 2009) utiliza *random walkers* que são agentes que caminham pelas arestas da rede e, através da avaliação do número de vezes que esses caminhantes passaram pelos mesmos trechos, determina-se quais grupos de vértices estão mais conectados na rede e, portanto, formam uma comunidade.

4.2 Exemplo

Nesta seção, são utilizadas as funções descritas nas seções anteriores para a análise de objetos de um fluxo de dados F e, assim, exemplificar todo o processo da metodologia Rede Protótipo para melhor entendimento.

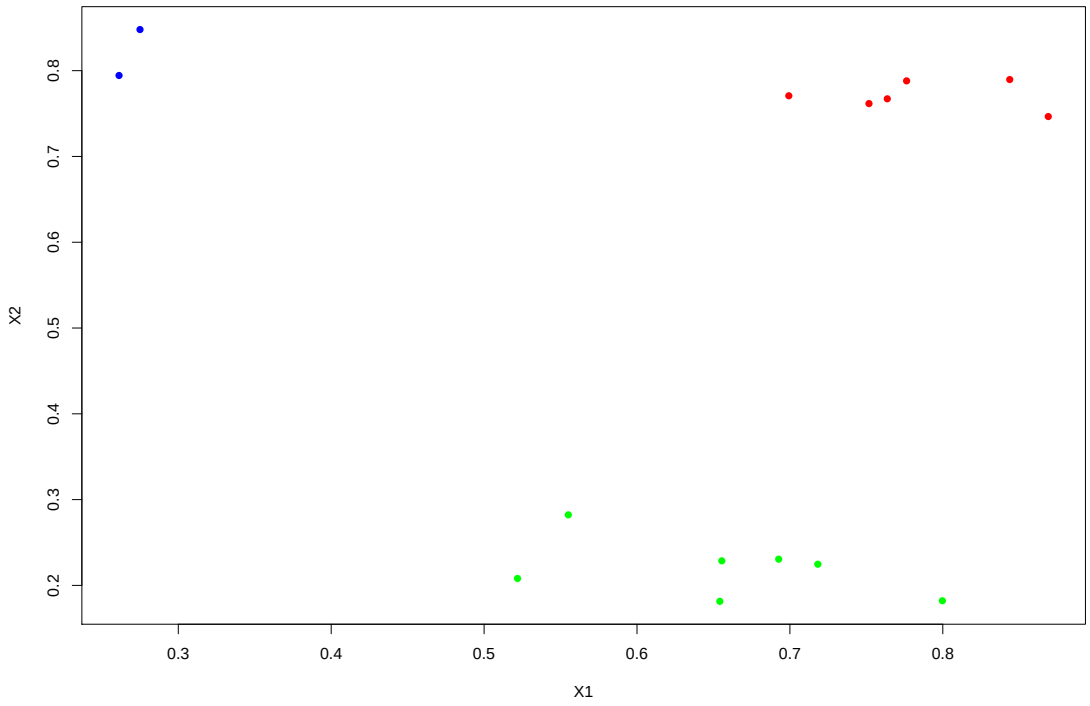
Na Figura 4.1 é exibida uma representação gráfica de uma RP, e esta é utilizada como exemplo para demonstrar a metodologia ao longo desta seção. Na Tabela 4.1 são reproduzidos os objetos do fluxo de dados F com seus respectivos atributos e classes que foram usados na construção da Rede Protótipo da Figura 4.1, e na Figura 4.4 são exibidos os objetos em um gráfico para melhor visualização.

Tabela 4.1 - Dados utilizados para construir a Rede Protótipo da Figura 4.1.

Objeto	Atributos		Classes
$\vec{x}^1$	0.76	0.77	Y_2
$\vec{x}^2$	0.56	0.28	Y_3
$\vec{x}^3$	0.65	0.18	Y_3
$\vec{x}^4$	0.78	0.79	Y_2
$\vec{x}^5$	0.72	0.22	Y_3
$\vec{x}^6$	0.8	0.18	Y_3
$\vec{x}^7$	0.27	0.85	Y_1
$\vec{x}^8$	0.87	0.75	Y_2
$\vec{x}^9$	0.84	0.79	Y_2
$\vec{x}^{10}$	0.26	0.79	Y_1
$\vec{x}^{11}$	0.66	0.23	Y_3
$\vec{x}^{12}$	0.52	0.21	Y_3
$\vec{x}^{13}$	0.75	0.76	Y_2
$\vec{x}^{14}$	0.7	0.77	Y_2
$\vec{x}^{15}$	0.69	0.23	Y_3

Supõe-se que os dados são gerados por um fluxo de dados F que contém três rótulos diferentes Y_1, Y_2 e Y_3 que nas imagens estão sendo representadas pelas cores azul, vermelho e verde, respectivamente. Cada rótulo corresponde a um conjunto de dados que tem uma distribuição gaussiana no espaço de dados de dimensão 2. Além disso, é considerado que os parâmetros da metodologia são definidos como $MAXV = 8$ e $H = 16$.

Figura 4.4 - Dados utilizados na construção da Rede Protótipo contida na Figura 4.1.



Dados da Tabela 4.1 exibidos em um gráfico para melhor visualização. As cores representam as respectivas classes dos pontos, sendo azul o rótulo Y_1, vermelho o rótulo Y_2 e verde o rótulo Y_3.

Fonte: Produção da Autora.

Tabela 4.2 - Resultados do cálculo da similaridade do novo vértice v_9 com os outros vértices da rede.

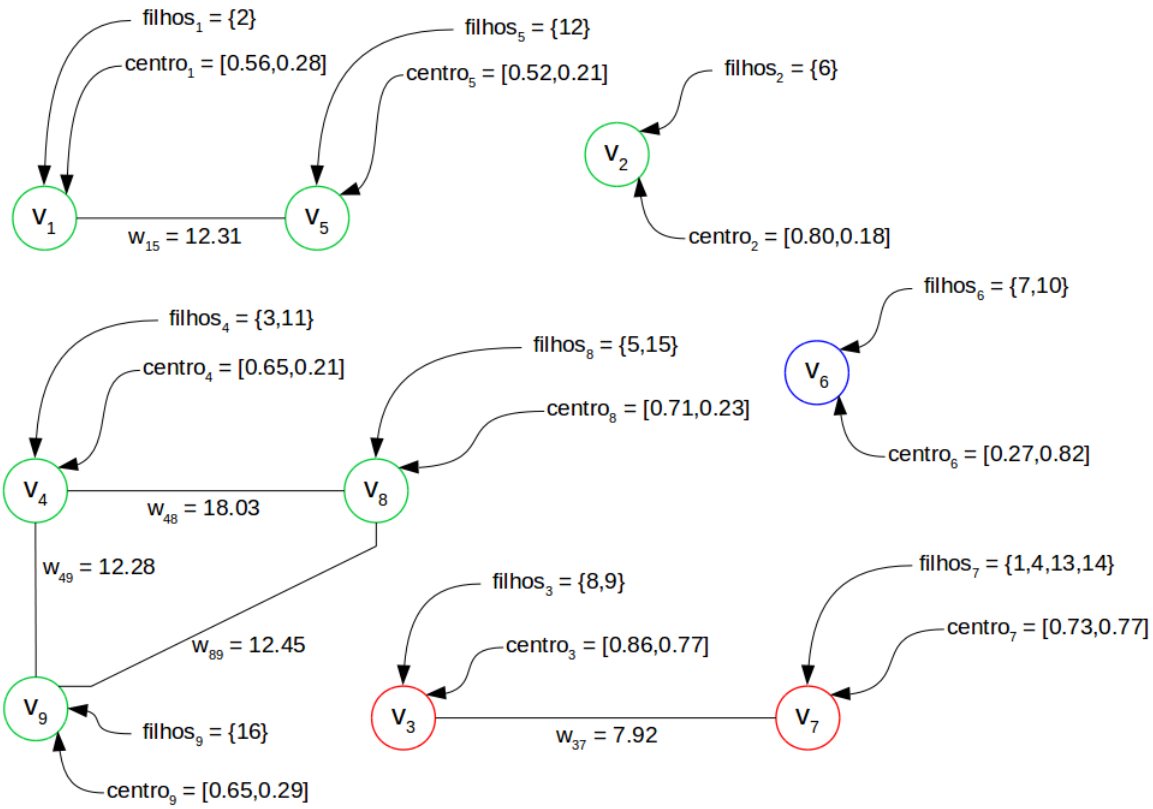
$sim(v_9, v_k)$	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
v_9	10.45	5.49	1.90	12.28	6.63	1.52	2.03	12.45

Para exemplificar a metodologia RP, utilizando os objetos e rede já apresentados nas Figuras 4.1 e 4.4, supõe-se a observação de um novo objeto $\vec{x}^{16}$ e faz-se uma descrição dos efeitos desse novo objeto na rede. Considerando, então, a rede que fora construída conforme os objetos da Tabela 4.1 foram sendo observados, supõe-se agora a observação

de um novo objeto $x^{16} = [0.65, 0.29]$ do rótulo Y_3.

Seguindo as instruções relatadas na Seção 4.1.1, primeiramente o objeto é colocado em um novo vértice, nesse caso o vértice v_9, onde $filhos_9 = \{16\}$ e $centro_9 = [0.65, 0.29]$. Após essa etapa, o centroide $centro_9$ é comparado com os outros centroides dos vértices já existentes na rede através do cálculo das similaridades, conforme Equação 4.1. Os resultados da comparação podem ser vistos na Tabela 4.2.

Figura 4.5 - Resultado da aplicação das funções *novoVertice()* e *escolheVizinhanca()* na RP usada como exemplo.



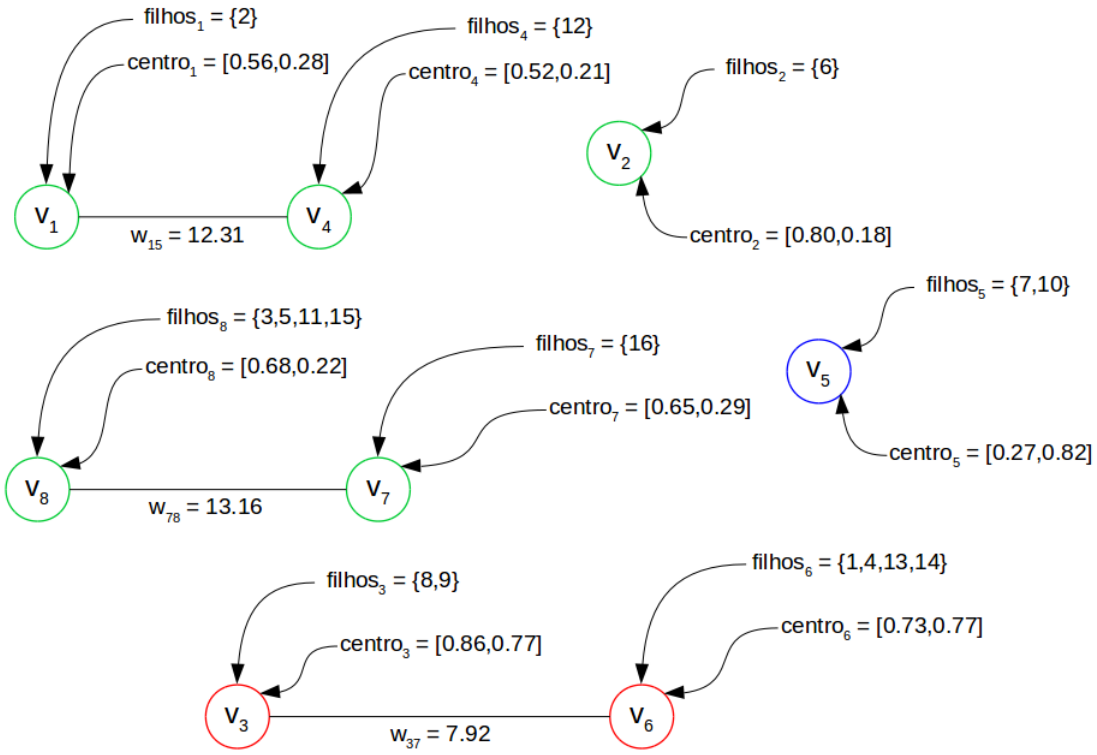
Rede Protótipo com 9 vértices depois da inclusão do objeto x^{16}, e com mais duas arestas depois de escolhida a vizinhança.

Fonte: Produção da Autora.

Com os valores das similaridades calculados, aplica-se o algoritmo ε-*cut-ondemand*. Em um primeiro momento, seguindo as Equações 4.4 e 4.5, o valor da média e desvio-padrão

são iguais a $\mu_9 = 6.60$ e a $\rho_9 = 4.65$, respectivamente. Logo, o valor do limite é $\varepsilon_9 = 6.60 + 4.65 = 11.25$ e, portanto, analisando as similaridades calculadas expostas na Tabela 4.2, são escolhidos os vértices que serão os vizinhos.

Figura 4.6 - Resultado da aplicação da função *aglutina()* na RP usada como exemplo.



Rede Protótipo com 8 vértices e 3 arestas depois da eliminação dos antigos vértices v_4 e v_8 e adição do novo vértice aglutinado v_8.

Fonte: Produção da Autora.

Os vértices v_4 e v_8 têm similaridade maior que o valor do limite, e, como resultado, são escolhidos como vizinhos do novo vértice v_9. A vizinhança do novo vértice com v_4 e v_8 é definida com a inclusão de duas novas arestas: uma entre v_4 e v_9, e outra entre v_8 e v_9, sendo que o peso das arestas é o valor das similaridades que já foram calculadas, onde $w_{49} = sim(v_4, v_9) = 12.28$ e $w_{89} = sim(v_8, v_9) = 12.45$. Na Figura 4.5 é possível ver como as inclusões do novo vértice e, consequentemente, das duas arestas modificam a rede.

Neste ponto da execução da metogologia RP, a rede exemplo da Figura 4.5 tem 9 vértices,

5 arestas e 16 filhos. Com esses valores e os parâmetros $MAXV$ e H, é verificado dois aspectos da rede: a) se o número de vértices $|V|$ está além do definido pelo parâmetro $MAXV$ e; b) se o número total de filhos $nfilhos(RP)$ é maior que o parâmetro H. No primeiro caso, se $|V| > MAXV$, aplica-se a função *aglutina()*, e, no segundo caso, se $nfilhos(RP) > H$, aplica-se a função *eliminaAntigo()*.

Considerando novamente a rede exemplo na Figura 4.5 e os valores dos parâmetros $MAXV$ e H como sendo 8 e 16, respectivamente, percebe-se que após a inclusão do objeto $\vec{x}^{16}$ a rede começou a possuir 9 vértices e, portanto, passou a violar o parâmetro $MAXV$. No entanto, isso não significou que a rede também passou a violar o parâmetro H, que continua sendo maior ou igual ao número de filhos. Portanto, a rede, segundo o fluxograma da Figura 4.2, passa pelo processo de aglutinação.

Tabela 4.3 - Arestas e Pesos da Rede Protótipo da Figura 4.5.

Arestas	Pesos
$(1,5)$	$w_{15} = 12.31$
$(3,7)$	$w_{37} = 7.92$
$(4,8)$	$w_{48} = \mathbf{18.03}$
$(4,9)$	$w_{49} = 12.28$
$(8,9)$	$w_{89} = 12.45$

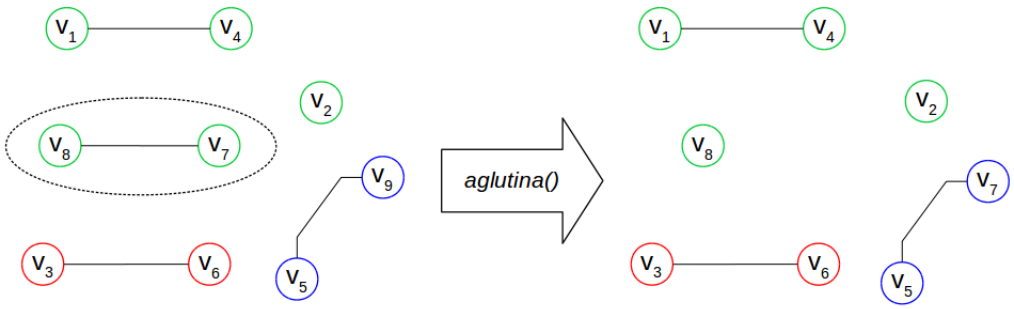
Seguindo as instruções relatadas na Seção 4.1.2, inicialmente observa-se o conjunto dos pesos das arestas e verifica-se qual contém peso $w_{ij} = \max(W)$. No caso da rede exemplo, a aresta com maior peso é a aresta $(4,8)$, conforme pode ser observado na Tabela 4.3, logo, os vértices a serem aglutinados são v_4 e v_8. Os filhos de v_4 são $filhos_4 = \{3,11\}$ e o centroide $centro_4 = [0.65, 0.21]$; os filhos de v_8 são $filhos_8 = \{5,15\}$ com centroide $centro_9 = [0.71, 0.23]$. É interessante notar que todos os objetos x^3, x^{11}, x^5 e x^{15} possuem o mesmo rótulo, conforme pode ser visto na Tabela 4.1.

O novo vértice aglutinado v_a tem $filhos_a = \{3,5,11,15\}$ e $centro_a = [0.68, 0.21]$. A eliminação dos dois vértices elimina também três arestas, a aresta $(4,8)$ entre eles próprios, como também as arestas $(4,9)$ e $(8,9)$, deixando v_9 sem vizinhos. Ajustando os índices dos vértices, a nova rede conterá 7 vértices depois da eliminação, e o v_a será incluído como v_8, finalizando o processo de aglutinação.

A inclusão do vértice aglutinado envolve escolher uma vizinhança similar ao método já relatado na Seção 4.1.1, portanto, aplicando o algoritmo ε-*cut-ondemand*. O resultado

pode ser visto na Figura 4.6 e, não por acaso, o novo vértice v_8, resultado da aglutinação entre os antigos v_4 e v_8, será vizinho do vértice v_7. Além disso, o antigo vértice v_9 e a aresta entre eles terá peso $w_{78} = 13.16$, o maior peso da rede.

Figura 4.7 - Resumo gráfico das configurações da rede antes e depois da execução da função *aglutina()* causada pela observação do novo objeto $\vec{x}^{17}$.



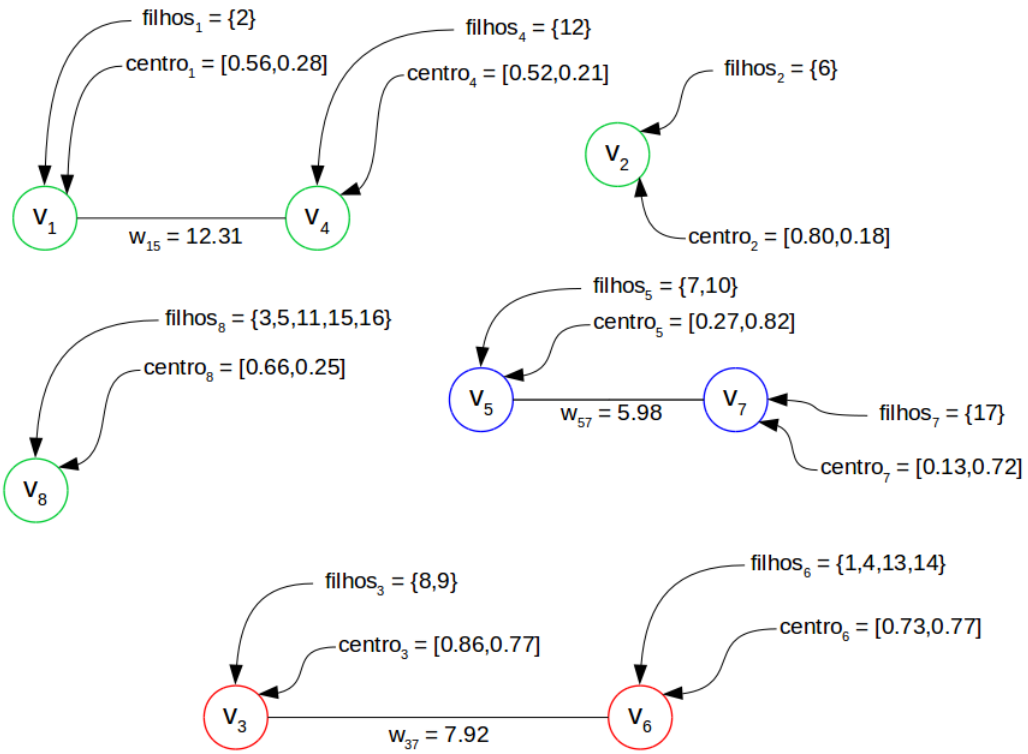
O antes e depois da execução da função *aglutina()* após observação do novo objeto $\vec{x}^{17}$. Antes a rede possuía nove vértices, portanto violando o parâmetro $MAXV$, e após a aglutinação dos vértices v_7 e v_8 e ajustes dos índices dos vértices, chega-se à rede à direita.

Fonte: Produção da Autora.

A rede exemplo da Figura 4.6 tem agora 8 vértices, 2 arestas e 16 filhos. Por enquanto nem o parâmetro $MAXV$ nem o parâmetro H estão sendo violados, mas a chegada de um novo objeto $\vec{x}^{17}$ faz a rede passar pela aglutinação novamente e depois ter que eliminar o vértice mais antigo.

O novo objeto $\vec{x}^{17} = [0.13, 0.72]$ possui rótulo Y_1 (azul) e, após calcular as similaridades e aplicar o algoritmo ε-*cut-ondemand*, uma aresta $(5,9)$ entre o novo vértice v_9 e o vértice v_5 é adicionada com peso $w_{59} = 5.98$, sendo $filhos_9 = \{17\}$ e $centro_9 = [0.13, 0.72]$. Essa inclusão leva a uma aglutinação do par mais próximo da rede que é o par (v_7, v_8) com peso $w_{78} = 13.16$, que leva a um novo ajuste de índices e termina em uma nova configuração onde a rede possui 8 vértices, 3 arestas e 17 filhos. Todo esse processo está ilustrado na Figura 4.7 e o resultado final da rede na Figura 4.8.

Figura 4.8 - Resultado da rede após observação do novo objeto $\vec{x}^{17}$ e execução da função *aglutina()*.



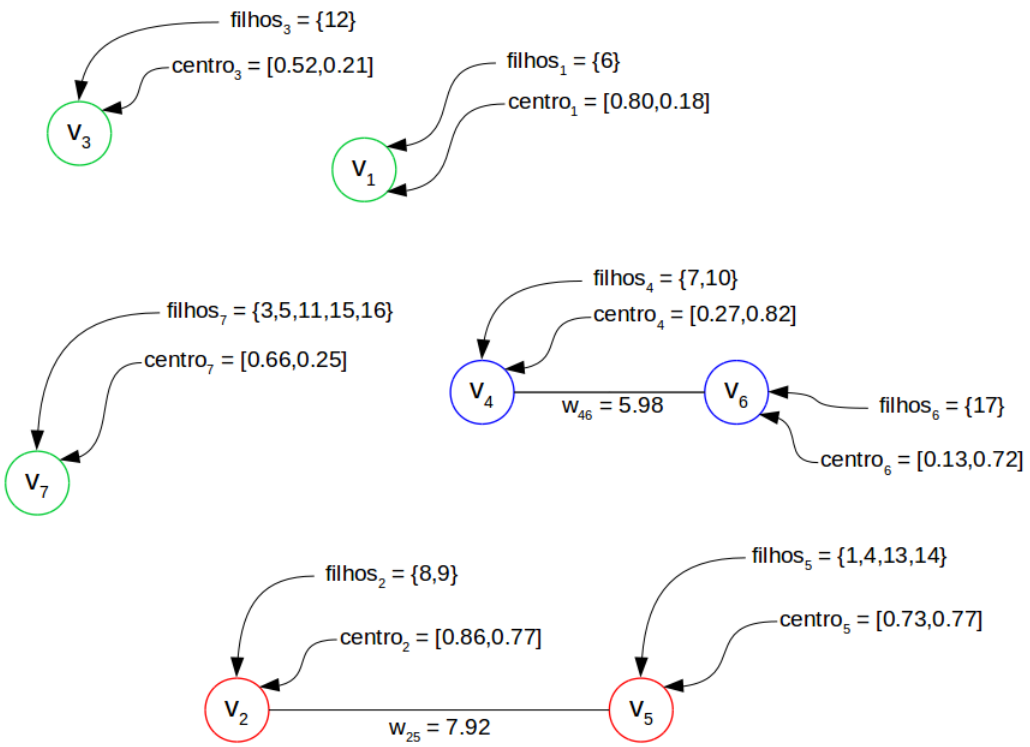
Rede protótipo com 8 vértices, 3 arestas e 17 filhos depois da aglutinação entre os antigos vértices v_7 e v_8 e inclusão do vértice resultante v_8.

Fonte: Produção da Autora.

O número total de filhos nesse ponto passa a violar o parâmetro H, portanto, a rede deve eliminar o vértice mais antigo. A função *eliminaAntigo()* retira da rede tanto o vértice v_1 como todas suas arestas e, consequentemente, retira também os objetos representados em $filhos_1$ e a possível influência que este vértice poderia exercer nas próximas escolhas de vizinhança da rede.

O resultado final da eliminação do vértice antigo, já com os ajustes de índices, pode ser observado na Figura 4.9. Enquanto os vértices que tem objetos dos rótulos Y_1 (azul) e Y_2 (vermelho) estão conectados, os representantes do rótulo Y_3 (verde) estão dispersos devido às últimas transformações causadas pela entrada dos objetos $\vec{x}^{16}$ e $\vec{x}^{17}$.

Figura 4.9 - Resultado final da rede após observação do novo objeto $\vec{x}^{17}$, execução da função *aglutina()* e da função *eliminaAntigo()*



Rede protótipo com 7 vértices, 2 arestas e 16 filhos depois da eliminação do antigo vértice v_1 pela função *eliminaAntigo()*.

Fonte: Produção da Autora.

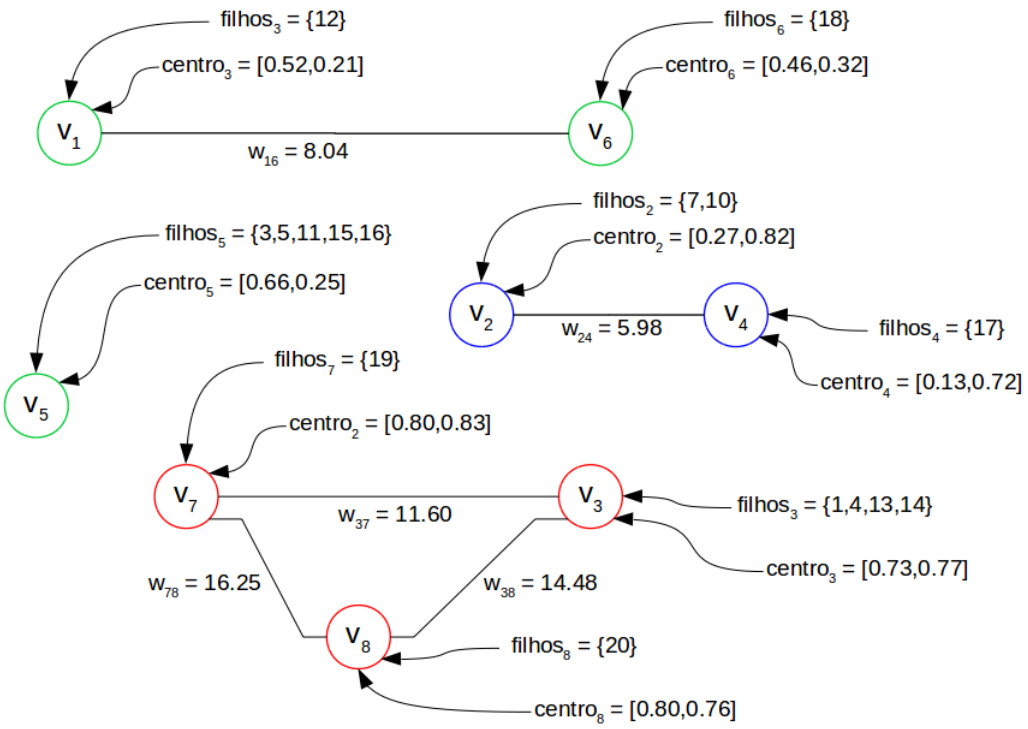
Tabela 4.4 - Novos objetos observados pela rede exemplo.

Objetos	Atributos		Classe
$\vec{x}^{18}$	0.46	0.32	3
$\vec{x}^{19}$	0.80	0.82	2
$\vec{x}^{20}$	0.80	0.76	2

Avançando um pouco no fluxo de dados F, a Figura 4.10 mostra o resultado final da rede após a observação até o objeto $\vec{x}^{20}$. Os novos objetos $\vec{x}^{18}$, $\vec{x}^{19}$ e $\vec{x}^{20}$, pertencentes aos rótulos Y_3 (verde) e Y_2 (vermelho), com atributos conforme Tabela 4.4, modificam a rede

em vários pontos, tanto pela ação da função *aglutina()* como pela função *eliminaAntigo()*.

Figura 4.10 - Resultado final da rede após observação até o objeto $\vec{x}^{20}$ do fluxo de dados F.



Rede protótipo com 8 vértices, 5 arestas e 16 filhos depois da observação até o objeto $\vec{x}^{20}$ do fluxo de dados F.

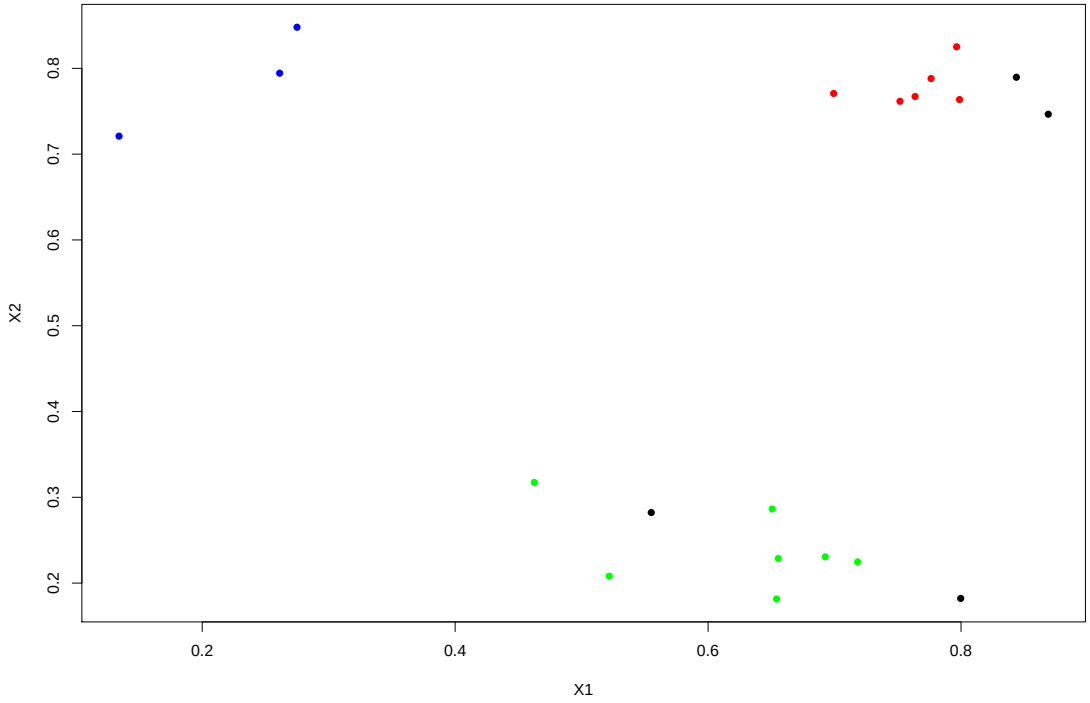
Fonte: Produção da Autora.

O gráfico da Figura 4.11 traz uma melhor visualização de todos os objetos já observados até então no espaço de dados. Em preto estão os objetos já "esquecidos" pela rede, são eles o x^2, x^6, x^8 e x^9, os dois primeiros do rótulo Y_3 (verde) e os dois últimos do rótulo Y_2 (vermelho). Percebe-se que os primeiros objetos a serem esquecidos são pertencentes aos dois grupos mais volumosos, e os objetos do rótulo Y_1 (azul) permanecem na rede.

A rede que está sendo utilizada como exemplo nesta seção, como pode ser observado na Figura 4.10, é um grafo desconexo e cada parte conectada contém um grafo completo, ou seja, contém arestas entre todos os vértices da parte. Neste caso, o algoritmo *Infomap* é levado a um caso trivial, onde, independente de onde os *random walkers* iniciem suas

caminhadas, eles irão percorrer sempre os mesmos trechos, logo, cada parte será definida como uma comunidade. Portanto, a detecção de comunidades nesse ponto da rede divide os vértices em 4 comunidades C_i^v, onde $C_1^v = \{v_1, v_6\}$, $C_2^v = \{v_2, v_4\}$, $C_3^v = \{v_3, v_7, v_8\}$ e $C_4^v = \{v_5\}$.

Figura 4.11 - Dados utilizados na construção da Rede Protótipo contida na Figura 4.10.



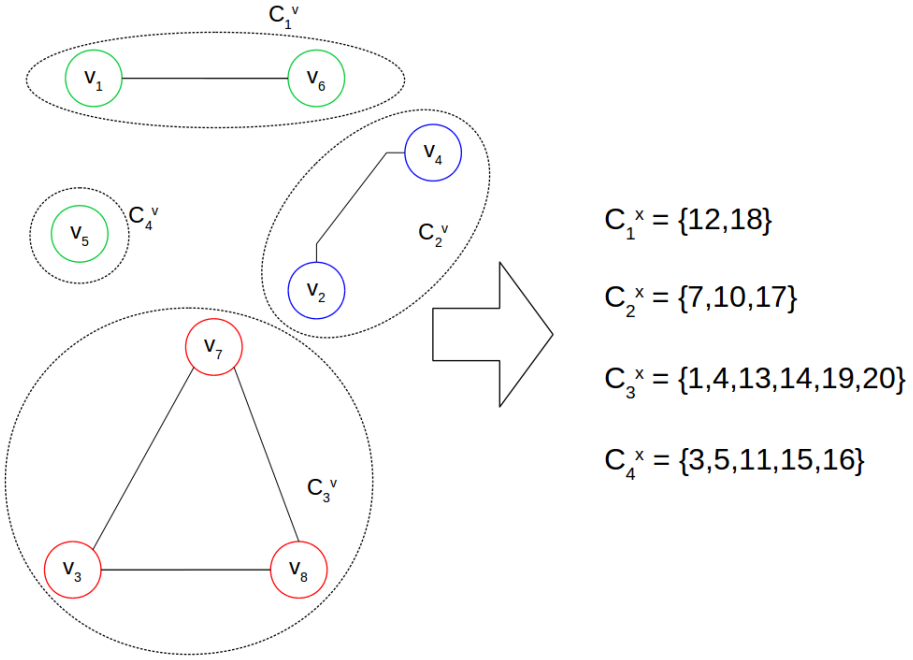
Dados observados pela rede exemplo até o objeto x^{20}. Os objetos em preto são os que já foram "esquecidos" pela rede, o restante das cores representam os rótulos dos objetos.

Fonte: Produção da Autora.

Esse resultado é utilizado para definir os grupos (ou *clusters*) de cada objeto $\vec{x^i} \in RP$, onde para cada comunidade C_i^v existe um *cluster* C_i^x tal que $C_i^x = \bigcup_j filhos_j, \forall v_j \in C_i^v$. Assim, considerando a rede exemplo, os *clusters* formados são: $C_1^x = \{12, 18\}$, $C_2^x = \{7, 10, 17\}$, $C_3^x = \{1, 4, 13, 14, 19, 20\}$ e $C_4^x = \{3, 5, 11, 15, 16\}$. Esse processo pode ser visto na Figura 4.12. Para avaliar se esses *clusters* representam a realidade dos dados, eles devem ser comparados com os verdadeiros rótulos Y_k do fluxo de dados F.

Primeiramente, como visto até agora, os rótulos Y_k dos objetos podem ser três, são eles: $Y_1 = \{x^7, x^{10}, x^{17}\}$, $Y_2 = \{x^1, x^4, x^{13}, x^{14}, x^{19}, x^{20}\}$ e $Y_3 = \{x^3, x^5, x^{11}, x^{12}, x^{15}, x^{18}\}$, onde o rótulo Y_1 é o de cor azul nas imagens, o Y_2 é o de cor vermelha e o Y_3, o de cor verde. Por se tratar de um exemplo pequeno, verifica-se que o *cluster* C_2^x corresponde a Y_1, C_3^x corresponde a Y_2 e que a junção dos *clusters* C_1^x e C_4^x corresponde a Y_3.

Figura 4.12 - Processo de agrupamento dos objetos $\vec{x}^i$ em *clusters* C_i^x através do uso do resultado da divisão em comunidades C_i^v da rede.



Divisão dos objetos sendo representados pela rede exemplo da Figura 4.10 em *cluters* utilizando o resultado da aplicação do algoritmo *Infomap*.

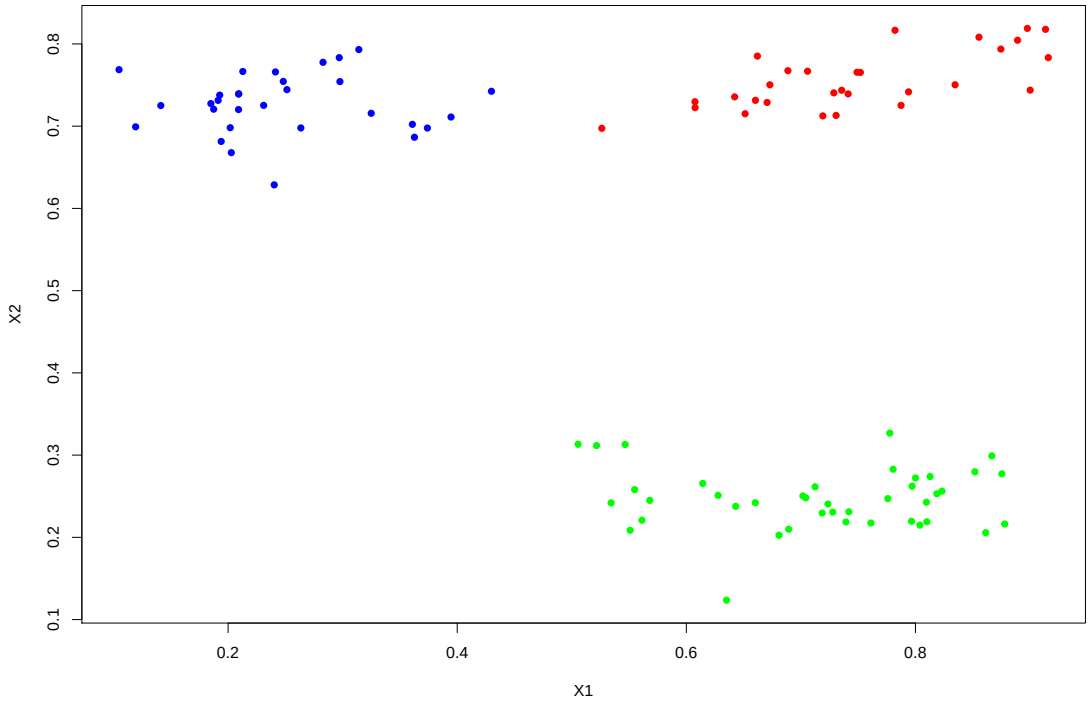
Fonte: Produção da Autora.

No caso do exemplo, todos os *clusters* tem todos os elementos pertencentes ao rótulo dominante, ou seja acurácia 100%, no entanto, o número de *clusters* n_C é maior que o número de rótulos k, assim, o resultado da pureza nesse caso é $purity = \frac{4}{3} \times 100\% \approx 133\%$, exatamente porque os elementos do rótulo Y_3 estão divididos em dois *clusters* C_1^x e C_4^x.

Para analisar a função *clusteringRede()* com um exemplo mais complexo, considera-se

agora uma rede construída a partir da observação de 100 objetos do mesmo fluxo de dados F, mas com parâmetros $MAXV = 30$ e $H = 80$. Os dados podem ser observados na Figura 4.13.

Figura 4.13 - 100 objetos do fluxo de dados F.

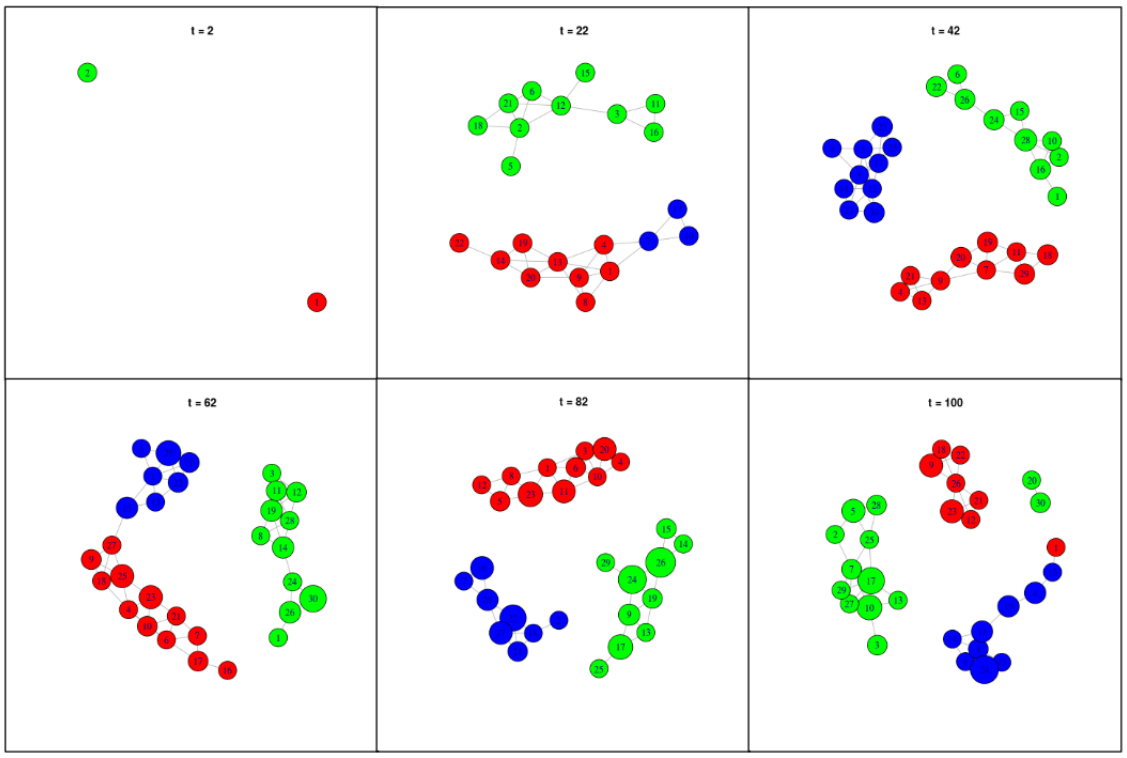


Objetos do fluxo de dados F observados até x^{100}. As cores representam as diferentes classes dos objetos.

Fonte: Produção da Autora.

Na Figura 4.14 é mostrada uma sequência de *snapshots*, ou seja, estados da rede em diversos momentos, durante a observação dos 100 objetos. As cores dos vértices só são atribuídas se todos os objetos representados por este são do mesmo rótulo, e o tamanho do vértice é proporcional à quantidade de objetos em seu respectivo conjunto $filhos$.

Figura 4.14 - Rede Protótipo em diferentes momentos no tempo.

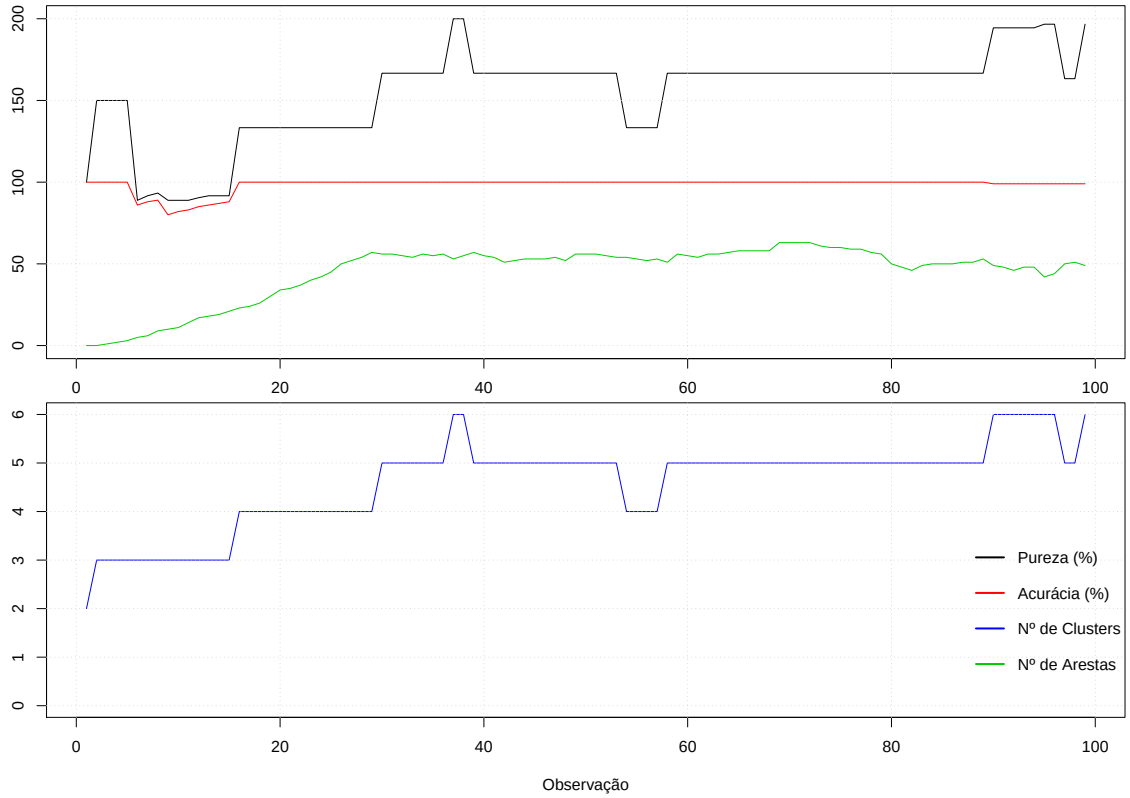


Rede Protótipo em diversos momentos de observação dos dados da Figura 4.13, o t indica o último objeto observado em cada rede. As cores dos vértices só são atribuídas caso todos os objetos representados sejam da mesma classe, e o tamanho do vértice é proporcional a quantidade de objetos no conjunto $filhos$.

Fonte: Produção da Autora.

Por fim, na Figura 4.15 mostra-se a evolução dos valores de algumas métricas como a pureza, a acurácia, o número de *clusters* e o número de arestas totais na rede durante a execução desse último exemplo a cada nova observação. Os valores de pureza se mantiveram na maioria do tempo acima dos 100% indicando que há a formação de mais do que três *clusters* (que seria o ideal nesse exemplo). Fazendo uma melhor análise dos resultados de pureza do exemplo dado, é possível verificar que valores iguais a 133%, 166%, 200% e assim por diante significam que, apesar da quantidade de *clusters* ser maior que o ideal, os *clusters* formados foram todos homogêneos.

Figura 4.15 - Evolução das métricas de avaliação a cada observação.



Gráficos mostrando os valores para *purity*, *accuracy*, número de *clusters* e de arestas. Os valores da pureza encontram-se quase sempre acima dos 100% com algumas poucas exceções indicando que o agrupamento na RP forma mais *clusters* do que classes. Essa informações é confirmada pelos valores de acurácia e de número de *clusters*, tanto que os desenhos das linhas formam o mesmo padrão . O número de arestas desse exemplo se mantém estável depois da observação do objeto 30 por causa da ação das funções *aglutina()* e *eliminaAntigo()*.

Fonte: Produção da Autora.

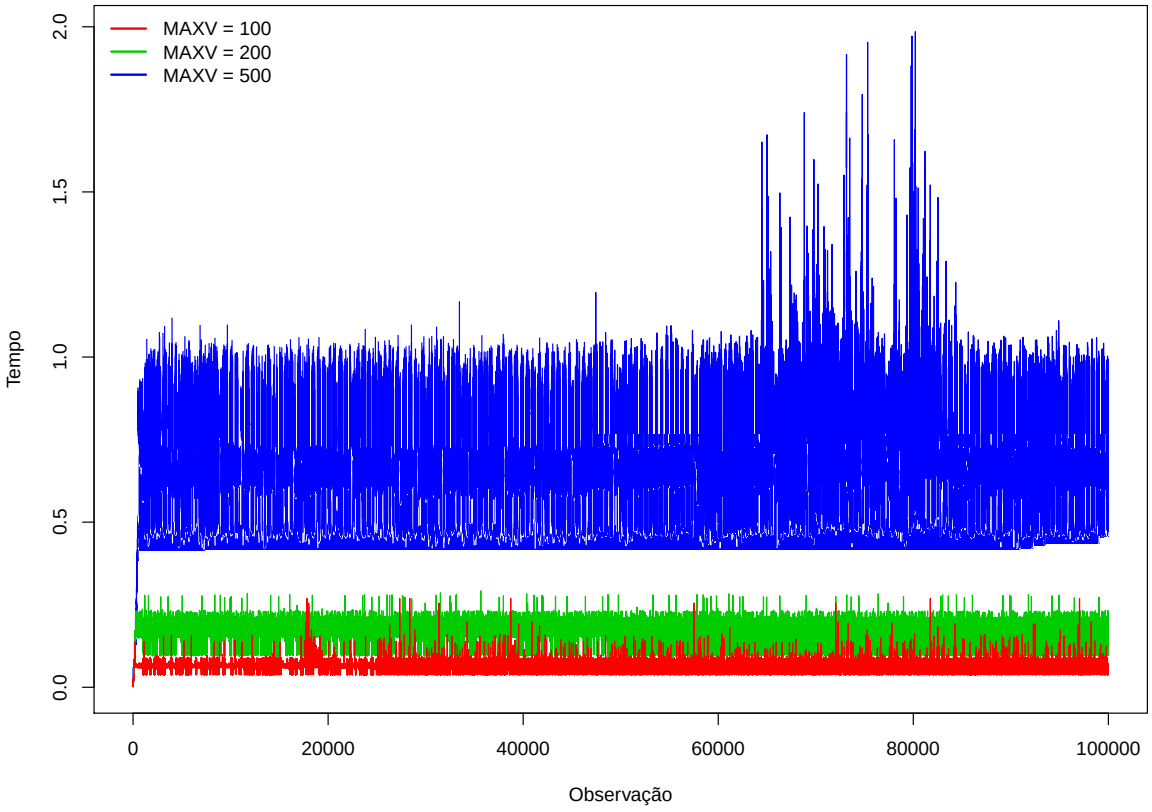
Essa informação é confirmada pelo valor da acurácia que se mantém em 100% a não ser por um breve período no início e no fim das observações. A evolução do número de arestas mostra que independente do número de observações, o conjunto das arestas se mantém relativamente estável. Por último, a evolução do número de *clusters* confirma o que pode ser observado pelos valores de pureza e acurácia, tanto que o desenho das linhas de pureza e número de *clusters* são similares.

4.3 Avaliação dos parâmetros

Nesta seção são apresentados resultados das avaliações do tempo de processamento e desempenho de acurácia da metodologia RP para diferentes valores de parâmetros.

Primeiramente verificou-se como diferentes valores do parâmetro $MAXV$, que é o responsável por definir o tamanho máximo da estrutura RP, afeta o tempo de processamento considerando um fluxo de dados com tamanho $N = 100000$, dimensão $d = 20$, número de *clusters* $k = 5$ e parâmetro da janela $H = 1000$. Na Figura 4.16 pode ser visto os valores de medição do tempo que a metodologia leva para processar cada uma das cem mil observações do fluxo.

Figura 4.16 - Tempo de execução da metodologia RP



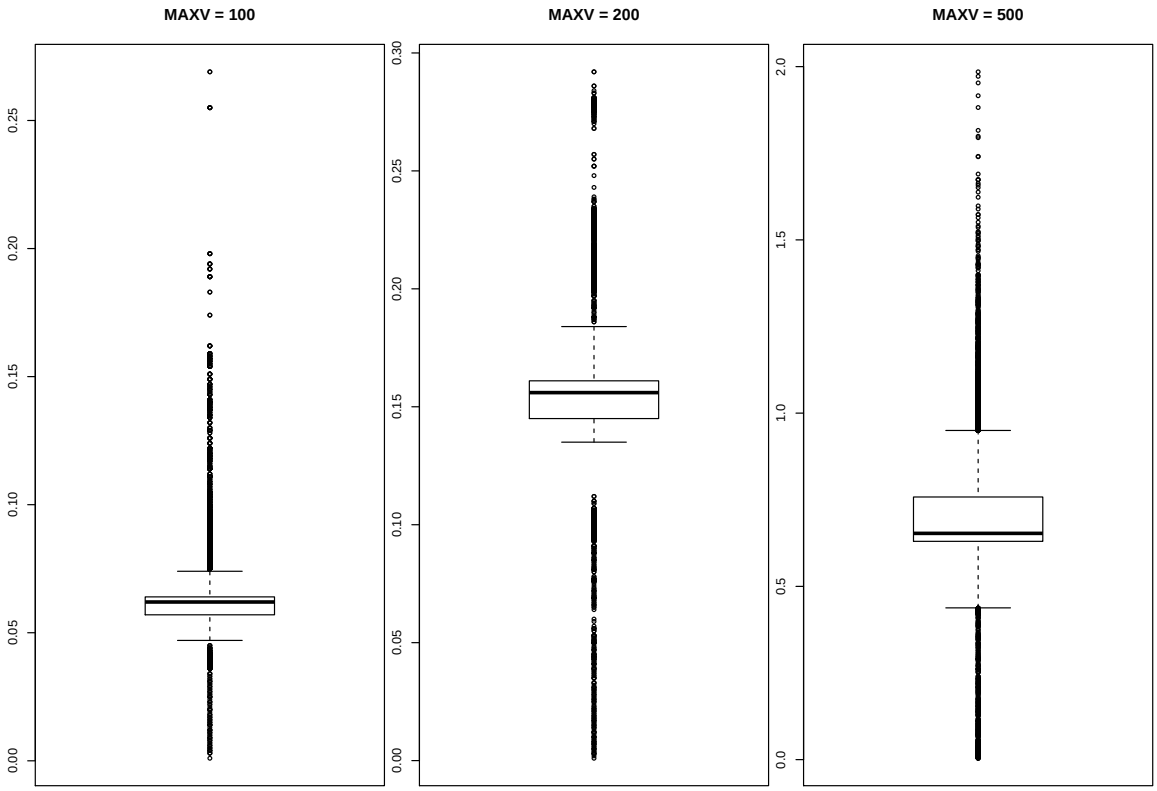
Tempo de processamento de cada observação da metodologia RP para diferentes valores do parâmetro $MAXV$. Os outros parâmetros são definidos $d = 20$, $k = 5$ e $H = 1000$.

Fonte: Produção da Autora.

Os resultados desse gráfico mostram que o tempo para esse processamento se mantêm de certa forma constante independente do tamanho do fluxo, variando apenas quando se varia o valor do parâmetro *MAXV*. Isso se deve ao fato que para cada observação o processamento mais custoso da metodologia é fazer o cálculo da similaridade com todos os outros vértices atuais da rede e a quantidade de vértices que passam por esse cálculo é justamente o valor do parâmetro *MAXV*.

Na Figura 4.17 é apresentado um resumo dos valores coletados de tempo de execução considerando média, desvio-padrão e valores extremos. Todos os experimentos deste capítulo e dos próximos foram realizados em um PC com processador Intel i3-6100 3.70 GHz com 4 núcleos, 8 Gb de memória RAM e executando sistema operacional Ubuntu 16.04.5 LTS 64-bit.

Figura 4.17 - Tempo médio de execução da metodologia RP



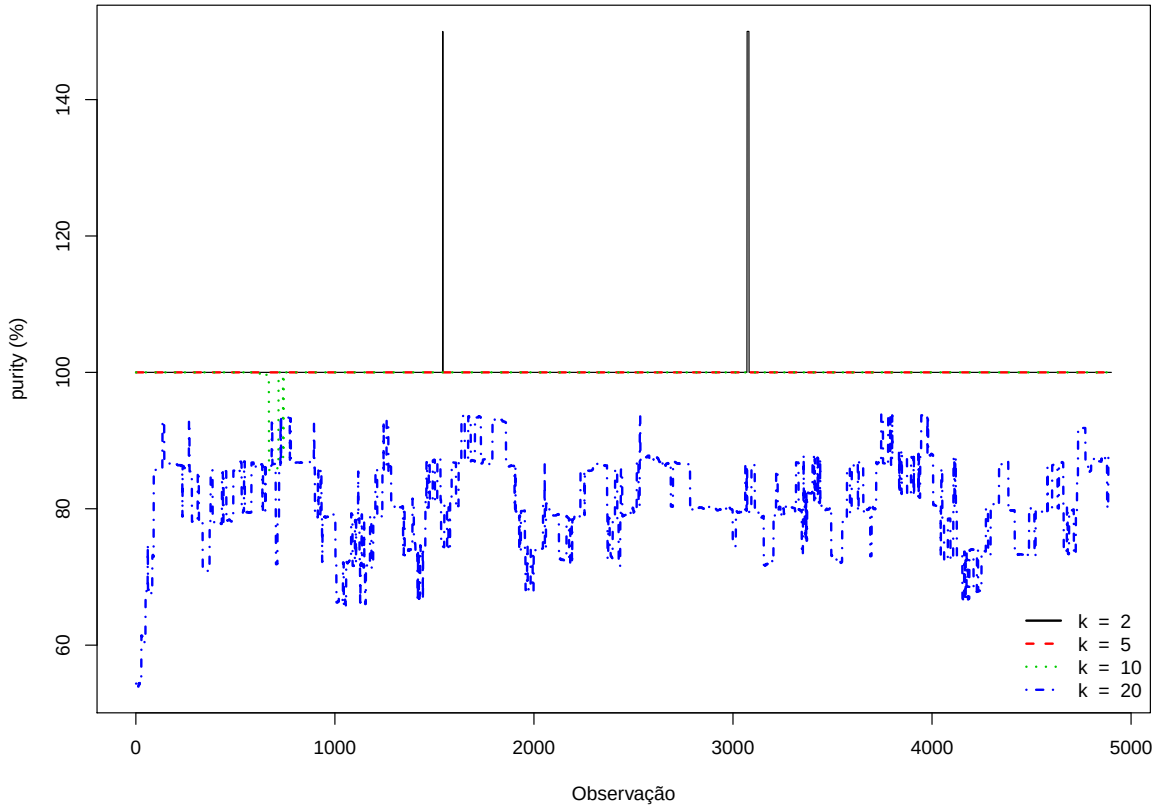
Resumo dos valores de tempo de execução para cada valor diferente do parâmetro *MAXV*. Na Figura pode ser observado a média, desvio-padrão e valores extremos.

Fonte: Produção da Autora.

Todo o desenvolvimento da metodologia foi realizado na linguagem R (R Core Team, 2018) no RStudio versão 1.1.456. O pacote *igraph* (CSARDI; NEPUSZ, 2006) foi utilizado em todo o tratamento, manipulação, criação e análise das redes complexas e dos gráficos de visualização. Por fim, foram utilizados os pacotes *stream* (HAHSLER et al., 2017) e *streamMOA* (HAHSLER et al., 2018) que têm implementações de várias funções e algoritmos da literatura relacionados aos fluxos de dados.

Além da verificação do tempo de execução, testou-se a pureza dos *clusters* formados para diferentes valores de dimensão d e número de *clusters* k. Na Figura 4.18 são mostrados os resultados para quando o valor de k é definido como $2, 5, 10$ ou 20, e os outros parâmetros são $MAXV = 200$, $d = 20$, $N = 5000$ e $H = 500$.

Figura 4.18 - Valores de *purity* para diferentes valores do parâmetro k.

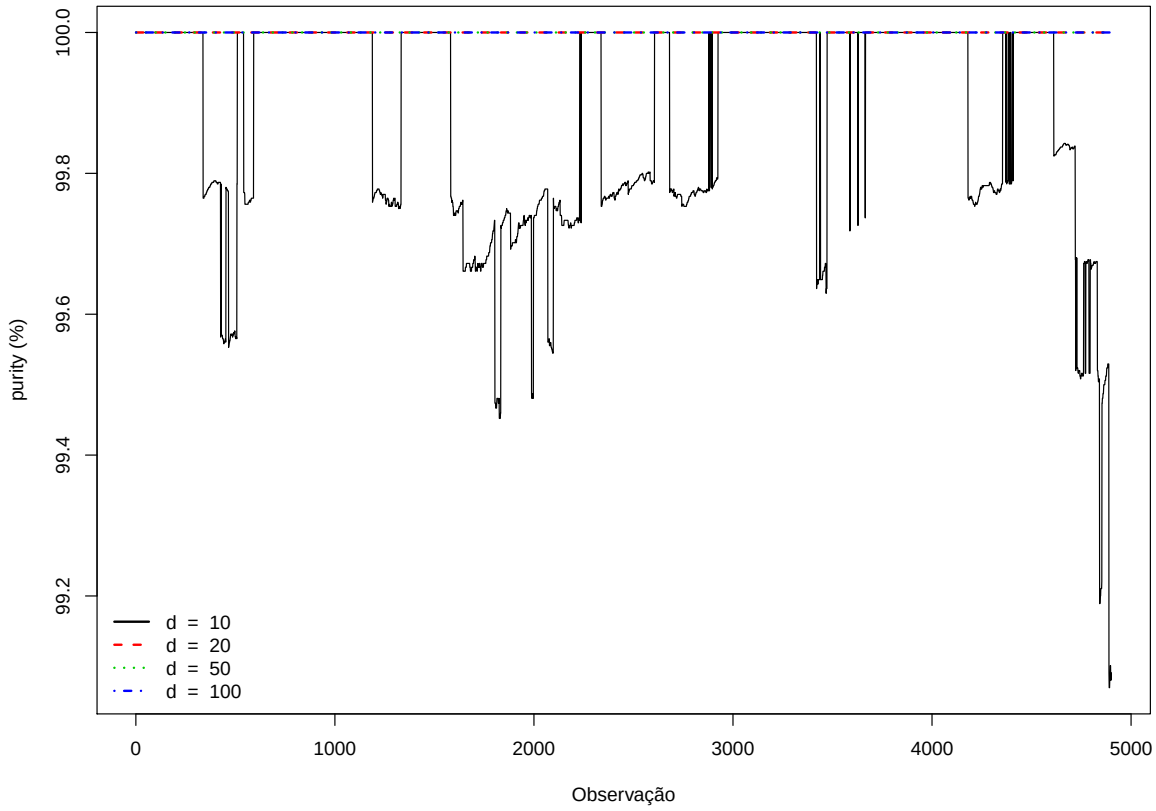


Valores de pureza obtidos pela metodologia considerando diferentes valores do parâmetro k. Os outros parâmetros são definidos $MAXV = 200$, $d = 20$, $N = 5000$ e $H = 500$.

Fonte: Produção da Autora.

Observa-se que quando há apenas dois *clusters*, $k = 2$, por duas vezes a metodologia forma somente um *cluster* e o valor da pureza altera-se, portanto. Para os valores de k em 5 e 10, a pureza se mantém em 100% em todo o processamento, mas para $k = 20$ a metodologia já não consegue ter desempenho perfeito, mantendo valores de pureza entre 60% e 90%.

Figura 4.19 - Valores de *purity* para diferentes valores do parâmetro d.



Valores de pureza obtidos pela metodologia considerando diferentes valores do parâmetro d. Os outros parâmetros são definidos $k = 5$, $MAXV = 200$, $N = 5000$ e $H = 500$.

Fonte: Produção da Autora.

Na Figura 4.19 são mostrados os resultado obtidos para quando os valores de dimensão são $10, 20, 50$ e 100 e os outros parâmetros são definidos $k = 5$, $MAXV = 200$, $N = 5000$ e $H = 500$. Como pode ser visto, apenas quando $d = 10$ o desempenho da metodologia não é perfeito, mesmo assim o valor de pureza se mantêm acima dos 99%.

4.4 Considerações finais

A metodologia Rede Protótipo é um conjunto de funções que operam uma rede complexa. Na rede são mantidos três conjuntos: o dos vértices, o das arestas e o dos pesos. Cada conjunto é preenchido conforme a metodologia é invocada pela observação de um novo objeto vindo do fluxo de dados com o qual se está trabalhando.

A construção e manutenção da RP, com a ajuda do método ε-*cut-ondemand*, inclui e altera os elementos na rede para refletir a observação desse novo objeto. Além disso, toda vez que solicitado, a metodologia oferece uma partição em *clusters* dos objetos na rede, com o auxílio do algoritmo *Infomap*.

Os vértices e arestas vão sendo adicionados ou eliminados conforme as funções *novoVertice()*, *escolheVizinhanca()*, *aglutina()* e *eliminaAntigo()* vão sendo executadas. Sendo que as duas primeiras funções tratam da inserção de um novo vértice, oriundo do fluxo ou de alguma aglutinação, a terceira mantém o número de vértices abaixo do valor do parâmetro $MAXV$ e a quarta e última função mantém o número total de objetos mantidos na RP abaixo do valor do parâmetro H, que faz parte do mecanismo de janela deslizante.

O que a metodologia tem a oferecer é uma exploração maior das possibilidades extras de estruturas do tipo rede complexa para a tarefa do agrupamento em fluxo de dados. Uma vantagem é o uso do algoritmo *Infomap* que consegue separar os vértices em comunidades sem precisar definir nenhum parâmetro, já que a própria rede é o parâmetro de entrada do método. As comunidades, nesse caso, refletem os *clusters* dos dados porque as funções de construção e manutenção da estrutura foram implementadas com esse intuito.

Outra vantagem que pode ser citada é a facilidade de manutenção da estrutura que é formada por três conjuntos V, E e W. No conjunto dos vértices V são mantidas as abstrações dos dados representados por protótipos e por um conjunto de *timestamps*. Nos conjuntos de arestas E e de pesos W são mantidas as relações e os valores de similaridade entre os protótipos dos vértices, sendo que o método ε-*cut-ondemand* é utilizado para determinar quais arestas serão estabelecidas e os valores das similaridades referentes às arestas são armazenados no conjunto dos pesos.

Outra vantagem do uso da rede complexa é que as operações de aglutinação e de eliminação do vértice mais antigo alteram os vértices na rede, mas, ao mesmo tempo, o conjunto dos vértices mantêm um índice que indica quais são os mais novos e quais são mais antigos, exatamente para executar a função *eliminaAntigo()*. Isso é resolvido mantendo-se um valor para cada vértice, semelhante ao *timestamp* dos objetos, que é

atualizado a cada eliminação. Assim, a rede automaticamente mantém o vértice mais antigo da rede na posição 1 e toda vez que dois vértices são aglutinados e, portanto, eliminados da rede, eles voltam à rede com o maior índice disponível no momento.

A metodologia RP será submetida a vários experimentos nos dois próximos capítulos assim como os outros algoritmos apresentados ao longo desta tese *CluStream*, *DenStream* e *SNCStream*. Medidas de *silhouette*, *purity* e *accuracy* serão comparadas entre todos os algoritmos. Além disso, outras medidas também serão comparadas como número de *clusters*, número de *microclusters*, número de arestas e graus dos vértices.

Os experimentos são realizados com dados sintéticos que simulam mudanças de conceito dos tipos *shift* abrupto, gradual e incremental. Além disso, também testam os algoritmos com dados sintéticos que sofrem evoluções de conceito dos tipos *New* e *Disappear*. Por último, a metodologia RP é aplicada a um cenário de dados que correspondem a séries temporais de observação da Terra.

5 EXPERIMENTOS

Neste capítulo, os algoritmos *CluStream*, *DenStream*, *SNCStream* e a metodologia Rede Protótipo são testados e comparados com dados sintéticos que simulam vários tipos de alterações nos conceitos em cinco experimentos.

5.1 Descrição dos experimentos

Nos três primeiros experimentos é utilizado um fluxo de dados que contêm objetos de dimensão 2 divididos entre três *clusters*. Os três *clusters* são gaussianas, sendo que dois são fixos no espaço de dados e o terceiro sofre uma mudança de conceito do tipo *shift*. No primeiro experimento, o terceiro *cluster* sofre uma mudança do tipo *shift* abrupto, no experimento 2, o *cluster* sofre um *shift* gradual e no experimento 3, um *shift* incremental.

Nos experimentos 4 e 5, o fluxo de dados observado contêm objetos de dimensão 2 e pode conter de 3 a 7 *clusters*. No experimento 4, o fluxo inicia com 3 *clusters* e por 4 vezes surge um novo *cluster* no espaço de dados até um total de 7. No experimento 5, o fluxo inicia com 6 *clusters* e por 3 vezes um *cluster* deixa de existir, restando apenas 3 ao final.

Nos três primeiros experimentos, os parâmetros relacionados ao mecanismo de janela deslizante, H e λ, são testados com duas versões. Na primeira versão, os algoritmos terão tempo para se adaptar ao novo cenário e, na segunda, os algoritmos irão lembrar de todo o histórico dos dados, inclusive da posição anterior do terceiro *cluster*.

Nas Tabelas 5.1, 5.2 e 5.3 são resumidas as informações sobre os experimentos realizados neste capítulo. Os dados sofrem diferentes movimentos em cada experimento e são observados pelos algoritmos considerando os parâmetros do mecanismo de janela deslizante. Os parâmetros dos algoritmos foram escolhidos a partir da recomendação dos próprios autores em (AGGARWAL, 2003; CAO et al., 2006; BARDDAL et al., 2015).

Tabela 5.1 - Parâmetros dos Fluxos de Dados

Experimento	Parâmetros	Movimento
1	$N = 1000$, $d = 2$, $k = 3$	*Shift* Abrupto
2	$N = 1000$, $d = 2$, $k = 3$	*Shift* Gradual
3	$N = 1000$, $d = 2$, $k = 3$	*Shift* Incremental
4	$N = 3000$, $d = 2$, $k = 3:7$	*New* (4 vezes)
5	$N = 3000$, $d = 2$, $k = 6:3$	*Disappear* (3 vezes)

Os três primeiros experimentos consideram os primeiros mil dados de um fluxo F_1 divididos em 3 *clusters*. Sendo que um dos *clusters* sofre uma mudança de conceito do tipo *shift*. Os dois últimos experimentos consideram os primeiros três mil dados de um fluxo F_2, que pode conter de 3 a 7 *clusters*.

Tabela 5.2 - Parâmetros do mecanismo de Janela.

Experimento	Parâmetros
1	$H = 300$ e $\lambda = 0.1$ ou $H = 1000$ e $\lambda = 0.01$
2	$H = 300$ e $\lambda = 0.1$ ou $H = 1000$ e $\lambda = 0.01$
3	$H = 300$ e $\lambda = 0.1$ ou $H = 1000$ e $\lambda = 0.01$
4	$H = 1000$ e $\lambda = 0.01$
5	$H = 1000$ e $\lambda = 0.01$

Os parâmetros do mecanismo de janela deslizante são de dois tipos: o H considera a quantidade de objetos sendo observada e o λ considera o peso da influência de um objeto, quanto mais antigo, menos influência. Os três primeiros experimentos consideraram duas versões dos parâmetros da janela: $H = 300$ e $\lambda = 0.1$, dessa forma, levando em consideração só um passado mais recente, e $H = 1000$ e $\lambda = 0.01$, para verificar o comportamento dos algoritmos considerando todo o passado dos dados.

Tabela 5.3 - Parâmetros dos Algoritmos

Algoritmo	Parâmetros
Clu	$m = 100$, $k = 3$, $init = 120$
Den	$\varepsilon = 0.05$, $\mu = 1$, $\beta = 0.2$, $\eta = 2$, $init = 120$
SNC	$\varepsilon = 0.05$, $\mu = 1$, $\beta = 0.2$, $\eta = 2$, $\omega = 4$
RP	$MAXV = 100$

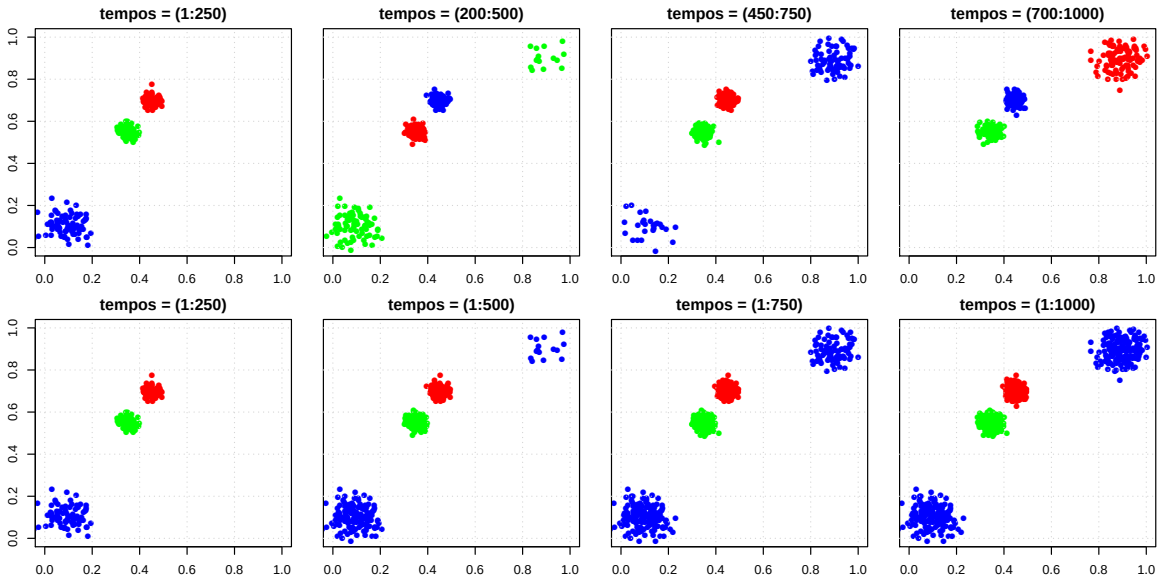
Os algoritmos tiveram seus parâmetros definidos considerando as próprias recomendações de seus respectivos autores. Os algoritmos *CluStream* e *DenStream* utilizados foram os contidos no pacote *stream* (HAHSLER et al., 2017; HAHSLER et al., 2018) da linguagem R (R Core Team, 2018). O algoritmo *SNCStream* foi implementado pela autora na linguagem R seguindo as instruções dos autores em (BARDDAL et al., 2015; BARDDAL et al., 2019).

Os dados também foram formados utilizando o pacote *stream*, especificamente as funções *DSD_MG()*, *MGC_Static()* e *DSD_Memory()*. As funções permitem controlar os centros

e raios dos *clusters* que irão compor o fluxo de dados e também usar outros formatos para os *clusters* como retângulos ao invés de gaussianas. Ao adicionar ou excluir *clusters* em tempos pré-determinados simula-se o efeito das evoluções *New* e *Disappear* e ao mover o centro de um *cluster* simula-se o efeito das mudanças de conceito do tipo *shift*.

O 1º experimento foi feito considerando um fluxo de dados de dimensão $d = 2$, com número de *clusters* $k = 3$ e quantidade de objetos final $N = 1000$, sendo que um dos *clusters* sofre um *concept drift* do tipo *shift* abrupto no tempo $t = 400$, ou seja, entre as observações 400 e 401, o centro do *cluster* se movimenta como indicado nas Figuras 5.1, na primeira linha de imagens, os dados são apresentados considerando a janela H = 300 e, na segunda, a janela $H = 1000$.

Figura 5.1 - Experimento 1 - Movimento de *Shift* Abrupto.



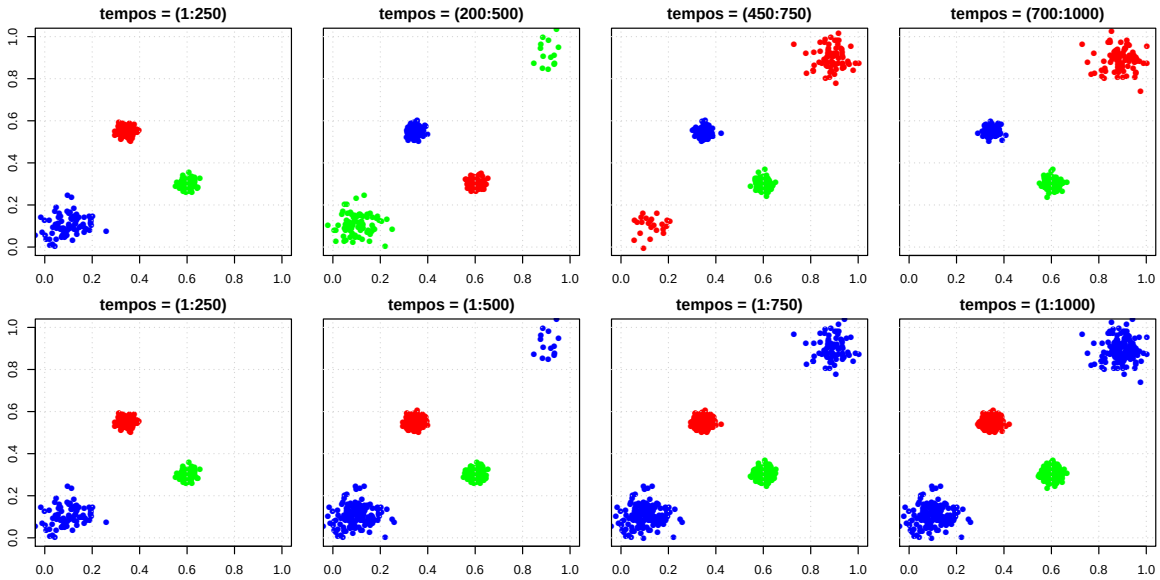
O *cluster* que inicia com a cor azul é o que sofre o *shift*. Os *snapshots* da linha acima na figura mostram os dados com janela $H = 300$ nos tempos $t = 250$, $t = 500$, $t = 750$ e $t = 1000$. Abaixo, estão os *snapshots* dos dados com janela $H = 1000$ nos mesmos tempos.

Fonte: Produção da Autora.

O segundo experimento foi feito considerando que um fluxo emite objetos de dimensão $d = 2$ que pertencem a um dos três *clusters* nos dados e a quantidade final de objetos é $N = 1000$. Dois *clusters* são fixos no espaço de dados e o terceiro sofre um movimento

do tipo *shift* gradual durante os tempos $t = 400$ e $t = 600$, conforme pode ser visto na Figura 5.2. Ou seja, durante esse período os objetos do *cluster* que está se movimentando podem aparecer tanto em volta da antiga como da nova posição do centro do *cluster*. O experimento também foi feito em duas versões considerando os parâmetros do mecanismo de janela deslizante. Na primeira versão, os valores são $H = 300$ e $\lambda = 0.1$ e, na segunda versão, os valores são $H = 1000$ e $\lambda = 0.01$.

Figura 5.2 - Experimento 2 - Movimento de *Shift* Gradual

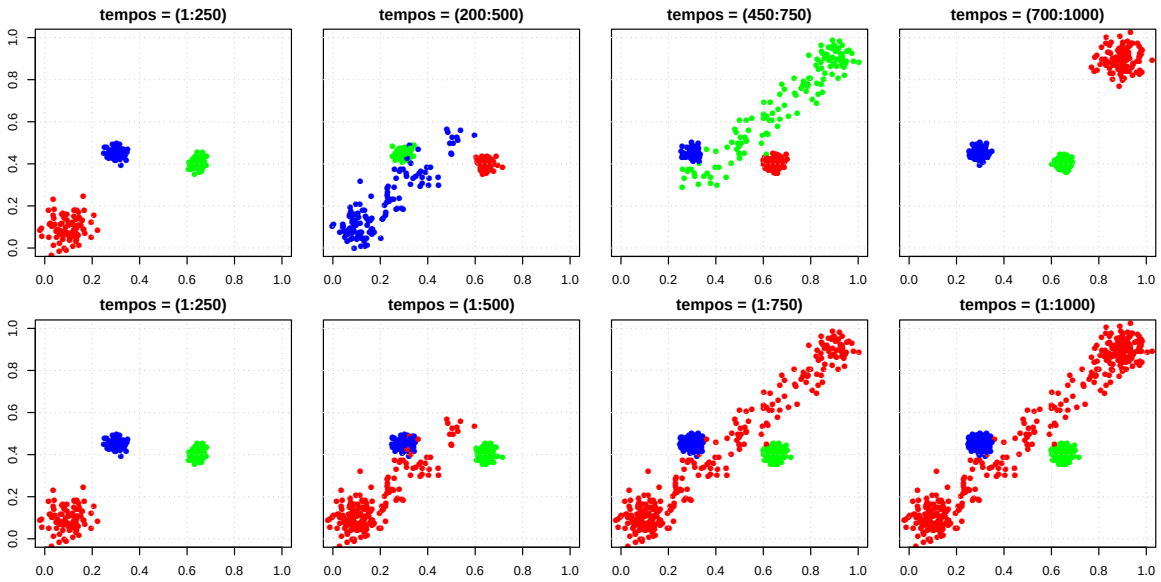


O *cluster* no canto inferior esquerdo sofre um *shift* gradual entre os tempos $t = 400$ e $t = 600$. A sequência de *snapshots* acima na figura mostra os dados quando o tamanho da janela $H = 300$ e abaixo quando $H = 1000$, nos tempos $t = 250$, $t = 500$, $t = 750$ e $t = 1000$.

Fonte: Produção da Autora.

O fluxo de dados do experimento 3 contém três *clusters* com objetos de dimensão $d = 2$. Dois *clusters* se mantêm fixos no espaço de dados e um terceiro sofre um movimento do tipo *shift* incremental durante os tempos $t = 400$ e $t = 600$. O *shift* incremental faz com que o centro do *cluster* se movimente lentamente na direção da nova posição deixando um rastro de objetos no caminho conforme pode ser visto na Figura 5.3. O experimento 3 também considera as duas versões dos parâmetros de janela deslizante.

Figura 5.3 - Experimento 3 - Movimento de *Shift* Incremental.



O *cluster* no canto inferior esquerdo sofre um *shift* incremental entre os tempos $t = 400$ e $t = 600$. A sequência de *snapshots* acima na figura mostra os dados quando a janela tem tamanho $H = 300$ e abaixo quando $H = 1000$, nos tempos $t = 250$, $t = 500$, $t = 750$ e $t = 1000$.

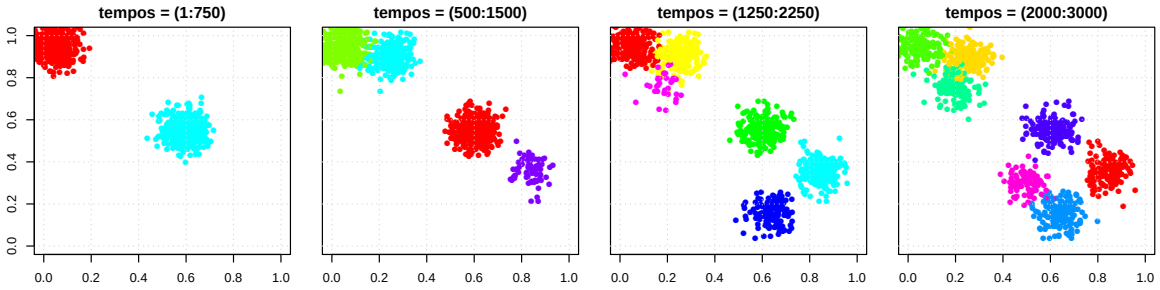
Fonte: Produção da Autora.

O quarto experimento com os algoritmos *CluStream*, *DenStream*, *SNCStream* e a metodologia RP foi feito com um fluxo de dados que emite objetos de dimensão $d = 2$, onde inicialmente podem ser observados três *clusters* e ao final são observados sete *clusters*, como pode ser visto na Figura 5.4. A quantidade total de objetos observados ao final da execução é $N = 3000$ e os *snapshots* mostrados na figura são nos tempos $t = 750$, $t = 1500$, $t = 2250$ e $t = 3000$, e a partir do tempo $t = 800$ a cada 400 observações nasce um novo *cluster*. O experimento considera apenas um tamanho de janela $H = 1000$ e valor de $\lambda = 0.01$.

O quinto experimento testa os algoritmos em dados que sofrem evoluções de conceito do tipo *Disappear*. O fluxo emite objetos de dimensão $d = 2$, inicialmente divididos em seis *clusters* e, após três movimentos do tipo *Disappear*, em apenas três *clusters*, como pode ser observado na Figura 5.5. A quantidade total de objetos observados ao final da execução é $N = 3000$ e os *snapshots* mostrados na figura são nos tempos $t = 750$, $t = 1500$, $t = 2250$ e $t = 3000$. A partir do tempo $t = 1000$, a cada 500 observações, um *cluster*

deixa de existir. O experimento considera apenas uma versão do experimento quanto aos parâmetros da janela com $H = 1000$ e $\lambda = 0.01$.

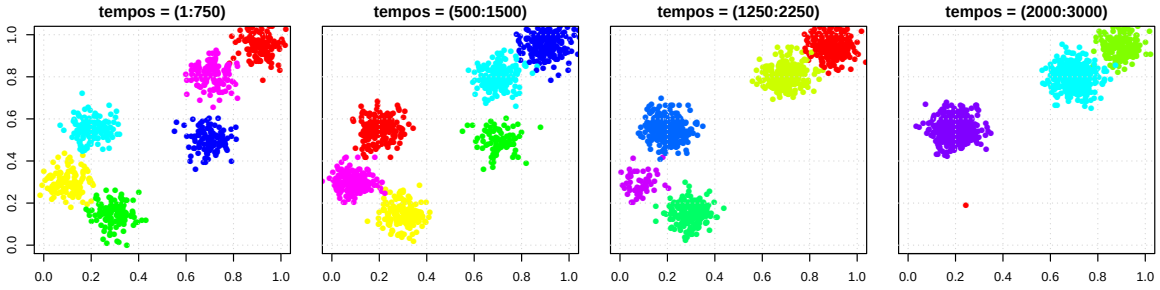
Figura 5.4 - Experimento 4 - Movimentos do tipo *New*.



Sequência de *snapshots* nos tempos $t = 750$, $t = 1500$, $t = 2250$ e $t = 3000$ considerando uma janela de tamanho $H = 1000$. É possível ver os novos *clusters* surgindo (movimento *New*) ao longo das observações, até que sete são formados ao final.

Fonte: Produção da Autora.

Figura 5.5 - Experimento 5 - Movimentos do tipo *Disappear*.



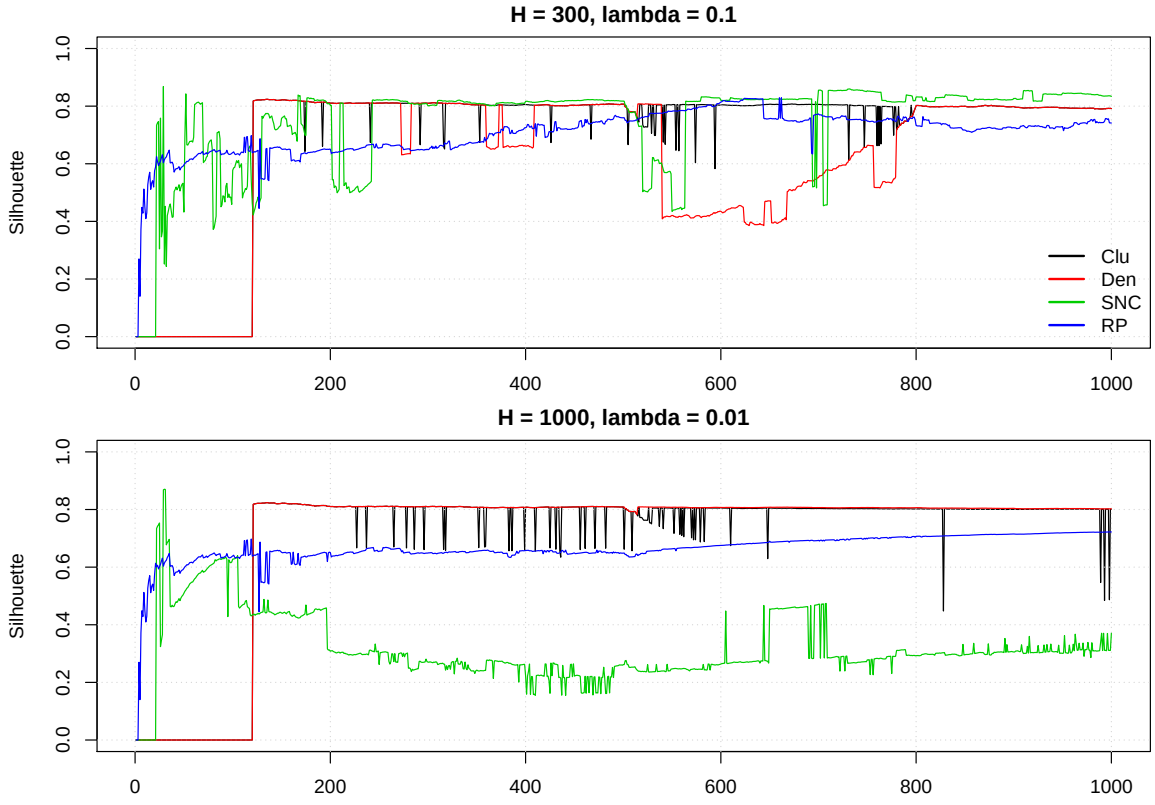
Sequência de *snapshots* nos tempos $t = 750$, $t = 1500$, $t = 2250$ e $t = 3000$ considerando uma janela de tamanho $H = 1000$. É possível ver os *clusters* deixando de existir (movimento *Disappear*) ao longo das observações, até restar apenas três ao final.

Fonte: Produção da Autora.

5.2 Resultados e considerações

Nesta seção são apresentados os resultados obtidos pelos quatro algoritmos nos cinco experimentos descritos na seção anterior. Inicialmente serão apresentados os valores de *silhouette* para os três primeiros experimentos. Depois os valores de pureza e acurácia obtidos no cinco experimentos. Após isso, algumas métricas como números de *clusters*, números de *microclusters*, número de arestas e graus serão apresentados. Por fim, alguns gráficos mostram como foi a evolução de formação de *clusters* e *microclusters* dos quatro algoritmos durante os cinco experimentos descritos neste capítulo.

Figura 5.6 - Experimento 1 - *Silhouette*.



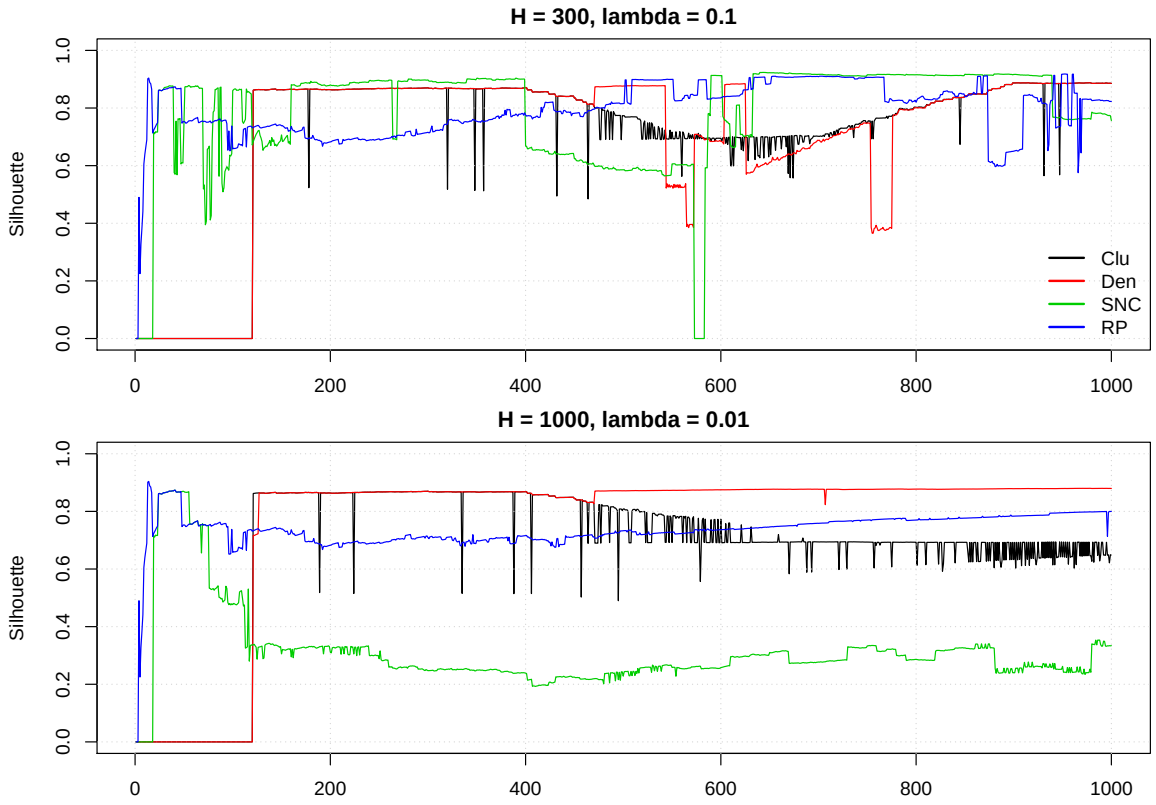
Valores médio de *silhouette* para os quatro algoritmos nas duas versões do experimento 1. Acima quando $H = 300$ e $\lambda = 0.1$, abaixo $H = 1000$ e $\lambda = 0.01$.

Fonte: Produção da Autora.

Nas Figuras 5.6, 5.7 e 5.8 são apresentados os resultados da medida de *silhouette* para

os três primeiros experimentos. Devido ao custo que é calcular essa medida, esta não foi calculada para os dois últimos.

Figura 5.7 - Experimento 2 - *Silhouette*.

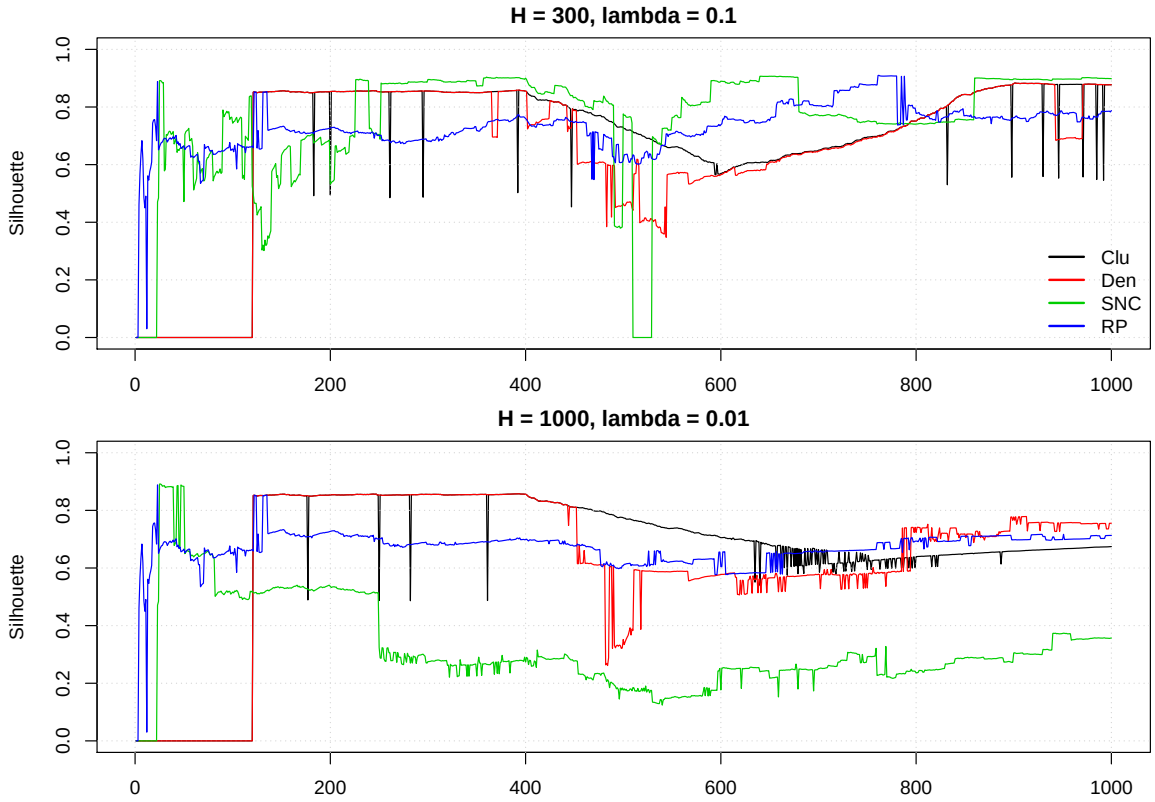


Valores médio de *silhouette* para os quatro algoritmos nas duas versões do experimento 2. Acima quando $H = 300$ e $\lambda = 0.1$, abaixo $H = 1000$ e $\lambda = 0.01$.

Fonte: Produção da Autora.

O que observa-se nesses resultados é que na primeira versão dos três experimentos, considerando a janela $H = 300$ e $lambda = 0.1$, todos os algoritmos obtém desempenho parecido, com alterações fortes nos valores quando ocorrem os movimentos de *shift*, o que é previsível considerando a mudança de conceito.

Figura 5.8 - Experimento 3 - *Silhouette*.



Valores médio de *silhouette* para os quatro algoritmos nas duas versões do experimento 3. Acima quando $H = 300$ e $\lambda = 0.1$, abaixo $H = 1000$ e $\lambda = 0.01$.

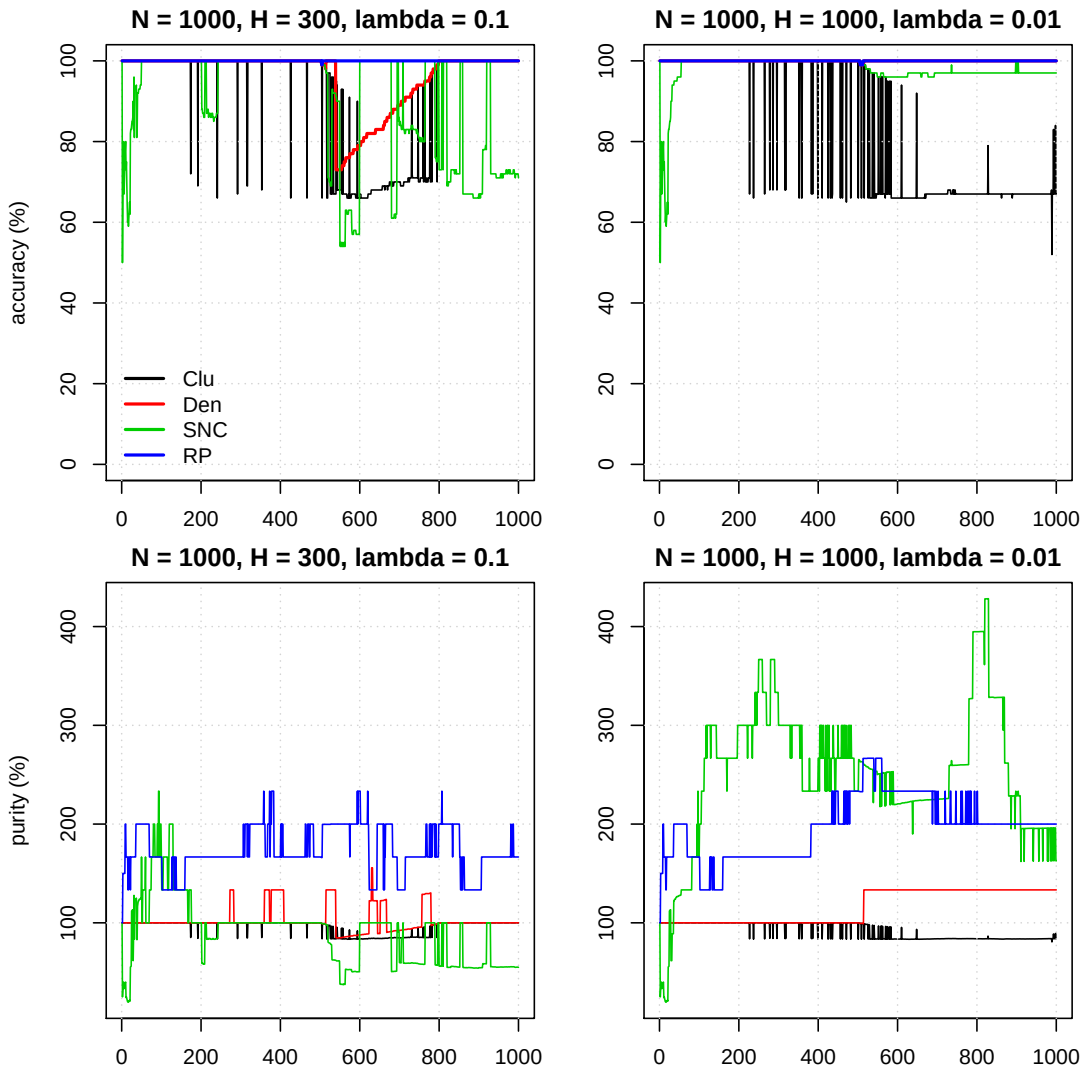
Fonte: Produção da Autora.

Dentre os três experimentos, considerando a primeira versão, o último, com mudança de conceito incremental é o que mais afeta os valores de *silhouette* dos algoritmos. O algoritmo *SNCStream* chega a obter valor zero em determinado momento. A metodologia RP obtêm valor entre 0.6 e 0.9 em quase toda a execução dos três experimentos para a primeira versão de janela deslizante e não parece afetar-se tanto com as mudanças de conceito quanto o algoritmo *DenStream*, por exemplo.

Já na segunda versão de janela deslizante dos três primeiros experimentos mostram uma mudança dos valores de *sihouette*. O algoritmo *SNCStream* tem pior desempenho nos três, com valores que se mantém entre 0.1 e 0.4. Os outros três algoritmos mantêm-se acima de 0.6 durante a maioria do tempo da execução, sendo que no terceiro experimento os

valores são mais baixos que os dois primeiros.

Figura 5.9 - Experimento 1 - Pureza e acurácia.



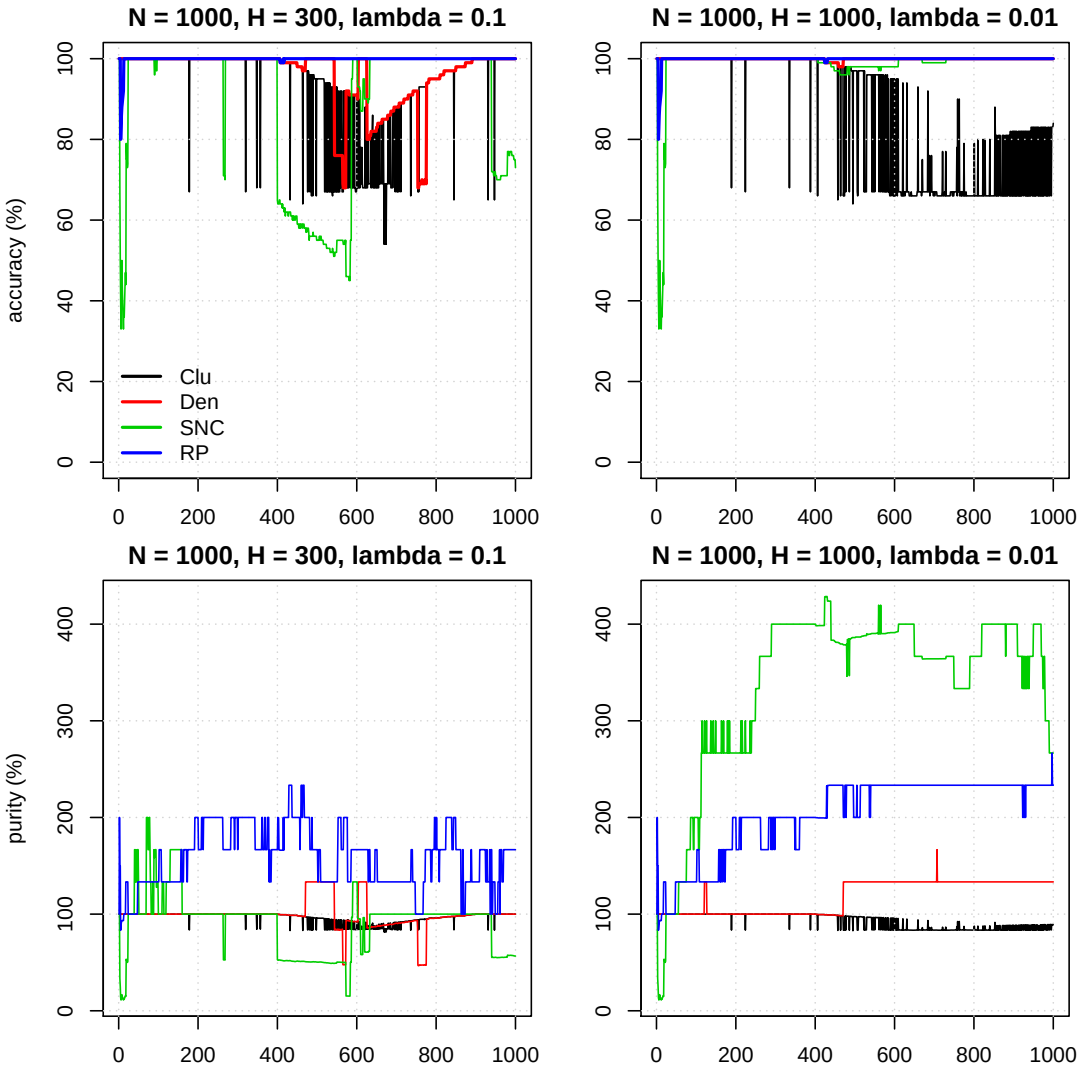
Acima na figura estão os valores de acurácia para as duas versões do experimento 1, abaixo estão os valores de pureza. A metodologia RP teve um bom desempenho de acurácia mas a pureza se manteve acima dos 100% na maior parte da execução das duas versões, indicando que há a formação de mais *clusters* que o necessário.

Fonte: Produção da Autora.

Na Figura 5.9 são mostrado os valores de pureza e acurácia para as duas versões do experimento 1. Em acurácia, a metodologia RP teve melhor desempenho em todo o

experimento, para as duas versões. Em pureza, o algoritmo *DenStream* teve valores melhores, portanto, conseguiu identificar melhor o número correto de *clusters*.

Figura 5.10 - Experimento 2 - Pureza e acurácia.



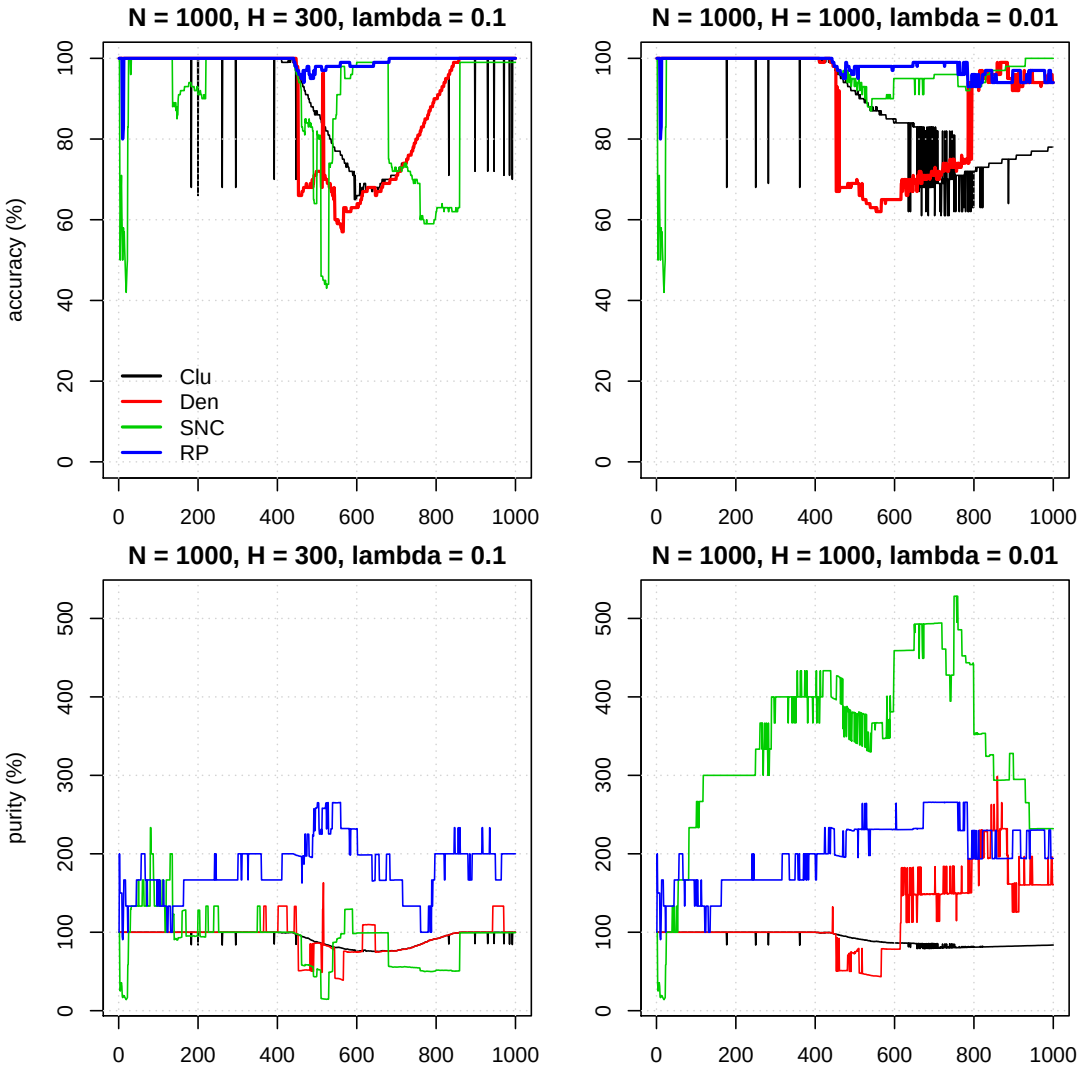
Acima na figura estão os valores de acurácia dos algoritmo para as duas versões do experimento 2. Nas duas, a metodologia RP obteve bons resultados, mesmo durante o processo de transição. Abaixo na figura estão os valores de pureza, apesar dos bons valores de acurácia, a metodologia mais uma vez forma mais *clusters* que o necessário.

Fonte: Produção da Autora.

Na Figura 5.10, os valores de acurácia da metodologia RP se mantêm em 100% na

maior parte do experimento 2, enquanto os outros algoritmos perdem a acurácia durante a transição dos dados entre os tempos $t = 400$ e $t = 600$. No entanto, os valores de pureza mostram que os algoritmos com número de *clusters* variável tiveram dificuldades em manter o valor certo, principalmente na segunda versão do experimento.

Figura 5.11 - Experimento 3 - Pureza e acurácia.

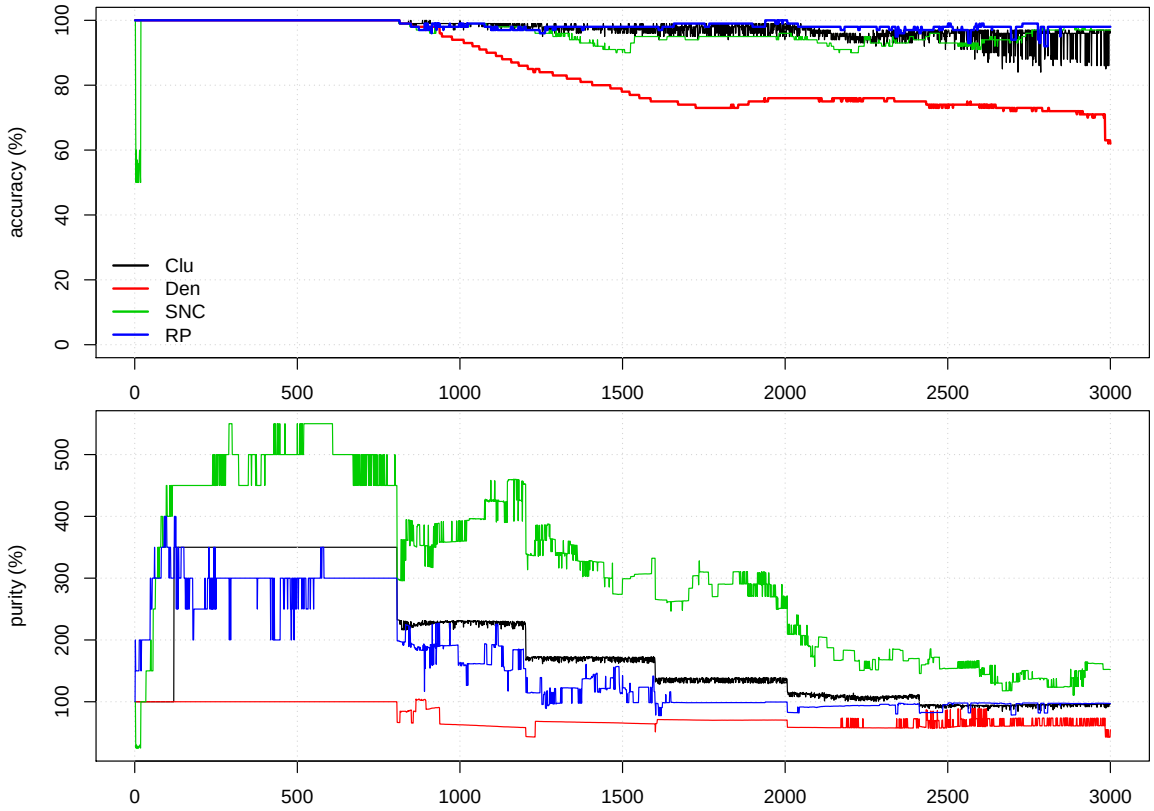


Acima na figura estão os valores de acurácia para as duas versões do experimento 3 e abaixo estão os valores de pureza também para as duas versões. Todos os algoritmos tiveram dificuldades no período de transição dos dados. Para as duas versões do experimento, a metodologia RP, apesar de ter os bons valores para a acurácia, não teve bom desempenho na pureza.

Fonte: Produção da Autora.

Os valores de pureza e acurácia obtidos pelos métodos no experimento 3 podem ser vistos na Figura 5.11. A metodologia RP novamente tem os melhores valores de acurácia, mas perde no desempenho de pureza. A segunda versão do experimento se mostrou especialmente difícil para os algoritmos.

Figura 5.12 - Experimento 4 - Pureza e acurácia.



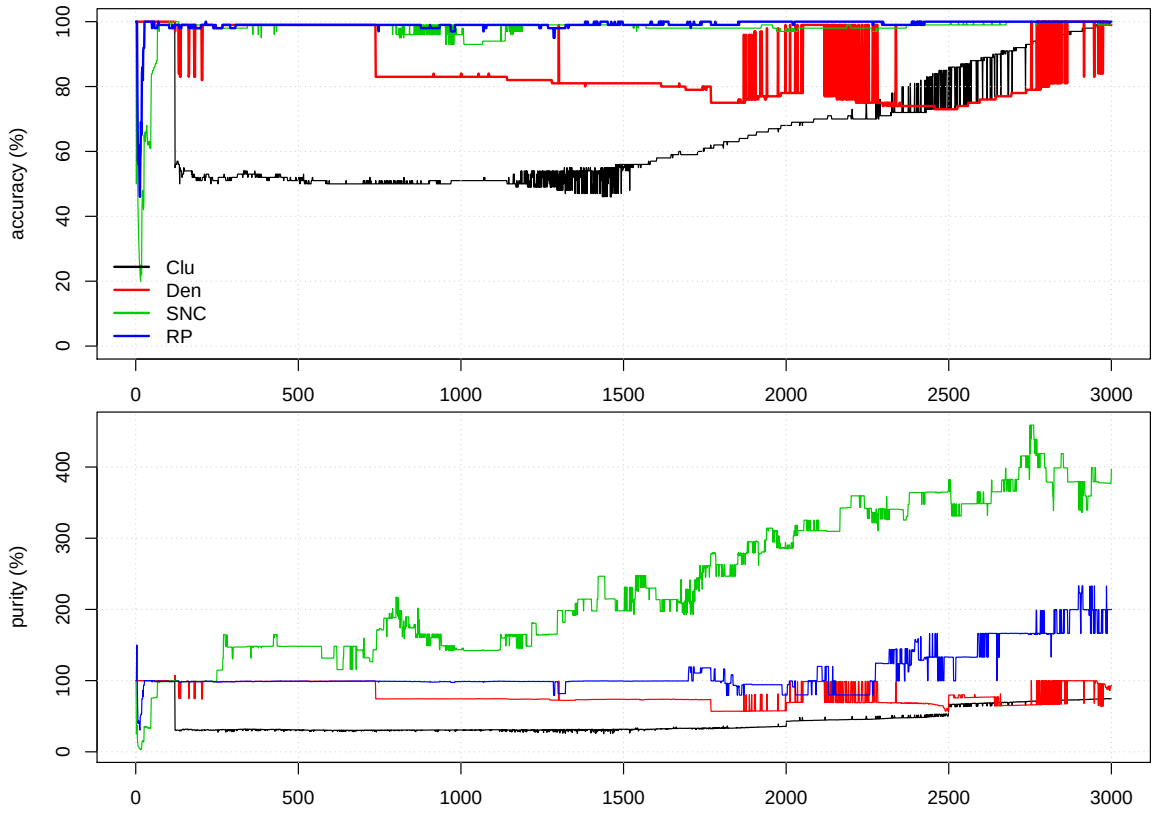
Medidas de SSQ, pureza e acurácia para o experimento 4. Observa-se que os algoritmos tem desempenho similar considerando a medida SSQ, no entanto, a acurácia e pureza mostram que os algoritmos criaram *clusters* com acurácia abaixo de 100% e que também formaram mais ou menos *clusters* que o necessário. Pelos valores, a metodologia RP teve um bom desempenho principalmente ao final.

Fonte: Produção da Autora.

Na Figura 5.12 são mostrado os valores das medidas pureza e acurácia para todos os algoritmos para o experimento 4. O algoritmo *DenStream* tem o pior desempenho em acurácia, não conseguindo separar corretamente os *clusters* do fluxo. No gráfico da

pureza, percebe-se que apesar do valor de acurácia ser bom para os outros algoritmos, eles formam mais *clusters* que o necessário.

Figura 5.13 - Experimento 5 - Pureza e acurácia.

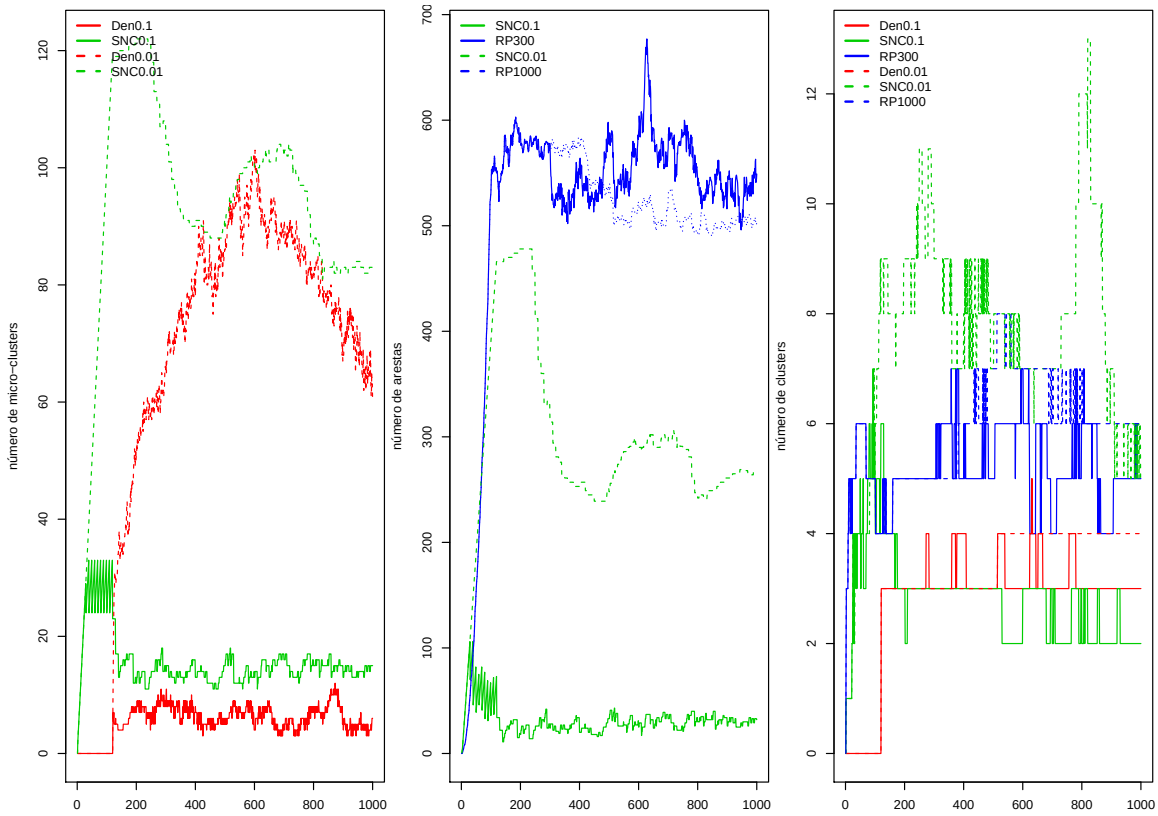


Medidas de SSQ, pureza e acurácia para os quatro algoritmos ao longo do experimento 5. Novamente as medidas de SSQ são similares entre todo os algoritmos e os valores de acurácia e pureza mostram que a metodologia RP e o algoritmo *SNCStream* formaram *clusters* amis homogêneos mas em maior quantidade que o necessário.

Fonte: Produção da Autora.

Os resultados para as pureza e acurácia dos quatro algoritmos para o experimento 5 podem ser observados na Figura 5.13. Os valores de acurácia para o algoritmo *SNCStream* e para a metodologia RP permanecem mais próximos a 100%, mas os valores de pureza mostram que estes formam mais *clusters* do que o necessário.

Figura 5.14 - Experimento 1 - Número de *microclusters*, arestas e *clusters*.



À esquerda, o gráfico contêm uma comparação entre os algoritmos *DenStream* e *SNCStream* quanto ao número de *microclusters*, o algoritmo *CluStream* e a metodologia RP não são considerados pois têm esse valor fixo em 100. Ao meio, a quantidade de arestas é comparada entre o algoritmo *SNCStream* e a metodologia RP, que são os métodos que utilizam redes complexas. À direita, o número de *clusters* inferido pelos métodos durante a execução, exceto *CluStream* que tem esse valor fixo. Percebe-se que o comportamento dos números de *clusters* é similar ao valor de pureza capturado na figura anterior.

Fonte: Produção da Autora.

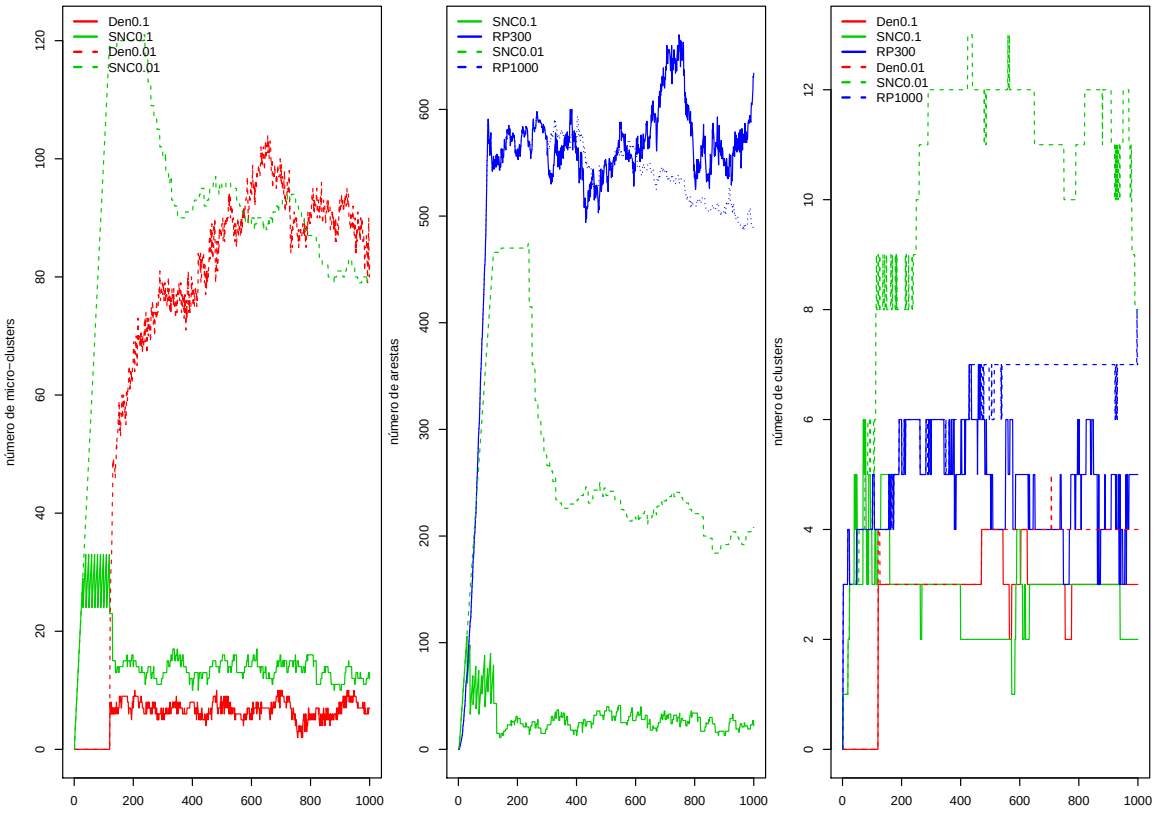
Na Figura 5.14 é possível observar as medidas quanto ao número de *microclusters*, número de arestas e número de *clusters*. No primeiro gráfico é comparado o número de *microclusters* dos algoritmos *DenStream* e *SNCStream*, já que esse número varia para os dois. O algoritmo *CluStream* e a metodologia RP mantêm esse número fixo.

Para os dois valores de λ, os dois algoritmos, *DenStream* e *SNCStream*, têm valores semelhantes, indicando que os dois identificaram potencial-*mc* de forma similar.

No segundo gráfico é comparado o número de arestas entre o algoritmo *SNCStream* e a metodologia RP e percebe-se que o número de arestas para a RP é bem maior nas duas versões dos algoritmos.

Por fim, no terceiro e último gráfico é feita a comparação do número de *clusters* que mostra exatamente o que foi inferido pelos valores de pureza e acurácia: o algoritmo *DenStream* tem melhor desempenho em manter três *clusters*. O *CluStream* não é considerado nessas medições pois o número de *clusters* $k = 3$ é um dos parâmetros do algoritmo, logo, não altera-se durante o experimento.

Figura 5.15 - Experimento 2 - Número de *microclusters*, arestas e *clusters*.

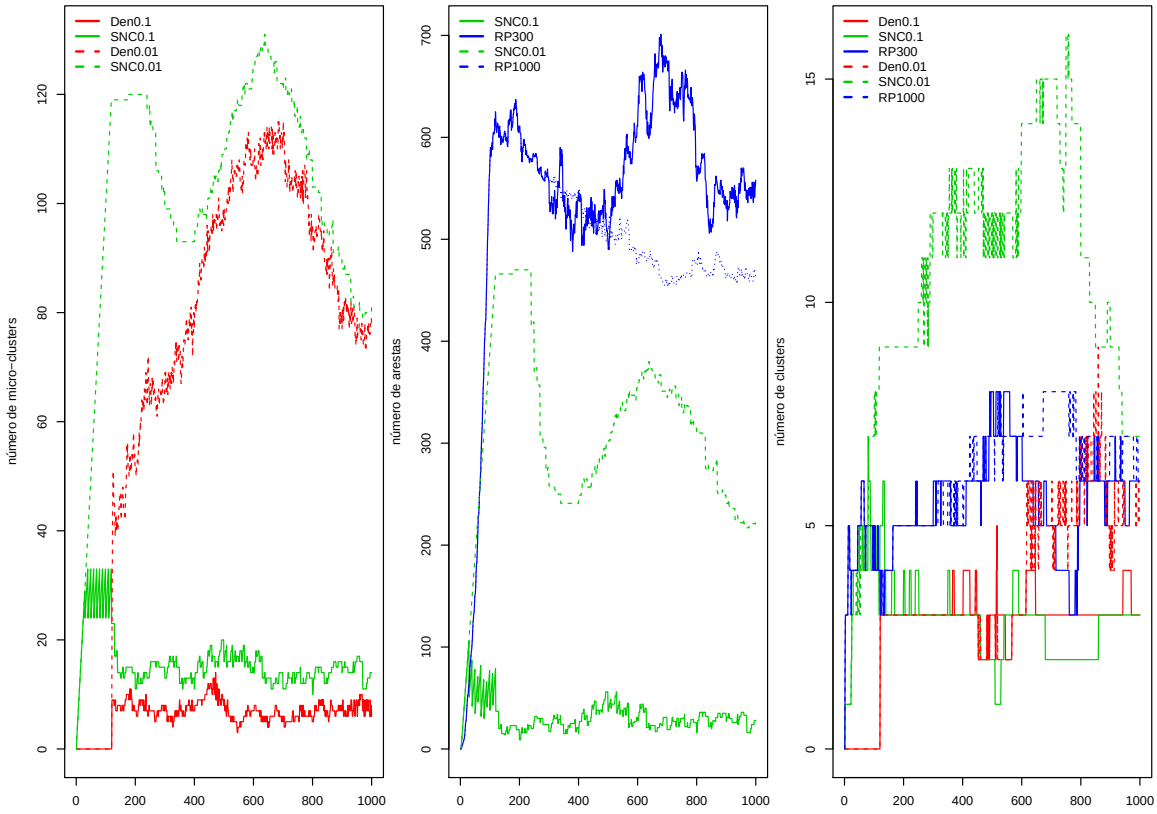


À esquerda, compara-se a quantidade de *microclusters* entre os algoritmos *DenStream* e *SNCStream*. Ao meio, compara-se o número de arestas entre o algoritmo *SNCStream* e a metodologia RP. À direita, o número de *clusters*. Os resultados deses testes são parecidos com os obtidos pelo experimento 1, indicando que as estruturas mantêm o formato mesmo neste outro cenário.

Fonte: Produção da Autora.

Na Figura 5.15, observa-se que os resultados de número de *microclusters*, arestas e *clusters* são semelhantes ao do experimento 1 nas duas versões. Os algoritmos *DenStream* e *SNCStream* mantém os valores parecidos de *microclusters*, a metodologia RP forma novamente mais arestas que o *SNCStream* e, por fim, o número de *clusters* reflete o que já havia sido visto pelos valores de pureza.

Figura 5.16 - Experimento 3 - Número de *microclusters*, arestas e *clusters*.

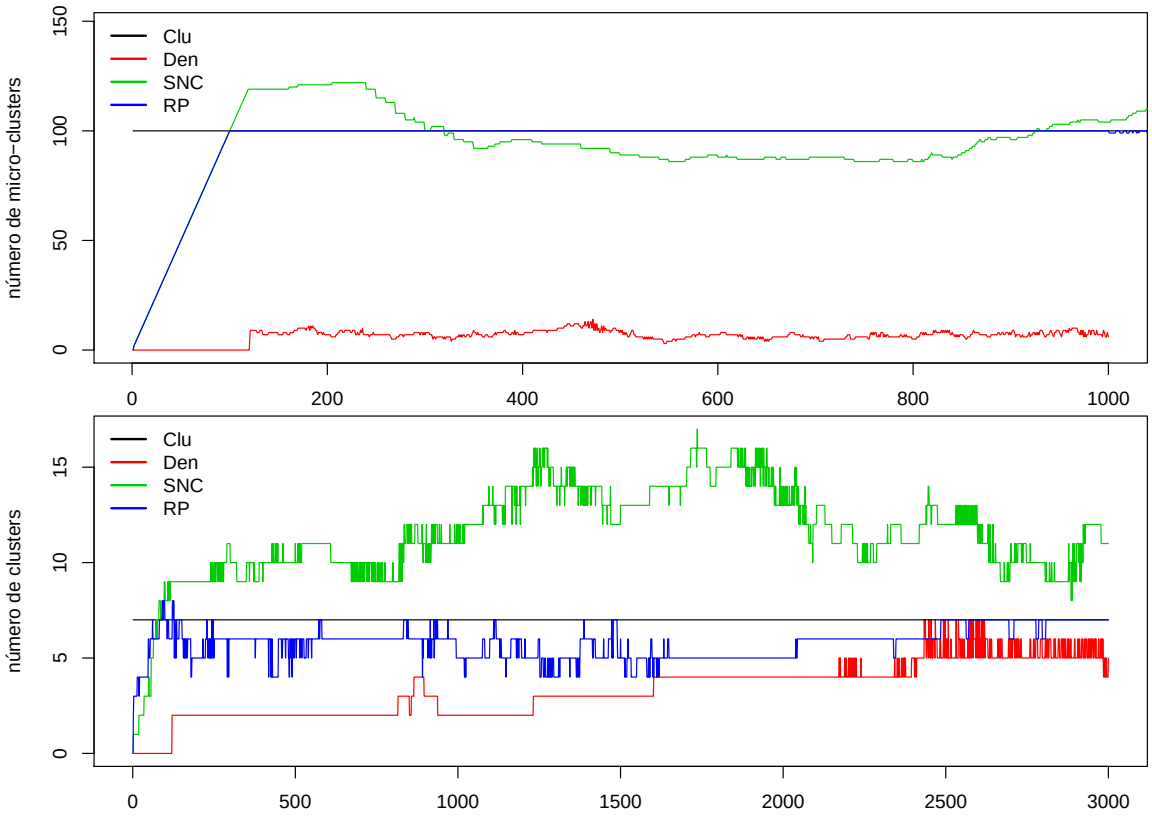


Medições para a quantidade de *microclusters*, arestas e *clusters* dos algoritmos nas duas versões do experimento 3. Os valores que podem ser observados mostram que o formato das estruturas são similares aos outros experimentos, mesmo considerando outro cenário de dados.

Fonte: Produção da Autora.

Na Figura 5.16, os valores para o número de *microclusters*, arestas e *clusters* tem desempenhos similares aos experimentos anteriores, apesar do desempenho de acurácia e pureza terem sido diferentes.

Figura 5.17 - Experimento 4 - Número de *microclusters* e *clusters*.

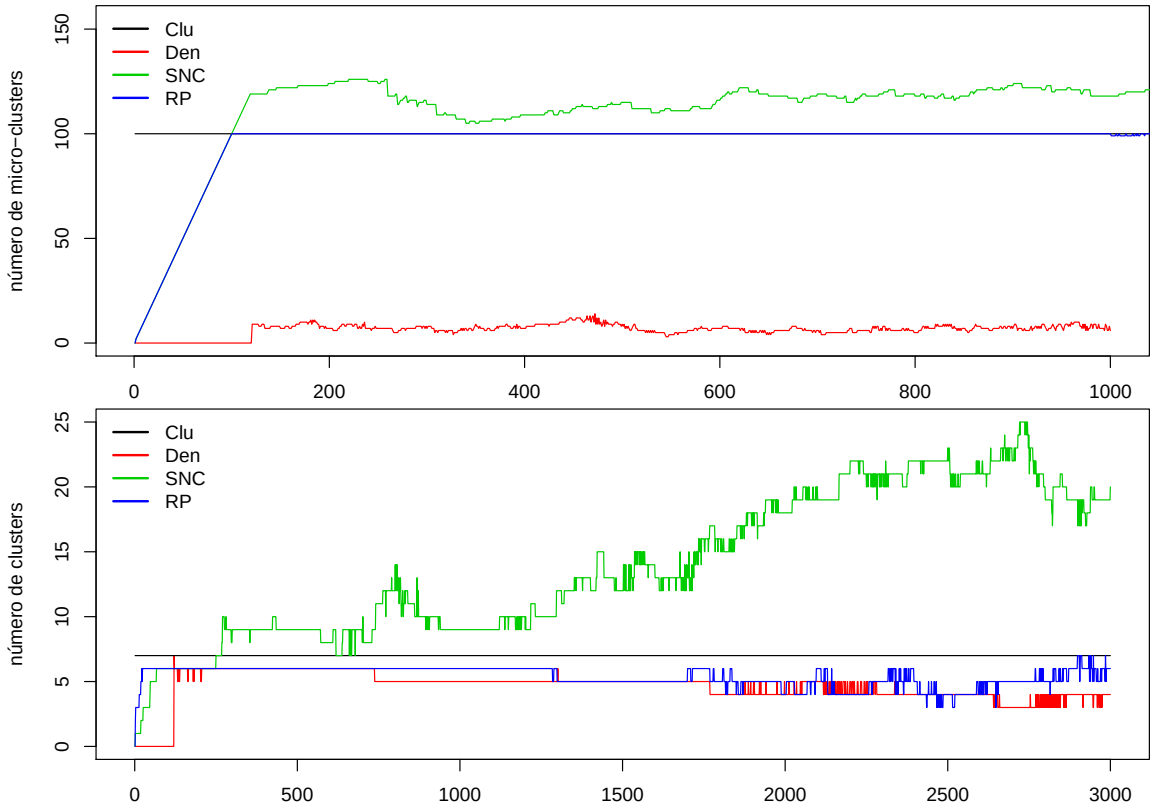


Acima na figura são mostradas as quantidades de *microclusters* de todos os algoritmos ao longo da execução do experimento. Abaixo as quantidades de *clusters* formados. Os poucos *microclusters* formados pelo algoritmo *DenStream* acabam prejudicando a formação correta de *clusters* ao final da execução. A metodologia RP, ao final, consegue obter corretamente o número de *clusters*, assim como o algoritmo *CluStream*, mas este último tem esse valor como parâmetro.

Fonte: Produção da Autora.

As quantidades de *microclusters* e *clusters* formados pelos algoritmos podem ser observadas na Figura 5.17. O algoritmo *DenStream* forma bem menos *microclusters* que os outros algoritmos (até porque o algoritmo *CluStream* e a metodologia RP têm esse valor fixo), por outro lado, o algoritmo *DenStream* e a metodologia RP, neste experimento, variam em número de *clusters* acompanhando, de certa forma, o nascimento ao longo da execução. O algoritmo *CluStream* tem esse valor fixo desde o início em $k = 7$ e o algoritmo *SNCStream* acaba formando sempre mais *clusters* que o necessário.

Figura 5.18 - Experimento 5 - Número de *microclusters* e *clusters*.

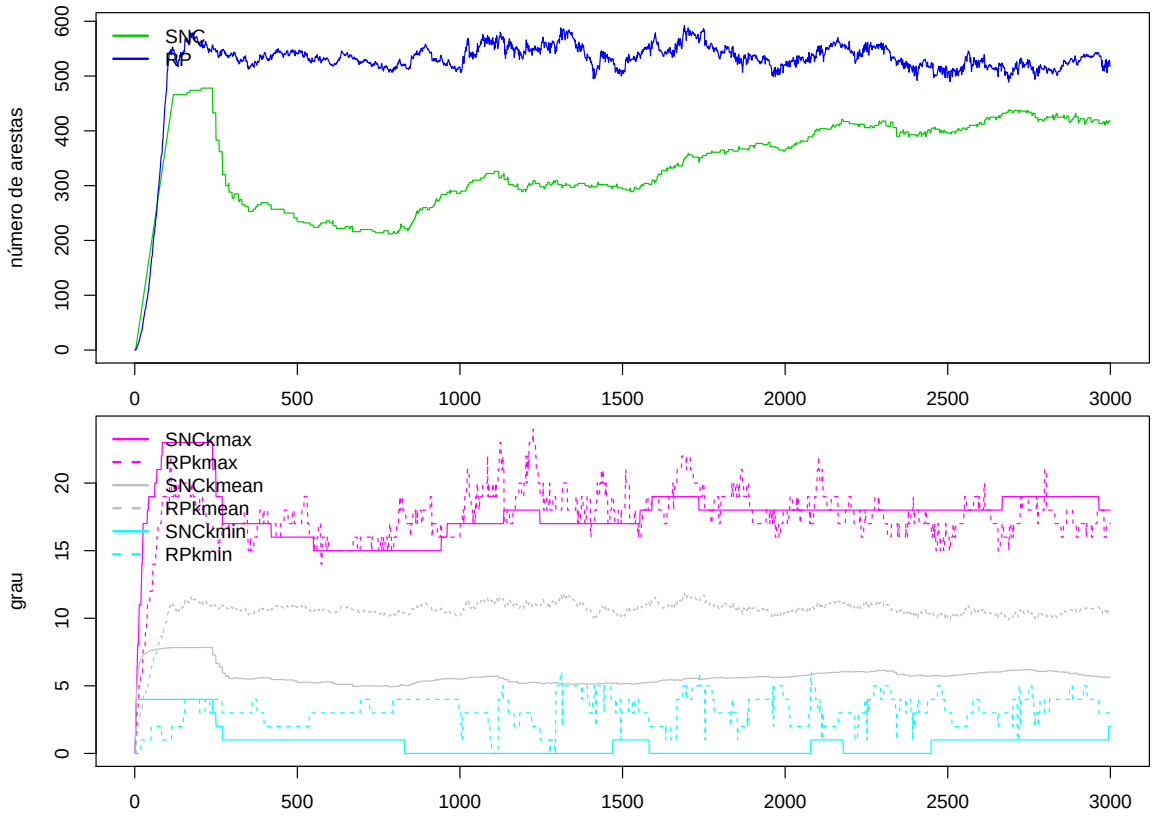


Acima na figura, é mostrada a comparação entre os números de *microclusters* dos algoritmos. Abaixo, a comparação entre os números de *clusters*. Os valores mostram que o algoritmo *DenStream* e a metodologia RP diminuem o número de *clusters* a princípio, mas ao final, só o *DenStream* forma três *clusters*.

Fonte: Produção da Autora.

Na Figura 5.18, são comparados os números de *microclusters* e *clusters* entre os métodos. O algoritmo *DenStream* tem o melhor desempenho considerando essas medidas, pois mantém menos *microclusters*, mas forma na maioria das vezes o número correto de *clusters*.

Figura 5.19 - Experimento 4 - Número de arestas e graus máximo, mínimo e médio dos vértices.



Acima na figura, é mostrada a comparação do número de arestas entre o algoritmo *SNCStream* e a metodologia RP. Abaixo, são mostrados os valores de grau máximo, mínimo e médio, também para os mesmos dois métodos, ao longo do experimento. O fato do grau médio da rede da metodologia RP ser maior que o da rede do algoritmo *SNCStream* justifica o número maior de arestas.

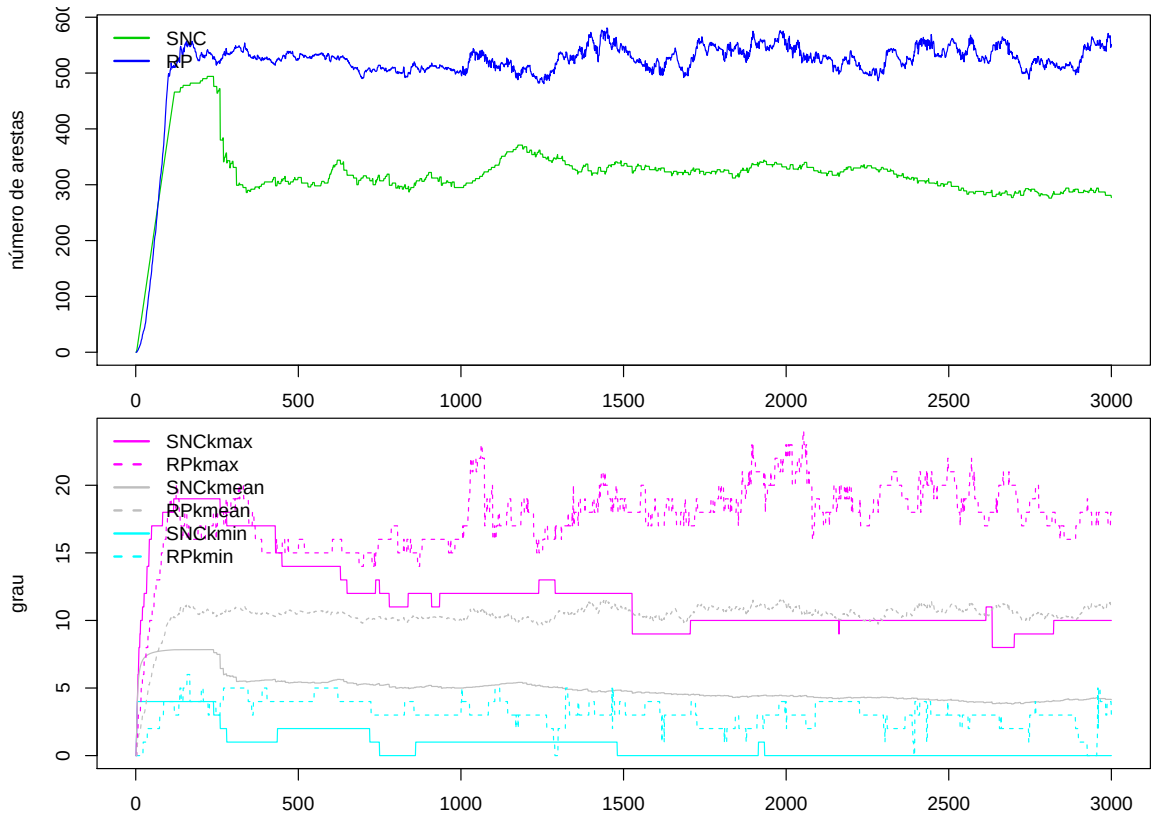
Fonte: Produção da Autora.

Na Figura 5.19 é comparado o número de arestas entre o algoritmo *SNCStream* e a metodologia RP e também os graus máximo, mínimo e médio. A metodologia RP mantêm, assim como nos experimentos anteriores, mais arestas que o algoritmo *SNCStream*, no entanto, observa-se que o número de arestas da RP é estável ao contrário do algoritmo *SNCStream* que parece crescer esse número junto com os dados observados.

O número elevado de arestas da metodologia pode ser justificada pelos valores de grau médio que são maiores na rede RP do que na rede do algoritmo *SNCStream*. Ou seja, o gráfico abaixo na Figura 5.19 mostra que a rede da metodologia mantêm mais conexões entre os vértices apesar de não definir nenhum parâmetro para esse propósito como é o

caso do parâmetro ω do algoritmo *SNCStream*.

Figura 5.20 - Experimento 5 - Número de arestas e graus máximo, mínimo e médio dos vértices.

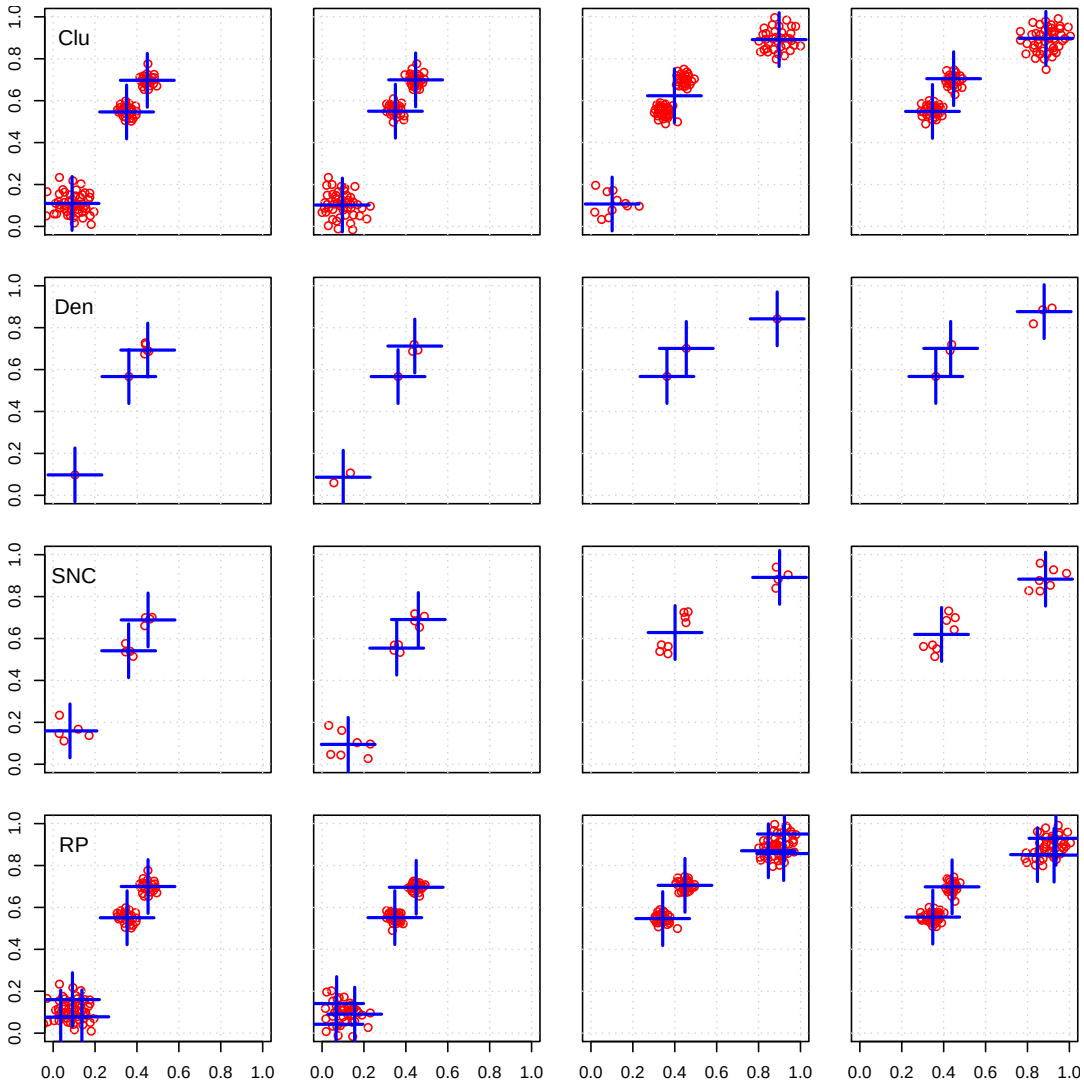


Comparação entre o número de arestas e os graus máximo, mínimo e médio das redes do algoritmo *SNCStream* e da metodologia RP. Assim como no experimento anterior, observa-se que a metodologia forma bem mais arestas que são justificadas pelo grau médio que é maior.

Fonte: Produção da Autora.

As comparações entre arestas e graus dos vértices das redes do algoritmo *SNCStream* e da metodologia RP são mostradas na Figura 5.20. Novamente, a metodologia mantém bem mais arestas que o algoritmo *SNCStream* que é justificado pelo maior número de grau médio ao longo da execução.

Figura 5.21 - Experimento 1 - Formação de *microclusters* e *clusters* (H = 300, $\lambda = 0.1$).

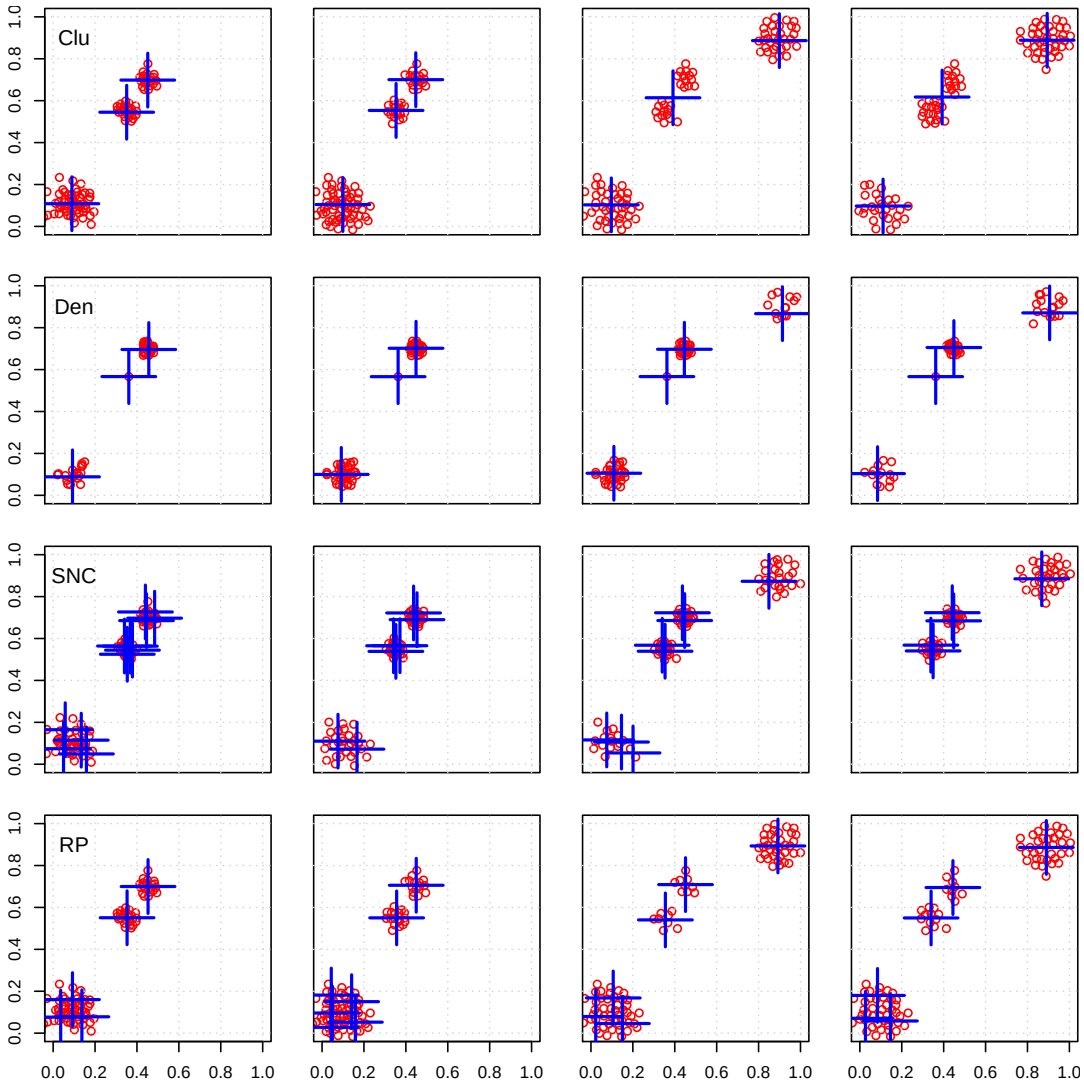


As sequências de quatro *snapshots* ($t = 250$, $t = 500$, $t = 750$ e $t = 1000$) em cada linha refletem os *microclusters* (círculos) e *clusters* (cruzes) para cada algoritmo durante a primeira versão do experimento 1. Ao final, todos menos o algoritmo *SNCStream* conseguiram capturar o novo conceito dos dados. Entre o *CluStream*, *DenStream* e a metodologia RP, o segundo utilizou menos recursos de máquina (espaço) e obteve o melhor desempenho entre todos quanto ao número de *clusters*.

Fonte: Produção da Autora.

Nas Figuras 5.21 e 5.22 são mostradas quais são as configurações de *microclusters* e *clusters* de cada algoritmo nos tempos $t = 250$, $t = 500$, $t = 750$ e $t = 1000$. Cada *microcluster* é representado por círculos vermelhos e cada *cluster* é representado por uma cruz azul.

Figura 5.22 - Experimento 1 - Formação de *microclusters* e *clusters* (H = 1000, $\lambda = 0.01$).



As sequências de quatro *snapshots* ($t = 250$, $t = 500$, $t = 750$ e $t = 1000$) em cada linha refletem os *microclusters* (círculos) e *clusters* (cruzes) para cada algoritmo durante a segunda versão do experimento 1. Ao contrário da primeira versão, o algoritmo *SNCStream* foi o que melhor apreendeu o novo conceito primeiramente, no entanto, manteve mais *clusters* que o necessário. O algoritmo *DenStream* e a metodologia RP no tempo $t = 1000$ ainda mantinham *clusters* referentes à antiga posição do *clusters* que sofreu o movimento de *shift*, mas o *DenStream* teve melhor desempenho no número de *clusters*. O algoritmo *CluStream*, por ter o parâmetro *k* fixo em 3, mesclou os dois *clusters* fixos em apenas um.

Fonte: Produção da Autora.

Quando $H = 300$ e $\lambda = 0.1$, os quatro métodos conseguem "esquecer" a antiga posição

do *cluster* que sofre o *shift* e se adaptar ao novo conceito, já a partir do tempo $t = 500$. No entanto, o algoritmo *SNCStream*, como pode ser visto no tempo $t = 750$, considera que os dois outros *clusters* são um só após o *shift*.

Ainda observando a Figura 5.21, verifica-se também a diferença entre o número de *microclusters* entre os métodos *DenStream*/*SNCStream* e *CluStream*/RP. Os dois primeiros tem essa quantidade variável, enquanto os dois últimos têm esse valor fixo.

Quando $H = 1000$ e $\lambda = 0.01$, os métodos *DenStream* e *SNCStream* acabam formando mais *microclusters* já que há mais informação a ser mantida. O único método que conseguiu esquecer a antiga posição do *cluster* que sofre o *shift* até o tempo $t = 1000$, foi o *SNCStream*.

A metodologia RP no tempo $t = 1000$ ainda mantêm 4 *clusters* na antiga posição do *cluster*, isso porque, ao contrário dos métodos *DenStream* e *SNCStream*, não utiliza uma função que decai a importância dos objetos ao longo do tempo dentro da estrutura.

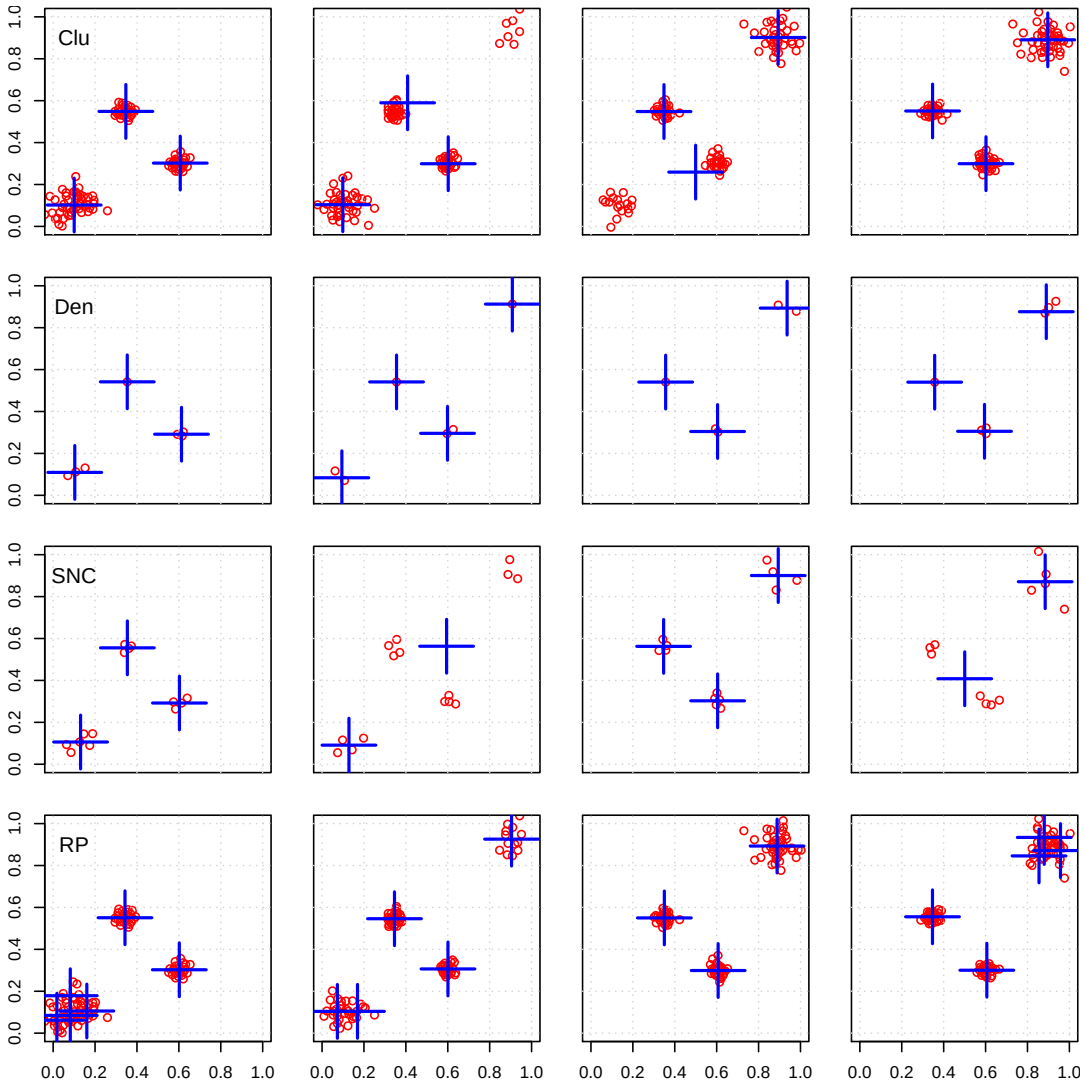
O algoritmo *CluStream* como sabe *a priori* que são formados três *clusters* tem um desempenho fraco, já que não tem tempo de "esquecer" a antiga posição e acaba, por esse motivo, juntando os outros dois *clusters* em um só.

As posições dos *microclusters* e *clusters* formados nas duas versões do experimento 2 podem ser vistos nas Figuras 5.23 e 5.24. A primeira figura contém os gráficos para quando os valores referentes aos parâmetros da janela são $H = 300$ e $\lambda = 0.1$, e, na segunda, quando $H = 1000$ e $\lambda = 0.01$.

Novamente, assim como no experimento 1, na primeira versão o algoritmo *DenStream* se sai melhor que os outros, a metodologia RP forma mais *clusters* que o necessário, o algoritmo *SNCStream* junta os dois *clusters* fixos em um só e o *CluStream* é o que demora mais a esquecer a antiga posição do *cluster* que sofre o *shift*.

Na segunda versão do experimento 2, também, assim como no experimento 1, o algoritmo *SNCStream* é o único que esquece a antiga posição ao final da execução, no entanto, forma muito mais *clusters* que o necessário. Os algoritmos *CluStream* e *DenStream* e a metodologia RP continuam a manter formação de *microclusters* e *clusters* na antiga posição do *cluster*, sendo que o *CluStream* forma apenas *microclusters* na posição já que seu parâmetro $k = 3$ é fixo. No tempo $t = 1000$, o algoritmo *DenStream* ainda mantêm um *cluster* na posição antiga e a metodologia RP mantêm diversos *clusters* na antiga posição, apesar de manter só um na posição nova.

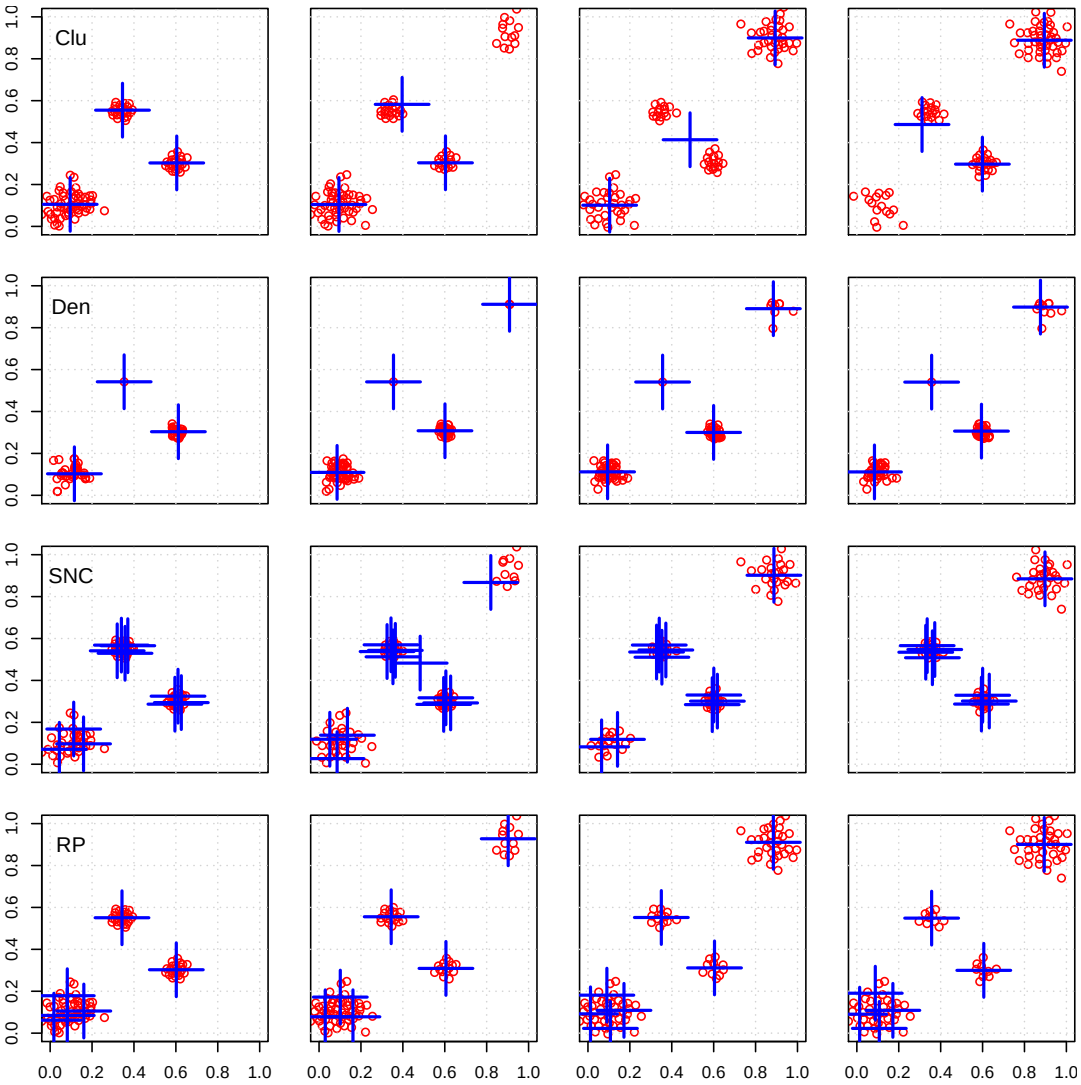
Figura 5.23 - Experimento 2 - Formação de *microclusters* e *clusters* (H = 300, $\lambda = 0.1$)



As quatro sequências de *snapshots* em cada linha refletem os *microclusters* (círculos) e *clusters* (cruzes) para cada algoritmo durante a execução do experimento 2 na primeira versão. A metodologia RP recuperou-se da antiga posição do *cluster* já no tempo $t = 750$ mas no tempo $t = 1000$ inferia mais *clusters* que o necessário. O algoritmo *SNCStream* acaba formando apenas dois *clusters* ao final da execução. Os algoritmos *Den Stream* e *CluStream* ao final da execução formam três *clusters* corretamente.

Fonte: Produção da Autora.

Figura 5.24 - Experimento 2 - Formação de *microclusters* e *clusters* (H = 1000, $\lambda = 0.01$)

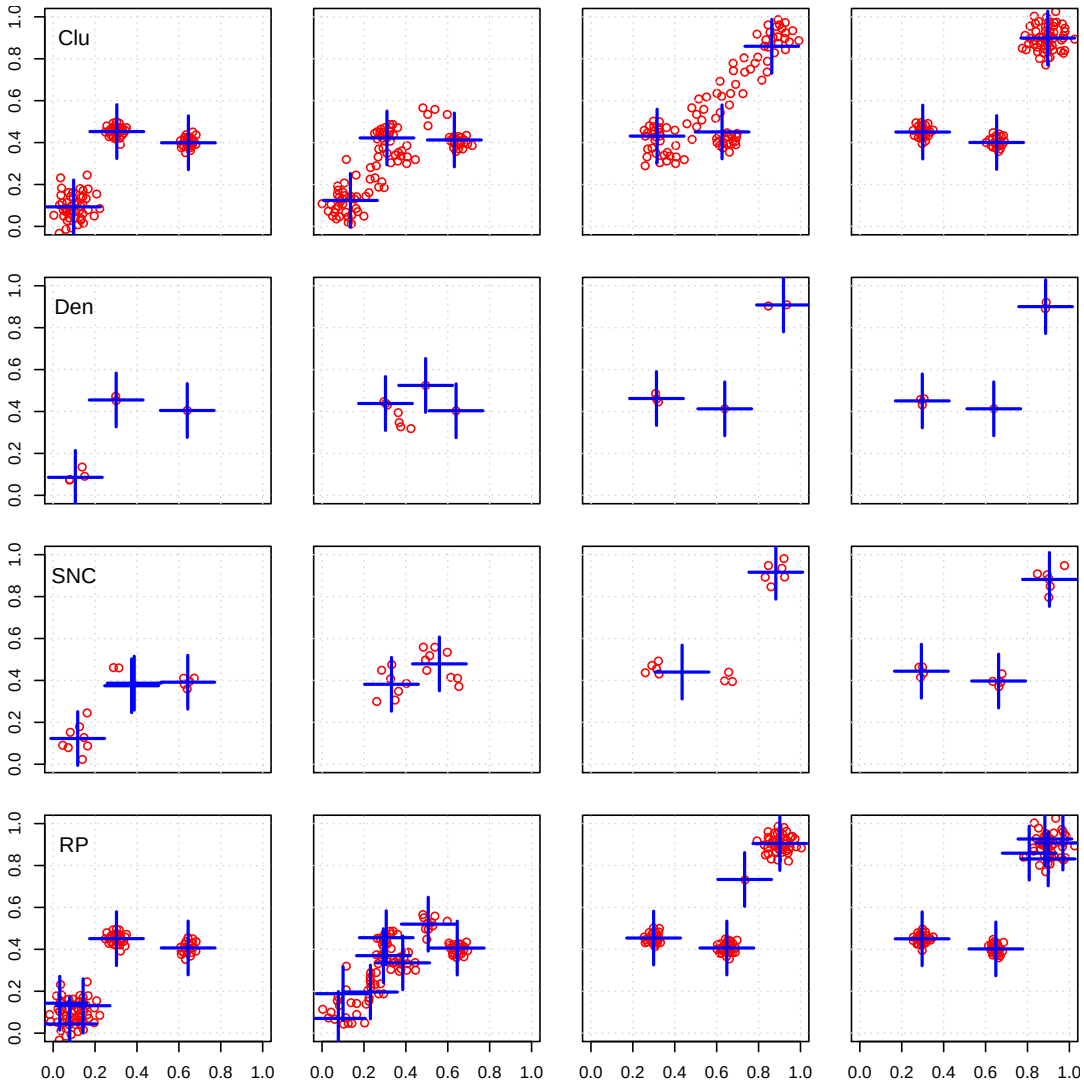


As quatro sequências de *snapshots* em cada linha refletem os *microclusters* (círculos) e *clusters* (cruzes) para cada algoritmo durante a execução do experimento 2 na segunda versão. Assim como no experimento 2, somente o algoritmo *SNCStream* consegue esquecer a antiga posição do *cluster* que sofre o movimento de *shift*, mas forma muito mais *clusters* que o necessário. A metodologia RP mantêm ainda mais *clusters* na antiga posição do que no experimento anterior. O *DenStream* e o *CluStream* tiveram melhores desempenhos nesta versão do experimento.

Fonte: Produção da Autora.

Nas Figuras 5.25 e 5.26 podem ser vistas as formações de *microclusters* e *clusters* dos algoritmos para o experimento 3. A primeira figura considera quando os parâmetros de janela são $H = 300$ e $\lambda = 0.1$ e a segunda figura quando $H = 1000$ e $\lambda = 0.01$.

Figura 5.25 - Experimento 3 - Formação de *microclusters* e *clusters* (H = 300, $\lambda = 0.1$).



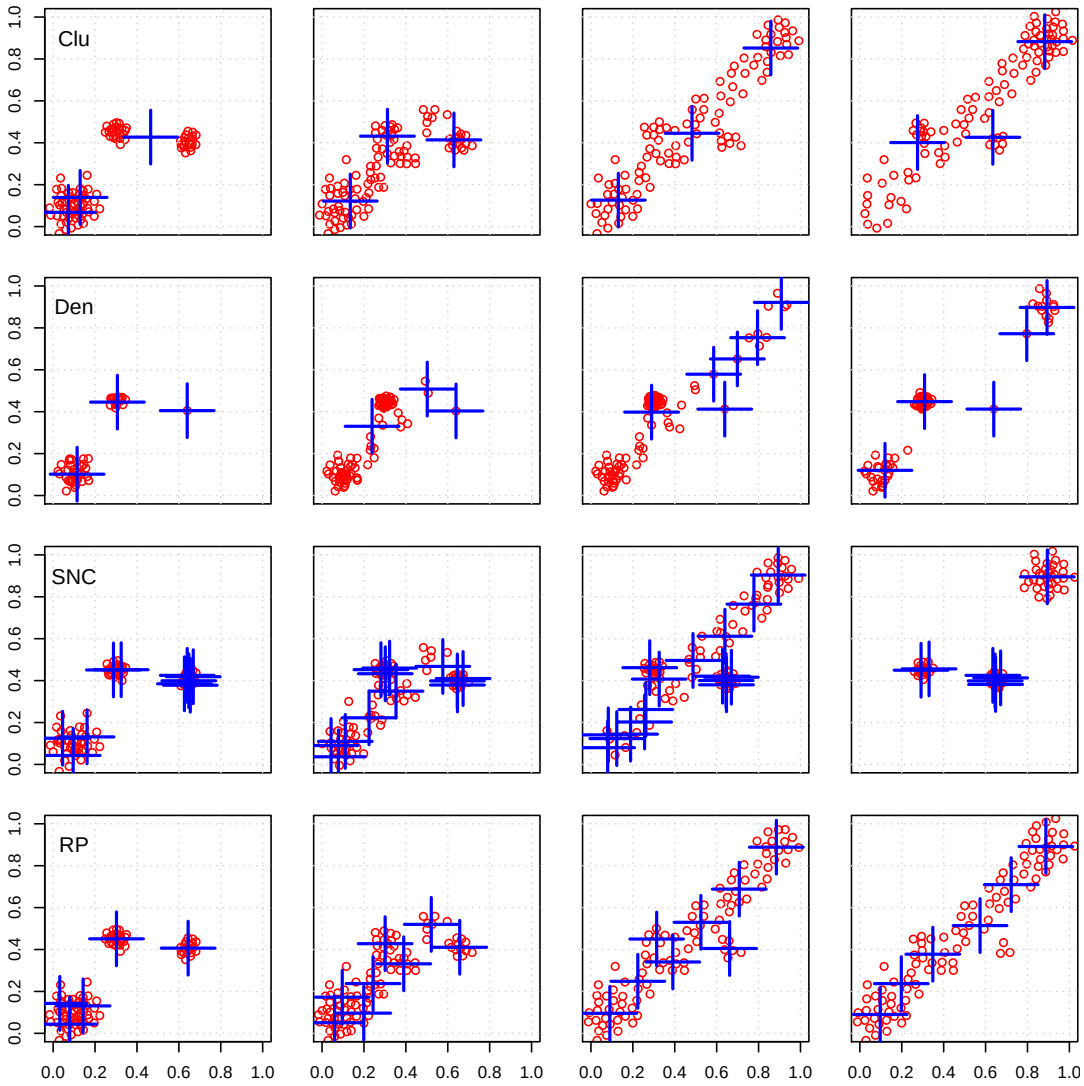
Em cada linha uma sequência de quatro *snapshots* nos tempos $t = 250$, $t = 500$, $t = 750$ e $t = 1000$ para cada algoritmo durante a primeira versão do experimento 3. Os *microclusters* são representados pelos círculos e os *clusters* por cruzes. Todos os algoritmos conseguem se adaptar ao novo conceito dos dados ao final da execução, mas a metodologia RP mantêm, mais uma vez, mais *clusters* do que o necessário.

Fonte: Produção da Autora.

As sequências de *snapshots* observadas na Figura 5.25, mostram que apesar da mudança de conceito do tipo *shift* incremental causar mais contratempos no espaço de dados, a janela menor ajuda os algoritmos a se adaptarem ao novo conceito mais facilmente, com todos marcando as três posições corretamento ao final. No entanto, a metodologia RP

mantém um número bem maior de *clusters* que o necessário.

Figura 5.26 - Experimento 3 - Formação de *microclusters* e *clusters* (H = 1000, $\lambda = 0.01$).



Em cada linha uma sequência de quatro *snapshots* nos tempos $t = 250$, $t = 500$, $t = 750$ e $t = 1000$ para cada algoritmo durante a segunda versão do experimento 3. Os *microclusters* são representados pelos círculos e os *clusters* por cruzes. Os algoritmos tem maior dificuldade para se adaptar ao novo conceito nesta versão do experimento, somente o algoritmo *SNCStream* ao final chega às três posições dos *clusters* corretamente mas forma mais *clusters* que o necessário. O *CluStream* chega às posições corretas dos *clusters* mas mantêm vários *microclusters* ao longo do caminho do movimento. Nem o *DenStream* nem a metodologia RP conseguem esquecer a antiga posição do *cluster*, mas a metodologia RP tem desempenho ainda pior.

Fonte: Produção da Autora.

Na primeira versão do experimento 3, é interessante notar na segunda coluna da Figura 5.25 o comportamento dos algoritmos no que se refere ao "caminho" que o centro do *cluster* percorre ao se movimentar.

Na Figura 5.26, quando é considerado um tamanho maior para a janela, $H = 1000$, os algoritmos têm desempenhos diversos. No tempo $t = 1000$, o algoritmo *CluStream* forma as posições dos 3 *clusters* corretamente, mas mantém vários *microclusters* ao longo do caminho do movimento.

O algoritmo *DenStream*, ao final da execução, forma cinco *clusters*, três das posições corretas, um na posição anterior do *cluster* que se movimenta e um na posição imediatamente anterior do caminho percorrido. O algoritmo *SNCStream* no tempo $t = 750$ forma vários *clusters* ao longo de todo o caminho do movimento mas se recupera no tempo $t = 1000$, apesar de ainda manter mais *clusters* que o necessário.

Por fim, a metodologia RP não consegue acompanhar o movimento corretamente até o tempo $t = 1000$ nessa versão do experimento 3, seria necessário mais tempo acompanhando os *clusters* formados para verificar se a metodologia iria eventualmente se adaptar ao novo conceito dos dados.

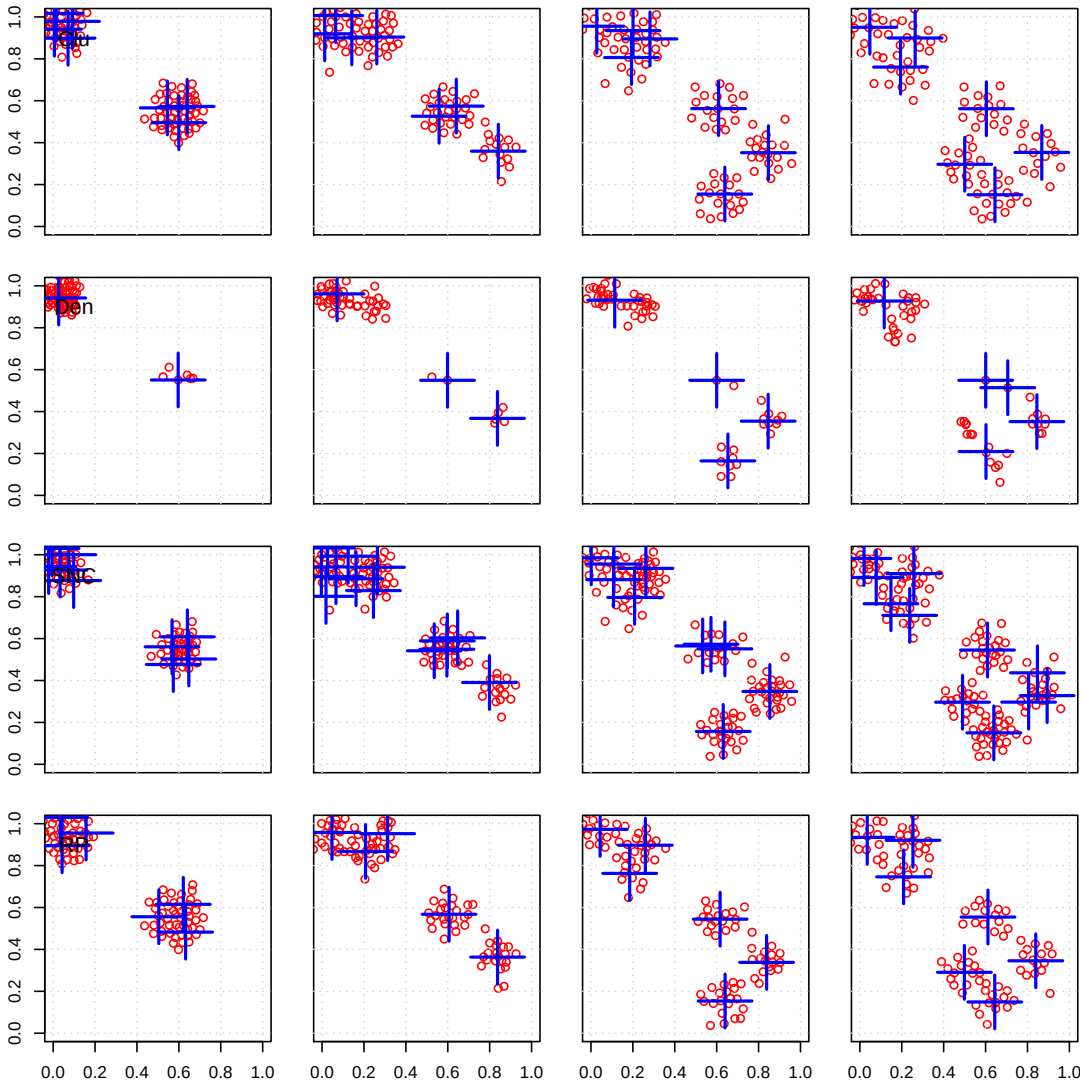
Na Figura 5.27, são mostrada as formações de *microcluters* e *clusters* dos algoritmos ao longo do experimento 4. O algoritmo *CluStream* tem parâmetro $k = 7$ definido desde o início da execução e, ao final, os 7 *clusters* são identificados corretamente.

O algoritmo *DenStream* não consegue identificar os sete *clusters* ao final da execução, marcando os três *clusters* do canto superior esquerdo como um só. O algoritmo *SNCStream* forma bem mais *clusters* que o necessário ao final da execução, como já havia sido percebido pelos suas medidas de pureza.

Por fim, a metodologia RP foi o único método que conseguiu formar corretamente todos os *clusters* ao londo de toda a execução, apesar que a acurácia não é 100% ao final, indicando que há objetos que foram mal agrupados.

As formações de *microclusters* e *clusters*, que podem ser vistas na Figura 5.28, confirmam o que as medidas anteriores mostraram. O *CluStream* manteve três *clusters* desde o início e identifica a posição dos restantes corretamente. O algoritmo *DenStream* mantém uma divisão em *clusters* satisfatória mas, às vezes, une mais de um *cluster*.

Figura 5.27 - Experimento 4 - Formação de *microclusters* e *clusters*.

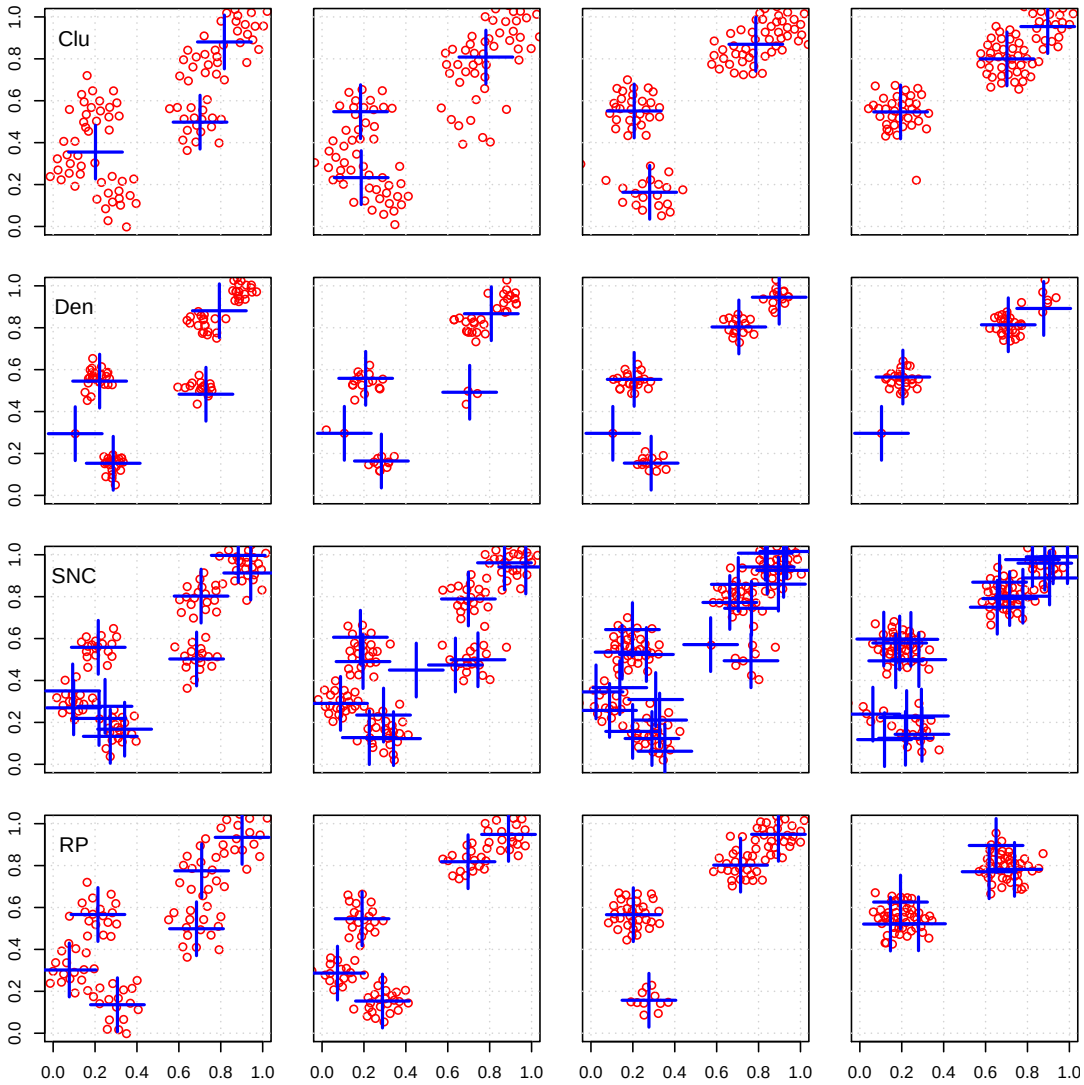


Cada linha da figura mostra a formação de *microclusters* e *clusters* dos quatro algoritmos em *snapshots* nos tempos $t = 750$, $t = 1500$, $t = 2250$ e $t = 3000$. Neste experimento, a metodologia RP teve desempenho melhor que os outros pois identificou corretamente a posição dos sete *clusters* finais sem a necessidade de parâmetros para essa finalidade.

Fonte: Produção da Autora.

O algoritmo *SNCStream* ao longo de todo o experimento forma muito mais *clusters* que o necessário, como já havia sido mostrado pelos valores de pureza. Por fim, a metodologia RP mantém boa divisão de *clusters* inicialmente mas ao final acaba unindo dois em um só.

Figura 5.28 - Experimento 5 - Formação de *microclusters* e *clusters*.



Cada linha da figura mostra a formação de *microclusters* e *clusters* dos quatro algoritmos em *snapshots* nos tempos $t = 750$, $t = 1500$, $t = 2250$ e $t = 3000$. Neste experimento, nenhum algoritmo teve desempenho satisfatório, todos formaram quantidades erradas de *clusters* ao longo dos experimentos.

Fonte: Produção da Autora.

6 ESTUDO DE CASO: DADOS DE USO E COBERTURA DA TERRA

Neste capítulo, a metodologia Rede Protótipo é aplicada a dados reais de observação da Terra. Os dados são provenientes do instrumento MODIS a bordo dos satélites Terra e Aqua que são administrados pela NASA. Eles descrevem a evolução do uso e cobertura da terra no estado do Mato Grosso no Brasil.

6.1 Descrição dos dados

O acesso aos dados deste estudo de caso são provenientes dos esforços de governos e instituições para analisar os possíveis fatores relacionados às mudanças climáticas. Estudos demonstram que os componentes atmosféricos estão profundamente integrados com os oceanos e a superfície terrestre, os quais afetam a atmosfera em contrapartida (BORIAH et al., 2008; BORIAH, 2010).

Ou seja, para entender o planeta e o clima é preciso considerar e observar a atmosfera terrestre do ponto de vista de fora do globo, algo que só é possível com o uso de tecnologia espacial (GOMEZ et al., 2016). Os dados são utilizados como alimento para a construção de modelos de representação dinâmicos para ajudar a entender o planeta como uma integração de três partes, oceanos, superfície terrestre e atmosfera, como um único ecossistema (FREITAS et al., 2011; PETITJEAN; WEBER, 2014).

Os dados são coletados a partir de diversos instrumentos a bordo de diversos satélites, trazendo como resultado uma captura de vários fatores atmosféricos terrestres, geralmente como uma série temporal, onde um mesmo tipo de dado é lido ao longo de intervalos fixos no tempo, e essas informações e novas leituras mantêm-se ativas indeterminadamente (BOULILA et al., 2011).

Esses instrumentos coletam uma gama de informações diária e orquestradamente por um longo período de tempo, de maneira a obter uma representação de cada região dos oceanos, atmosfera e superfície terrestre (LAMBIN; LINDERMAN, 2006).

No caso dos dados que serão utilizados neste capítulo, a coleta ocorre desde fevereiro de 2000, quando o satélite Terra foi posto em órbita. Esse satélite pertence a um conjunto de satélites coordenados pelo *EOS - Earth Observing System*, financiado pelo comitê espacial Norte Americano (NASA). Esse sistema gera vários produtos que são distribuídos pela *internet* para estudo e uso das organizações, governos e institutos (LUNETTA et al., 2006; VERBESSELT et al., 2012; SHIMABUKURO et al., 2014).

Entre os produtos distribuídos há o MOD 13 que são dados gerados pelo sensor *Mo-*

derate Resolution Imaging Spectroradiometer (MODIS) e contêm valores de índices de vegetação. O MOD 13 tem valores que são formados a partir de transformações espectrais em uma ou mais bandas, esses valores formam alguns índices em um mesmo ponto do planeta por vários anos mas, dentre esses índices, dois são largamente utilizados na literatura (MAUS et al., 2019), são eles: *Normalized Difference Vegetation Index* (NDVI) (TUCKER, 1979) e *Enhanced Vegetation Index* (EVI) (JIANG et al., 2008).

Enquanto o primeiro é mais sensível à clorofila, o segundo é mais responsivo às variações do dossel da vegetação, analisando-os em conjunto é possível construir um modelo que possa prever cenários de variação na vegetação por ações naturais ou humanas (JUSTICE et al., 2002).

Com a coleta desses índices ao longo do tempo é possível acompanhar as mudanças sazonais, de curto- e longo-prazo, na vegetação. Comparando esses índices entre áreas vizinhas é possível verificar o alastre de um tipo de vegetação na superfície, por exemplo.

Toda essa análise dos dados no tempo e espaço gera ainda mais informações que podem ser utilizadas para os mais diversos fins, inclusive alimentar o próprio modelo. Essa enorme quantidade de dados continua a ser atualizada e pode alimentar sistemas que queiram operar, agrupar ou classificar os novos dados.

Os dados são formados por 11742 observações, cada uma composta por seis séries temporais, latitude e longitude, data de início e fim da coleta dos dados e a classe *ground truth*.

Tabela 6.1 - Quantidade de observações por classe.

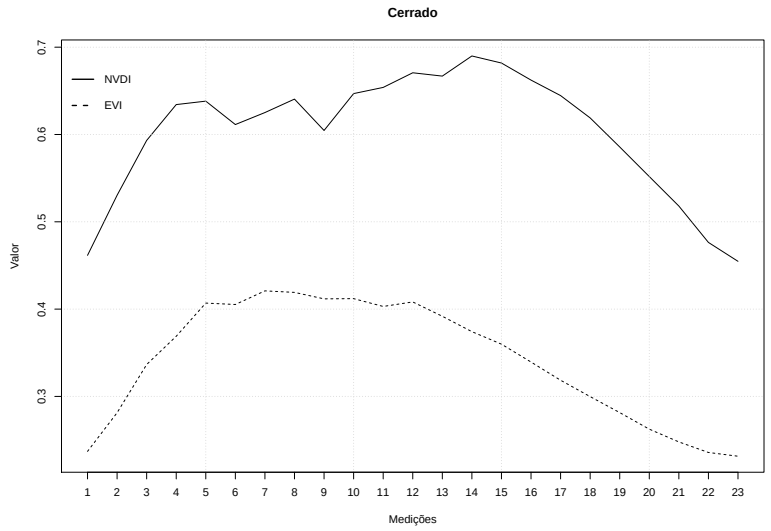
Identificador	Classe	Número de Observações
1	Cerrado	400
2	Cerrado_Campo	1255
3	Cerrado_Rupestre	597
4	Corn_Cotton	36
5	Fallow_Cotton	475
6	Forest	138
7	Millet_Cotton	242
8	Pasture	346
9	Soy_Corn	4514
10	Soy_Cotton	3301
11	Soy_Fallow	139
12	Soy_Millet	246
13	Soy_Sunflower	53

Cada série temporal acompanha um índice diferente, inclusive o NVDI e o EVI, sobre a vegetação, cada série é constituída por 23 medições desse índice em uma mesma latitude e longitude por 1 ano, obtidas a intervalos de 16 dias. Ou seja, cada conjunto de seis séries temporais está relacionado com um mesmo ponto na superfície terrestre.

As medições em cada série temporal foram coletadas nos mesmos dias durante o mesmo ano agrícola, que vai de setembro de um ano até agosto do ano seguinte. Além disso, cada observação possui uma classe *ground truth* que é a validação por terra do rótulo desse ponto.

São 6481 longitudes e 4967 latitudes diferentes combinados em 11742 pontos de observação capturados durante 16 anos a partir do ano 2000 no estado do Mato Grosso no Brasil. As observações estão divididas em 13 classes diferentes: *Cerrado*, *Cerrado_-Campo*, *Cerrado_ Rupestre*, *Corn_Cotton*, *Fallow_Cotton*, *Forest*, *Millet_Cotton*, *Pasture*, *Soy_Corn*, *Soy_Cotton*, *Soy_Fallow*, *Soy_Millet* e *Soy_Sunflower*, na Tabela 6.1 é exibida a quantidade de observações por classe.

Figura 6.1 - Média das séries temporais dos índices NVDI e EVI da classe Cerrado.



Média dos valores de NVDI e EVI nas 23 medições ao longo do ano agrícola da classe Cerrado. Percebe-se que as séries têm um formato em arco.

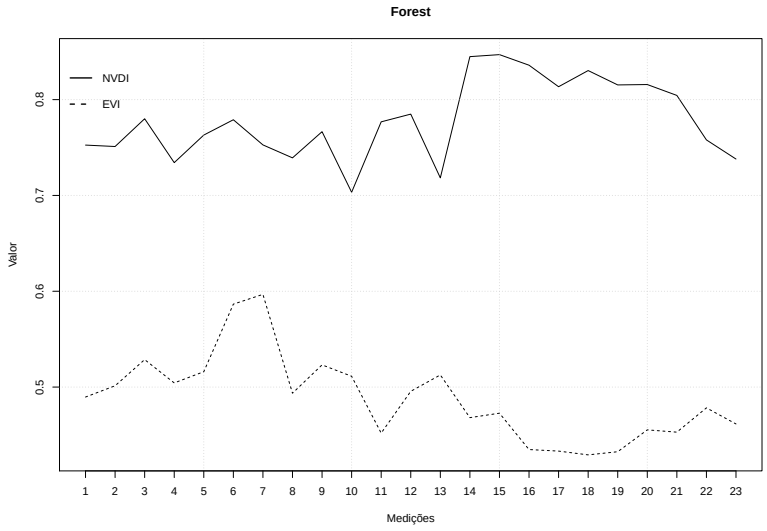
Fonte: Produção da Autora.

Cada série temporal representa um índice de vegetação coletado de 16 em 16 dias durante 1 ano resultando em 23 medições em um mesmo ponto de observação. Cada ponto de observação é analisado e medido para seis diferentes índices, dentre esses índices, este trabalho foca em dois que estão entre os mais utilizados na literatura: NVDI e EVI.

Esses dois valores variam de formas diferentes ao longo do ano dependendo do tipo de vegetação. Por exemplo, nas Figuras 6.1, 6.2 e 6.3 são mostradas as médias das séries temporais dos valores de NVDI e EVI das classes *Cerrado*, *Forest* e *Corn_Cotton*, respectivamente, ao longo do ano agrícola que inicia no mês de setembro e termina em agosto do ano seguinte. Analisando-se os gráficos dessas três classes visualmente percebe-se que é um desafio dividir as séries temporais em classes diferentes, pois os valores são muito similares.

Apesar de cada classe ter seus valores peculiares de NVDI e EVI ao longo do ano agrícola, é difícil obter uma divisão clara entre eles, algo essencial para a tarefa de classificação e/ou agrupamento. No entanto, fazendo-se uma análise mais detalhada, é possível dividir essas classes quanto ao comportamento das séries temporais em 3 *macrogrupos* que se parecem entre si.

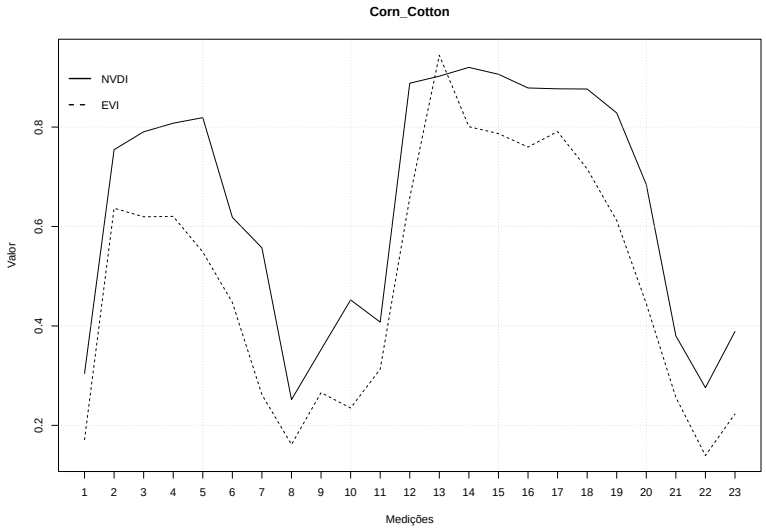
Figura 6.2 - Média das séries temporais dos índices NVDI e EVI da classe *Forest*.



Média dos valores de NVDI e EVI nas 23 medições ao longo do ano agrícola da classe *Forest*. Percebe-se que nos primeiros seis meses as séries tem um comportamento diferente dos últimos seis meses.

Fonte: Produção da Autora.

Figura 6.3 - Média das séries temporais dos índices NVDI e EVI da classe *Corn_Cotton*.



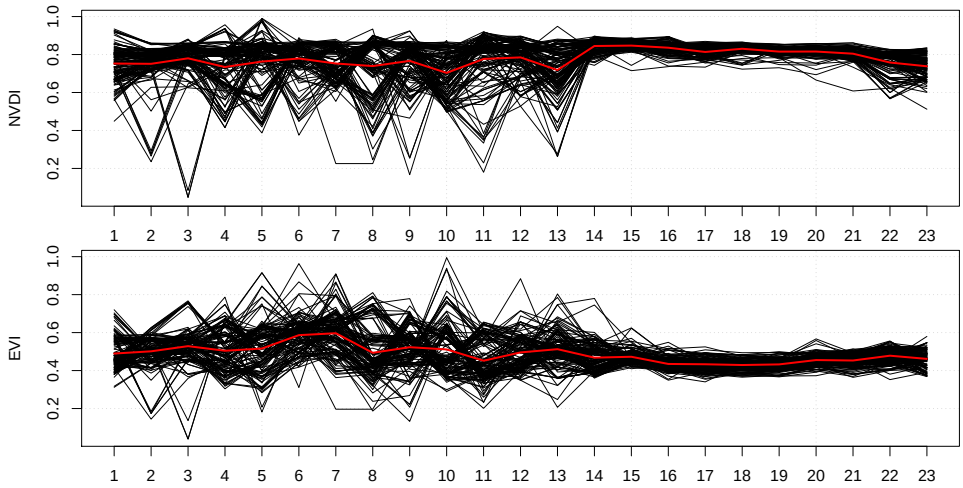
Média dos valores de NVDI EVI nas 23 medições ao longo do ano agrícola da classe *Corn_-Cotton*. O formato das séries tem como característica dois picos e dois vales, indicando plantação e colheita.

Fonte: Produção da Autora.

Por exemplo, olhando-se a Figura 6.4 onde é comparada a série temporal média de NVDI e EVI com todas as 138 observações da classe *Forest*, percebe-se que enquanto nos primeiros seis meses, quando é primavera e verão, as séries têm um comportamento um tanto errático, já nos seis meses seguintes, outono e inverno, o valor de NVDI sobe a média enquanto o EVI desce um pouco a média, ao mesmo tempo que as séries se comportam mais homogeneamente. A classe *Forest* (Floresta) forma o primeiro macrogrupo.

O segundo macrogrupo é em relação às séries temporais que descrevem dupla-colheita. Ou seja, naquele ponto de observação o ano agrícola é dividido em dois tipos de plantação entre *Soy* (Soja), *Cotton* (Algodão), *Corn* (Milho), *Millet* (Milheto) e *Sunflower* (Girassol), além de *Fallow*, que é quando uma terra está em desuso, chamado de pousio.

Figura 6.4 - Classe *Forest* - média versus observações.



Comparação de todas as 138 observações da Classe *Forest* com a série temporal média para os índices NVDI e EVI.

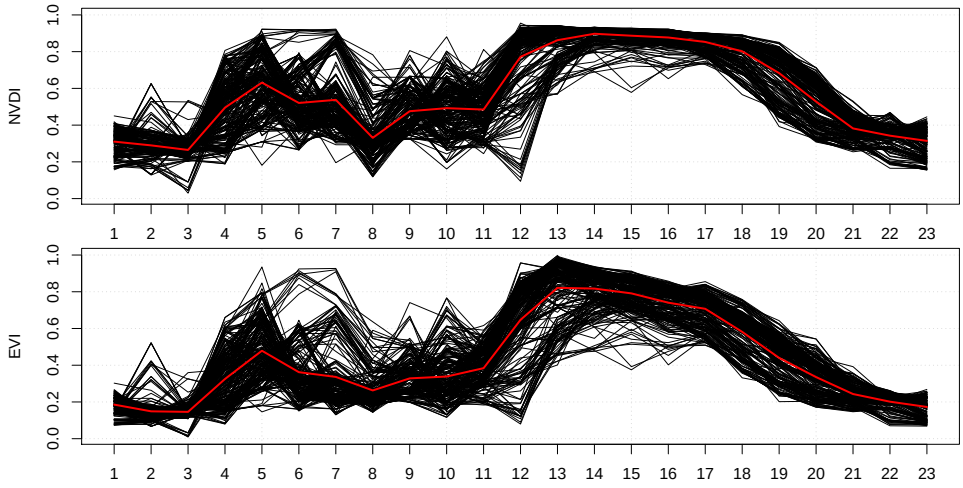
Fonte: Produção da Autora.

O comportamento das séries temporais do segundo macrogrupo tem dois picos que indicam quando a plantação está em alta e dois vales que indicam que a plantação foi colhida. Por fim, o terceiro macrogrupo é composto pelos *Cerrados* e *Pasture* (Pasto) onde o comportamento das séries é mais homogêneo durante o ano com uma leve alteração durante o verão e outono.

Nas Figuras 6.5 e 6.6 são mostrados as comparações de todas as observações das classes *Millet_Cotton* e *Pasture* com as médias de NVDI e EVI, respectivamente.

Na primeira é possível verificar que os valores dos índices para a plantação de Algodão no segundo semestre tem um comportamento mais homogêneo que a plantação de Milheto no primeiro semestre. Na segunda, verifica-se que apesar da série média dos índices da classe *Pasture* ter um comportamento estável durante o ano agrícola, os dados na verdade são bastante ruidosos.

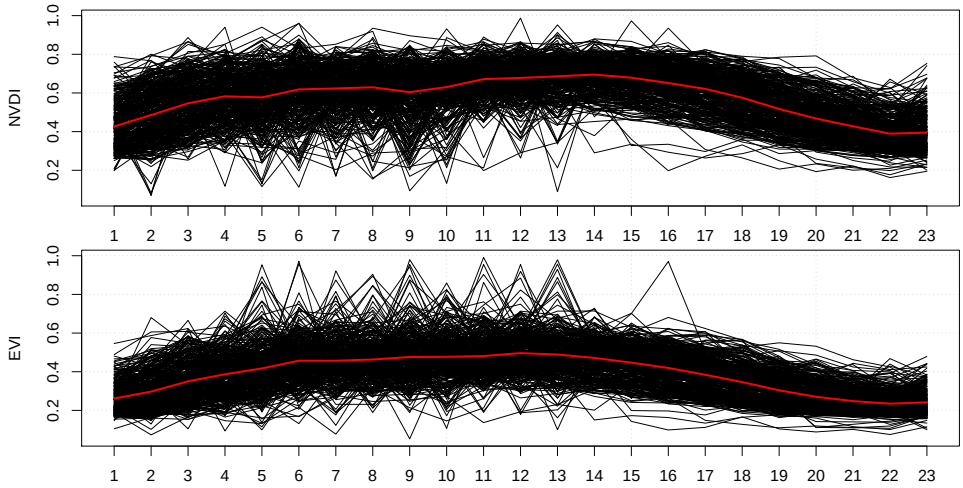
Figura 6.5 - Classe *Millet_Cotton* - média versus observações.



Comparação de todas as 242 observações da Classe *Millet_Cotton* com a média para os índices NVDI e EVI.

Fonte: Produção da Autora.

Figura 6.6 - Classe *Pasture* - média versus observações.

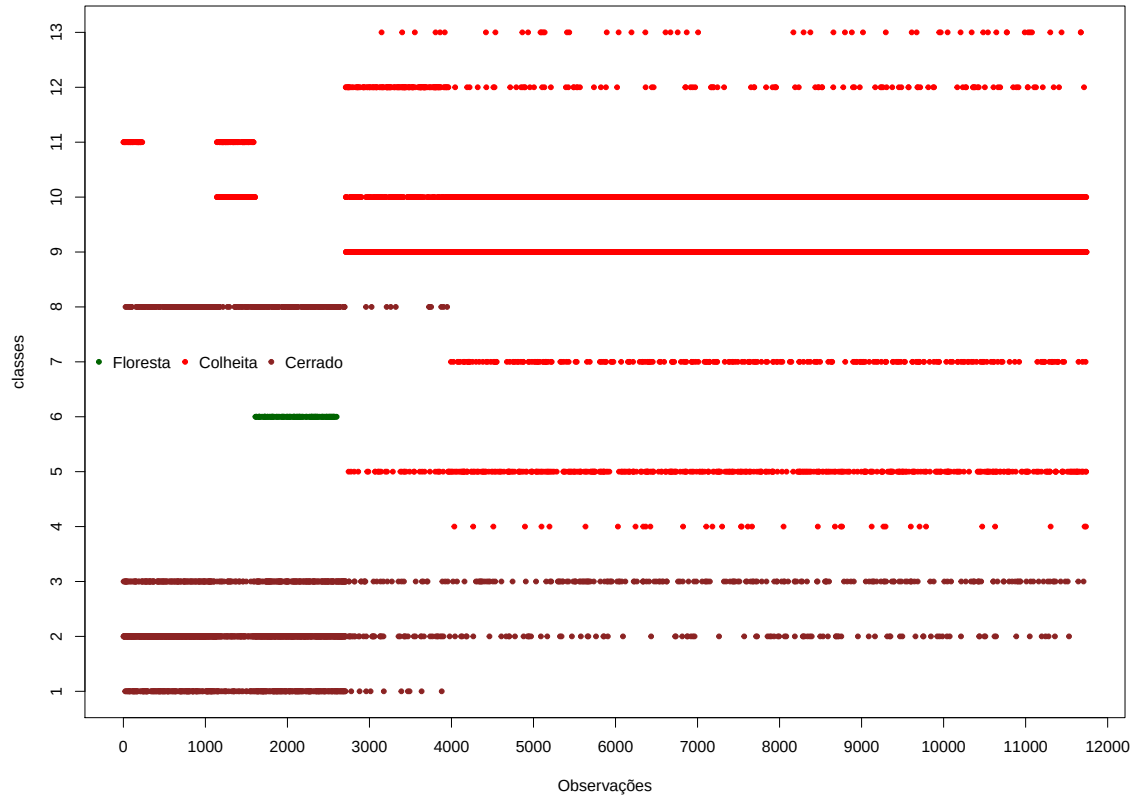


Comparação de todas as 346 observações da Classe *Pasture* com a média para os índices NVDI e EVI.

Fonte: Produção da Autora.

Por fim, na Figura 6.7 é mostrada a ordem de aparição dos dados considerando a data final da coleta das 11742 observações. Inicialmente, são coletados dados que são das classes 1,2,3 e 8, que são as classes do macrogrupo 3, Cerrado e Pasto. Por volta da observação 2700, os dados coletados mostram uma evolução para dados das classes de colheita, apesar de continuar aparecendo dados do macrogrupo de Cerrado. A classe e macrogrupo de Floresta aparece rapidamente no intervalo entre a observação 1500 e 3000.

Figura 6.7 - Ordem de aparição e tipos das 11742 observações.



Cada ponto corresponde a uma observação do conjunto de dados. Os dados estão separados no eixo y pela classe e separados pelas cores nos três macrogrupos. Percebe-se que por volta da observação 2800 os pontos de observação que eram Cerrado e Pasto dão lugar à Colheita.

Fonte: Produção da Autora.

6.1.1 DTW (*Dynamic Time Warping*)

Como elucidado na seção anterior, os dados que são utilizados nos experimentos deste capítulo não são pontos em um espaço R^n, mas sim séries temporais que descrevem as

variações de dois índices de vegetação, NVDI e EVI, em vários pontos da superfície terrestre do estado do Mato Grosso no Brasil.

São dados que foram coletados durante 16 anos formando 11742 observações, onde cada observação é a reunião de vários elementos como longitude, latitude, datas de início e fim da coleta, seis séries temporais, dentre elas as dos índices NVDI e EVI, além da classe *ground truth*.

Também foi relatado na seção anterior como as séries temporais, mesmo aquelas pertencentes à mesma classe, diferem bastante nos valores dos índices. No entanto, o comportamento das séries durante o ano agrícola pode ser encarado como uma melhor forma de identificar as classes das observações, ou ainda dos macrogrupos floresta, dupla-colheita e cerrados/pasto.

Algoritmos e métodos de agrupamento ou classificação necessitam calcular a similaridade entre as observações de tempos em tempos. Quando se trata de pontos no espaço R^n, as distâncias tradicionais, como euclidiana e de Manhattan, são bastante utilizadas, mas quando a similaridade a ser calculada é entre séries temporais algumas considerações devem ser feitas *a priori*.

Na literatura, o agrupamento de séries temporais supõe três cenários diferentes: agrupamento entre um conjunto de várias séries temporais, agrupamento de sub-sequências de uma mesma série ou agrupamento dos pontos das séries (PETITJEAN et al., 2011).

No primeiro cenário, o objetivo é analisar um conjunto de diferentes séries para agrupá-las quanto as suas similaridades. No segundo cenário, o objetivo é encontrar sub-sequências similares dentro de uma mesma série temporal, extraídas a partir de um esquema de janela deslizante, ou seja, agrupar segmentos de mesmo tamanho da série temporal. Por último, no terceiro cenário, também a partir de uma mesma série, o objetivo é agrupar os pontos individuais de acordo com seus valores e também proximidade temporal.

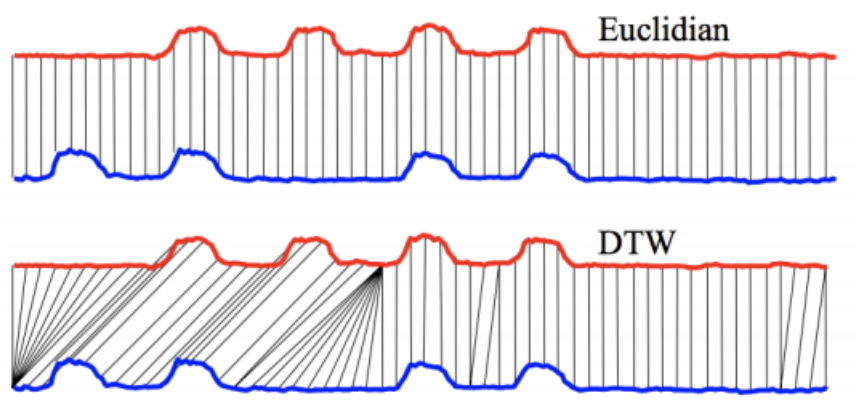
Dentre as abordagens para agrupamento de um conjunto de séries diferentes, existe aquela que considera o formato da série como o aspecto mais importante, ou seja, os valores, apesar de considerados, não são a maior influência, mas sim o desenho que esses valores formam ao longo do tempo.

Para capturar a similaridade de formato entre duas séries, a distância euclidiana, apesar de poder ser utilizada para esse fim, não é a ideal por fazer uma correspondência ponto-a-ponto que acaba por não capturar a semelhança de formato (MAUS, 2015; MAUS

et al., 2019). Portanto, uma medida de distância apropriada deve ser utilizada.

Na literatura, uma das medidas mais utilizadas é a DTW (*dynamic time warping*) (SAKOE; CHIBA, 1971; SAKOE; CHIBA, 1978), que foi pensada originalmente para capturar semelhanças entre falas. Ou seja, o que importa no contexto das falas é *o que* e *como* foi dito, mas não precisa ser exatamente no mesmo ritmo. Na Figura 6.8 é comparada as diferenças entre os cálculos da distância euclidiana e DTW.

Figura 6.8 - Comparação entre o cálculo da distância euclidiana e DTW.



As linhas vermelhas e azuis são as séries temporais sendo comparadas, as linhas pretas indicam quais foram as correspondências entre os pontos. Enquanto a distância euclidiana compara pares de pontos em sequência, a distância DTW distorce o eixo do tempo para dar mais ênfase ao formato da série.

Fonte: Maus (2016).

Quando o formato ou desenho das séries é o aspecto mais importante a ser analisado, é preciso considerar que a ocorrência de padrões de comportamento (como picos e vales, por exemplo) não necessariamente ocorrem no mesmo período do tempo das séries que estão sendo comparadas.

A medida DTW, portanto, verifica se as séries temporais tem padrões de comportamento similares alongando ou contraindo o eixo do tempo com objetivo de tornar trechos das séries o mais semelhante possível. Feito o ajuste do eixo temporal, a distância é calculada

somando-se as distâncias ponto-a-ponto, assim como a distância euclidiana (MAUS et al., 2019).

Assim sendo, neste experimento o cálculo da similaridade será feito segundo Equação 6.1 abaixo:

$$sim(v_i, v_j) = 1/((dtw(nvdi_i, nvdi_j) + dtw(evi_i, evi_j))/2), \tag{6.1}$$

onde v_i e v_j são os vértices da Rede Protótipo e *nvdi* e *evi* são as séries temporais relacionadas aos índices NVDI e EVI, respectivamente. Ou seja, neste experimento, cada vértice contém dois atributos que formam o centróide que são as duas séries temporais de NVDI e EVI.

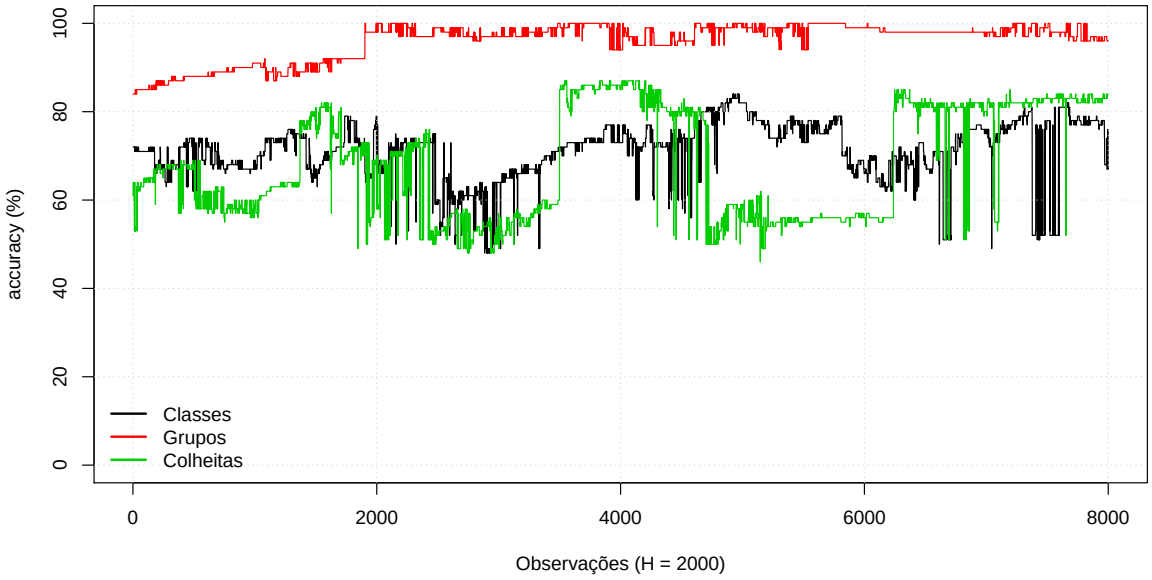
6.2 Experimentos, resultados e considerações

Foram realizados três experimentos com a metodologia RP e os dados de uso e cobertura da Terra. No primeiro experimento, a metodologia foi aplicada à totalidade dos dados e os resultados foram coletados considerando a divisão em 13 classes. No segundo experimento, a metodologia também foi aplicada à totalidade dos dados, mas os resultados foram coletados considerando a divisão em 3 macrogrupos. Por fim, no terceiro experimento, a metodologia foi aplicada somente aos dados que pertencem ao macrogrupo 2 e os resultados foram coletados considerando as 8 classes do macrogrupo de colheita.

Em todos os experimento a metodologia teve o parâmetro $MAXV$ definido como 250, outros experimentos com valores maiores foram realizados, mas verificou-se que os resultados pioravam quando utilizavam-se muitos vértices. De todos os testes, os melhores resultados foram obtidos quando $MAXV = 250$. O parâmetro do tamanho da janela foi definido em $H = 2000$ para todos os experimentos.

A medição do valor de *silhouette* não foi realizada porque é demasiadamente custoso mensurar os erros dos 11742 dados com os centroides. Logo, foram verificadas apenas as purezas e acurácias obtidas. Além disso, foram verificados os números de *clusters* formados para se comparar com os valores verdadeiros e também foram feitos diversos *snapshots* da estrutura em rede, para visualização da atuação da metodologia neste cenário de dados.

Figura 6.9 - Observação da Terra - Acurácia.



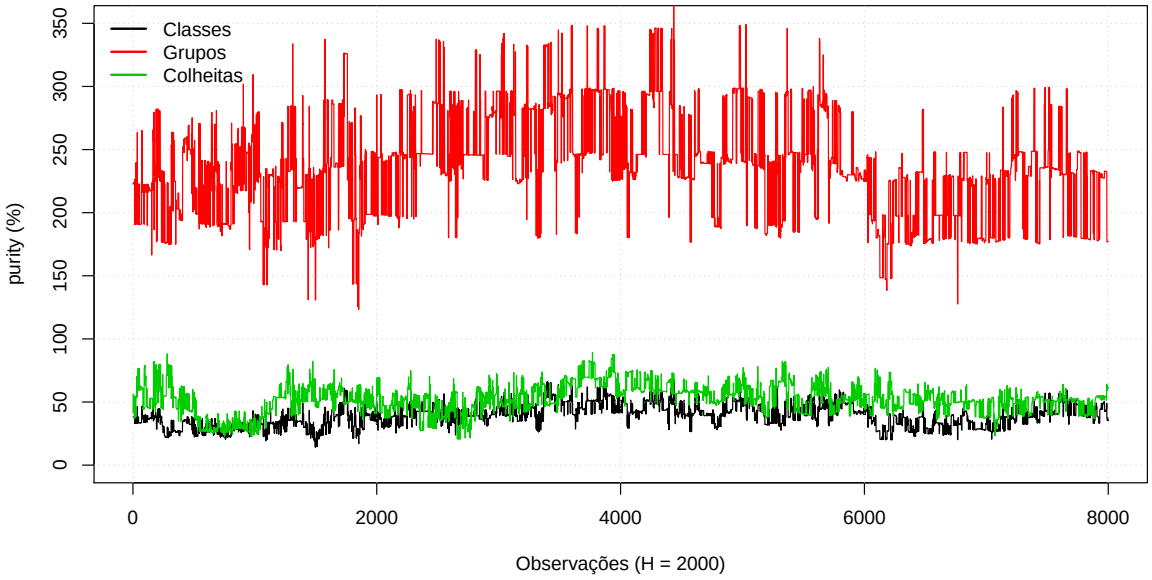
A acurácia dos três experimentos para as últimas 5000 observações com uma janela tamanho $H = 2000$. O resultado para a divisão em grupos mostra que a RP consegue lidar melhor com essa separação, enquanto a divisão em classes e a divisão entre as classes de colheita tiveram desempenho fraco.

Fonte: Produção da Autora.

Na Figura 6.9, é possível verificar os valores de acurácia para os três experimentos. Somente o segundo experimento que considera a divisão em grupos teve resultados bons. A divisão em classes e em classes de colheita obtiveram acurácia mediana durante a execução, entre 50% e 90%.

Na Figura 6.10, são comparadas as medidas de pureza dos três experimentos com os dados de uso e cobertura da Terra. Novamente, apesar da metodologia ter tido bons valores de acurácia para o segundo experimento, a pureza mostra que eram formados mais *clusters* que o necessário. Os valores de pureza para os outros dois experimentos tem desempenho parecido com os de acurácia.

Figura 6.10 - Observação da Terra - Pureza.



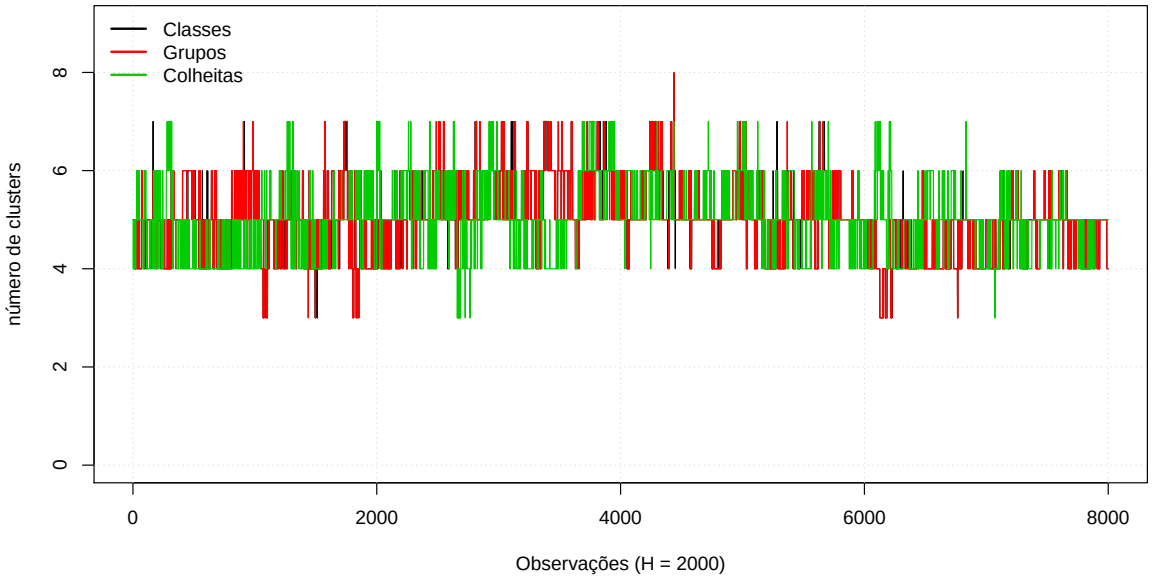
Valores de pureza para as últimas 5000 observações considerando os três experimentos e uma janela tamanho $H = 1000$. Apesar da boa acurácia obtida pela RP para a divisão em grupos, a pureza mostra que eram formados mais *clusters* que o necessário. Para a divisão em classes e em classes de colheita os resultados foram fracos.

Fonte: Produção da Autora.

Confirma-se a informação inferida pelos valores de pureza do segundo experimento na Figura 6.11, onde é possível ver que a quantidade de *clusters* fica entre 4 e 8, quando deveria ser apenas 3. É interessante notar que independente dos dados considerados, a metodologia os divide de forma parecida entre 4 e 8 *clusters*.

Nas Figuras 6.12 e 6.13, são sequenciados vários *snapshots* da estrutura em rede dos segundo e terceiro experimento, respectivamente. Na primeira figura, os vértices só têm cor se todos os objetos representados por ele sejam do mesmo grupo, sendo assim, percebe-se pela imagem que a maioria dos vértices obedece a essa condição.

Figura 6.11 - Observação da Terra - Número de *clusters*.



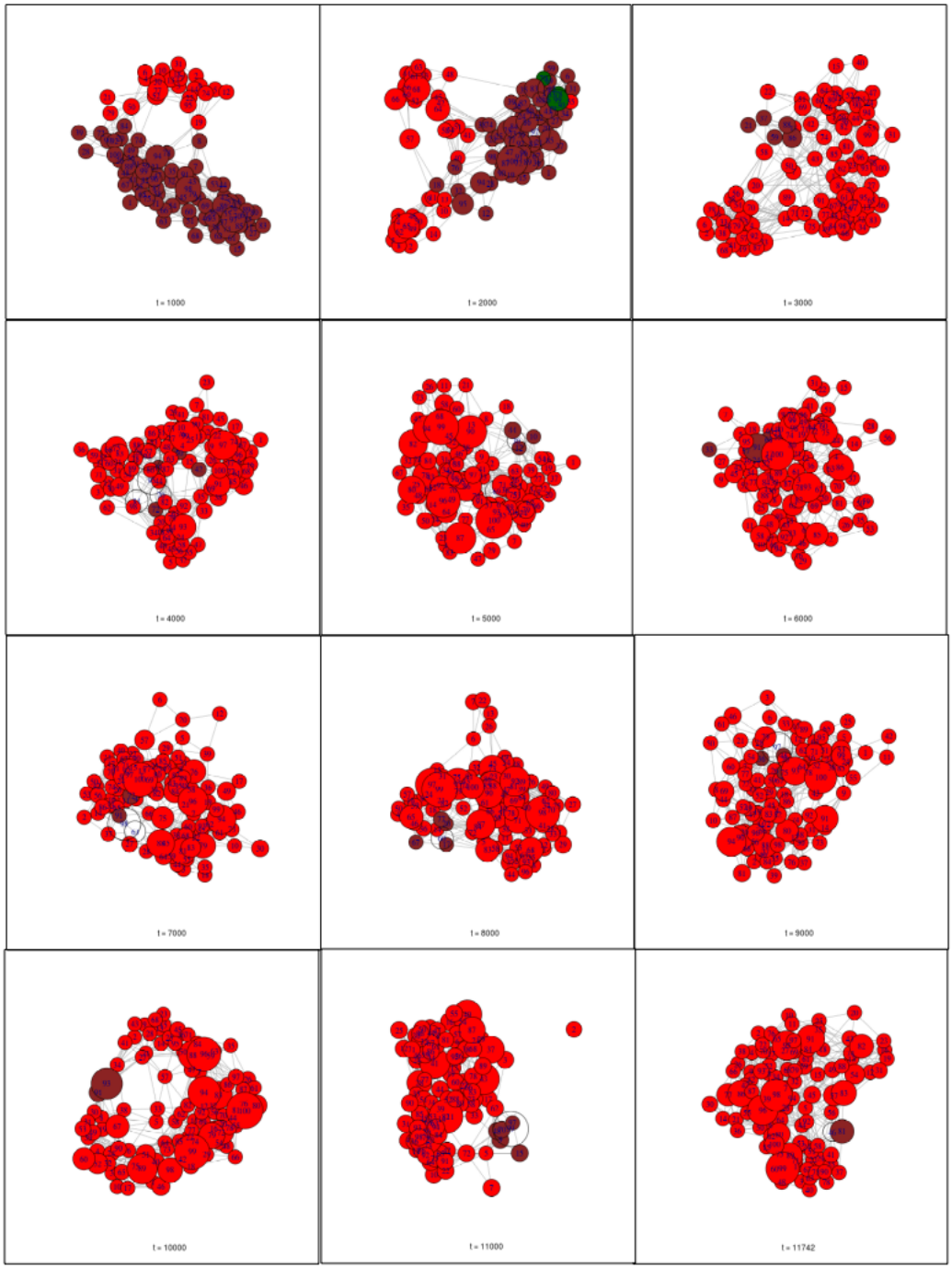
A quantidade de *clusters* formados pela metodologia RP para as últimas 5000 observações com uma janela tamanho $H = 1000$. Independente do experimento, a metodologia forma uma quantidade similar de *clusters*, entre 3 e 7.

Fonte: Produção da Autora.

Na Figura 6.12, também é possível notar como inicialmente os vértices representam em maior número os dados referentes às classes do Cerrado e Pasto, no segundo *snapshot* aparecem vértices representantes do grupo de Floresta e, a partir do terceiro *snapshot*, os vértices são dominados pelos dados de colheita.

Na Figura 6.13, a estrutura em rede reflete bem o quão difíceis são os dados, enquanto há uma dominância muito forte da maior classe dos dados, a *Soy_Corn* (roxo), os outros dados acabam mesclados nos vértices, sem a metodologia conseguir uma separação clara entre eles.

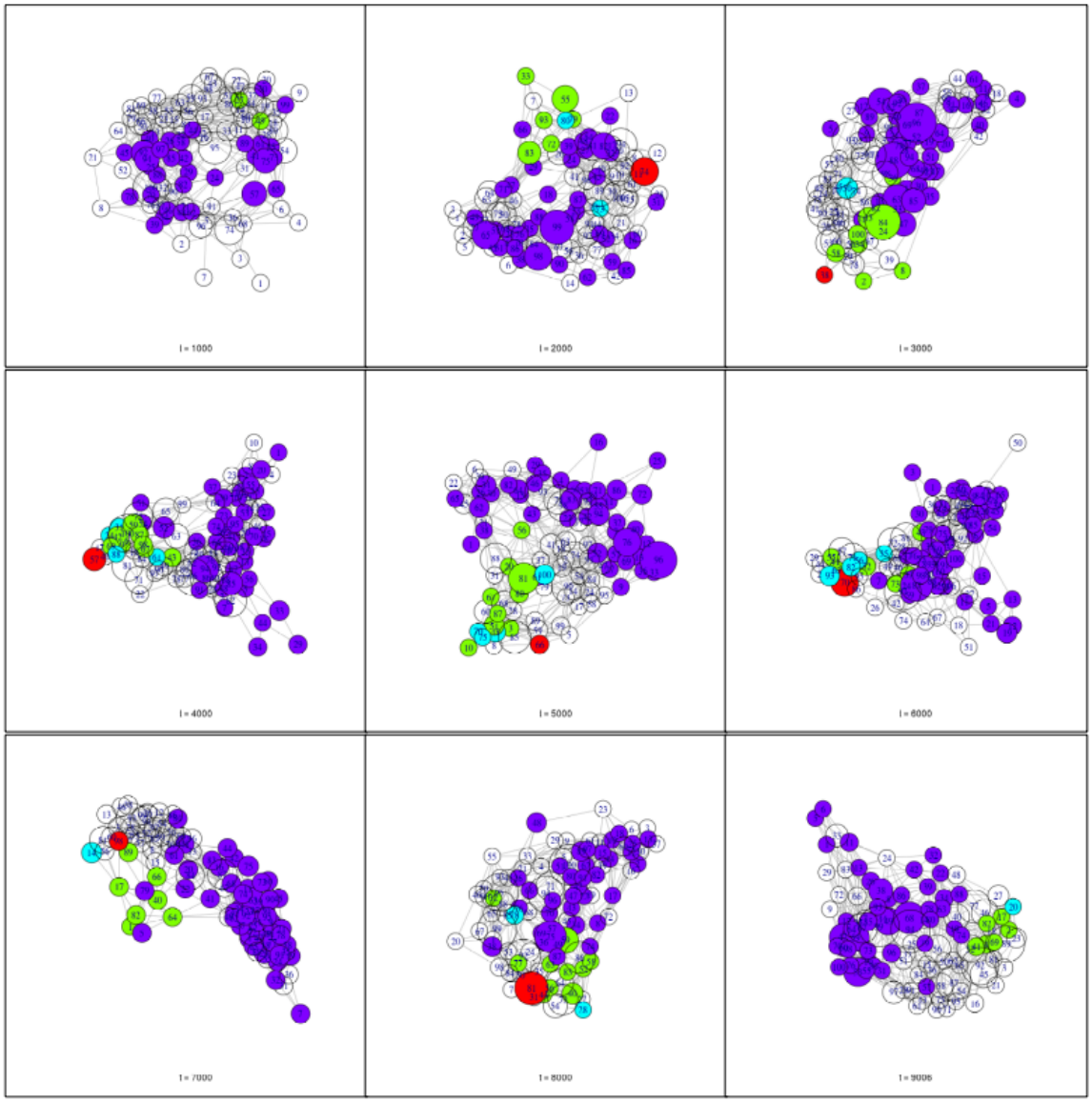
Figura 6.12 - Observação da Terra - *Snapshots* do experimento 2.



Snapshots da rede ao longo do experimento considerando os três grupos dos dados. As cores dos vértices só são definidas se todos os objetos representados são do mesmo grupo. No segundo *snapshot*, é possível ver a formação de alguns vértices verdes que correspondem ao Grupo 1 - Floresta. No resto da execução, os vértices do Grupo 2 dominam a estrutura que contém apenas alguns poucos representantes do Grupo 3.

Fonte: Produção da Autora.

Figura 6.13 - Observação da Terra - *Snapshots* do experimento 3.



Snapshots da rede ao longo do experimento considerando as classes de colheita. As cores dos vértices só são definidas se todos os objetos representados são do mesmo grupo. Durante toda a execução há a formação de mais vértices da cor roxa, que representam objetos da classe *Soy_Corn*, que é a mais numerosa. Além desses, somente alguns poucos vértices com objetos da mesma classe são formados e a outra parte são vértices transparentes, ou seja, que tem objetos de classes diferentes sendo representados.

Fonte: Produção da Autora.

7 CONCLUSÕES

Nas últimas décadas, o mundo observou a tecnologia aumentar seu potencial de forma exponencial e junto com esse aumento vieram novos problemas, novos conceitos, novos dados e novas análises de dados. Dentre esses, há um novo tipo de dado que é chamado fluxo de dados, onde há uma fonte emitindo objetos de forma sequencial e ilimitada. Os exemplos estão no dia-a-dia: redes sociais, trajetórias de GPS, internet das coisas, entre tantos.

Um novo tipo de dado precisa de um novo tipo de análise, sendo que os objetivos continuam os mesmos: classificar, segmentar, agrupar e etc. Mas as soluções propostas precisam se adaptar às diferenças que existem entre fluxos de dados e bases de dados tradicionais.

A primeira e mais crucial diferença é o acesso aos dados, que deve ser sequencial, ou seja, a leitura e análise dos dados precisam ser feitas ao mesmo tempo e provavelmente em tempo real. Outra diferença também drástica é que nos problemas tradicionais o objetivo é dividir os dados considerando que eles tem uma probabilidade de distribuição fixa, em fluxos de dados essa probabilidade pode mudar ao longo do tempo.

Dentro deste contexto, este trabalho teve o objetivo de desenvolver uma metodologia para fazer agrupamento em fluxo de dados sem precisar da intervenção do usuário. A estratégia deu-se a partir do uso de uma estrutura do tipo rede complexa, que oferece muitas funcionalidades que são úteis para o agrupamento, como a divisão em comunidades, e que dispensa que muitos parâmetros sejam definidos.

Apresentada no Capítulo 4, a metodologia Rede Protótipo agrupa objetos provenientes de fluxos de dados com a ajuda de uma estrutura do tipo rede complexa. A metodologia depende de apenas dois parâmetros e é flexível para diferentes contextos, inclusive *concept drift* e *concept evolution*. Com isso, o objetivo principal deste trabalho foi alcançado.

A metodologia conta com cinco funções que são responsáveis por alterar a rede complexa de forma que a estrutura em comunidades da rede seja reflexo da divisão em *clusters* dos dados mais recentes do fluxo. A partir dos experimentos com fluxos sintéticos que continham mudanças e evoluções de conceitos, chega-se a conclusão que a metodologia RP é capaz de lidar com essas alterações nos dados.

A partir dos experimentos com os dados e cobertura da Terra, conclui-se também que a metodologia consegue lidar com um grande volume de dados utilizando poucos recursos de memória. A metodologia também mostrou-se capaz de lidar com diferentes tipos de

dados e medidas de similaridade, isto é, séries temporais e a medida de similaridade DTW.

A metodologia teve o desempenho avaliado ao ser comparada com algoritmos da literatura que também se propõem a fazer o agrupamento de fluxos de dados, sendo eles: *CluStream*, *DenStream* e *SNCStream*. A comparação deu-se através das medidas de *silhouette*, pureza e acurácia, além de comparação de outros valores como número de *microclusters*, arestas, *clusters* e graus máximo, mínimo e médio.

Os experimentos mostraram que todos os métodos que foram testados tiverem facilidades e dificuldades nos mesmos experimentos. Por exemplo, o *shift* incremental considerando o tamanho da janela em $H = 1000$ foi particularmente difícil para todos os métodos.

A evolução *New* também mostrou-se uma tarefa difícil para os métodos, ainda mais considerando que surgiram 7 *clusters* dentro de um espaço de dados de dimensão 2. As fronteiras entre os *clusters* eram muito próximas e os métodos tiveram dificuldade para separá-los.

De forma geral, os experimentos mostraram que a metodologia tem um bom desempenho mesmo não dependendo de vários parâmetros e não utilizando muitos recursos de memória. O potencial maior da metodologia é a possibilidade de extensão em trabalhos futuros, onde a RP possa ser incrementada com mais recursos como o aprendizado ativo.

Também podem ser implementados diferentes cálculos de centroides, ou então, diferentes tipos de abstrações que possam ser colocadas nos vértices. Além disso, a rede complexa pode conter diferentes categorias para os vértices, por exemplo, a inclusão de vértices específicos para o tratamento de *outliers*.

Dessa forma, pode-se enumerar como trabalhos futuros desta tese:

- Adaptação para o uso do aprendizado ativo, utilizando as métricas da rede para definir quais objetos devem ser apresentados aos especialistas.
- Adaptação para agrupamento considerando o aprendizado semissupervisionado, se beneficiando do aprendizado ativo.
- Utilização de abstrações de dados diferentes a serem representados pelos vértices.
- Utilização de diferentes categorias para os vértices, possibilitando a detecção de *outliers*.

REFERÊNCIAS BIBLIOGRÁFICAS

ABDULLATIF, A.; MASULLI, F.; ROVETTA, S. Clustering of nonstationary data streams: a survey of fuzzy partitional methods. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, v. 8, n. 4, p. e1258, 2018. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1258>. 1, 3, 8, 21

AGGARWAL, C. C. A framework for diagnosing changes in evolving data streams. In: INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA. **Proceedings...** New York: ACM, 2003. p. 575–586. ISBN 1-58113-634-X. Disponível em: <http://doi.acm.org/10.1145/872757.872826>. 4, 13, 16, 19, 21, 22, 73

____. **Data streams: models and algorithms (Advances in Database Systems)**. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 0387287590. 14, 21, 22

____. An introduction to data streams. In: ____. **Data streams: models and algorithms**. Boston, MA: Springer US, 2007. p. 1–8. ISBN 978-0-387-47534-9. 1, 2, 16

AGGARWAL, C. C.; HAN, J.; WANG, J.; YU, P. S. A framework for clustering evolving data streams. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, 29., 2003. **Proceedings...** VLDB Endowment, 2003. p. 81–92. ISBN 0-12-722442-4. Disponível em: <http://dl.acm.org/citation.cfm?id=1315451.1315460>. 16, 19, 21, 22

AGGARWAL, C. C.; REDDY, C. K. **Data Clustering: Algorithms and Applications**. 1st. ed. [S.l.]: Chapman & Hall/CRC, 2013. ISBN 1466558210, 9781466558212. 2, 7, 8, 11

AGGARWAL, C. C.; YU, P. S. On clustering massive text and categorical data streams. **Knowledge and Information Systems**, v. 24, n. 2, p. 171–196, Aug 2010. ISSN 0219-3116. Disponível em: <https://doi.org/10.1007/s10115-009-0241-z>. 1, 8, 13

AHMED, M. Data summarization: a survey. **Knowledge and Information Systems**, v. 58, n. 2, p. 249–273, Mar 2018. ISSN 0219-3116. Disponível em: <https://doi.org/10.1007/s10115-018-1183-0>. 13

ALMEIDA, P. R.; OLIVEIRA, L. S.; BRITTO, A. S.; SABOURIN, R. Adapting dynamic classifier selection for concept drift. **Expert Systems with Applications**, v. 104, p. 67 – 85, 2018. ISSN 0957-4174. Disponível em:

<http://www.sciencedirect.com/science/article/pii/S0957417418301611>. 1, 8

AMINI, A.; WAH, T. Y.; SABOOHI, H. On density-based data streams clustering algorithms: a survey. **Journal of Computer Science and Technology**, v. 29, n. 1, p. 116–141, Jan 2014. ISSN 1860-4749. Disponível em: <https://doi.org/10.1007/s11390-014-1416-y>. 1, 13, 14, 21, 24

ASUR, S.; PARTHASARATHY, S.; UCAR, D. An event-based framework for characterizing the evolutionary behavior of interaction graphs. **TKDD**, v. 3, n. 4, 2009. 29

BALCAN, M.-F.; URNER, R. Active learning – modern learning theory. In: ____. **Encyclopedia of algorithms**. New York, NY: Springer New York, 2016. p. 8–13. ISBN 978-1-4939-2864-4. 10

BARABÁSI, A. L. Scale-free networks: a decade and beyond. **Science**, v. 325, p. 412, 2009. 33, 34

BARABÁSI, A. L.; ALBERT, R. Emergence of scaling in random networks. **Science**, v. 286, p. 509–512, 1999. 31, 32

BARABÁSI, A. L.; ALBERT, R.; JEONG, H. Scale-free characteristics of random networks: the topology of the world wide web. **Physica A**, v. 272, p. 173–187, 2000. 33

BARBER, D. **Bayesian reasoning and machine learning**. [S.l.]: Cambridge University Press, 2012. 3

BARDDAL, J. P.; GOMES, H. M.; ENEMBRECK, F. Sncstream: a social network-based data stream clustering algorithm. In: ANNUAL ACM SYMPOSIUM ON APPLIED COMPUTING, 30., 2015. **Proceedings...** New York: ACM, 2015. p. 935–940. ISBN 978-1-4503-3196-8. Disponível em: <http://doi.acm.org/10.1145/2695664.2695674>. 4, 43, 73, 74

____. On social network-based algorithms for data stream clustering. In: ____. **Learning from data streams in evolving environments: methods and applications**. Cham: Springer, 2019. p. 297–317. ISBN 978-3-319-89803-2. 11, 15, 16, 21, 43, 74

BIFET, A.; GAVALDÀ, R. Adaptive learning from evolving data streams. In: ADAMS, N. M.; ROBARDET, C.; SIEBES, A.; BOULICAUT, J.-F. (Ed.). **Advances in intelligent data analysis VIII**. Berlin, Heidelberg: Springer, 2009. p. 249–260. ISBN 978-3-642-03915-7. 7

BISHOP, C. M. **Pattern recognition and machine learning (Information Science and Statistics)**. Secaucus, NJ, USA: Springer-Verlag, 2006. ISBN 0387310738. 7

BOLLOBÁS, B. **Modern graph theory**. [S.l.]: Springer, 1998. (Graduate texts in mathematics). 29, 31

BORIAH, S. **Time series change detection: algorithms for land cover change**. Minneapolis: University of Minnesota: [s.n.], 2010. ISBN 9781109768558. 105

BORIAH, S.; KUMAR, V.; STEINBACH, M.; POTTER, C.; KLOOSTER, S. Land cover change detection: a case study. In: INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING. **Proceedings...** New York, NY, USA: ACM, 2008. (KDD '08, v. 14), p. 857–865. ISBN 978-1-60558-193-4. Disponível em: <http://doi.acm.org/10.1145/1401890.1401993>. 105

BOULILA, W.; FARAH, I.; ETTABAA, K.; SOLAIMAN, B.; GHEZALA, H. B. A data mining based approach to predict spatiotemporal changes in satellite images. **International Journal Applied Earth Observation and Geoinformation**, v. 13, p. 386–395, 06 2011. 105

CAO, F.; ESTER, M.; QIAN, W.; ZHOU, A. Density-based clustering over an evolving data stream with noise. In: SIAM CONFERENCE ON DATA MINING. **Proceedings...** [S.l.], 2006. p. 328–339. 1, 4, 13, 14, 16, 19, 21, 22, 24, 73

CARNEIN, M.; TRAUTMANN, H. Optimizing data stream representation: an extensive survey on stream clustering algorithms. **Business & Information Systems Engineering**, Jan 2019. ISSN 1867-0202. Disponível em: <https://doi.org/10.1007/s12599-019-00576-5>. 12, 15, 19, 21, 25

CHAPELLE, O.; SCHÖLKOPF, B.; ZIEN, A. **Semi-supervised learning**. Cambridge, MA: MIT Press, 2006. Disponível em: <http://www.kyb.tuebingen.mpg.de/ssl-book>. 7, 8, 9

CHEN, C. P.; ZHANG, C.-Y. Data-intensive applications, challenges, techniques and technologies: a survey on big data. **Information Sciences**, v. 275, p. 314 – 347, 2014. ISSN 0020-0255. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0020025514000346>. 21

CHEN, Y.; TU, L. Density-based clustering for real-time stream data. In: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING. **Proceedings...** New York: ACM, 2007. p. 133–142. ISBN

978-1-59593-609-7. Disponível em:
<http://doi.acm.org/10.1145/1281192.1281210>. 21, 22

CHENG, J.; KE, Y.; NG, W. A survey on algorithms for mining frequent itemsets over data streams. **Knowledge and Information Systems**, v. 16, n. 1, p. 1–27, Jul 2008. ISSN 0219-3116. Disponível em:
<https://doi.org/10.1007/s10115-007-0092-4>. 1, 8, 14

CLAUSET, A.; NEWMAN, M. E. J.; MOORE, C. Finding community structure in very large networks. **Physical Review E**, v. 70, p. 066111, 2004. Disponível em:
<http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0408187>. 4, 40, 41

CONWAY, D.; WHITE, J. M. **Machine learning for hackers**. Sebastopol, CA: O'Reilly Media, 2012. Disponível em: <http:
//www.amazon.com/Machine-Learning-Hackers-Drew-Conway/dp/1449303714>. 2

COSTA, L.; RODRIGUES, F.; TRAVIESO, G.; BOAS, P. Characterization of complex networks: a survey of measurements. **Advances in Physics**, v. 56, n. 1, p. 167–242, 2007. 3, 31

CSARDI, G.; NEPUSZ, T. The igraph software package for complex network research. **InterJournal**, Complex Systems, p. 1695, 2006. Disponível em:
<http://igraph.sf.net>. 68

DIESTEL, R. **Graph theory**. 3. ed. [S.l.]: Springer-Verlag Heidelberg, 2005. 29

DUDA, P.; JAWORSKI, M.; RUTKOWSKI, L. Knowledge discovery in data streams with the orthogonal series-based generalized regression neural networks. **Information Sciences**, 2017. ISSN 0020-0255. Disponível em:
<http://www.sciencedirect.com/science/article/pii/S0020025517308344>. 1

ERDOS, P.; RÉNYI, A. On random graphs i. **Publicationes Mathematicae (Debrecen)**, v. 6, p. 290, 1959. 31

____. On the evolution of random graphs. **Publication of the Mathematical Institute of the Hungarian Academy of Sciences**, v. 5, p. 17, 1960. 31

FAN, Y.; LI, M.; ZHANG, P.; WU, J.; DI, Z. Accuracy and precision of methods for community identification in weighted networks. **Physica A: Statistical Mechanics and**

its Applications, v. 377, n. 1, p. 363–372, abr. 2007. Disponível em: <http://www.sciencedirect.com/science/article/B6TVG-4MHP94W-4/1/00988db3212bd4667da5ee3cf99401ff>. 31

FEOFILOFF, P.; KOHAYAKAWA, Y.; WAKABAYASHI, Y. **Uma introdução sucinta à teoria dos grafos**. [S.l.: s.n.], 2011. 29

FLACH, P. **Machine learning: the art and science of algorithms that make sense of data**. New York, NY, USA: Cambridge University Press, 2012. ISBN 1107422221, 9781107422223. 2, 7, 15

FORESTIERO, A.; PIZZUTI, C.; SPEZZANO, G. Flockstream: a bio-inspired algorithm for clustering evolving data streams. In: INTERNATIONAL CONFERENCE ON TOOLS WITH ARTIFICIAL INTELLIGENCE, 21., 2009. **Proceedings...** Washington, DC, USA: IEEE, 2009. p. 1–8. ISBN 978-0-7695-3920-1. Disponível em: <https://doi.org/10.1109/ICTAI.2009.60>. 21, 22

FORTUNATO, S. Community detection in graphs. **Physics Reports**, v. 486, p. 75 – 174, 2010. 3, 29, 31, 39, 40

____. **Benchmark graphs to test community detection algorithms**. 2013. Disponível em: <https://sites.google.com/site/santofortunato/inthepress2>. 3

FRANKE, M.; GEYER-SCHULZ, A. An update algorithm for restricted random walk clustering for dynamic data sets. **Advances in Data Analysis and Classification**, v. 3, n. 1, p. 63–92, 2009. 41

FREITAS, R. de; ARAI, E.; ADAMI, M.; SOUZA, A.; SATO, F. Y.; SHIMABUKURO, Y.; ROSA, R.; ANDERSON, L.; RUDORFF, B. Virtual laboratory of remote sensing time series: visualization of modis evi2 data set over south america. **Journal of Computational Interdisciplinary Sciences**, v. 2, p. 57–68, 01 2011. 105

GABER, M. M.; ZASLAVSKY, A.; KRISHNASWAMY, S. A survey of classification methods in data streams. In: ____. **Data streams: models and algorithms**. Boston, MA: Springer US, 2007. p. 39–59. ISBN 978-0-387-47534-9. 1, 3, 8, 11, 14, 16, 21

GAMA, J. A survey on learning from data streams: current and future trends. **Progress in Artificial Intelligence**, v. 1, n. 1, p. 45–55, Apr 2012. ISSN 2192-6360. Disponível em: <https://doi.org/10.1007/s13748-011-0002-6>. 1, 8, 11, 12, 13, 14, 16, 17, 19, 21

GETZ, G.; SHENTAL, N.; DOMANY, E. Semi-supervised learning – a statistical physics approach. **CoRR**, 2006. 7, 8, 9

GOMEZ, C.; WHITE, J.; WULDER, M. Optical remotely sensed time series data for land cover classification: a review. **ISPRS Journal of Photogrammetry and Remote Sensing**, v. 116, p. 55–72, 03 2016. 105

GUHA, S.; MISHRA, N.; MOTWANI, R.; O'CALLAGHAN, L. Clustering data streams. In: ANNUAL SYMPOSIUM ON FOUNDATIONS OF COMPUTER SCIENCE, 41., 2000. **Proceedings of the 41st**. Washington: IEEE, 2000. ISBN 0-7695-0850-2. Disponível em: <http://dl.acm.org/citation.cfm?id=795666.796588>. 1, 2, 12, 21

GUPTA, M.; GAO, J.; AGGARWAL, C. C.; HAN, J. Outlier detection for temporal data: a survey. **IEEE Transactions on Knowledge and Data Engineering**, v. 26, n. 9, p. 2250–2267, Sep. 2014. ISSN 1041-4347. 1, 21

HAHSLER, M.; BOLAÑOS, M.; FORREST, J. Introduction to stream: an extensible framework for data stream clustering research with R. **Journal of Statistical Software**, v. 76, n. 14, p. 1–50, 2017. 68, 74

HAHSLER, M.; BOLANOS, M.; FORREST, J. **Stream: infrastructure for data stream mining**. [s.n.], 2018. R package version 1.3-0. Disponível em: <https://CRAN.R-project.org/package=stream>. 21, 68, 74

HAIDAR, D.; GABER, M. Data stream clustering for real-time anomaly detection: an application to insider threats. In: ____. **Clustering methods for big data analysis**. Berlin: Springer, 2018. p. 115–144. 1, 3, 8, 13, 15, 19, 21

IBRAHIM, O. A.; DU, Y.; KELLER, J. Robust on-line streaming clustering. In: MEDINA, J.; OJEDA-ACIEGO, M.; VERDEGAY, J. L.; PELTA, D. A.; CABRERA, I. P.; BOUCHON-MEUNIER, B.; YAGER, R. R. (Ed.). **Information processing and management of uncertainty in knowledge-based systems. Theory and foundations**. Cham: Springer International Publishing, 2018. p. 467–478. ISBN 978-3-319-91473-2. 1, 13, 19

JEBARA, T.; SHCHOGOLEV, V. B-matching for spectral clustering. In: FURNKRANZ, J.; SCHEFFER, T.; SPILIOPOULOU, M. (Ed.). **Machine learning: ECML 2006**. [S.l.]: Springer Berlin Heidelberg, 2006, (Lecture Notes in Computer Science, v. 4212). p. 679–686. ISBN 978-3-540-45375-8. 35

JIANG, Z.; HUETE, A.; DIDAN, K.; MIURA, T. Development of a two-band enhanced vegetation index without a blue band. **Remote Sensing of Environment**, v. 112, p. 3833–3845, 10 2008. 106

JOSHI, P. **Artificial intelligence with Python**. Packt Publishing, 2017. ISBN 9781786464392. Disponível em: <https://books.google.com.br/books?id=hbUgvgAACAAJ>. 2, 3, 8, 10, 11, 14

JUSTICE, C.; TOWNSHEND, J.; VERMOTE, E.; ED, M.; WOLFE, R.; SALEOUS, N.; ROY, D.; MORISETTE, J. An overview of modis land data processing and product status. **Remote Sensing of Environment**, v. 83, p. 3–15, 11 2002. 106

KELLEHER, J. D.; NAMEE, B. M.; DARCY, A. **Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies**. [S.l.]: The MIT Press, 2015. 2, 3, 8, 10

KIRKPATRICK, S.; GELATT, C.; VECCHI, M. Optimization by simulated annealing. **Science**, v. 220, n. 4598, p. 671, 1983. 43

KRANEN, P.; ASSENT, I.; BALDAUF, C.; SEIDL, T. The clustree: indexing micro-clusters for anytime stream mining. **Knowledge and Information Systems**, v. 29, n. 2, p. 249–272, Nov 2011. ISSN 0219-3116. Disponível em: <https://doi.org/10.1007/s10115-010-0342-8>. 21, 22

KRAWCZYK, B.; MINKU, L. L.; GAMA, J.; STEFANOWSKI, J.; WOźNIAK, M. Ensemble learning for data stream analysis: a survey. **Information Fusion**, v. 37, p. 132 – 156, 2017. ISSN 1566-2535. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1566253516302329>. 13

LAMBIN, E. F.; LINDERMAN, M. Time series of remote sensing data for land change science. **IEEE Transactions on Geoscience and Remote Sensing**, v. 44, p. 1926 – 1928, 08 2006. 105

LANCICHINETTI, A.; FORTUNATO, S. Community detection algorithms: a comparative analysis. **Physical Review E**, American Physical Society, v. 80, p. 056117, Nov 2009. Disponível em: <http://link.aps.org/doi/10.1103/PhysRevE.80.056117>. 39, 40

LARRANAGA, P. et al. Machine learning in bioinformatics. **Briefings in bioinformatics**, v. 7, n. 1, p. 86–112, 2006. 2

LIN, Y.-R.; CHI, Y.; ZHU, S.; SUNDARAM, H.; TSENG, B. L. Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. In: INTERNATIONAL CONFERENCE ON WORLD WIDE WEB, 2008. **Proceedings...** New York: ACM, 2008. p. 685–694. ISBN 978-1-60558-085-2. 41

LUNETTA, R.; KNIGHT, J.; EDIRIWICKREMA, J.; LYON, J. G.; WORTHY, L. D. Land-cover change detection using multi-temporal modis ndvi data. **Remote Sensing of Environment**, v. 105, p. 142–154, 11 2006. 105

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In: BERKELEY SYMPOSIUM ON MATHEMATICAL STATISTICS AND PROBABILITY, 5., 1967. **Proceedings...** Berkeley, California: University of California: University of California Press, 1967. p. 281–297. 21

MAINI, V.; SABRI, S. **Machine learning for humans**. [S.l.: s.n.], 2017. 2, 3, 10, 11, 14

MANSALIS, S.; NTOUTSI, E.; PELEKIS, N.; THEODORIDIS, Y. An evaluation of data stream clustering algorithms. **Statistical Analysis and Data Mining: The ASA Data Science Journal**, v. 11, 06 2018. 8, 11, 15, 16, 19, 21, 25

MAO, J.; SONG, Q.; JIN, C.; ZHANG, Z.; ZHOU, A. Online clustering of streaming trajectories. **Frontiers of Computer Science**, v. 12, n. 2, p. 245–263, Apr 2018. ISSN 2095-2236. Disponível em: <https://doi.org/10.1007/s11704-017-6325-0>. 1, 12, 14, 15

MARCU, O.-C.; COSTAN, A.; ANTONIU, G.; PÉREZ-HERNÁNDEZ, M. S.; TUDORAN, R.; BORTOLI, S.; NICOLAE, B. **Storage and ingestion systems in support of stream processing: a Survey**. [s.n.], Nov 2018. 1-33 p. Disponível em: <https://hal.inria.fr/hal-01939280>. 21

MARSLAND, S. **Machine learning: an algorithmic perspective**. [S.l.]: Chapman & Hall/CRC, 2009. ISBN 1420067184, 9781420067187. 2, 7

MAUS, V. Open boundary dynamic time warping for satellite image time series classification. In: INTERNATIONAL GEOSCIENCE AND REMOTE SENSING SYMPOSIUM. **Proceedings...** Milan, Italy, 2015. 113, 114

MAUS, V.; CÂMARA, G.; APPEL, M.; PEBESMA, E. dtwsat : Time-weighted dynamic time warping for satellite image time series analysis in r. **Journal of Statistical Software**, v. 88, 01 2019. 106, 113, 114, 115

MAUS, V. W. **Land use and land cover monitoring using remote sensing image time series**. Tese (Doutorado) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2016. 114

MESSNER, E.; DJAMALI, J.; TERHORST, Y.; SCHULLER, B.; CUMMINS, N.; SALAMON, G.; HUNGER, C.; BAUMEISTER, H. How did you like 2017? detection

of language markers of depression and narcissism in personal narratives. In: INTERSPEECH. **Proceedings...** Hyderabad, 2018. p. 3388–3392. 1, 12

MILGRAM, S. The small world problem. **Psychology Today**, v. 2, p. 60–67, 1967. 31

MITCHELL, T. M. **Machine learning**. [S.l.]: WCB McGraw-Hill, 1997. 2, 7

MOULTON, R. H.; VIKTOR, H. L.; JAPKOWICZ, N.; GAMA, J. Clustering in the presence of concept drift. In: BERLINGERIO, M.; BONCHI, F.; GÄRTNER, T.; HURLEY, N.; IFRIM, G. (Ed.). **Machine learning and knowledge discovery in databases**. Cham: Springer International Publishing, 2019. p. 339–355. ISBN 978-3-030-10925-7. 2, 14, 18, 19, 21, 25

NEWMAN, M. E. J.; GIRVAN, M. Finding and evaluating community structure in networks. **Physical Review E**, v. 69, n. 2, p. 026113, fev. 2004. Disponível em: <http://link.aps.org/doi/10.1103/PhysRevE.69.026113>. 29, 39

NGUYEN, H.-L.; WOON, Y.-K.; NG, W.-K. A survey on data stream clustering and classification. **Knowledge and Information Systems**, v. 45, n. 3, p. 535–569, Dec 2015. ISSN 0219-3116. Disponível em: <https://doi.org/10.1007/s10115-014-0808-1>. 1, 8, 11, 21

NGUYEN, L. T. T.; NGUYEN, N.-T.; VO, B.; NGUYEN, H. S. Efficient method for updating class association rules in dynamic datasets with record deletion. **Applied Intelligence**, v. 48, n. 6, p. 1491–1505, Jun 2018. ISSN 1573-7497. Disponível em: <https://doi.org/10.1007/s10489-017-1023-z>. 19

NGUYEN, T. T. T.; NGUYEN, T. T.; LIEW, A. W.-C.; WANG, S.-L. Variational inference based bayes online classifiers with concept drift adaptation. **Pattern Recognition**, v. 81, p. 280 – 293, 2018. ISSN 0031-3203. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0031320318301419>. 8, 13

ORMAN, G. K.; LABATUT, V.; CHERIFI, H. Qualitative comparison of community detection algorithms. **CoRR**, 2012. Disponível em: <https://arxiv.org/abs/1207.3603>. 29, 39

PALLA, G.; BARABASI, A.-L.; VICSEK, T. Quantifying social group evolution. **Nature**, v. 446, n. 7136, p. 664–667, April 2007. Disponível em: <http://dx.doi.org/10.1038/nature05670>. 41

PETITJEAN, F.; INGLADA, J.; GANCARSKI, P. Clustering of satellite image time series under time warping. In: INTERNATIONAL WORKSHOP ON THE ANALYSIS OF MULTI-TEMPORAL REMOTE SENSING IMAGES, 2011. **Proceedins...** [S.l.], 2011. p. 69 – 72. 113

PETITJEAN, F.; WEBER, J. Efficient satellite image time series analysis under time warping. **Geoscience and Remote Sensing Letters**, v. 11, p. 1143–1147, 06 2014. 105

PHRIDVIRAJ, M.; GURURAO, C. Data mining – past, present and future – a typical survey on data streams. **Procedia Technology**, v. 12, p. 255 – 263, 2014. ISSN 2212-0173. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2212017313006683>. 3, 7, 8

PONS, P.; LATAPY, M. Computing communities in large networks using random walks. **Journal of Graph Algorithms and Applications**, v. 10, n. 2, p. 191–218, 2006. 4, 41

PUSCHMANN, D.; BARNAGHI, P.; TAFAZOLLI, R. Adaptive clustering for dynamic iot data streams. **IEEE Internet of Things Journal**, v. 4, n. 1, p. 64–74, Feb 2017. ISSN 2327-4662. 1, 8, 11, 12, 15, 25

QIN, Y.; SHENG, Q. Z.; FALKNER, N. J.; DUSTDAR, S.; WANG, H.; VASILAKOS, A. V. When things matter: a survey on data-centric internet of things. **Journal of Network and Computer Applications**, v. 64, p. 137 – 153, 2016. ISSN 1084-8045. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1084804516000606>. 21

R Core Team. **R: a language and environment for statistical computing**. Vienna, Austria, 2018. Disponível em: <https://www.R-project.org/>. 68, 74

RAGHAVAN, U. N.; ALBERT, R.; KUMARA, S. Near linear time algorithm to detect community structures in large-scale networks. **Physical Review E**, v. 76, n. arXiv:0709.2938, p. 036106. 12 p, Sep 2007. 4, 41

RODRIGUES, F. A. **Caracterização, classificação e análise de redes complexas**. Tese (Doutorado) — Instituto de Física de São Paulo, Universidade de São Paulo, São Paulo, 2007. 3, 30, 31

RODRIGUES, P. P.; ARAúJO, J.; GAMA, J.; LOPES, L. A local algorithm to approximate the global clustering of streams generated in ubiquitous sensor networks. **International Journal of Distributed Sensor Networks**, v. 14, n. 10, p.

1550147718808239, 2018. Disponível em: <https://doi.org/10.1177/1550147718808239>. 19, 21, 24

ROSTEN, E.; DRUMMOND, T. Machine learning for high-speed corner detection. In: LEONARDIS, A.; BISCHOF, H.; PINZ, A. (Ed.). **Computer vision–ECCV 2006**. [S.l.]: Springer, 2006. p. 430–443. 2

ROSVALL, M.; AXELSSON, D.; BERGSTROM, C. T. The map equation. **The European Physical Journal Special Topics**, v. 178, n. 1, p. 13–23, 2009. Disponível em: <http://dx.doi.org/10.1140/epjst/e2010-01179-1>. 4, 41, 42, 51

ROSVALL, M.; BERGSTROM, C. T. Maps of random walks on complex networks reveal community structure. **National Academy of Sciences**, v. 105, n. 4, p. 1118–1123, 2008. 41, 42, 43, 51

ROUSSEEUW, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. **Journal of Computational and Applied Mathematics**, v. 20, p. 53 – 65, 1987. ISSN 0377-0427. Disponível em: <http://www.sciencedirect.com/science/article/pii/0377042787901257>. 25

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3rd. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009. ISBN 0136042597, 9780136042594. 2, 3, 7, 8, 11

SAKOE, H.; CHIBA, S. A dynamic programming approach to continuous speech recognition. In: INTERNATIONAL CONGRESS ON ACOUSTICS, BUDAPEST, 7., 1971. **Proceedings...** Budapest: Akadémiai Kiadó, 1971. v. 3, p. 65–69. 114

____. Dynamic programming algorithm optimization for spoken word recognition. **IEEE Transactions on Acoustics Speech and Signal Processing**, v. 26, p. 43–49, 1978. 114

SCRUCCA, L.; FOP, M.; MURPHY, T. B.; RAFTERY, A. E. mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. **The R Journal**, v. 8, n. 1, p. 205–233, 2017. Disponível em: <https://journal.r-project.org/archive/2017/RJ-2017-008/RJ-2017-008.pdf>. 3

SEBASTIANI, F. Machine learning in automated text categorization. **ACM computing surveys (CSUR)**, v. 34, n. 1, p. 1–47, 2002. 2

SETTLES, B. **Active learning literature survey**. [S.l.: s.n.], 2009. 10

____. **Active learning**. [S.l.]: Morgan and Claypool Publishers, 2012. 7, 10

SHAO, J.; TAN, Y.; GAO, L.; YANG, Q.; PLANT, C.; ASSENT, I. Synchronization-based clustering on evolving data stream. **Information Sciences**, 2018. ISSN 0020-0255. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0020025518307400>. 10, 12, 15, 19, 21, 25

SHIMABUKURO; BEUCHLE, R.; GRECCHI, R.; FRÉDÉRIC, A. Assessment of forest degradation in brazilian Amazon due to selective logging and fires using time series of fraction images derived from Landsat ETM+ images. **Remote Sensing Letters**, v. 5, 09 2014. 105

SILVA, J. A.; FARIA, E. R.; BARROS, R. C.; HRUSCHKA, E. R.; CARVALHO, A.; GAMA, J. Data stream clustering: a survey. **ACM Computing Surveys**, v. 46, n. 1, p. 13:1–13:31, jul. 2013. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/2522968.2522981>. 1, 3

SILVA, J. de A.; HRUSCHKA, E. R.; GAMA, J. An evolutionary algorithm for clustering data streams with a variable number of clusters. **Expert Systems with Applications**, v. 67, p. 228 – 238, 2017. ISSN 0957-4174. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417416304985>. 8, 13, 15

SUN, Y.; TANG, K.; ZHU, Z.; YAO, X. Concept drift adaptation by exploiting historical knowledge. **IEEE Transactions on Neural Networks and Learning Systems**, p. 1–11, 2018. ISSN 2162-237X. 3, 12

TRUDEAU, R. **Introduction to graph theory**. [S.I.]: Dover, 1993. (Dover Books on Mathematics Series). ISBN 9780486678702. Disponível em: <http://books.google.com.br/books?id=8nYH5OYEW24C>. 29

TUCKER, C. J. Red and photographic infrared linear combinations for monitoring vegetation. **Remote Sensing of Environment**, v. 8, n. 2, p. 127 – 150, 1979. ISSN 0034-4257. Disponível em: <http://www.sciencedirect.com/science/article/pii/0034425779900130>. 106

VELOSO, A.; OTEY, M. E.; PARTHASARATHY, S.; MEIRA, W. Parallel and distributed frequent itemset mining on dynamic datasets. In: PINKSTON, T. M.; PRASANNA, V. K. (Ed.). **High Performance Computing - HiPC 2003**. Berlin, Heidelberg: Springer, 2003. p. 184–193. ISBN 978-3-540-24596-4. 1, 9

VERBESSELT, J.; ZEILEIS, A.; HEROLD, M. Near real-time disturbance detection using satellite image time series. **Remote Sensing of Environment**, v. 123, 08 2012. 105

VIANA, M. P. **A metodologia das redes complexas para caracterizaćão do sistema de havers**. Tese (Doutorado) — Universidade de São Paulo, São Paulo, 2007. 32

WANG, S.; MINKU, L. L.; YAO, X. A systematic study of online class imbalance learning with concept drift. **IEEE Transactions on Neural Networks and Learning Systems**, p. 1–20, 2018. ISSN 2162-237X. 8, 13, 19

WATTS, D. J.; STROGATZ, S. H. Collective dynamics of small-world networks. **Nature**, v. 393, n. 6684, p. 440–442, June 1998. 31

WU, X.; ZHU, X.; WU, G. Q.; DING, W. Data mining with big data. **IEEE Transactions on Knowledge and Data Engineering**, v. 26, n. 1, p. 97–107, Jan 2014. ISSN 1041-4347. 3

ZHOU, A.; CAO, F.; QIAN, W.; JIN, C. Tracking clusters in evolving data streams over sliding windows. **Knowledge and Information Systems**, v. 15, n. 2, p. 181–214, May 2008. ISSN 0219-3116. Disponível em: <https://doi.org/10.1007/s10115-007-0070-x>. 10

ZHU, X. **Semi-Supervised learning literature survey**. [S.l.: s.n.], 2005. 2, 7, 35

ZHU, X. **Semi-supervised learning with graphs**. Tese (Doutorado), Carnegie Mellon University, Pittsburgh, PA, USA, 2005. 2, 7, 9, 10, 35

ZHU, X.; GHAHRAMANI, Z.; LAFFERTY, J. Semi-supervised learning using gaussian fields and harmonic functions. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 20., 2003. **Proceedings...** [S.l.], 2003. p. 912–919. 7

ZYTKOW, J.; RAUCH, J. **Principles of data mining and knowledge discovery: Third European Conference**. Berlin: Springer, 1999. (Lecture Notes in Computer Science). ISBN 9783540664901. Disponível em: <https://books.google.co.uk/books?id=uTzeRZFmaBgC>. 7