UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Monitoramento de transições em agrupamento de fluxos de dados

Afonso Matheus Sousa Lima

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

Afonso Matheus Sousa Lima

Monitoramento de transições em agrupamento de fluxos de dados

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Profa. Dra. Elaine Parros Machado de Sousa

USP – São Carlos
Agosto de 2022

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Lima, Afonso Matheus Sousa
L732m Monitoramento de transições em agrupamento de fluxos de dados / Afonso Matheus Sousa Lima; orientadora Elaine Parros Machado de Sousa. -- São Carlos, 2022.
101 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2022.

1. Monitoramento de Transições. 2. Fluxos de Dados. 3. Agrupamento. I. Sousa, Elaine Parros Machado de, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Afonso Matheus Sousa Lima

Cluster Tracking for Clustering of Streaming Data Sources

Master dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Profa. Dra. Elaine Parros Machado de Sousa

USP – São Carlos
August 2022

Este trabalho é dedicado à mim,
que carrega pedaços da família, de amigos e de mestres,
sem os quais não tornariam isto possível.

AGRADECIMENTOS

À minha família, pelo apoio incondicional. Ao meu pai Afonso Odério Nogueira Lima e à minha mãe Jackselene Maria de Sousa, por me mostrarem que a educação é o único caminho. À minha irmã Rebecca Sousa Lima, principal responsável pela pessoa que sou hoje. Aos meus tios e tias, por se importarem e se orgulharem com o que faço.

À Prof. Dra. Elaine Parros Machado de Sousa pela orientação, disponibilidade, interesse e participação ao orientar este trabalho. Sobretudo, agradeço principalmente pelo exemplo de empatia e pelas palavras de conforto, sou eternamente grato.

Aos meus amigos, que há muito tempo me acompanham. Henrique Alves e Samuel Moura, pelas discussões enriquecedoras e seus duelos. Antônio Ribeiro, Erik Henrique e Paulo Filho, pelas jogatinas e *realities*. Bruno Mendonça, João Gabriel Macedo e Danielly Gomes, pelas aventuras e completa falta de noção. Piero Capelo, Thalys Batista e Franco Saraiva, por fazerem de São Carlos parte do nordeste. Marcos Alencar e Marina Rocha, pelos conselhos e positividade inabalável.

Ao meu vizinho José Antônio e à minha amiga Eliane Karasawa, por serem as melhores companhias em épocas nada fáceis.

Aos meus colegas do Grupo de Base de Dados e Imagens, especialmente José Maria Clementino Junior, Bruno Squizato Faiçal e João Victor de Oliveira Novaes, pela receptividade, conversas e amizade.

À CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) e ao CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), pelo apoio financeiro a este projeto de mestrado.

Aos professores e servidores do Programa de Pós-Graduação em Ciências da Computação e Matemática Computacional, pelo excelente serviço prestado durante o trabalho.

Ao Instituto de Ciências Matemáticas e de Computação e à Universidade de São Paulo, por tornar possível que pesquisas de qualidade sejam feitas para o desenvolvimento profissional de seus alunos.

À Deus, que escreve certo por linhas tortas.

“Vou vencer distâncias.”
(Hércules, 1997)

RESUMO

LIMA, A. M. S. **Monitoramento de transições em agrupamento de fluxos de dados**. 2022. 101 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2022.

A disponibilização de grandes volumes de dados em diferentes áreas do conhecimento impulsiona o desenvolvimento de novas técnicas computacionais para processar massivas quantidades de dados, considerando as limitações de recursos disponíveis e tempo. Em particular, há domínios de problemas em que os dados são gerados e recebidos constantemente, sendo necessário realizar um processamento contínuo para que a análise possa refletir, com maior exatidão possível, o contexto atual dos dados. Os desafios inerentes a esse cenário motivam trabalhos na área de descoberta de conhecimento em fluxos de dados, definidos como sequências potencialmente infinitas de dados que são gerados continuamente, em geral em alta velocidade, com uma grande capacidade evolutiva, ou seja, mudanças ocorrem em seu comportamento ao longo do tempo.

Dentre as tarefas de descoberta de conhecimento em fluxos de dados, uma das mais abordadas na literatura é o agrupamento, que engloba tanto o agrupamento de pontos (objetos ou itens de dado provenientes de um ou mais fluxos de dados), quanto o agrupamento de fluxos de dados (ou seja, das próprias fontes geradoras dos fluxos). Embora diversos métodos de agrupamento desenvolvidos para fluxos de dados suportem evolução dos dados e adaptação de grupos, eles normalmente não são capazes de rastrear as mudanças ocorridas nos grupos ao longo do tempo. Entender como e quando os grupos mudam, conforme os fluxos de dados são processados, pode gerar conhecimento adicional relevante para o entendimento do problema, como padrões de mudança e sazonalidade. Esse rastreamento das mudanças em agrupamentos é chamado de monitoramento de transições. A maioria dos métodos presentes na literatura foram concebidos para serem usados em bases de dados convencionais com características temporais, sendo poucos os direcionados para tarefas com fluxos de dados, principalmente as que buscam agrupar os fluxos de dados em si.

Por isso, no escopo deste trabalho, foi desenvolvido a técnica CETra (*Cluster Evolution Tracker*) para monitoramento e detecção de transições que leva em consideração as características das tarefas de agrupamento de fluxos de dados. Essa técnica detecta diversos tipos de transições intra e inter grupos, considera a evolução gradual inerente aos fluxos de dados e é aplicável qualquer algoritmo de agrupamento de fluxos de dados que gere grupos disjuntos não sumarizados. CETra possui complexidade de tempo de processamento linear, o que a torna mais eficiente que métodos correlatos da literatura. A avaliação experimental realizada com dados sintéticos e dados reais mostram que a CETra é até duas vezes mais rápida que o método correlato aplicável a agrupamento de fluxos de dados. Além disso, CETra detecta transições que métodos correlatos não conseguem detectar pois esses não consideram a evolução gradual dos dados. Por fim, o

estudo com dados reais junto a um algoritmo de agrupamento de fluxos de dados mostra que CETra é capaz de acompanhar o processamento e formação de novos agrupamentos sem impactar significativamente no tempo geral dessa tarefa.

Palavras-chave: Monitoramento de transições, Fluxo de dados, Agrupamento.

ABSTRACT

LIMA, A. M. S. **Cluster Tracking for Clustering of Streaming Data Sources**. 2022. 101 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2022.

The availability of large volumes of data in different areas of knowledge drives the development of new computational techniques to process massive amounts of data, considering limitations of resources and time. In particular, problem domains where data is constantly generated and received, requiring continuous processing so that the analysis can reflect, as accurately as possible, the current context of the data. The challenges inherent to this scenario motivate work in the area of knowledge discovery in data streams, defined as potentially infinite sequences of data that are generated continuously, generally at high speed, with a great evolutionary capacity, that is, changes occur in their behavior over time.

Among the knowledge discovery tasks in data streams, one of the most discussed in the literature is clustering, which encompasses both the clustering of streaming data objects (data items coming from one or more data streams) and the clustering of streaming data sources (the sources generating the streams). While many clustering methods developed for data streams supports data evolution and cluster adaptation, they are typically not able to track changes in clusters over time. Understanding how and when clusters change as data streams are processed can generate additional knowledge relevant to understanding the problem, such as change's patterns and seasonality. This detection of changes in clusters is called cluster tracking. Most methods present in literature were designed to be used in conventional databases, with few being directed to data streams' tasks, especially those that seek to cluster streaming data sources.

Therefore, in the scope of this work, the CETra (*Cluster Evolution Tracker*) technique was developed for monitoring and detecting transitions, which takes into account characteristics of streaming data sources clustering tasks. This technique detects different types of intra and inter-cluster transitions, considers data streams' gradual evolution, and any streaming data sources clustering algorithm that generates non-summarized disjoint clusters is applicable. CETra has linear processing time complexity, which makes it more efficient than related methods in the literature. The experimental evaluation carried out with synthetic data and real data shows that CETra is twice as fast as the applicable related method. Furthermore, CETra detects transitions that correlated methods cannot detect because they do not consider the gradual evolution of data. Finally, the study with real data together with a streaming data sources clustering algorithm shows that CETra is able to follow processing and formation of new clusters without significantly impacting task's overall time.

Keywords: Cluster tracking, Data stream, Clustering.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ALGORITMOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CETra	*Cluster Evolution Tracker*
ECM	*Evolving Clustering Method*
iFCDS	*evolving Fractal-Based Clustering of Data Streams*
MEC	*Monitor of the Evolution of Clusters*
MONIC	*MONItoring Clusters*
PodClus	*Probability and Distribution-based Clustering*
SSE	*Sum Square Error*
TRACE	*TRAcking and validating Clusters Evolution using Regression analysis*

LISTA DE SÍMBOLOS

x^i — Objeto unidimensional

x_n^i — Objeto n-dimensional

S — Fluxo de dados unidimensional

DS — Fluxo de dados multidimensional

$\mathbb{S}$ — Conjunto de fluxos de dados unidimensionais

$\mathbb{S}^D$ — Conjunto de fluxos de dados multidimensionais

ζ^{t_i} — Agrupamento no instante de tempo t_i

K — Quantidade de agrupamentos

ζ_{ref} — Agrupamento referencial

$\zeta_{ref}^{t_i}$ — Agrupamento referencial no tempo t_i

ζ_{evo} — Agrupamento evolutivo

$\zeta_{evo}^{t_j}$ — Agrupamento evolutivo no tempo t_j

E — Quantidade de agrupamentos evolutivos

$C_x^{t_i}$ — Grupo referencial x no tempo t_i

$C_y^{t_j}$ — Grupo evolutivo y no tempo t_j

p — Quantidade total p de grupos

p_i — Quantidade total p_i de grupos referenciais no tempo t_i

p_j — Quantidade total p_j de grupos evolutivos no tempo t_j

N — Quantidade de elementos

N^{t_i} — Quantidade de elementos no tempo t_i de grupos

N^{t_j} — Quantidade de elementos no tempo t_j de grupos

τ — Limiar de sobreposição

τ_{split} — Limiar de separação

τ_{NEW} — Limiar de fontes de dados novas

τ_{MISS} — Limiar de fontes de dados desaparecidas

Int_i — Estatística interna

$\delta(Int)$ — Limiar de estatística interna

S — Quantidade de estatísticas internas

$I_{C_i}^{Int_i}$ — Valor da estatística interna Int_i pertencente ao grupo C_i

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

A disponibilização de dados e informação em diferentes áreas do conhecimento atingiu patamares impressionantes, algo que não era visto desde a invenção da tecnologia da informação no início do século XIX (GANTZ; REINSEL, 2012; WU *et al.*, 2013; AGGARWAL *et al.*, 2015). Duas das características mais relevantes desses dados são a diversidade e a quantidade. Diariamente, milhões de comentários e opiniões são escritos em diversas redes sociais. Milhares de sensores são usados para monitoramento de produção em variadas indústrias. Uma abundância de produtos são disponibilizados para venda e milhares de compras são feitas por inúmeras pessoas ao redor do mundo. Esses são alguns exemplos que representam esse crescimento imensurável na disponibilização de dados, que ainda tende a crescer com o passar dos anos. Esse cenário atual aumenta a necessidade de investir na criação de novas técnicas computacionais que trabalhem com esses dados de modo a obter informação relevante não óbvia que auxilie na execução de tarefas em diferentes domínios de aplicação.

Várias técnicas já são consolidadas na área de computação para manipular e analisar dados. No entanto, devido a esse extremo crescimento na quantidade dados que devem ser processados, mantendo a qualidade em tempo viável, muitas dessas técnicas já não são suficientes. Essa situação é agravada ainda mais quando trabalha-se com domínios de aplicação onde novos dados são gerados e recebidos constantemente. Nesse caso, atualizações devem ser feitas para que a análise possa refletir, com maior exatidão possível, o contexto atual dos dados. Essa problemática é abordada em pesquisas que trabalham com fluxos de dados (do inglês *data streams*), que são justamente definidos como uma sequência potencialmente infinita de dados que são gerados continuamente, em geral em alta velocidade, com uma grande capacidade evolutiva, ou seja, mudanças ocorrem em seu comportamento ao longo do tempo (SILVA *et al.*, 2013).

Essa característica evolutiva em fluxo de dados é um dos aspectos mais abordados em pesquisas nessa área (GAMA *et al.*, 2014; WEBB *et al.*, 2016; ANDERSON; KOH; DOBBIE, 2018; MOULTON *et al.*, 2018). Um exemplo dessa evolução é a mudança de interesse de

usuários sobre notícias. Em um determinado momento, um usuário pode estar interessado em notícias a respeito da copa do mundo, então é interessante recomendar páginas *web* que abordem esse conteúdo. Em outro momento, o usuário pode estar interessado em política, portanto é necessário atualizar as recomendações de acordo com o novo interesse do usuário. Vale ressaltar que as particularidades dessa evolução está diretamente relacionada com o contexto do problema abordado, ou seja, a frequência e a intensidade das mudanças é dependente do domínio de aplicação. Essa característica afeta as tarefas de mineração em fluxos de dados, pois os algoritmos devem ser capazes de tratar qualquer intensidade e tipo de mudança, mantendo os requisitos para processar esse tipo de dado com custo computacional factível.

Uma das tarefas de mineração mais abordadas em fluxos de dados consiste em realizar um particionamento dos objetos[1] do fluxo, gerando assim um agrupamento. Tendo o mesmo objetivo de quando se trabalha com base de dados convencionais (estáticas), essa tarefa consiste em formar um conjunto finito de grupos tal que, objetos pertencentes a um grupo são mais semelhantes entre si do que com objetos pertencentes a outros grupos (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996). Um exemplo desse tipo de tarefa em fluxo de dados é continuamente agrupar os resultados de pesquisas *web* para acelerar a navegação dos usuários ao realizarem novas consultas (ZENG *et al.*, 2004). No entanto, devido às características particulares de fluxos de dados, os algoritmos de agrupamento devem ser capazes de processá-los incrementalmente, ser escaláveis, gerar modelos compactos, detectar rapidamente a presença de *outliers*, suportar diferentes tipos de dados e adaptar-se às evoluções que ocorrem nos dados (SILVA *et al.*, 2013).

Além disso, devido a esse potencial de constante evolução no agrupamento, somente realizar a adaptação dos grupos para refletir o estado atual dos dados pode não ser suficiente para que haja um conhecimento aprofundado sobre o problema. Conforme os dados são processados, grupos novos podem surgir, outros podem desaparecer, podem se unir ou separar e suas características internas podem mudar. Entender como e quando esses grupos evoluem ao longo do tempo pode auxiliar a descoberta novos padrões para apoio às tomadas de decisão. Essa tarefa é diretamente relacionada ao conceito de monitoramento de transições de grupos (do inglês *cluster tracking*) (SILVA *et al.*, 2013) e tem motivado o desenvolvimento de novas técnicas que auxiliem na percepção dessas transições para proporcionar uma análise mais completa. Esse é o tópico de interesse deste trabalho de mestrado.

1.1 Contextualização e Motivação

Devido ao avanço da tecnologia, diversas aplicações atuais geram enormes quantidades de dados em velocidades muito altas. Logo, pesquisas voltadas à análise e descoberta de conhecimento em fluxos de dados tornaram-se bastante relevantes. Um fluxo de dados pode ser definido como uma massiva sequência de objetos $x^1, x^2, x^3, \ldots$ potencialmente infinita (SILVA *et*

[1] Também denominados: pontos, instâncias, elementos, exemplos, itens, entre outros.

al., 2013). Além disso, um fluxo de dados está constantemente suscetível a mudanças, uma vez que a maioria dos problemas do mundo real são dinâmicos. No entanto, as técnicas utilizadas em mineração de dados tradicional, usualmente aplicadas para descoberta de conhecimento (WITTEN; FRANK, 2002), em geral não são eficazes para análise de fluxos de dados, pois se baseiam em dados estáticos e não são capazes de trabalhar com a natureza dinâmica do fluxo (OLIVEIRA; GAMA, 2010). Por outro lado, técnicas específicas de análise e mineração de fluxos de dados visam o tratamento adequado das particularidades desse tipo de dado (GAMA, 2010).

Nesse contexto, o foco deste trabalho de mestrado é o problema de monitoramento de transições para agrupamento em fluxo de dados. Embora diversos métodos de agrupamento em fluxo de dados suportem evolução dos dados e consequentemente de grupos, é necessário incluir o rastreamento e o entendimento da própria evolução do grupo para obter informações adicionais sobre os dados e apoiar decisões estratégicas (SILVA *et al.*, 2013). Em outras palavras, é necessário fornecer informações sobre a natureza da mudança de um grupo. Será que um grupo está desaparecendo ou seus membros estão migrando para outros grupos? Um grupo emergente reflete um novo perfil de objetos de dados ou consiste em objetos antigos cujas características evoluíram? Essas são algumas perguntas que podem ser respondidas com o monitoramento dos grupos.

O monitoramento de transições é relevante em diversas aplicações, pois promove a criação de conhecimento sustentável sobre os fenômenos estudados e, consequentemente, a adoção de atitudes pró-ativas (OLIVEIRA; GAMA, 2010). Estudos que aplicaram o monitoramento de transições em bases de dados estáticas, associadas a um período de tempo, conseguiram obter conhecimento não óbvio sobre os problemas abordados (SPILIOPOULOU *et al.*, 2006; OLIVEIRA; GAMA, 2010; PEREIRA; MENDES, 2016). Um exemplo de aplicação em que o monitoramento de transições é útil para a descoberta de conhecimento, no contexto de empresas de telecomunicações, é apresentado no trabalho de Pereira e Mendes (2016), que estuda a evolução do comportamento dos usuários por meio de suas chamadas telefônicas durante um período de tempo de 15 dias. Para isso, foram detectadas as transições entre agrupamentos formados em períodos pré definidos. Por meio das transições, foi possível detectar características de sazonalidade nesse período de tempo, localizando transições de grupos que apresentavam padrões similares em períodos de tempo distintos. Com esse conhecimento, é possível traçar um perfil de comportamento dos usuários. Embora os autores tenham utilizado uma base de dados estática, o problema é facilmente traduzido para fluxos de dados, uma vez que é plausível realizar essa análise continuamente, detectando novas transições conforme novos dados forem gerados e processados (no caso, chamadas telefônicas que são disponibilizadas por uma fonte geradora de dados, os clientes).

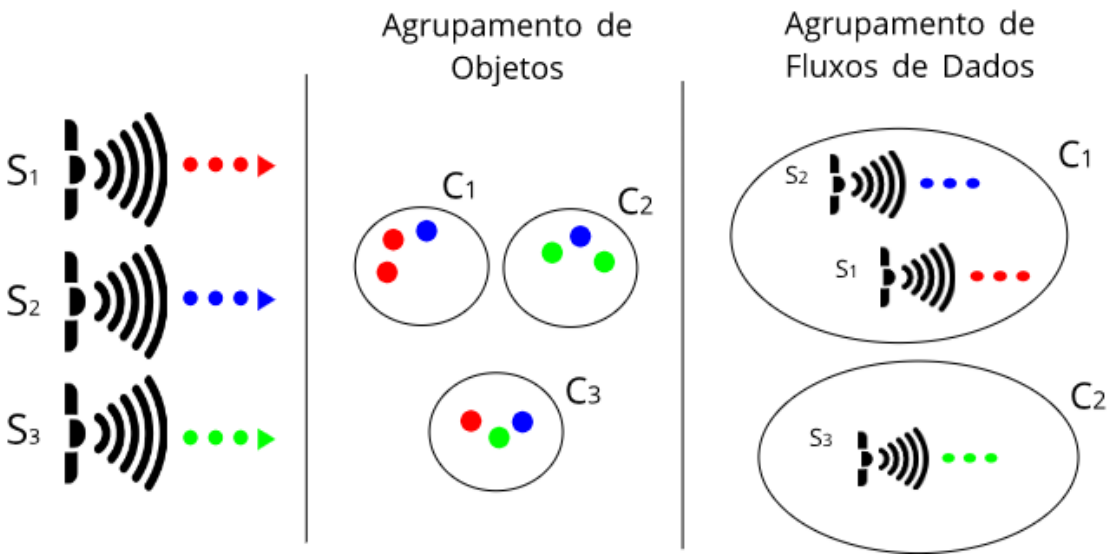
Executar esse monitoramento em paralelo à geração de novos dados pode resultar em análises mais precisas e confiáveis, permitindo também que uma transição potencialmente impor-

tante seja rapidamente capturada e apresentada ao especialista do domínio de aplicação. Além disso, manter um histórico de transições de grupos pode beneficiar tanto especialistas como os desenvolvedores de algoritmos. Esse potencial pode ser observado no trabalho de Alves, Barioni e Faria (2017), que utilizou as transições detectadas por um algoritmo de monitoramento para gerenciar a evolução de rótulos em uma abordagem de classificação semi-supervisionada. Assim, a detecção de transições também pode ser utilizada para desenvolver e aprimorar algoritmos.

1.2 Definição do Problema

Assim como na mineração de dados tradicional, tarefas em fluxos de dados podem ser designadas como: supervisionadas (e.g. classificação) ou não supervisionadas (e.g. agrupamento). Segundo Nguyen, Woon e Ng (2015), a tarefa de classificação consiste em utilizar dados conhecidos previamente para rotular novos objetos. Para a tarefa de agrupamento, o objetivo mais usual é agrupar os objetos de um ou mais fluxos de dados em subconjuntos chamados grupos (do inglês *clusters*), em que objetos de dados em um grupo são similares entre si e dissimilares a objetos de dados nos outros grupos, considerando uma determinada medida de similaridade (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996).

Figura 1 – Abordagens de agrupamento em fluxos de dados.



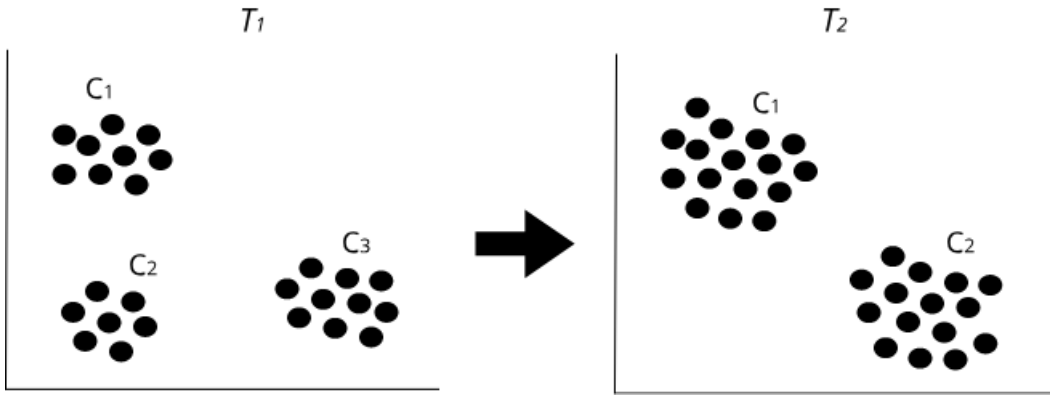
Fonte: Adaptada de Bones, Romani e Sousa (2016).

Para o contexto de fluxo de dados, existem duas abordagens principais para realizar a tarefa de agrupamento. A primeira abordagem, comumente chamada de agrupamento de objetos (em inglês *Clustering by Examples*), cada objeto é analisado individualmente, independente do fluxo de origem, e atribuído a um grupo. A segunda abordagem, discutida em mais detalhes na Seção 2.2.2, consiste em não mais agrupar os objetos individuais, mas sim identificar fluxos com comportamentos similares ao longo do tempo e agrupá-los (GAMA, 2010; RODRIGUES;

GAMA, 2014). Ambas abordagens são ilustradas na Figura 1. Tem-se três fontes geradoras de fluxos de dados (e.g. sensores) S_1, S_2, S_3, cada um contendo três objetos. Na primeira abordagem, cada um desses objetos é agrupado baseado em semelhança em um dos grupos C_1, C_2, C_3. Na segunda abordagem, os três objetos presentes em cada fluxo de dados é utilizado para caracterizá-lo e assim os próprios fluxos são agrupados nos grupos C_1, C_2.

Um exemplo de análise seguindo essa segunda abordagem é identificar zonas climáticas semelhantes por meio de grupos de sensores meteorológicos que estão constantemente coletando medidas de variáveis climáticas. Portanto, ao invés de agrupar as medidas captadas por esses sensores, esses dados são processados a fim de identificar as características de cada sensor (por exemplo, distribuição dos dados, estatísticas, medidas de sumarização, entre outros) para então associá-lo ao grupo de sensores com características mais similares. Este trabalho de mestrado abordará esse segundo tipo de agrupamento, sendo adotado o termo agrupamento **de** fluxos de dados (em inglês *Clustering by Variables* ou *Clustering of Streaming Data Sources* (GAMA, 2010; SILVA *et al.*, 2013; RODRIGUES; GAMA, 2014)).

Figura 2 – Agrupamentos formados em dois tempos distintos.

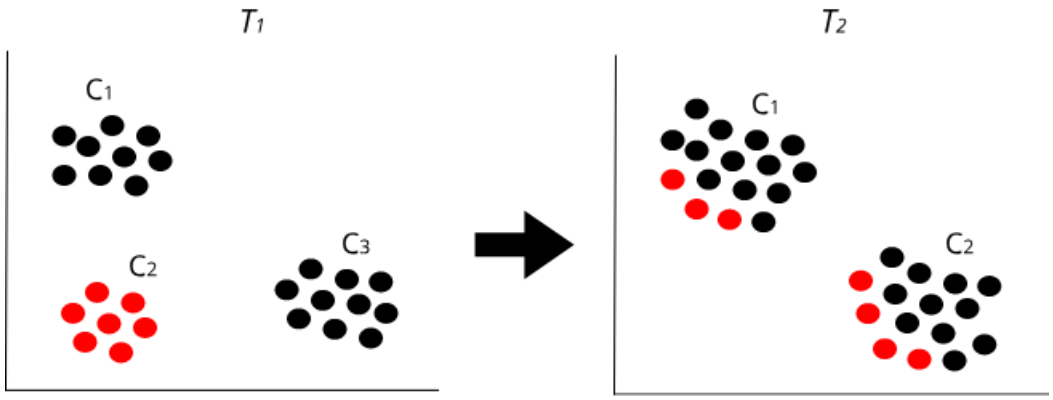


Fonte: Elaborada pelo autor.

Em tarefas de agrupamento de modo geral, tanto aplicadas a bases de dados convencionais quanto a fluxos de dados, o monitoramento de transições tem como objetivo detectar as mudanças que ocorrem em grupos conforme novos agrupamentos vão sendo realizados em tempos distintos. Dependendo das características evolutivas do problema, um grupo formado no tempo T_1 pode se dividir em um ou mais grupos, unir-se completamente ou parcialmente com outros grupos ou até mesmo desaparecer em um tempo posterior T_2. Há também a possibilidade do grupo se manter aparentemente inalterado em um tempo T_2, porém as suas características internas podem ter mudado. Um exemplo ilustrativo é apresentado na Figura 2, sendo constituído de dois agrupamentos de objetos bidimensionais, formados em tempos distintos. Vale notar que usualmente os rótulos dos grupos são redefinidos para cada agrupamento, logo não há correspondência direta entre rótulos iguais definidos em agrupamentos de tempos distintos. Percebe-se que o grupo C_2 do tempo T_1 não existe mais no tempo T_2 e que os demais grupos tiveram um aumento na quantidade de elementos. No entanto, não é possível precisar o que

aconteceu com os elementos do grupo que desapareceu. Observando a Figura 3, percebe-se que os elementos, agora destacados em vermelho, migraram para os outros grupos no tempo T_2, ou seja, esse grupo não simplesmente desapareceu, mas foi dividido e absorvido pelos demais. Houve então uma evolução desse grupo tal que seus elementos passaram a adquirir características mais semelhantes a outros grupos em T_2. Com esse conhecimento, é possível, por exemplo, realizar uma análise contextual dos motivos que acarretaram essa transição.

Figura 3 – Localização dos elementos do grupo C_2 em T_2.



Fonte: Elaborada pelo autor.

É relevante ressaltar que os algoritmos de agrupamento em fluxo de dados atuais, que em sua maioria oferecem suporte à evolução do agrupamento ao longo do tempo, geralmente detectam os grupos em cada um dos tempos T_1 e T_2 de modo eficaz, como no cenário ilustrado na Figura 2. No entanto, esses algoritmos não informam precisamente as mudanças ocorridas em qualquer um dos grupos entre T_1 e T_2. Identificar as particularidades dessas mudanças nos agrupamentos em tempos distintos é o objetivo do monitoramento de transições, executado como uma tarefa complementar ao agrupamento. Em outras palavras, no contexto de fluxos de dados, os algoritmos de agrupamento de objetos ou de agrupamentos de fluxos são responsáveis pela criação e adaptação do grupos ao longo do tempo, enquanto os algoritmos de monitoramento de transição visam descrever as mudanças ocorridas no grupos, atendendo os requisitos de processamento de fluxos de dados (descritos em mais detalhes na Seção 2.2.2).

Na literatura da área, as soluções mais relevantes propostas para monitoramento de transições são os *frameworks MONItoring Clusters* (MONIC) (SPILIOPOULOU *et al.*, 2006) e o *Monitor of the Evolution of Clusters* (MEC) (OLIVEIRA; GAMA, 2010; OLIVEIRA; GAMA, 2012). Tanto o MONIC como o MEC utilizam uma representação não sumarizada de grupos e realizam o monitoramento das transições comparando os grupos de dois agrupamentos gerados em tempos distintos. Para isso, é necessário identificar em que grupo do agrupamento no tempo T_2 estão localizados os elementos de cada um dos grupos do agrupamento do tempo T_1. A priori, essa abordagem de monitoramento de transições é aplicada a bases de dados convencionais que possuem informação de tempo associada. Outros trabalhos que abordaram o problema de monitoramento de transições em base de dados estáticas são focados em aprimorar os

anteriores (NTOUTSI; SPILIOPOULOU; THEODORIDIS, 2009; NTOUTSI; SPILIOPOULOU; THEODORIDIS, 2012; PEREIRA; MENDES, 2016) ou utilizar as transições detectadas por eles como uma informação adicional para outras tarefas (ALVES; BARIONI; FARIA, 2017).

O monitoramento de transições também é explorado para o contexto de fluxo de dados. O trabalho de Hawwash e Nasraoui (2012) apresenta um *framework* completo para minerar, monitorar e validar os grupos em fluxos de dados utilizando regressão linear para sumarizar e assim acompanhar as mudanças nas estatísticas desses grupos enquanto o fluxo de dados é processado. No trabalho de Namitha, Saju e Kumar (2018), é utilizado um mecanismo de monitoramento de transições que utiliza sumarizações dos grupos por meio de estatísticas e realiza comparações de distância entre os grupos de agrupamentos gerados consecutivamente. Ambos os métodos usam representações sumarizadas de grupos, portanto as transições são obtidas por meio de comparações entre as estatísticas dos grupos sumarizados. Além disso, esses métodos foram concebidos para tarefas de agrupamento de objetos e não para agrupamentos de fluxos de dados. Até o momento de finalização desta dissertação de mestrado, não foram encontrados trabalhos em que um mecanismo de monitoramento de transições tenha sido concebido para abordar as particularidades das tarefas de agrupamentos **de** fluxos de dados.

Em suma, as abordagens consolidadas de monitoramento de transições para base de dados convencionais consistem em realizar comparações entre objetos de grupos não sumarizados. Isso permite um monitoramento cujas transições detectadas refletem com maior exatidão as mudanças ocorridas entre grupos, já que cada objeto de dado do agrupamento é monitorado, ao custo de um processamento mais elevado. Já as abordagens de monitoramento de transições para fluxos de dados trabalham com versões sumarizadas dos grupos para manter a eficiência de processamento, realizando comparações entre estatísticas que resultam em um monitoramento de transições mais heurístico.

Para tarefas de agrupamento de objetos dos fluxos de dados pode ser inviável monitorar as transições em grupos não sumarizados, uma vez que objetos mais antigos tendem a ser descartados à medida que novos objetos são processados e pelo fato de a maioria dos métodos de agrupamento já necessitam sumarizar esses objetos. Todavia, para tarefas de agrupamento de fluxos de dados, os objetos processados são utilizados para caracterizar uma fonte geradora de dados que tende a permanecer no agrupamento ao longo do tempo. Por exemplo, em agrupamento de sensores, cada sensor tende a permanecer ativo por um período significativo de tempo e, portanto, estará presente em agrupamentos distintos gerados em instantes de tempos. Isso motiva o desenvolvimento de uma técnica para o monitoramento de transições específica para tarefas de agrupamento de fluxos de dados que utilize uma representação não sumarizada de grupos, buscando atender as características e requisitos de processamento para essa tarefa de agrupamento, o que até então não foi abordado na literatura.

1.3 Objetivos e Contribuições do Trabalho

Este trabalho de mestrado explora as características das tarefas de agrupamento de fluxos de dados para desenvolvimento de uma técnica de monitoramento de transições específica para esse contexto, visando uma abordagem que gere um conjunto de transições mais exatas e que sejam obtidas em tempo viável para esse tipo de problema. Para isso, é necessário adaptar e melhorar abordagens presentes na literatura e elaborar uma nova dinâmica para o processamento dos agrupamentos de fluxos de dados que são gerados continuamente ao longo do tempo.

A partir das considerações apresentadas neste Capítulo, foram formuladas as seguintes questões de pesquisa:

- Os tipos de transições e a acurácia da abordagem baseados em grupos não sumarizados podem ser mantidos para tarefas de agrupamento de fluxos de dados?
- Como a característica de evolução gradual dos fluxos e dos agrupamentos interfere nas transições detectadas?
- É possível reduzir a complexidade computacional da operação de detecção de transições, apresentada nas abordagens atuais, para atender o requisito de eficiência em tempo de processamento, fundamental para fluxos de dados?

Portanto, o objetivo principal deste trabalho de mestrado é propor e implementar uma técnica de monitoramento de transições específica para tarefas de agrupamento de fluxos de dados, tratando os aspectos levantados nas questões de pesquisa. Para tanto, foram estudados os fundamentos da área de fluxos de dados e as características das tarefas e dos algoritmos de agrupamento para esse tipo de dado, principalmente os que agrupam os fluxos. Foi realizando também um estudo aprofundado das técnicas presentes na literatura para monitoramento de transições, tanto para base de dados convencionais como para fluxos de dados. Com esse estudo, pôde-se conceber e implementar a técnica *Cluster Evolution Tracker* (CETra), que atende ao objetivo principal deste trabalho de mestrado.

A técnica de monitoramento de transições CETra, sendo a principal contribuição deste trabalho, atende aos requisitos de processamento, mudança gradual e detecção de vários tipos de transições que o contexto de fluxo de dados necessita. Quando comparada com outras técnicas de monitoramento da literatura que utilizam uma abordagem não sumarizada de grupos, a CETra é duas vezes mais rápida em termos de tempo de processamento para realizar todo processo de detecção de transições. Além disso, a técnica é capaz de obter um conjunto de transições que refletem melhor as mudanças graduais dos fluxos de dados e é aplicável a qualquer algoritmo que gere agrupamentos não sumarizados.

1.4 Organização do Trabalho

O restante desta dissertação de mestrado está dividida como segue:

- Capítulo 2: As propriedades e particularidades sobre fluxos de dados são apresentadas nesse Capítulo, além das características das transições em agrupamentos que evoluem ao longo do tempo.
- Capítulo 3: Trabalhos que exploraram o problema de monitoramento de transições em agrupamentos são introduzidos nesse Capítulo, assim como suas extensões e aprimoramentos.
- Capítulo 4: Apresentação e detalhamento do método de monitoramento de transições para agrupamento de fluxos de dados criado neste projeto de mestrado.
- Capítulo 5: Experimentos realizados e discutidos para validar as características da técnica proposta junto a outras abordagens da literatura são apresentados.
- Capítulo 6: Nesse Capítulo é apresentada a conclusão do trabalho desenvolvido e as sugestões para trabalhos futuros.

CAPÍTULO

2

FLUXO DE DADOS

Dados e algoritmos de aprendizado de máquina são utilizados há várias décadas para identificação de padrões e descoberta de conhecimento. Originalmente, pesquisas de aprendizado de máquina focavam no aprendizado em lote utilizando pequenas bases de dados estáticas (GAMA, 2010; NGUYEN; WOON; NG, 2015). Com a chegada da era *Big Data*, tem-se como maior desafio explorar grandes volumes de dados para extrair informações ou conhecimento útil para ações futuras, mantendo esse processo eficiente e em tempo real, uma vez que é praticamente impossível armazenar todos os dados observados (WU *et al.*, 2013). Além disso, há um crescente interesse em gerenciar massivas e potencialmente infinitas sequências de dados que são continuamente geradas normalmente em alta velocidade, os chamados fluxos de dados (SILVA *et al.*, 2013).

Segundo Gama (2010), fluxos de dados podem ser vistos como processos estocásticos, com eventos ocorrendo de modo contínuo e independente entre si. No trabalho de Muthukrishnan (2005), o fenômeno de fluxos de dados é apresentado como uma sequência de dados de entrada que chegam em altas taxas de velocidade, sendo capaz de estressar a infraestrutura computacional e comunicativa. Nguyen, Woon e Ng (2015) citam as características intrínsecas presentes em fluxo de dados, como a possibilidade de volume infinito, ordem cronológica e mudanças dinâmicas. Um exemplo cotidiano é o Google, que processa milhões de buscas diariamente, cada qual associada com um período tempo, e essas buscas mudam de acordo com os tópicos relevantes do momento.

Aprimorar a análise de fluxos de dados é relevante, visto que atualmente várias organizações geram uma quantidade grande de dados com uma velocidade enorme (NGUYEN; WOON; NG, 2015). Exemplos de fontes desses dados incluem: tráfego TCP/IP, GPS, chamadas telefônicas, *emails*, *etc.* (GAMA, 2010). Além disso, detectar a mudança de comportamento em fluxos de dados é útil em diversas aplicações, como: rastreamento na evolução de propagação de doenças e na carga de trabalho de servidores, análises meteorológicas, padrões em tráfego de rede, informações de sensores, fluxo em redes sociais e em motores de busca, entre outros

(BARBARA, 2002; NGUYEN; WOON; NG, 2015).

Conceitos básicos relacionados a fluxos de dados são apresentados na Seção 2.1. Na Seção 2.2 são sintetizadas técnicas de mineração de fluxos de dados, com ênfase na tarefa de agrupamento.

2.1 Conceitos Básicos

Formalmente, um fluxo de dados S é uma massiva sequência de objetos $\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^N$, ou seja, $S = \{\mathbf{x}^i\}_{i=1}^{N}$, onde N é potencialmente infinito ($N \to \infty$) (SILVA *et al.*, 2013). Essa representação é suficiente para definir um **fluxo de dados unidimensional**, que pode ser encontrado em sensores que geram informações únicas, como temperatura no contexto de sensores meteorológicos, por exemplo. No entanto, muitos trabalhos abordam problemas multidimensionais no contexto de fluxo de dados (SOUSA *et al.*, 2006; NUNES *et al.*, 2013; SILVA *et al.*, 2016; CYGANEK, 2018). Define-se, para um **fluxo de dados multidimensional**, que cada objeto $\mathbf{x}^i$ do fluxo multidimensional DS é descrito por um vetor de atributos n-dimensional $\mathbf{x}^i = [x_j^i]_{j=1}^{n}$ pertencendo a um espaço de atributos que pode ser contínuo, categórico ou ambos (SILVA *et al.*, 2013). Portanto, um único fluxo de dados passa a gerar múltiplas informações. Por exemplo, no contexto de sensores meteorológicos, cada sensor passa a gerar múltiplas informações como temperatura máxima, temperatura mínima e umidade, sendo assim um fluxo de dados com três dimensões.

Gama (2010) cita quatro características de um fluxo de dados, sendo elas:

1. Os dados de um fluxo chegam em tempo real.
2. Não há controle sobre a ordem em que os dados chegam, seja em um único fluxo ou entre fluxos.
3. Fluxos de dados, potencialmente, não têm limite de tamanho.
4. Uma vez que um elemento de um fluxo de dados é processado, ele é descartado ou arquivado. Ele não pode ser recuperado facilmente a não ser que ele esteja explicitamente armazenado em memória, sendo esta pequena em relação ao tamanho dos fluxos de dados.

Essas características fazem com que o processamento de fluxos de dados seja diferente do processamento feito para bases de dados estáticas. Segundo Gama (2010) e Nguyen, Woon e Ng (2015), em mineração de dados tradicional é possível ler os dados múltiplas vezes, processar com uma quantidade possivelmente ilimitada de tempo e memória, não há evolução nos dados e seus resultados precisam ter uma alta acurácia. Para mineração de fluxo de dados, os resultados são geralmente aproximados e os métodos devem satisfazer algumas restrições, como leitura única dos dados, processamento em tempo real, memória principal limitada e detecção da evolução

dos dados. Algoritmos de aprendizado adaptativo devem ter como propriedade a habilidade de incorporar novos dados e adaptar-se à evolução dos dados ao longo do tempo, a qual é chamada de mudança de conceito (do inglês, *concept drift*) (GAMA *et al.*, 2014). Sendo assim, um dos principais desafios para fluxos de dados é projetar algoritmos que possam detectar mudanças de conceito de maneira incremental sem necessitar de uma demanda crescente sobre os recursos de memória e processamento (BARBARA, 2002).

Um exemplo para ilustrar a mudança de conceito é a detecção de interesse de usuários. Em uma tarefa onde o objetivo é recomendar páginas *Web* para um usuário que está interessado em ler notícias sobre a copa do mundo, é necessário classificar essas páginas em relevantes e não relevantes de acordo com esse assunto. Portanto, todas as páginas que falam sobre a copa do mundo são relevantes, enquanto páginas que falam sobre outros assuntos são irrelevantes. Posteriormente, o usuário passa a querer saber mais sobre a situação política do país. Com isso, páginas com conteúdo político tornam-se relevantes, enquanto páginas sobre a copa do mundo e outros assuntos passam a ser irrelevantes para esse usuário. Assim, o algoritmo deve ser capaz detectar essa mudança de interesse, fazendo isso de modo incremental para continuar atendendo os requisitos para processamento de um fluxo de dados.

Para entender a definição de mudança de conceito, é preciso inicialmente definir o que é conceito. Webb *et al.* (2016) apresenta a necessidade de uma definição probabilística sobre conceito, uma vez que outras definições da literatura não são precisas dentro do contexto de aprendizado em fluxo. Baseado na definição de Gama *et al.* (2014) para mineração em fluxo, Webb *et al.* (2016) definem:

$$\begin{aligned} Conceito &= P(X,y), \text{para problemas supervisionados} \\ Conceito &= P(\chi), \text{para problemas não supervisionados} \end{aligned} \tag{2.1}$$

Ou seja, conceito é a distribuição conjunta de um conjunto X de valores de atributos com uma classe y. $P(\chi)$ denota simplesmente a distribuição probabilística do vetor de valores.

Como mineração de dados tradicional trabalha somente com um conceito, a distribuição da base de treinamento e da base de teste é a mesma durante todo o processo de aprendizagem (NGUYEN; WOON; NG, 2015). A mudança de conceito é justamente a modificação dessa probabilidade ao longo do tempo, ocorrendo em ambientes dinâmicos. Formalmente, Webb *et al.* (2016) definem que a mudança de conceito, entre um tempo t_0 e um tempo t_1, apresenta-se do seguinte modo:

$$\begin{aligned} P_{t0}(X,y) &\neq P_{t1}(X,y), \text{para problemas supervisionados} \\ P_{t0}(\chi) &\neq P_{t1}(\chi), \text{para problemas não supervisionados} \end{aligned} \tag{2.2}$$

Como as mudanças na distribuição de dados podem se manifestar de diferentes formas,

algoritmos de detecção de mudanças devem ser capazes de tratar vários casos. Gama *et al.* (2014) considera que os dados podem mudar de modo:

- **Repentino:** Mudando de um conceito para outro em um período de tempo curto.
- **Incremental:** Consistentemente mudando de conceito ao longo do tempo.
- **Gradual:** Alterna gradualmente entre um conceito antigo e um conceito novo.
- **Recorrente:** Conceitos que já existiram podem reaparecer conforme geração do fluxo.

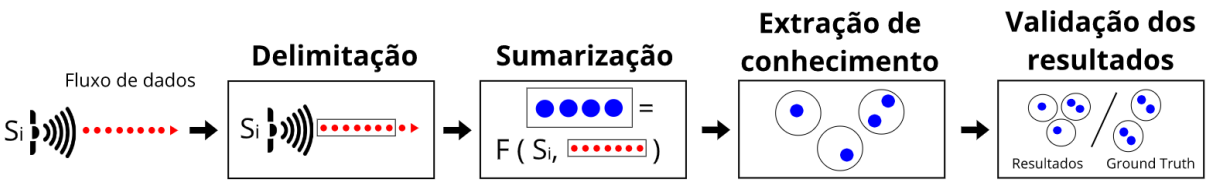
Além disso, um dos maiores desafios para algoritmos de detecção da mudança de conceito é não caracterizar um *outlier* como uma mudança, sendo considerado uma anomalia no fluxo. A maioria das técnicas de aprendizado adaptativo especializa-se em um conjunto de mudanças específicas.

Por fim, a mudança de conceito está diretamente relacionada com as alterações que ocorrem em agrupamentos. Já que os objetos do fluxo estão em constante evolução, é esperado que isso seja refletido na configuração dos grupos com o passar do tempo.

2.2 Mineração de Fluxo de Dados

Devido às características particulares que diferenciam fluxos de dados de bases de dados convencionais, existe um tratamento diferenciado para o processamento e a extração de conhecimento dessas sequências potencialmente infinitas de dados. Resumidamente, como apresentado na Figura 4, um algoritmo que tenha o objetivo de processar um fluxo de dados dever ser capaz de delimitar a leitura, sumarizar os dados e então extrair conhecimento deles (GAMA, 2010).

Figura 4 – Etapas para realização de mineração em fluxos de dados.

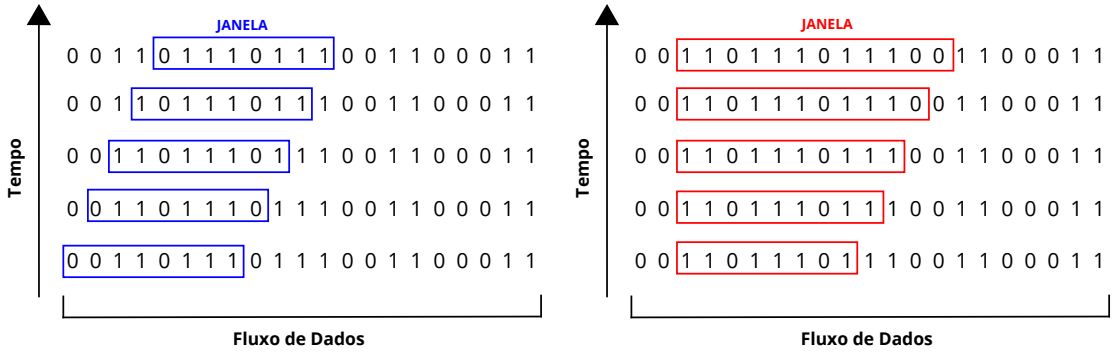


Fonte: Elaborada pelo autor.

O primeiro passo é realizar a **delimitação dos dados**. Muitos problemas têm como meta tomar decisões baseadas em avaliações estatísticas ou modelos obtidos a partir de um conjunto de dados sempre disponível. No entanto, em fluxos de dados há uma constante necessidade de processar novos objetos, fazendo com que seja impraticável o armazenamento em memória de todo o fluxo, além de não ser possível realizar mais de uma leitura dos mesmo. Para tratar esse

problema, uma estratégia amplamente empregada é utilizar janelas deslizantes (do inglês, *sliding windows*) (DATAR *et al.*, 2002; GABER; ZASLAVSKY; KRISHNASWAMY, 2005; GAMA, 2010). Essa técnica consiste em restringir o tamanho do conjunto de dados a ser analisado, buscando observar uma quantidade N de objetos mais recentes (BIFET; GAVALDA, 2007). Com essa implementação, é possível manter estatísticas contínuas em intervalos de tempo pré-definidos adaptando o tamanho N da janela para o contexto do problema que está sendo trabalhado. Dentre as diversas abordagens para implementar uma janela deslizante, a mais básica consiste em definir um tamanho fixo N que retrata o número de objetos que se deseja trabalhar (GAMA, 2010), como ilustrado na Figura 5a. Assim, quando um novo objeto recebido excede o tamanho N, o objeto mais antigo da janela é descartado, semelhante a uma estrutura de fila. Outra implementação empregada é chamada de *Landmark Window* (GAMA, 2010), que consiste em fixar um marco inicial para início da janela. Esse tipo de janela é utilizado, por exemplo, quando se deseja observar algum evento específico que ocorre nos dados, por um determinado período de tempo. Um exemplo desse tipo de janela é apresentado na Figura 5b.

Figura 5 – Exemplos de janelas deslizantes.



(a) Janela deslizante de tamanho fixo (b) Janela deslizante com início fixado no tempo

Fonte: Elaborada pelo autor.

Como fluxos de dados estão diretamente relacionados com a gestão de conjuntos massivos de dados, muitas vezes é necessário **sumarizar** esses dados. Algumas técnicas de sumarização comuns são as baseadas em histogramas, amostragem, *Sketchs* e *Wavelets* (GAMA, 2010; CORMODE *et al.*, 2011). As técnicas de histogramas consistem em representar a frequência de distribuição dos valores dos conjuntos de dados fazendo uma divisão de classes (*buckets*). As técnicas de amostragem consistem em fazer uma seleção de um subconjunto dos dados disponíveis para serem analisados em intervalos regulares. É interessante que esse subconjunto de dados seja uma amostra que represente bem o conjunto de dados total. Os *Sketchs* sumarizam os dados aplicando uma função de transformação nos dados de entrada. Assim, ao invés de utilizar só uma pequena parte dos dados, essa técnica permite que todos os dados sejam visualizados de forma compacta. As *Wavelets* são transformadas que buscam capturar a evolução ou tendências de funções numéricas realizando a decomposição do sinal em um conjunto de coeficientes, sinal esse que pode ser reconstruído utilizando-se todos os coeficientes do conjunto sem perda

significativa de informação. A escolha, tanto de técnicas de delimitação como de sumarização, depende do contexto do problema trabalhado.

A etapa seguinte, de **extração de conhecimento**, envolve principalmente tarefas de aprendizado supervisionado e aprendizado não supervisionado. Alguns exemplos dessas tarefas para fluxo de dados são:

- Classificação: tarefa supervisionada na qual são utilizados dados conhecidos previamente para rotular novos objetos do fluxo de dados, realizando a adaptação do modelo de classificação para a mudança de conceito (GABER; ZASLAVSKY; KRISHNASWAMY, 2007; NGUYEN; WOON; NG, 2015).
- Mineração de padrões frequentes: tarefa que consiste em identificar sequências similares de dados entre fluxos, mesmo com cada um deles evoluindo independentemente (TOYODA; SAKURAI; ISHIKAWA, 2013).
- Detecção de *outliers*: tarefa focada em identificar e tratar objetos com características muito discrepantes dos demais, para assim não influenciar tendenciosamente em outras tarefas de extração de conhecimento (BARBARA, 2002).
- Agrupamento: tarefa não supervisionada que consiste em particionar os objetos de um ou mais fluxos, ou os próprios fluxos de dados, em grupos, sendo que os elementos pertencentes a um mesmo grupo são mais semelhantes entre si do que com elementos pertencentes a outros (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996; SILVA *et al.*, 2013).

Como este trabalho aborda o monitoramento de transições em agrupamento de dados, esse tipo de tarefa será detalhado, tanto para agrupamento de objetos como para agrupamento de fluxos, nas Seções 2.2.1 e 2.2.2.

No contexto de tarefas de agrupamento, a **validação dos resultados** é bastante importante para mensurar a qualidade dos grupos obtidos entre os diferentes métodos, uma vez que uma técnica de agrupamento ideal para todos os problemas não existe (ARBELAITZ *et al.*, 2013). Essa avaliação de qualidade pode ser categorizada de acordo com a presença ou ausência de um conjunto de validação (do inglês, *ground-truth*). Se não houver um conjunto de validação, são utilizados índices internos, que consistem em usar das informações contidas nos próprios grupos formados pelos agrupamentos. Parte-se da premissa que um grupo é formado por objetos com características mais próximas entre si do que com objetos que estão mais distantes. Um dos índices internos mais comuns é o *Sum Square Error* (SSE), que mensura o total da soma dos erros ao quadrado das distâncias de um determinado objeto em relação a um ponto central do próprio grupo (BRUN *et al.*, 2007). Com isso, quando o menor o índice SSE melhor é a qualidade do agrupamento. Outros índices internos, que buscam medir o quão coesos estão

os elementos do outro em um agrupamento, são o Coeficiente de Silhueta e o Índice de Dunn (BRUN *et al.*, 2007).

No caso de existir um conjunto de validação, pode-se utilizar índices externos para mensurar a qualidade do agrupamento. Para isso, compara-se o resultado obtido pela técnica de agrupamento com a informação do conjunto de validação disponível. Ou seja, quanto mais objetos estiverem em um mesmo grupo tanto no agrupamento gerado pela técnica quanto no agrupamento de validação disponível, mais confiável é a técnica empregada. Diversos índices externos foram propostos na literatura (ARBELAITZ *et al.*, 2013), dois que são comumente empregados são o Índice de Precisão, que mensura qual a porcentagem dos objetos foram corretamente agrupados em cada grupo em relação à quantidade total de objetos associados ao grupo, e o Índice de Revocação, que mensura qual a porcentagem dos objetos foram corretamente agrupados em cada grupo em relação a quantidade total real de objetos que deveria ter sido associado ao grupo. Vale ressaltar que esses índices internos e externos podem ser utilizados tanto para objetos do fluxo como para os próprios fluxos de dados.

2.2.1 Agrupamento de Dados

A tarefa de agrupamento tem como objetivo agrupar um conjunto de dados em subconjuntos chamados grupos (do inglês *clusters*), em que objetos de dados em um grupo são similares entre si e dissimilares a objetos de dados nos outros grupos, em relação a uma determinada medida de similaridade (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996).

Existem várias abordagens aplicadas para agrupar dados, como: construção de uma hierarquia de grupos, particionamento de dados, densidade de objetos ou estruturas de grade (BERKHIN, 2006). A abordagem hierárquica consiste em construir uma hierarquia de grupos que geralmente é representada por um dendograma. Essa abordagem pode ser tanto aglomerativa, combinando objetos em grupos e recursivamente recombinando esses grupos em grupos cada vez maiores, como divisiva, iniciando com um único grupo que engloba todos os objetos e divisões são feitas recursivamente para formar novos grupos menores. Em uma abordagem de particionamento de dados, o foco é separar os objetos em grupos levando em consideração medidas de distância entre objetos ou assumindo distribuições probabilísticas. As técnicas de densidades baseiam-se na definição que um grupo é uma componente densamente conectada, e assim ele cresce em qualquer direção que a densidade leva, sendo então capaz de descobrir grupos com diferentes formatos. Por fim, usar uma estrutura de grade consiste em dividir o espaço de dados em células que formam uma estrutura de grade, e o agrupamento dos dados é feito de acordo com a distribuição de dados nas grades.

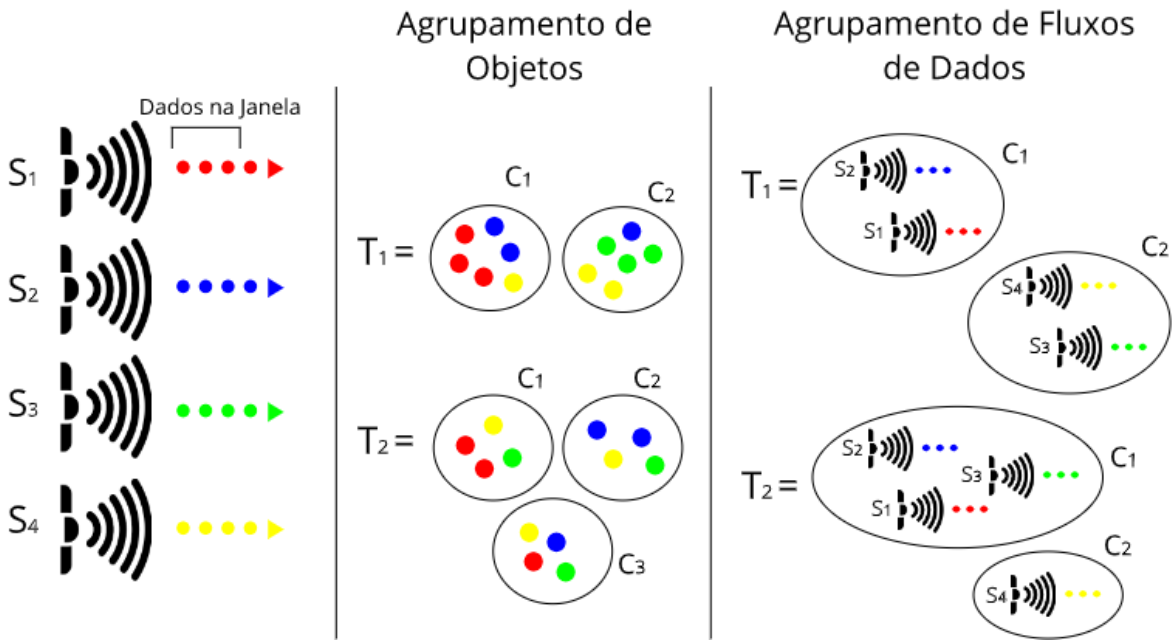
Levando em consideração uma tarefa de agrupamento no contexto de fluxo de dados, alguns requisitos fundamentais devem ser atendidos pelos algoritmos de agrupamento de maneira contínua, evolutiva, concisa e *online*. São eles:

- Representação de tamanho compacto (BARBARA, 2002; GAMA, 2010): como muitas vezes não é possível armazenar todos os objetos em memória principal, os grupos devem ser sumarizados de modo a não exceder a quantidade de memória disponível.
- Processamento rápido e incremental dos novos objetos (BARBARA, 2002; GAMA, 2010): o tempo gasto na atualização no modelo de agrupamento com a chegada de novos objetos não deve exceder a taxa de chegada dos dados.
- Identificar e reagir a mudanças (GAMA, 2010; WEBB *et al.*, 2016): como um fluxo de dados em aplicações do mundo real comporta-se de maneira dinâmica, é necessário identificar quando as mudanças ocorrem. Com isso, os grupos podem ser adequados à nova configuração da realidade.
- Rastreabilidade das mudanças ocorridas nos grupos (GAMA, 2010): deve ser possível identificar quais grupos foram criados, quais deixaram de existir e quais persistem ao longo do tempo, ou seja, monitorar as transições dos grupos.
- Rápida e clara identificação de *outliers* (BARBARA, 2002; GAMA, 2010): identificar *outliers* para que esses não influenciem negativamente na configuração do modelo.

Para fluxos de dados, há dois tipos de agrupamento (GAMA, 2010; SILVA *et al.*, 2013; RODRIGUES; GAMA, 2014). Pode-se agrupar os objetos dos fluxos de dados (em inglês *Clustering by Examples*), ou pode-se agrupar os próprios fluxos de dados (em inglês *Clustering by Variables* ou *Clustering of Streaming Data Sources*). A Figura 6 apresenta, de modo similar à Figura 1, a diferença entre agrupar objetos de um fluxo e agrupar fluxos de dados. Tem-se quatro fontes geradoras de fluxos de dados (e.g. sensores) S_1, S_2, S_3, S_4, cada uma com três objetos dentro de uma janela de dados. No primeiro caso, cada um dos três objetos pertencente aos fluxos são agrupados baseado em métricas de semelhança. No tempo T_1, os objetos são agrupados em dois grupos C_1 e C_2, já no tempo T_2 os dados contidos na janela são atualizados fazendo com que os objetos sejam reagrupados em três grupos C_1,C_2 e C_3. No segundo caso, os objetos presentes nas janelas de cada fluxo de dados são utilizados para caracterizar a fonte de dados e assim agrupá-las. No tempo T_1, as fontes de dados são agrupados em dois grupos C_1 e C_2, já no tempo T_2 os dados contidos na janela são atualizados fazendo com que as fontes de dados sejam reagrupados, gerando os dois grupos C_1 e C_2 com um particionamento diferente do presente em T_1.

Sobre o agrupamento de objetos em fluxo de dados, existem diversas técnicas na literatura (NGUYEN; WOON; NG, 2015). Por exemplo, o REPSTREAM (LÜHR; LAZARESCU, 2009) consiste em uma abordagem hierárquica baseada em grafos; o algoritmo STREAM (GUHA *et al.*, 2003), baseado em particionamento de dados, é uma adaptação do algoritmo k-means que aprimora alguns de seus aspectos para contemplar o fluxo de dados, como o suporte a apenas uma leitura dos dados, o uso reduzido de memória e algumas garantias de performances de execução;

Figura 6 – Diferença entre agrupamento de objetos e agrupamento de fluxos de dados.



Fonte: Adaptada de Bones, Romani e Sousa (2016).

o algoritmo CluStream (AGGARWAL *et al.*, 2003) utiliza de uma representação sumarizada dos grupos para aplicar em um algoritmo tradicional de agrupamento; o DenStream (CAO *et al.*, 2006), o OPTICS-Stream (TASOULIS; ROSS; ADAMS, 2007) e o incPreDecon (KRIEGEL *et al.*, 2011) são baseados em densidade; o D-Stream (CHEN; TU, 2007) e o MR-Stream (WAN *et al.*, 2009) são baseados em grade.

Algoritmos para agrupamento de fluxos de dados, de interesse deste trabalho, são abordados na próxima seção.

2.2.2 Agrupamento de Fluxos de Dados

O agrupamento de fluxos de dados tem como objetivo encontrar grupos de fluxos de dados que tenham comportamento semelhante no mesmo intervalo de tempo (GAMA, 2010). Portanto, o objetivo é particionar o conjuntos de fontes geradoras de dados (e.g. sensores de uma rede) em grupos disjuntos de acordo com as suas similaridades. As Definições 1 e 2 (BONES, 2018) formalizam esse procedimento para fluxos de dados unidimensionais e multidimensionais, respectivamente.

Definição 1 (**Agrupamento de fluxos de dados unidimensionais**). Sendo $\mathbb{S} = \{S_1, S_2, ..., S_M\}$ um conjunto de fluxos de dados unidimensionais, com $S = \{\mathbf{x}^i\}_{i=1}^{N}$, onde N é potencialmente infinito. Uma coleção de grupos disjuntos $\mathbb{P} = \{C_1, ..., C_m\}$ de $\mathbb{S}$ é formado de tal modo que $\bigcup_{i=1}^{m} C_i = \mathbb{S}$ e $C_i \cap C_j = \{\varnothing | \forall i \neq j\}$.

Definição 2 (**Agrupamento de fluxos de dados multidimensionais**). Sendo um conjunto de fluxos de dados multidimensional formalizado como $\mathbb{S}^D = \{DS_1, DS_2, ..., DS_M\}$, sendo DS formado por objetos n-dimensionais $\mathbf{x}^i = [x^i_j]^n_{j=1}$. Uma coleção de grupos disjuntos multidimensionais $\mathbb{P} = \{C_1, ..., C_m\}$ de $\mathbb{S}^D$ é formado de tal modo que $\bigcup^m_{i=1} C_i = \mathbb{S}$ e $C_i \cap C_j = \{\varnothing | \forall i \neq j\}$.

Poucos trabalhos encontrados na literatura abordam agrupamento de fluxos de dados. De modo geral, esses métodos delimitam a quantidade de dados observada, utilizando uma janela deslizante de mesmo tamanho para cada fluxo de dados, realizam a sumarização e então agrupam os fluxos mais semelhantes. Alguns dos trabalhos apresentados na literatura são: o método de Beringuer e Hüllermeier (BERINGER; HÜLLERMEIER, 2006), o ODAC (RODRIGUES; GAMA; PEDROSO, 2006), o método *Evolving Clustering Method* (ECM) (SONG; KASABOV, 2001; WIDIPUTRA; PEARS; KASABOV, 2011), o *Probability and Distribution-based Clustering* (POD-Clus) (CHAOVALIT, 2009) e o *evolving Fractal-Based Clustering of Data Streams* (eFCDS) (BONES; ROMANI; SOUSA, 2016; BONES, 2018). A seguir, o três últimos métodos são brevemente apresentados.

ECM

O método ECM foi proposto por Song e Kasabov (2001) e adaptado por Widiputra, Pears e Kasabov (2011) para realizar a predição de séries temporais. Esse método constrói modelos locais em dois passos principais, que são a extração dos perfis de relacionamentos entre séries temporais e a detecção e agrupamento de tendências recorrentes na série temporal quando um determinado perfil emerge. O ECM calcula os perfis de relacionamento entre as séries temporais utilizando a correlação de Pearson, extraindo apenas os coeficientes mais significantes, obtidos mediante testes de significância estatística. Então, o ECM calcula a dissimilaridade entre as séries temporais por meio da equação *Rooted Normalized One-Minus-Correlation* (RNOMC) (RODRIGUES; GAMA; PEDROSO, 2008), apresentada na Equação 2.3.

$$rnomc(a,b) = \sqrt{\frac{1 - corr(a,b)}{2}} \tag{2.3}$$

Assim que calculadas as correlações de um conjunto de séries temporais, o algoritmo decide agrupá-las baseando-se em um conjunto de regras de decisão para criar, remover ou unir grupos. Embora seja um método desenvolvido para séries temporais, o ECM é aplicado em experimentos dentro do contexto de fluxos de dados por apresentar processamento rápido e incremental dos agrupamentos, representação compacta, rastreabilidade das mudanças e identificação de *outliers* (BONES, 2018).

Pod-Clus

O algoritmo Pod-Clus desenvolvido por Chaovalit (2009) realiza, entre outras abordagens, agrupamentos de fluxos de dados com acompanhamento da evolução dos grupos (CHAOVALIT,

2009). Ele busca manter resumos, por meio de distribuições normais, e descarta informações detalhadas dos pontos de dados para construir dois tipos de grupos, podendo ser classificado como grupo normal ou como grupo *outlier*. Primeiramente, ele utiliza uma quantidade pré-definida de grupos para gerar o primeiro agrupamento para então armazenar e atualizar estatísticas de cada um dos grupos, sendo elas: média, desvio padrão e matriz de covariância. Essas estatísticas são utilizadas para então mensurar a similaridade entre os fluxos de dados considerando cada atributo para assim construir e atualizar os grupos, sendo a distância entre fluxos de dados multidimensionais para o centroide de um grupo definida na Equação 2.4.

$$Dist_{DSxC} = \sum_{f=1}^{F} \frac{(\mu_{DS_f} - \mu_{C_f})^2}{\sigma^2_{DS_f}} \tag{2.4}$$

Onde DS representa o fluxo de dados, C representa o grupo, f a dimensão, μ_{DS_f} a média da dimensão f no fluxo de dados DS, σ_{DS_f} o desvio padrão da dimensão f no fluxo de dados DS e μ_{C_f} é a média da dimensão f do grupo C.

É importante ressaltar que o Pod-Clus busca acompanhar a evolução de cada agrupamento por meio de um conjunto de regras de decisão, buscando identificar o aparecimento de novos grupos, o desaparecimento de um grupo que existia, a união e a divisão de grupos. Essa capacidade do algoritmo difere daquela proposta pelos métodos de monitoramento de transições pois o objetivo do Pod-Clus é fazer com que o agrupamento esteja de acordo com a configuração mais atual dos objetos gerados pelos fluxos, não se propondo a monitorar e detectar as transições entre grupos, ou seja, não se sabe a qual dos grupos os fluxos que transitaram pertenciam antes da atualização do agrupamento. Essa diferença é mais abordada no Capítulo 3.

eFCDS

O método (eFCDS), desenvolvido por Bones, Romani e Sousa (2016), foi projetado para criar grupos disjuntos, amorfos (sem forma definida) de um conjunto de fluxo de dados multidimensionais. Além disso, o eFCDS também lida com problemas de: redução da sobreposição entre agrupamentos e detecção de *outliers* que não possam ser integrados com seus grupos próximos. O eFCDS possibilita agrupar fluxos de dados com comportamento similar em um intervalo de tempo, considerando as correlações entre as dimensões por meio do cálculo da dimensão fractal (MANDELBROT, 1983). O método proposto também segue a evolução dos fluxos de dados em que grupos podem desaparecer ou serem criados ao longo do tempo.

A dimensão fractal é utilizada pelo eFCDS para medir as correlações entre as dimensões do fluxo de dados. Ao modelar dados como fractais, suas propriedades permitem representar características como: autossimilaridade em diferentes níveis de escala e complexidade infinita de representação (MANDELBROT, 1983; SCHROEDER, 2009). O processo de cálculo da dimensão fractal de modo incremental é realizado pelo algoritmo SID-Meter (*data Stream Intrinsic Dimension meter*) (SOUSA *et al.*, 2006). A parte do fluxo delimitada pela janela é

sumarizada em um valor de dimensão fractal denominado pelo autor como ponto de dimensão fractal *DF*, representando o comportamento das correlações no período de tempo em questão.

2.3 Considerações Finais

Este capítulo abordou conceitos e técnicas relacionadas a fluxos de dados. Como apresentado, a descoberta de conhecimento em fluxos de dados, tanto para objetos como para os próprios fluxos, motiva o desenvolvimento de novas técnicas e algoritmos capazes de tratar adequadamente as características particulares desse tipo de dado, tal como a constante chegada de novos dados e a capacidade evolutiva dos mesmos, além de atender a requisitos de processamento e memória.

Com essas propriedades presentes em fluxos de dados, é esperado que ocorram diversas mudanças que acarretem em atualizações nos resultados das tarefas de mineração. Então, aplicar métodos de monitoramento de transições para tarefas de agrupamento pode gerar conhecimento adicional relevante, uma vez que espera-se que o agrupamento mude consideravelmente conforme os fluxos são processados.

Por isso o objetivo deste trabalho é propor um mecanismo de monitoramento de transições para o contexto de agrupamento de fluxos de dados, que é uma tarefa de mineração ainda pouco abordada na literatura, cujos métodos propostos não realizam o monitoramento e detecção de transições em suas propostas originais.

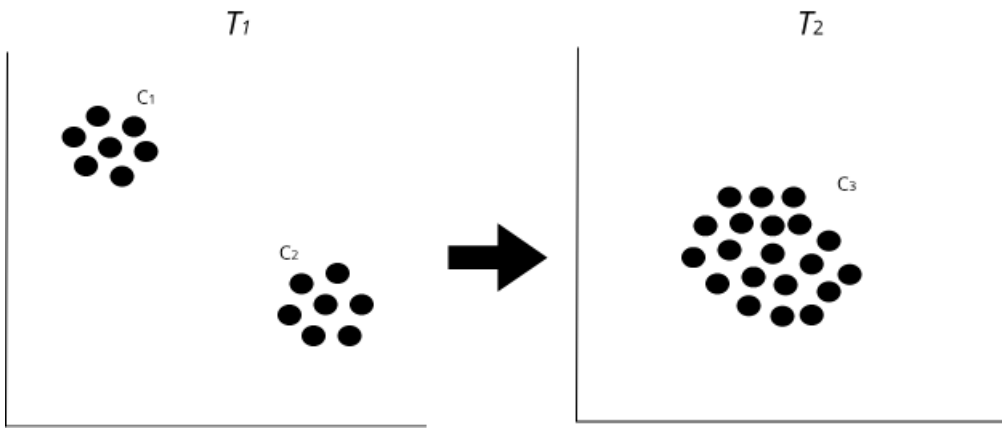
CAPÍTULO

3

MONITORAMENTO DE TRANSIÇÕES

Para Silva *et al.* (2013), a análise dos resultados de uma tarefa de agrupamento aplicada a fluxos de dados em diferentes intervalos de tempo pode fornecer aos usuários uma melhor compreensão sobre o comportamento dinâmico dos grupos. Enquanto muitos trabalhos focam em adaptar os grupos aos novos dados, é necessário incluir rastreamento e entendimento da própria evolução dos grupos, permitindo ganho de conhecimento e tomada de decisões. Para tanto, é necessário saber exatamente quais mudanças aconteceram nos grupos.

Figura 7 – Agrupamento em dois tempos distintos.

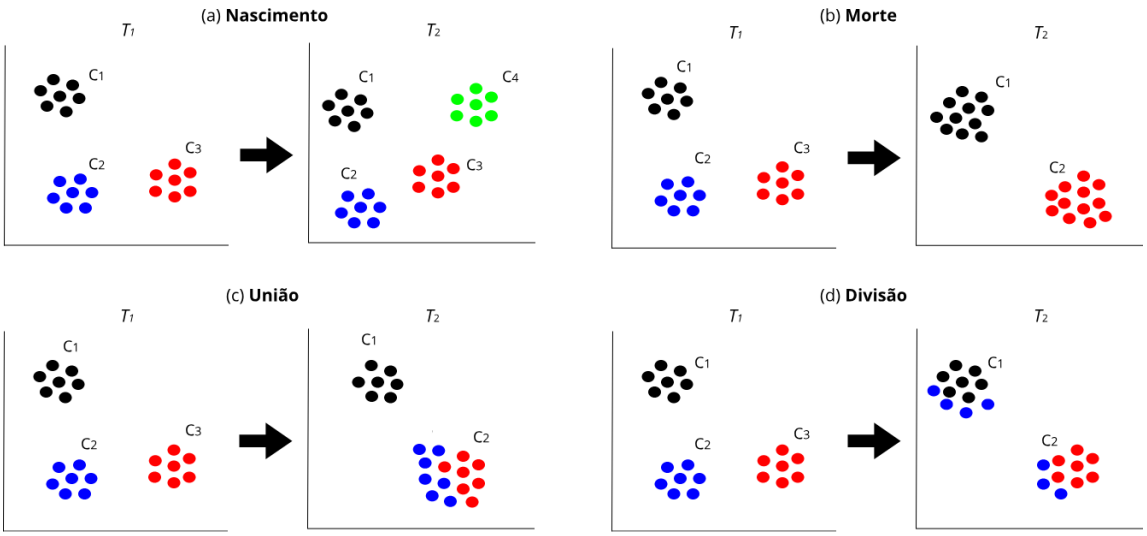


Fonte: Elaborada pelo autor.

Um exemplo pode ser observado na Figura 7, que ilustra uma mudança em um agrupamento de objetos bidimensionais em dois tempos distintos T_1 e T_2: no tempo T_1 existem dois grupos C_1 e C_2, enquanto no tempo T_2 existe somente um grupo C_3. A mudança no agrupamento é claramente percebida entre esses dois tempos, mas não é suficiente para responder às seguintes perguntas: qual foi o tipo de mudança que ocorreu entre os tempos T_1 e T_2? Será que os grupos C_1 e C_2 desapareceram e um novo grupo C_3 apareceu? Ou será que C_1 e C_2 uniram-se em um único grupo C_3? Ou C_3 representa objetos de C_1 ou C_2 cujas características evoluíram? As respectivas

respostas são obtidas utilizando o monitoramento de transições de grupos.

Figura 8 – Exemplos de cada tipo de transição externa.



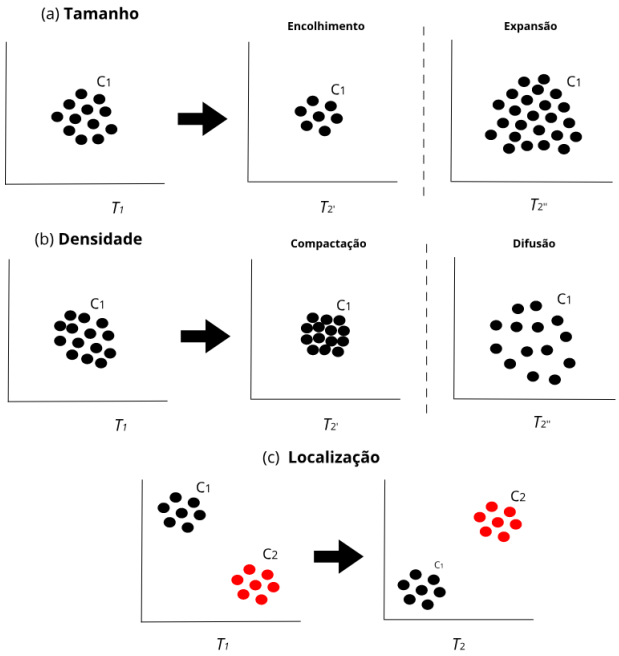
Fonte: Elaborada pelo autor.

Como indicado por Gama (2010) e Silva *et al.* (2013), é interessante que os algoritmos que se propõem em trabalhar com agrupamento em fluxo de dados sejam capazes de identificar quais foram as mudanças que ocorreram no agrupamento ao longo do tempo. Para identificar essas transições, a maioria das abordagens define uma taxonomia que engloba diversos tipos de transições (SPILIOPOULOU *et al.*, 2006; OLIVEIRA; GAMA, 2010). Segundo Oliveira e Gama (2010), existem no mínimo oito esquemas de taxonomia que classificam as transições de grupos na literatura. Pode-se destacar, no entanto, algumas transições que são comuns em todos esses esquemas, sendo elas:

- **Nascimento:** um novo grupo aparece (Figura 8a).
- **Morte:** um grupo previamente descoberto desaparece (Figura 8b).
- **União:** dois ou mais grupos unem-se em um único grupo (Figura 8c).
- **Divisão ou Separação:** um grupo é dividido e seus elementos são alocados em dois ou mais grupos (Figura 8d).
- **Sobrevivência:** um grupo não sofre nenhuma das outras transições.

Essas transições são classificadas como **externas**, pois estão relacionadas a mudanças na configuração do agrupamento. A Figura 8 exemplifica cada uma delas. Geralmente, as abordagens da literatura realizam comparações entre grupos obtidos em dois agrupamentos criados em tempos distintos, buscando descobrir equivalências entre eles. Além disso, também é possível caracterizar transições como **internas**, que são mudanças que ocorrem dentro de cada grupo. Para Oliveira e Gama (2010), as transições internas, exemplificadas pela Figura 9, são:

Figura 9 – Exemplos de cada tipo de transição interna.



Fonte: Elaborada pelo autor.

- **Tamanho:** a quantidade de objetos dentro de um grupo muda, podendo expandir ou encolher (Figura 9a).
- **Densidade:** a densidade de um grupo é alterada, podendo se tornar mais compacto ou mais difuso (Figura 9b).
- **Localização:** o grupo tem sua localização no espaço alterada (Figura 9c).

Existem poucos trabalhos na literatura que abordam o problema de monitoramento de transições. Dois que se destacam para bases de dados convencionais são: o *framework* de Spiliopoulou *et al.* (2006), chamado MONIC, e o de *framework* de Oliveira e Gama (2010), chamado MEC. Eles são capazes de realizar o monitoramento de diversos tipos de transições, tanto internas como externas, recebendo como entrada dois agrupamentos. Além disso, trabalhos posteriores sobre monitoramento de transições focam em aprimorá-los (NTOUTSI; SPILIOPOULOU; THEODORIDIS, 2009; NTOUTSI; SPILIOPOULOU; THEODORIDIS, 2012; PEREIRA; MENDES, 2016) ou utilizar as transições detectas por eles como uma informação adicional para outras tarefas (ALVES; BARIONI; FARIA, 2017). Em fluxos de dados, existem trabalhos que exploram o monitoramento de transições em tarefas de agrupamento de objetos (HAWWASH; NASRAOUI, 2012; NAMITHA; SAJU; KUMAR, 2018), sendo o *framework* de Hawwash e Nasraoui (2012) o mais interessante, chamado *Stream-Dashboard*.

Dito isso, as características e particularidades desses métodos serão apresentadas e discutidas ao longo deste Capítulo.

3.1 MONIC

No trabalho de Spiliopoulou *et al.* (2006) é proposto um *framework*, chamado *MONItoring Clusters* (MONIC), para modelar e monitorar transições de grupos. O seu modelo de transições engloba mudanças que envolvem mais de um grupo, gerando conhecimento de mudanças para todo o agrupamento.

3.1.1 Modelagem dos grupos

MONIC modela e monitora transição de grupos sobre dados que são coletados e particionados em agrupamentos ζ em tempos $t_1, ..., t_n$. Para trabalhar com isso, uma função de envelhecimento dos dados (*data ageing*) é usada para dar um peso menor para objetos de agrupamento antigos. MONIC assume reagrupamento ao invés de adaptação no grupo em cada tempo t_i, para que seja possível monitorar mudanças em grupos já existentes e em novos grupos. A função de envelhecimento dos dados é definida como:

Definição 3 (Função de Envelhecimento). Seja $t_1, ..., t_n$ a sequência de tempos em observação e seja D_i, com $i = 2...n$, o conjunto de objetos acumulados de t_{i-1} até t_i, sendo D_1 o conjunto de dados inicial e $D_i \cap D_j = \theta$ para $i \neq j$. Uma função de envelhecimento de dados atribui um peso $age(x,t_i) \in [0,1]$ para o registro de dados x no tempo t_i para cada $x \in \bigcup_{l=1}^{i} D_l$ e para cada t_i.

O pesos atribuídos pela função de envelhecimento determinam o impacto de um objeto sobre um agrupamento $\zeta_j \equiv \zeta_j(\bigcup_{t=1}^{i} D_t, age, t_i)$. Essa função pode ser aplicada em janelas deslizantes, em que os pesos dos objetos fora da janela são nulos.

Além disso, considerando um grupo $X \in \zeta_i$ descoberto em um tempo t_i , uma transição nesse grupo seria a mudança visível nele que aconteceria em um tempo posterior t_j. Para detectar essa transição, é necessário achar o grupo X no agrupamento ζ_j em t_j, isso se X ainda existir. É definido então, primeiramente, a sobreposição (do inglês, *overlap*) não simétrica entre grupos, e posteriormente o (melhor) "equivalente"(do inglês, *match*) para um grupo.

Definição 4 (Sobreposição de Grupos). Seja ζ_i, ζ_j, com $i < j$, os agrupamentos nos tempos t_i, t_j e seja $X \in \zeta_i, Y \in \zeta_j$ dois grupos. A "sobreposição de X para Y"é a soma normalizada dos pesos dos objetos na interseção de seus conjuntos:

$$overlap(X,Y) = \frac{\sum_{a \in X \cap Y} age(a,t_j)}{\sum_{x \in X} age(x,t_j)} \tag{3.1}$$

Definição 5 (Equivalência de Grupos). Seja X um grupo em um agrupamento ζ_i no tempo t_i e Y um grupo em um agrupamento ζ_j em um tempo $t_j > t_i$. Além disso, seja $\tau \in [0.5, 1]$ um

valor limiar. Y é um equivalente de X em ζ_j se e somente se $Y \in \zeta_j$ for o grupo com a máxima sobreposição para X e a sobreposição entre X e Y for no mínimo τ:

$$overlap(X,Y) = max_{U \in \zeta_j} overlap(X,U) \geq \tau \tag{3.2}$$

Se não existir tal $Y \in \zeta_j$, então $match_\tau(X, \zeta_j) = \varnothing$.

Pela Definição 5, ζ_j pode conter no máximo uma equivalência para cada grupo em ζ_i, embora o mesmo grupo em ζ_j pode ser equivalente com mais de um grupo em ζ_i. O limiar τ é restrito em um intervalo $[0.5, 1]$ para enfatizar que um grupo é equivalente somente se contém no mínimo metade do grupo pivô (e.g. metade dos seus membros, se os membros estiverem com pesos atribuídos iguais). Um empate pode ocorrer somente quando $\tau = 0.5$. Diferentes métodos podem ser usados para desempatar, por exemplo: escolher o Y com maior sobreposição reversa $overlap(Y,X)$ ou o Y que é mais próximo em tamanho de X.

3.1.2 Transições de grupos

Em MONIC, uma transição de grupo em certo tempo é um mudança experienciada por um grupo que foi descoberto em um tempo anterior. Essa transição pode ser **interna**, relacionada a mudanças no conteúdo e forma do grupo, ou **externa**, relacionada a mudança no seu relacionamento com o resto do agrupamento.

Tabela 1 – MONIC: Transição externa de grupos.

Tipo de transição	**Notação**	**Indicador**
O grupo sobrevive	$X \rightarrow Y$	$Y = match_\tau(X, \zeta_j) \quad AND \quad \nexists Z \in \zeta_i \setminus X : Y = match_\tau(Z, \zeta_j)$
O grupo é dividido	$X \overset{\subseteq}{\rightarrow} \{Y_1, ..., Y_p\}$	$(\forall_u 1...p : overlap(X,Y_u) \geq \tau_{split}) \wedge overlap(X, \bigcup_{u=1}^{p} Y_u) \geq \tau \wedge (\nexists Y \in \zeta_j \setminus \{Y_1 ..., Y_p\} : overlap(X,Y) \geq \tau_{split})$
O grupo é absorvido	$X \overset{\subseteq}{\rightarrow} Y$	$Y = match_\tau(X, \zeta_j) \quad AND \quad \exists Z \in \zeta_i \setminus \{X\} : Y = match_\tau(Z, \zeta_j)$
O grupo desaparece	$X \rightarrow \odot$	Nenhum dos casos acima servem para X
Um novo grupo aparece	$\odot \rightarrow Y$	Nenhum dos casos acima servem para Y

Fonte: Adaptada de Spiliopoulou *et al.* (2006).

As transições externas são apresentadas na Tabela 1. Um grupo $X \in \zeta_i$ sobrevive em ζ_j se há um equivalente a ele em ζ_j sujeito a τ e se essa equivalência não cobre outro grupo em ζ_i. Se a equivalência cobre mais de um grupo em ζ_i, então X foi absorvido. Se nenhuma equivalência existe, então uma divisão pode ter ocorrido, ou seja, o conteúdo de X está em mais de um grupo em ζ_j. Então, para esse caso, as sobreposições devem ser não menos que um novo limiar τ_{split}, com $\tau_{split} < \tau$, para prevenir casos degenerativos. Em suma, todos esses grupos juntos devem

formar uma equivalência para X. Se nenhum desses casos ocorrer, então X desapareceu. Novos grupos são detectados após traçar todas as transições externas para cada grupo de ζ_i, pois assim sabe-se que eles são grupos que não são o resultado de mudanças externas.

Tabela 2 – MONIC: Transição interna de grupos.

Tipo de transição	**Subtipo**	**Notação**	**Indicador**
1.Transição de tamanho	1a. o grupo diminui 1b. o grupo expande	$X \searrow Y$ $X \nearrow Y$	$\sum_{x\in X} age(x,t_i) > \sum_{y\in Y} age(y,t_j) + \varepsilon$ $\sum_{y\in Y} age(y,t_j) > \sum_{x\in X} age(x,t_i) + \varepsilon$
2.Transição de compactação	2a. o grupo torna-se mais compacto 2b. o grupo torna-se mais difuso	$X \xrightarrow{\bullet} Y$ $X \xrightarrow{*} Y$	$\sigma(Y) < \sigma(X) - \delta$ $\sigma(Y) > \sigma(X) + \delta$
3. Transição de localização	Mudança para o centro (I1) ou distribuição (I2)	$X \cdots \rightarrow Y$	I1. $\lvert\mu(X) - \mu(Y)\rvert > \tau_1$ //média I2. $\lvert\gamma(X) - \gamma(Y)\rvert > \tau_2$ //skewness
Sem mudança	-	$X \leftrightarrow Y$	Nenhum dos casos acima servem

Fonte: Adaptada de Spiliopoulou *et al.* (2006).

As transições internas são apresentadas na Tabela 2. É importante notar que um grupo pode sofrer mais de um tipo de transição interna. Por exemplo, um grupo $X \in \zeta_i$ equivalente a $Y \in \zeta_j$ pode se tornar maior em tamanho e também mais compacto. Para detectar transições de tamanho, compara-se os objetos de X e Y, levando em consideração os pesos dos objetos de X. Transições de compactação são detectadas por meio de valores obtidos a partir da distribuição dos dados, como o desvio padrão mostrado na Tabela 2. Outras métricas podem ser usadas, como a *kurtosis*. Transições de localização são movimentos dos grupos dentro do espaço, indicadas por mudanças na média e na *skewness*. É importante notar que existe, para cada tipo de transição interna, uma constante definida previamente que representa um valor de folga para que a transição seja significante.

MONIC ainda monitora o tempo de vida de grupos e agrupamentos para ganhar conhecimento sobre a evolução da população. Se a maioria dos grupos em um agrupamento sobrevivem de um período para o outro, então o conjunto de dados está passando por um período estático. Se as transições são frequentes, então a população é volátil e o agrupamento está tendo mudanças em sua configuração.

Definição 6. Seja X um grupo e t_i o primeiro ponto de tempo em que ele apareceu (como parte do agrupamento ζ_i). O tempo de vida de X é o número de instantes de tempo nos quais X sobreviveu. Define-se:

1. Um tempo de vida estrito (***lifetimeS***), representando o número de vezes seguidas que o grupo sobreviveu sem transições internas.

2. Um tempo de vida sujeito a transições internas (***lifetimeI***), representando todas as vezes que o grupo sobreviveu.

3. Um tempo de vida com absorções (***lifetimeA***), representando todas as vezes que o grupo sobreviveu, mesmo que ele tenha sido absorvido por outro.

O tempo de vida de um grupo é computado de trás para frente, ou seja, inicia-se com o agrupamento ζ_n e inicializa-se o tempo de vida de seus grupos com 1. Em um ponto no tempo mais cedo t_i o tempo de vida estrito de um grupo X é 1 se X não sobreviveu em t_{i+1}. Se há um $Y \in \zeta_{i+1}$ com $X \leftrightarrow Y$, então $lifetimeS(X) = lifetimeS(Y) + 1$. Se há um $Y \in \zeta_{i+1}$ com $X \rightarrow Y$, então o tempo de vida de X sob transições internas é $lifetimeI(X) = lifetimeI(Y) + 1$. Se há um $Y \in \zeta_{i+1}$ com $X \leftrightarrow Y$ ou $X \overset{\subseteq}{\rightarrow} Y$, então o tempo de vida de X com absorções é $lifetimeA(X) = lifetimeA(Y) + 1$. A sobrevivência de um agrupamento construído em t_i é refletido no número de grupos que sobreviveram ou foram absorvidos em um tempo t_{i+1}:

Definição 7. Seja ζ_i o agrupamento em $t_i, i...n-1$. Sua taxa de sobrevivência é a porção de seus grupos que sobreviveram (possivelmente com transições internas) em ζ_{i+1}. Sua taxa absorção é a porção de grupos absorvidos por grupos em ζ_{i+1}.

$$survivalRatio(\zeta_i) = \frac{|\{X \in \zeta_i | \exists Y \in \zeta_{i+1} : X \rightarrow Y\}|}{|\zeta_i|} \tag{3.3}$$

$$absorptionRatio(\zeta_i) = \frac{\left|\left\{X \in \zeta_i | \exists Y \in \zeta_{i+1} : X \overset{\subseteq}{\rightarrow} Y\right\}\right|}{|\zeta_i|} \tag{3.4}$$

onde $|\zeta_i|$ denota a quantidade de grupos do agrupamento ζ_i.

Para avaliar o funcionamento do MONIC, os autores realizaram um experimento utilizando publicações presentes em uma seção de uma biblioteca digital sobre "aplicações de base de dados", com o objetivo de ganhar conhecimento adicional sobre a evolução dos grupos e estudar o impacto de diferentes configurações de parâmetros. Esses documentos foram agrupados utilizando o *bisecting K-means*, com K=10, e uma janela deslizante de tamanho 2 que foi usada no processo de envelhecimento de dados. Um agrupamento foi gerado para cada ano do período abordado, os quais foram utilizados como entrada do método de monitoramento.

Nesse experimento, MONIC foi capaz de detectar diversas transições, as quais refletem mudanças nos tópicos de interesse da época. Por exemplo, nos primeiros quatro agrupamentos, há um grupo crescente constituído por documentos cujo assunto principal era "regras de associação", porém nos agrupamentos seguintes esse grupo é dividido e até mesmo desaparece. Investigando o contexto dessa mudança, percebeu-se que a partir de um certo ano muitas publicações sobre mineração de dados foram submetidas nessa seção específica. Isso mostra uma evolução nos tópicos de pesquisa da época, em que durante um período de quatro anos houve um constante crescimento de publicações sobre um assunto, porém ele foi perdendo relevância em favor de outro que se tornava cada vez mais popular. Portanto, a transição detectada teve valor semântico dentro do contexto do problema.

É importante notar que somente observar os agrupamentos gerados pelo *K-means* em cada ano não fornece o mesmo conhecimento sobre as mudanças detectadas pelo MONIC. Como

os métodos de agrupamento não têm o objetivo de manter informações sobre os grupos, só com o *k-means* não é possível saber que um mesmo grupo cresceu ao longo de um período e desapareceu em outro, por exemplo. Por fim, vale ressaltar que os autores não reportaram experimentos utilizando o MONIC em fluxos de dados e não descreveram um método de avaliação de corretude para as transições detectadas, dependendo de análises semânticas para validá-las.

3.2 MEC

Com o objetivo de obter uma maior compreensão dos processos evolutivos dos grupos, Oliveira e Gama (2010) desenvolveram o *framework Monitor of the Evolution of Clusters* (MEC), capaz de monitorar as transições de grupos ao longo do tempo através da identificação de relacionamentos temporais entre eles. Esse *framework* abrange uma taxonomia de vários tipos de transições de grupos, que podem ser externas ou internas, um mecanismo de rastreamento que depende da representação do grupo e um algoritmo de detecção de transições.

3.2.1 Modelagem dos grupos

MEC assume que os grupos podem ser representados por dois esquemas principais de representação. **Representação por enumeração** ($C_j(t) = \{\vec{x}_1, ..., \vec{x}_m\}$), a qual é a maneira mais usada e direta para definir grupos, sendo caracterizados pelos seus elementos. Esse tipo de representação não envolve perda de informação e permite o monitoramento de cada objeto ao longo do tempo, obtendo assim transições mais precisas e confiáveis. Esse é o mesmo tipo de representação usada pelo MONIC.

Outra forma de representação é chamada de **representação por compreensão** (do inglês *comprehension*), que sumariza os grupos em valores estatísticos (ver Definição 8). Opta-se por esse tipo de representação quando não é possível manter os elementos originais, tanto por motivos de armazenamento como de privacidade.

Definição 8. Um grupo C_j é um objeto temporal caracterizado pelas seguintes estatísticas:

$$C_j = \{ID, t, m, sup, r, p, \vec{c}\} \tag{3.5}$$

onde ID é o identificador único de C_j, com $j = 1, ..., k$, t é o ponto no tempo onde o grupo apareceu pela primeira vez, com $t = 1, ..., T$, m é o número de observações atribuídas para C_j, sup é o suporte de um grupo, usado para acessar a importância relativa do grupo, r é o raio do grupo, p representa a densidade do grupo e $\vec{c}$ representa o centroide do grupo.

Nesse esquema de compreensão, foi considerado uma medida de centralidade (centroide do grupo), uma medida de dispersão (raio do grupo) e uma medida de densidade (objetos

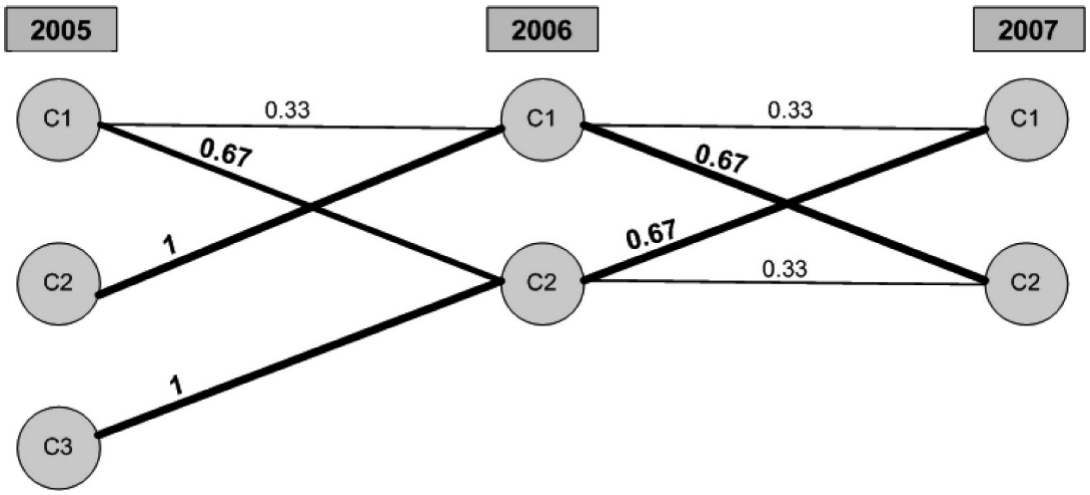
esféricos d-dimensionais). O lado negativo dessas medidas é que eles só conseguem descrever grupos esféricos ou circulares. Além disso, essa sumarização acarreta em perda de informação, podendo comprometer a acurácia do processo de descoberta de equivalências entre grupos de tempos diferentes.

3.2.2 Transições de grupos

O conceito de evolução adotado pelo MEC refere-se às transições sofridas por grupos durante o intervalo de tempo sob observação $[t_i, t_{i+\Delta t}]$, com $\Delta t = 1, ..., T - i$. O mecanismo de transição foi dividido em dois métodos diferentes, um para cada tipo de representação. Cada método faz um processo de mapeamento para descobrir as equivalências de grupos entre dois tempos distintos.

No método para grupos representados por enumeração, o processo de mapeamento explora o conceito de probabilidade condicional e é restrito por um limiar de sobrevivência pré-definido τ. Esse limiar serve para definir que uma equivalência entre grupos de agrupamentos distintos deve ter uma porcentagem mínima τ de elementos semelhantes. Para isso, os conjuntos de dados devem ser compostos por observações de elementos que permanecem em cada ponto do tempo. As equivalências são computadas para cada par de grupos em agrupamentos obtidos em diferentes tempos.

Figura 10 – Exemplo de uso de grafo bipartido para representação visual das transições.



Fonte: Oliveira e Gama (2012).

Um destaque na abordagem do MEC é o uso de grafos bipartidos para a representação visual das transições. A Figura 10 apresenta um exemplo sobre isso, de modo que os vértices do grafo representam os grupos, associados a um instante no tempo, e as arestas indicam o quão equivalente são os pares de grupos grupos. Por exemplo, os elementos do grupo C_1 em 2005 foram divididos para os grupos C_1 e C_2 em 2006, tendo o grupo C_2 uma porcentagem maior de

elementos. Optou-se por utilizar essa abordagem de grafos pela sua utilidade na modelagem de problemas de correspondência e por ser visualmente atraente. Os fundamentos do algoritmo de detecção de transição para esse tipo de representação baseia-se nessa ideia, que pode ser definida da seguinte forma:

Definição 9 (Grafos Bipartidos Pesados). Dado os agrupamentos $\zeta_i, \zeta_{i+\Delta t}$, obtidos em $t_i, t_{i+\Delta t}$, um grafo $G = (U,V,E)$ pode ser construído, onde U representa o primeiro subconjunto de vértices (grupos de t_i), e E denota um conjuntos de arestas pesadas entre qualquer par de grupos pertencentes a ζ_i e $\zeta_{i+\Delta t}$. Formalmente, o peso associado a cada aresta conectando os grupos $C_m(t_i)$ e $C_u(t_{i+\Delta t})$, com $m = 1,...,k_{t_i}$ e $m = 1,...,k_{t_{i+\Delta t}}$, sendo k_{t_i} e $k_{t_{i+\Delta t}}$ o número de grupos retornados por algum algoritmo de agrupamento nos pontos de tempo t_i e $t_{i+\Delta t}$, respectivamente, são estimados de acordo com a probabilidade condicional:

$$weight(C_m(t_i), C_u(t_{i+\Delta t})) = \frac{P(X \in C_u(t_{i+\Delta t}))}{X \in C_m(t_i)} = \frac{\sum P(x \in C_m(t_i) \cap C_u(t_{i+\Delta t}))}{\sum P(x \in C_m(t_i)} \tag{3.6}$$

Onde X é o conjunto de elementos presentes no grupo $C_m(t_i)$ e $\frac{P(X \in C_u(t_{i+\Delta t}))}{X \in C_m(t_i)}$ representa a probabilidade de X pertencer ao grupo C_u de $t_{i+\Delta t}$ sabendo que X pertence ao grupo C_m obtido em um tempo anterior t_i.

A taxonomia utilizada para a representação por enumeração é similar à utilizada pelo MONIC. Portanto, a taxonomia apresentada na Tabela 1 também é valida para o MEC, sendo a principal diferença a função usada para atribuir pesos aos objetos. Além disso, é utilizado um limiar de separação λ de forma análoga ao τ_{split} do MONIC, ou seja, é utilizado para detectar transições sobre divisão de grupos.

No método para grupos representados por compreensão, a similaridade entre grupos é descoberta ao encontrar uma região de intersecção no espaço, formado por pares de grupos em diferentes tempos. Esse procedimento permite que seja encontrado equivalências entre grupos de dois agrupamentos distintos. Para isso, é computado a distância entre os centroides dos grupos, para todos os pares de grupos pertencentes aos agrupamentos gerados em tempos distintos.

Tabela 3 – MEC: Transição externa de grupos pela representação por compreensão.

Tipo de transição	**Notação**	**Definição formal**
Nascimento de grupo	$\theta \rightarrow C_u(t_{i+\Delta_t})$	$d(C_m(t_i), C_u(t_{i+\Delta_t})) \geq r_{C_m}(t_i) + r_{C_u}(t_{i+\Delta_t})^{\forall_m}$
Desaparecimento de grupo	$C_m(t_i) \rightarrow \theta$	$d(C_m(t_i), C_u(t_{i+\Delta_t})) \geq r_{C_m}(t_i) + r_{C_u}(t_{i+\Delta_t})^{\forall_u}$
Divisão de grupo	$C_m(t_i) \overset{\subseteq}{\rightarrow} \{C_1(t_{i+\Delta_t}),...,C_r(t_{i+\Delta_t})\}$	$(d(C_m(t_i), C_u(t_{i+\Delta_t})) < r_{C_m}(t_i) + r_{C_u}(t_{i+\Delta_t})) \quad \wedge \quad \exists C_r \in \zeta_{i+\Delta_t} \{C_u\} : d(C_m(t_i), C_r(t_{i+\Delta_t}) < r_{C_m}(t_i) + r_{C_r}(t_{i+\Delta_t})$
União de grupo	$\{C_1(t_i),...,C_p(t_i)\} \overset{\subseteq}{\rightarrow} C_u(t_{i+\Delta_t})$	$(d(C_m(t_i), C_u(t_{i+\Delta_t})) < r_{C_m}(t_i) + r_{C_u}(t_{i+\Delta_t})) \quad \wedge \quad \exists C_p \in \zeta_{i+\Delta_t} \{C_m\} : d(C_p(t_i), C_u(t_{i+\Delta_t}) < r_{C_p}(t_i) + r_{C_u}(t_{i+\Delta_t})$
Sobrevivência de grupo	$C_m(t_i) \rightarrow C_u(t_{i+\Delta_t})$	$(d(C_m(t_i), C_u(t_{i+\Delta_t})) < r_{C_m}(t_i) + r_{C_u}(t_{i+\Delta_t})) \quad \wedge \quad \nexists C_p \in \zeta_{i+\Delta_t} \{C_m\} : d(C_p(t_i), C_u(t_{i+\Delta_t}) < r_{C_p}(t_i) + r_{C_u}(t_{i+\Delta_t})$

Fonte: Adaptada de Oliveira e Gama (2010).

Posteriormente, para acessar a existência de sobreposição entre grupos, a distância entre os centroides dos grupos é comparada com a soma dos raios dos grupos. Se a distância entre os centroides é igual ou maior que a soma dos raios dos grupos, então pode-se deduzir que não há intersecção entre os grupos, logo o grupo em t_i não pode ser equivalente a um grupo em $t_{i+\Delta t}$. Senão, pode-se assumir que esse par de grupos intersectam-se no espaço e conclui-se que eles são equivalentes. A taxonomia de transições para esse tipo de representação é apresentada na Tabela 3.

O monitoramento de transições internas é realizado de maneira semelhante ao MONIC, sendo feito somente para grupos sobreviventes. A Tabela 4 apresenta esses tipos de transição. Para transições de tamanho, é armazenado previamente e comparado a quantidade de elementos dos grupos sobreviventes. Para transições de densidade, é observado a distribuição dos dados em ambos os tempos.

Tabela 4 – MEC: Transição interna de grupos pela representação por compreensão..

Tipo de transição	**Notação**	**Definição formal**
Expansão	$C_m(t_i) \nearrow C_u(t_{i+\Delta t})$	$\#C_m(t_i) < \#C_u(t_{i+\Delta t})$
Encolhimento	$C_m(t_i) \searrow C_u(t_{i+\Delta t})$	$\#C_m(t_i) \geq \#C_u(t_{i+\Delta t})$
Compressão	$C_m(t_i) \overset{\bullet}{\rightarrow} C_u(t_{i+\Delta t})$	$\lambda_{C_m(t_i)} < \lambda_{C_u(t_{i+\Delta t})}$
Dispersão	$C_m(t_i) \overset{*}{\rightarrow} C_u(t_{i+\Delta t})$	$\lambda_{C_m(t_i)} > \lambda_{C_u(t_{i+\Delta t})}$

Fonte: Adaptada de Oliveira e Gama (2010).

MEC foi avaliado pelos autores com conjuntos de dados reais disponibilizados pelo Banco de Portugal, composto por objetos que representam setores de atividade, cujos atributos são indicadores econômico-financeiros. O agrupamento foi realizado para cada ano disponível, sendo mantidos os mesmos setores de atividade para todos os anos analisados. O experimento foi conduzido utilizando dois algoritmos de agrupamento distintos, sendo um hierárquico aglomerativo e o outro foi o K-means. Houve também normalização dos atributos, devido a diferenças significativas em escala e dispersão. Por fim, ambas as representações foram testadas.

Usando a representação por enumeração, houve mudança nas transições identificadas entre as duas técnicas de agrupamento. Essa diferença foi atribuída ao número diferente de grupos assumidos por cada algoritmo. Se o número de grupos fosse o mesmo, as transições detectadas convergiriam. Para a representação por compressão, houve diferenças significativas nas transições detectadas. A maior diferença foi na detecção do surgimento e desaparecimento de grupos. Essa situação foi atribuída aos fundamentos de cada método e às diferentes estratégias usadas para mapear os grupos. Exemplificando, se um grupo se mover abruptamente no espaço de um ano para outro, o método não poderá detectar uma região de interseção e, portanto, uma sobrevivência, e considerará essa situação como um desaparecimento e surgimento de um grupo. Portanto, a representação por enumeração mostrou-se mais eficaz para detectar transições.

Uma transição interessante, que foi detectada pela representação por enumeração, foi

a união de três grupos que tornaram-se um mesmo grupo no tempo posterior. A inspeção do conjunto de dados sugeriu que os setores de atividade dos três grupos sofreram uma piora no desempenho econômico e financeiro, causando a união deles. Para explicar essa transição foi descoberto que, nessa época, houve uma reestruturação no processo de submissão das informações ao banco, tornando obrigatório a divulgação das informações comerciais dos setores, por exemplo. Com isso, os dados se tornaram mais confiáveis e completos, refletindo uma imagem mais realista do país.

3.3 Stream-Dashboard

Hawwash e Nasraoui (2012) propõem um *framework* chamado *Stream-Dashboard* que é capaz de minerar, monitorar e validar os grupos em fluxos de dados. Esse *framework* formado por dois componentes principais: um algoritmo qualquer que constrói e atualiza agrupamentos de objetos do fluxo, mantendo estatísticas dos mesmos, e um módulo especializado em detectar e validar as transições chamado de *TRAcking and validating Clusters Evolution using Regression analysis* (TRACER), que utiliza regressão linear para sumarizar e assim acompanhar as mudanças nas estatísticas desses grupos enquanto o fluxo de dados é processado. Os modelos de regressão linear criados para cada grupo são atualizados incrementalmente e são recriados somente quando uma mudança significativa, como uma transição externa, é detectada.

3.3.1 Modelagem dos grupos

Qualquer algoritmo de agrupamento de objetos pode ser utilizado como o primeiro componente do *framework* desde que seja capaz de: satisfazer os requisitos para agrupar objetos do fluxo de dados, atualizar incrementalmente os agrupamentos e sumarizar as características dos grupos usando um conjunto de propriedades. Essas propriedades podem ser genéricas ou específicas do algoritmo utilizado, uma vez que elas devem ser mantidas e atualizadas incrementalmente. Para o *framework* proposto, são utilizadas três métricas: cardinalidade, escala e densidade.

Para cada grupo detectado, um conjunto de modelos de regressão linear que refletem seu comportamento é construído e mantido ao longo do vida útil do grupo, sendo esses modelos formados a partir do conjunto de métricas apresentadas anteriormente. Os coeficientes de regressão armazenados são então usados pelo TRACER para modelar ou resumir o comportamento de cada grupo ao longo do tempo, podendo até prever o comportamento futuro do grupo ou detectar qualquer desvio (*deviation*) a partir dele. Esses desvios são detectados automaticamente com base no ângulo obtido ao se comparar dois modelos de regressão consecutivos, e os momentos em que esses desvios ocorrem são chamados marcos (*milestones*).

Definição 10 (**Marco**). Dado dois modelos de regressão criados a partir de uma métrica, construídos em dois períodos de tempo consecutivos, se o ângulo θ entre eles for maior que um

limiar θ_{max}, então um marco M é encontrado para esse período de tempo.

Se nenhum marco for detectado com a comparação de dois modelos gerados consecutivamente, pode-se atualizar o modelo de regressão linear por meio de cálculos simples fazendo uma combinação dos dois modelos comparados. Se um marco for encontrado, um novo modelo de regressão linear é construído, já que isso indica a presença de um novo comportamento no agrupamento. Após esse novo modelo ser construído, o algoritmo capaz de detectar transições realiza uma comparação entre os dois últimos marcos identificados. Toda essa operação permite realizar verificações e análises contínuas sobre o comportamento desses grupos.

3.3.2 Transições de grupos

O mecanismo de detecção de transições presente no *Stream-Dashboard* é capaz de detectar transições tanto externas como internas. Como ele continuamente constrói e atualiza modelos de regressão para cada métrica de grupo utilizada, as transições internas são identificadas a partir da comparação de dois marcos. Utilizando medidas de estabilidade previamente definidas para cada métrica, uma transição interna dessa métrica pode ser detectada se a curva obtida por meio da comparação entre os modelos de regressão linear dos marcos estiver fora de um intervalo de estabilidade pré-definido, se não, a métrica é considerada estável nesse período e não ocorreu transição interna.

Tabela 5 – Stream-Dashboard: Transição externa de grupos.

Tipo de Transição	**Condição**
União	$W^{+}_{i,[M_{t-1},M_t]} \wedge \sigma^{+}_{i,[M_{t-1},M_t]} \wedge (\neg C_i, [M_{t-1}, M_t])$
Separação	$W^{-}_{i,[M_{t-1},M_t]} \wedge \sigma^{-}_{i,[M_{t-1},M_t]} \wedge C_i(neighbor[M_t])$
Nascimento	$t_{start} \in [M_{t-1}, M_t]$
Morte	$t_{end} \in [M_{t-1}, M_t]$
Sobrevivência: Envelhecimento	$W^{-}_{i,[M_{t-1},M_t]}$
Sobrevivência: Ganho na Periferia	$W^{+}_{i,[M_{t-1},M_t]} \wedge \sigma^{+}_{i,[M_{t-1},M_t]} \wedge C_i[M_{t-1}, M_t])$
Sobrevivência: Ganho no Centro	$W^{+}_{i,[M_{t-1},M_t]} \wedge \sigma^{-}_{i,[M_{t-1},M_t]}$

Fonte: Adaptada de Hawwash e Nasraoui (2012).

Para transições externas, uma taxonomia foi criada para formalizar as regras de decisão para detecção de vários tipos de transição: união, separação, nascimento, desaparecimento e sobrevivência de grupos, sendo elas inferidas com o auxílio das métricas e transições internas. Essa taxonomia é apresentada na Tabela 5. As métricas definidas previamente como cardinalidade (W) e escala (σ) serão comparadas entre os dois marcos $([M_{t-1}, M_t])$. Nessa comparação, as métricas podem ter aumentado $(W^{+}_{i,[M_{t-1},M_t]}, \sigma^{+}_{i,[M_{t-1},M_t]})$ ou diminuído $(W^{-}_{i,[M_{t-1},M_t]}, \sigma^{-}_{i,[M_{t-1},M_t]})$. Um grupo C_i pode ter sofrido uma atualização entre os marcos $(C_i, [M_{t-1}, M_t])$. Nota-se que a taxonomia abrange diferentes tipos de transição de sobrevivência, diferenciando-as pela localização dos novos objetos no grupo. Um grupo pode ser identificado como vizinho de

um grupo C_i em um marco M_t usando um limiar de distância pré-definido pelo algoritmo de agrupamento $(C_i(neighbor[M_t]))$, indicando assim uma possível separação. Se ambas as métricas aumentarem e não haver grupo vizinho identificado, provavelmente ocorreu uma união entre grupos. Detectadas as transições, o algoritmo de agrupamento deve atualizar as métricas dos grupos que foram separados ou unidos, uma vez que eles são indicativos de um marco e serão usados para comparação com o próximo.

Os autores testaram o *Stream-Dashboard* com um conjunto de bases de dados sintéticas e utilizaram dois algoritmos de agrupamento de objetos como componentes do *framework*. O *Stream-Dashboard* conseguiu detectar as transições internas e externas, com a qualidade dos resultados sendo definida pelo algoritmo de agrupamento utilizado, e provou ser capaz de validar os grupos por meio da observação da evolução das métricas.

3.4 Outros Trabalhos

Trabalhos posteriores sobre monitoramento de transições realizam aprimoramentos nos algoritmos base, tanto para o MONIC quanto para o MEC. Para o MONIC, o trabalho de Ntoutsi, Spiliopoulou e Theodoridis (2009) propõe uma extensão, nomeada de MONIC+, que torna possível detectar transições para diferentes representações de grupo, permitindo capturar transições específicas para essas representações. Em um trabalho posterior, Ntoutsi, Spiliopoulou e Theodoridis (2012) propõem uma modelagem de transições de grupos em uma estrutura de grafos, de modo similar ao funcionamento do MEC, além de dois algoritmos que resumem essa estrutura inicial em uma versão sumarizada chamada "*fingerprint*", objetivando facilitar a observação das transições pelo usuário.

Para o MEC, o trabalho de Pereira e Mendes (2016) explora uma nova variante da técnica de compreensão usada para o monitoramento de transições do algoritmo inicial, que faz uso de uma área ao redor de cada grupo, chamada de "*área de controle*", no processo de detecção. Os experimentos apontaram que o resultado de sua técnica foram superiores aos do algoritmo base. No trabalho de Namitha, Saju e Kumar (2018), é utilizado, no contexto de fluxos de dados, um mecanismo de detecção de transições para grupos sumarizados similar ao apresentado pelo MEC. O algoritmo detecta transições externas utilizando sumarizações dos grupos por meio de estatísticas e realiza comparações de distância em agrupamentos consecutivos. Além disso, essas transições são utilizadas para prever futuras transições por meio de métodos de previsão em séries temporais.

3.5 Considerações Finais

Neste capítulo foram apresentados os principais trabalhos que abordam o problema de monitoramento de transições. Os três principais apresentados, o MONIC, o MEC e o *Stream-*

Dashboard, são capazes de detectar diversos tipos de transições, com taxonomias bem definidas em cada um deles, até mesmo explorando diferentes representações de grupos. Esses métodos também funcionam em sua maior parte de modo independente do algoritmo de agrupamento. No entanto, nenhum deles foi desenvolvido para agrupamento de fluxos de dados. O Quadro 1 sumariza as características dos métodos apresentados, inclusive da técnica de monitoramento proposta neste trabalho.

Quadro 1 – Características dos métodos de monitoramento de transições.

Método	**Tipo de Dado**	**Foco do Agrupamento**	**Representação de Grupos**
MONIC	Base de Dados Convencionais	Dados Convencionais	Não sumarizado
MEC	Base de Dados Convencionais	Dados Convencionais	Não sumarizado Sumarizado
Stream-Dashboard	Fluxo de Dados	Objetos do Fluxo	Sumarizado
CETra	Fluxo de Dados	Fluxos (Fontes) de Dados	Não sumarizado

Fonte: Elaborada pelo autor.

Os métodos da literatura que utilizam uma representação não sumarizada de grupos, obtendo resultados mais precisos ao monitorar transições, são inviáveis para agrupamento de objetos de fluxos de dados, uma vez que realizam a comparação entre todos os objetos de dois agrupamentos distintos. A abordagem sumarizada para agrupamentos de objetos é interessante pelo processamento rápido e acurácia satisfatória nos resultados. Em ambas as abordagens, não é detalhado uma métrica de avaliação para as transições detectadas, utilizando principalmente de análises empíricas sobre a semântica dos resultados.

Por fim, é interessante explorar uma solução que faça o monitoramento de transições especificamente para o contexto de agrupamento de fluxos de dados e, para alcançar os melhores resultados possíveis na detecção das transições, otimizar o processamento ao conceber uma técnica que utilize abordagens não sumarizadas de grupos.

CAPÍTULO

4

MONITORAMENTO DE TRANSIÇÕES EM AGRUPAMENTOS DE FLUXOS DE DADOS

A detecção de transições para tarefas de agrupamentos de fluxos de dados deve considerar, além de suas próprias características, as diferenças para com as tarefas de agrupamento de objetos. Em agrupamento de objetos, os objetos dentro da janela atual são agrupados em grupos disjuntos e, a cada movimentação da janela, objetos antigos são desconsiderados e novos são adicionados. No agrupamento de fluxo de dados, os objetos dentro da janela são sumarizados e utilizados para caracterizar a fonte geradora do fluxo e, a cada movimentação da janela, essa caracterização pode ser ser alterada conforme a entrada e saída de objetos na janela.

Logo, no agrupamento de objetos, os objetos agrupados estão em constante mudança, uma vez que esses estão recorrentemente saindo e entrando na janela conforme o fluxo é processado. No agrupamento de fluxo de dados, as fontes geradoras de dados tendem a permanecer nos agrupamentos por um período mais longo de tempo, uma vez que a delimitação e atualização de seus elementos são referentes aos dados presentes em cada fonte geradora, e não a fonte geradora em si.

Um método de detecção de transições que utiliza a comparação de elementos não sumarizados entre dois agrupamentos pode ser mais facilmente utilizado no contexto de agrupamento de fluxo de dados (e.g. MONIC), pois os elementos que formam os grupos tendem a continuar presentes ao longo da geração de novos agrupamentos. No entanto, um método de detecção de transições para agrupamentos de fluxos de dados também deve ser concebido considerando algumas características do processamento dessa tarefa:

1. Os fluxos de dados podem estar em gradual evolução, ou seja, tendem a ter pequenas mudanças ao longo do processamento de novos dados por meio da movimentação da janela. O método de detecção de transições deve ser capaz de detectar transições nesse contexto evolutivo.

2. Os fluxos de dados são geralmente processados em tempo real. O método de detecção de transições deve ser capaz de acompanhar o mais próximo possível esse requisito de tempo de processamento.

3. Os fluxos de dados podem gerar diversas configurações de agrupamento ao longo do seu processamento. O método de detecção de transições deve ser capaz de identificar uma configuração de agrupamento que difere significativamente da configuração anterior.

O mecanismo de detecção de transições para fluxos de dados que utilize uma representação não sumarizada de grupos deve ser capaz de trabalhar com as características mencionadas anteriormente. No entanto, os métodos presentes na literatura não abordaram essas particularidades, seja por não trabalharem diretamente com processamento de fluxos de dados, seja por utilizarem abordagens que sumarizam os grupos, ou por só abordarem o agrupamento de objetos. As soluções propostas neste trabalho para essas características são apresentadas a seguir.

O restante deste Capítulo é dividido do seguinte modo: na Seção 4.1 é discutida característica de mudança gradual; na Seção 4.2 é discutido o requisito de processamento; na Seção 4.3 é discutido os tipos de transição e mudanças significativas; na Seção 4.4 é apresentada a técnica CETra que atende as características abordadas ao longo do Capítulo.

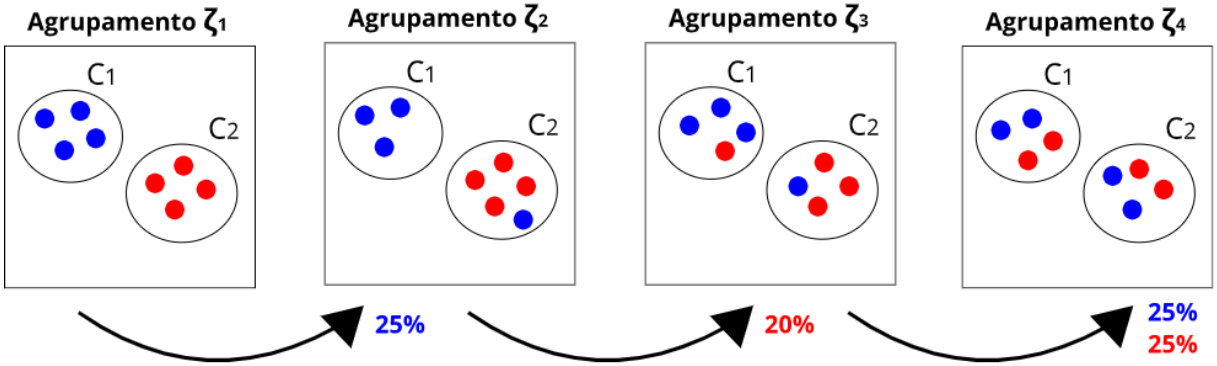
4.1 Trabalhando com a evolução gradual

Para detectar transições em grupos não sumarizados, a abordagem geral na literatura é identificar sobreposições de grupos entre dois agrupamentos ζ^{t_i}, ζ^{t_j} gerados em tempos distintos, tal que: $Track(\zeta^{t_i}, \zeta^{t_j})$ com $t_i < t_j$. Havendo um conjunto de agrupamentos $\zeta^{t_1}, \zeta^{t_2}, ..., \zeta^{t_K} = \{\zeta^{t_i}\}_{i=1}^{K}$ gerados sobre os dados em tempos consecutivos, detecta-se um conjunto de transições para cada comparação de agrupamentos distintos, definindo um conjunto de operações $\{Track(\zeta^{t_{i-1}}, \zeta^{t_i})\}_{i=2}^{K}$, sendo K o número total de agrupamentos gerados.

Em tarefas de agrupamento aplicadas a bases de dados convencionais com informação temporal associada, a detecção de transições ocorre entre agrupamentos gerados em tempos distintos considerando dados temporais da base. Por exemplo, o primeiro agrupamento é gerado para um mês e o agrupamento seguinte é gerado para o mês posterior. Sendo assim, já é esperado que mudanças significativas ocorram na configuração do agrupamento, por haver esse intervalo no tempo de evolução dos dados. Já no contexto de agrupamento de fluxo de dados, um novo agrupamento pode ser gerado a cada movimentação da janela, uma vez que novos objetos estão constantemente chegando e sendo processados, alterando a configuração dos grupos. Isso torna a evolução dos fluxos de dados muito mais sutil que em base de dados convencionais, com mudanças graduais ocorrendo de acordo com a movimentação da janela.

Nesse contexto, não é interessante que sejam feitas comparações entre agrupamentos consecutivos, por nem sempre ser possível distinguir uma mudança suficientemente significativa

Figura 11 – Exemplo de abordagem consecutiva de comparação de agrupamentos.



Fonte: Elaborada pelo autor.

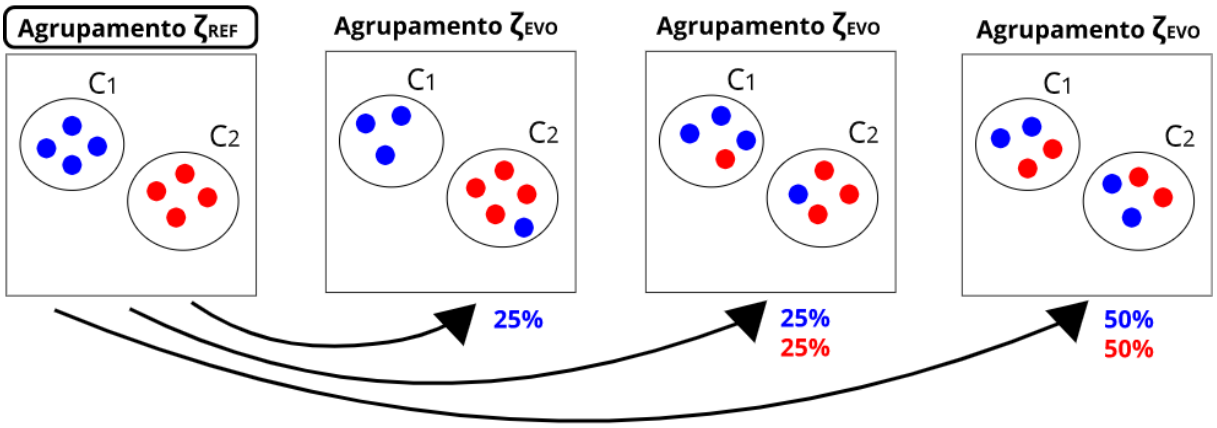
para ser caracterizada como transição em um período de tempo tão curto. Além disso, comparar agrupamentos consecutivos pode nunca gerar uma mudança significativa, uma vez que somente os dois agrupamentos mais recentes serão analisados. A Figura 11 apresenta um caso de um conjunto de dados cujos agrupamentos sofrem pequenas mudanças a cada readaptação com variações de no máximo 25% na configuração dos objetos dos grupos e, ao realizar comparações entre agrupamentos consecutivos, mudanças mais significativas quem implicam em uma transição podem nunca ser detectadas.

Portanto, é mais interessante para o contexto de fluxo de dados definir um **Agrupamento Referencial** (ζ_{ref}) que ficará armazenado em memória para ser comparado com os agrupamentos seguintes, chamados de **Agrupamentos Evolutivos** (ζ_{evo}). A Figura 12 apresenta um exemplo onde compara-se um mesmo agrupamento, definido como agrupamento referencial, com todos os agrupamentos seguintes, chamados de agrupamentos evolutivos. Desse modo, as mudanças vão se acumulando até o momento que existe uma variação de 50% na configuração dos objetos dos grupos, sendo agora possível detectar mudanças maiores e consequentemente novas transições. Assim, um mesmo agrupamento referencial permanecerá sendo usado para comparação com os agrupamentos seguintes até que uma mudança significativa ocorra em pelo menos um de seus grupos, causando a atualização de todo o agrupamento referencial. Tal mudança significativa é explicada na Seção 4.3.

Para cada agrupamento referencial ${\zeta_{ref}}^{t_i}$, há um conjunto de agrupamentos evolutivos $\{{\zeta_{evo}}^{t_j}\}_{j=i+1}^{E}$, sendo t_E o tempo do último agrupamento evolutivo comparado com ${\zeta_{ref}}^{t_i}$ em razão de uma mudança significativa ter sido detectada. Assim, o agrupamento evolutivo ${\zeta_{evo}}^{t_E}$ passa a ser o próximo agrupamento referencial ${\zeta_{ref}}^{t_i}$ com $i = E$, e será comparado com um conjunto de agrupamentos evolutivos posteriores a t_E. Portanto, tem-se um conjunto de operações entre o agrupamento referencial e seus demais agrupamentos evolutivos $\{Track({\zeta_{ref}}^{t_i}, {\zeta_{evo}}^{t_j})\}_{j=i+1}^{E}$ para todos os agrupamentos referenciais encontrados.

Com essa solução, pode-se acompanhar a evolução do agrupamento de maneira mais

Figura 12 – Exemplo de abordagem que utiliza a lógica de agrupamento referencial e evolutivo.



Fonte: Elaborada pelo autor.

apropriada para o contexto de fluxo de dados. Um determinado agrupamento referencial passa a ser observado e comparado até o momento que um agrupamento posterior divirja de maneira significativa, indicando que o mesmo está obsoleto e deve ser atualizado.

Por fim, como os agrupamentos são gerados a cada mudança na janela e a matriz deve ser construída para cada novo agrupamento evolutivo, é interessante que o tempo de processamento para a construção dessa estrutura seja otimizada já que em fluxos de dados espera-se que muitos janelamentos sejam feitos para processar continuamente seus objetos, o que é discutido a seguir.

4.2 Atendendo os requisitos de processamento

Para identificar sobreposições de grupos não sumarizados entre dois agrupamentos, os métodos da literatura procuram cada elemento presente no agrupamento de tempo t_i nos grupos do agrupamento posterior. Com isso, é possível construir a matriz de sobreposições que é utilizada para detectar se alguma transição externa ou interna ocorreu entre os agrupamentos observados.

A matriz de sobreposições é a principal estrutura lógica utilizada para entender quais mudanças ocorreram ao se comparar agrupamentos distintos. Para cada par de agrupamentos ζ^{t_i}, ζ^{t_j} com $t_i < t_j$, cada qual com um conjunto de grupos disjuntos $\zeta^{t_i} = \{C_1^{t_i}, C_2^{t_i}, ..., C_{p_i}^{t_i}\}$ e $\zeta^{t_j} = \{C_1^{t_j}, C_2^{t_j}, ..., C_{p_j}^{t_j}\}$, a matriz é construída calculando-se a intersecção dos elementos de cada grupo referencial $C_x^{t_i} \in \zeta^{t_i}$ com cada grupo evolutivo $C_y^{t_j} \in \zeta^{t_j}$ dividida pela intersecção total do grupo $C_x^{t_i}$, obtendo assim a sobreposição entre grupos. Junto a isso, a cada elemento do agrupamento pode ser atribuído um peso que impactará na sua relevância no contexto do problema explorado. A Equação 4.1 apresenta o cálculo da sobreposição entre dois grupos com

uma função de peso para seus elementos.

$$overlap(C_x^{t_i}, C_y^{t_j}) = \frac{\sum_{a \in C_x^{t_i} \cap C_y^{t_j}} weight(a)}{\sum_{b \in C_x^{t_i}} weight(b)} \tag{4.1}$$

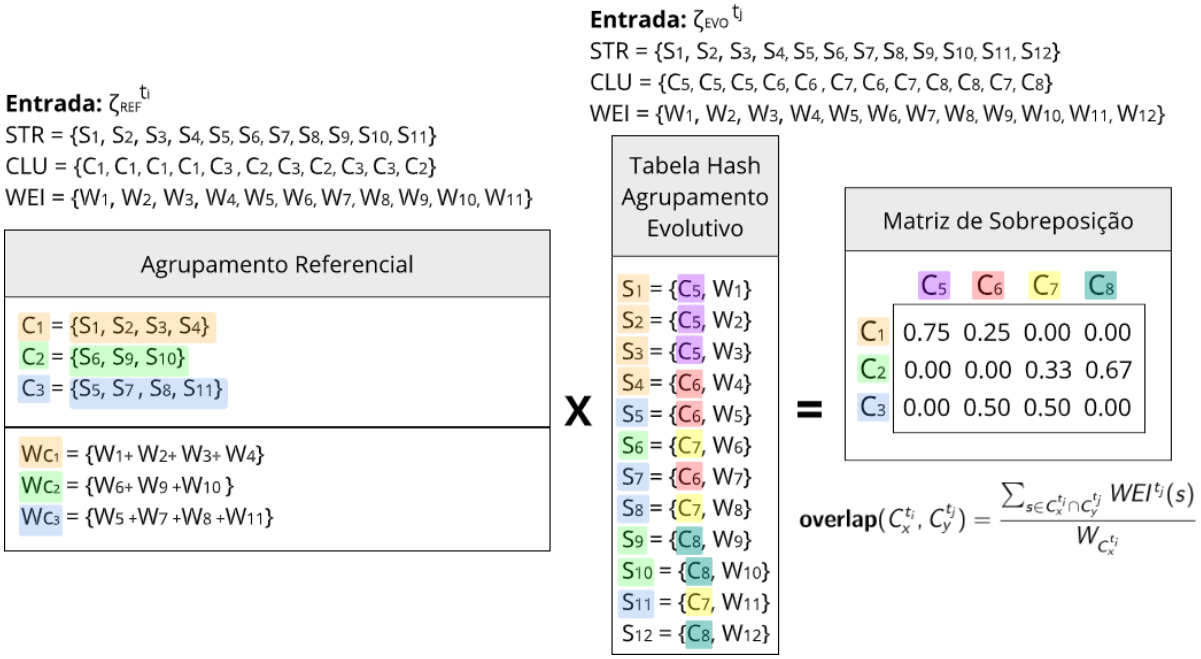
Cada elemento agrupado pode ser valorado igualmente (e.g. todos valem 1.0), tal que todos tenham um mesmo impacto no cálculo das sobreposições. Esquemas mais elaborados podem ser concebidos baseados no contexto em que se trabalha. Por exemplo, o MONIC atribui um peso menor para dados mais antigos na base de dados, impactando em sua relevância conforme novos agrupamentos são gerados em tempos posteriores. Assim, existem várias possibilidades de se trabalhar com modelos de pesos atrelados aos elementos do agrupamento.

Todavia, independentemente de se utilizar um esquema mais simples ou mais elaborado de atribuição de pesos, essa operação de construção da matriz, nos trabalhos da literatura, tem complexidade de $O(N^2)$ por realizar a comparação entre todos os elementos de um agrupamento e todos os elementos de outro agrupamento. Para fluxos de dados, esse custo torna a operação inviável para as tarefas que realizam a detecção de transições para cada atualização da janela. Como os métodos de agrupamento de fluxos de dados visam realizar o processamento em tempo real, ter uma operação $O(N^2)$ pode gerar gargalos entre movimentações de janelas e, consequentemente, a detecção de transições pode não refletir o estado mais atual do processo de agrupamento. Além disso, espera-se uma quantidade muito maior de agrupamentos em fluxos de dados, que devem ser analisados já que a cada movimentação da janela, independente da sua granularidade, a configuração dos grupos pode mudar.

Em abordagens de detecção de transições que utilizam versões sumarizadas de grupos, essa atualização a cada movimentação da janela é feita de modo mais rápido. Porém, a exatidão das transições detectadas é inferior à obtida utilizando a matriz de sobreposições. Logo, visando uma detecção de transições mais correta e que atenda aos requisitos de processamento de fluxos de dados, é preciso otimizar o processo de criação da matriz de sobreposições. Portanto, precisa-se de um modo mais eficiente de detectar a intersecção entre conjuntos. Isso pode ser obtido por meio de uma lógica de buscas baseada em tabela *hash*, para pesquisar os elementos de ζ_{ref} em uma tabela que represente o ζ_{evo}. A Figura 13 ilustra essa solução.

Três listas são usadas para representar um agrupamento: uma lista com os identificadores das fontes geradoras de dados, uma lista com os rótulos dos grupos referentes a cada fonte de dados e uma lista com os pesos referentes a cada fonte de dados (STR, CLU e WEI respectivamente na Figura 13). Primeiramente, o ζ_{ref} é armazenado, por ser utilizado para comparação com os demais agrupamentos evolutivos até o momento que uma mudança significativa seja detectada. A somatória dos pesos de cada grupo de ζ_{ref} também precisa ser armazenada, uma vez que essa é uma estatística utilizada para calcular as sobreposições entre grupos. O primeiro agrupamento gerado por um método de agrupamento de fluxos de dados é armazenado como o primeiro ${\zeta_{ref}}^{t_1}$. A partir desse momento, todos os agrupamentos gerados posteriormente com

Figura 13 – Exemplo de matriz de sobreposições entre um ζ_{ref} e um ζ_{evo} (considerando todos os pesos W como 1.0).



Fonte: Elaborada pelo autor.

a movimentação da janela são considerados agrupamentos evolutivos $\{\zeta_{evo}{}^{t_j}\}_{j=2}^{E}$ até que uma mudança significativa seja detectada no E-ézimo tempo. Para cada $\zeta_{evo}{}^{t_j}$, seus elementos (fontes de dados) são processados e assim é construída uma tabela *hash* com os N^{t_j} elementos do agrupamento, tendo como chave o identificador da fonte de dados e como valores uma tupla com o identificador do grupo ao qual o elemento pertence e o peso atribuído a essa fonte geradora. Nessa tabela, pode-se executar uma busca dos elementos do $\zeta_{ref}^{t_1}$ e detectar a qual grupo cada elemento pertence no $\zeta_{evo}{}^{t_j}$. Como a busca por chave em tabelas hash tem complexidade $O(1)$, a matriz de sobreposições é construída com complexidade $O(N^{t_i})$, onde N^{t_i} é a quantidade de elementos presentes no $\zeta_{ref}^{t_i}$ que são usados para a busca em $\zeta_{evo}{}^{t_j}$.

Portanto, a cada movimentação de janela em que o agrupamento seja atualizado, detectar as transições dos grupos requer uma leitura no agrupamento $\zeta_{evo}{}^{t_j}$, constituído por N^{t_j} elementos, para gerar uma tabela hash que é utilizada para buscar os N^{t_i} elementos do agrupamento ζ_{ref}. Logo, a cada movimentação da janela é executado um processamento $O(N^{t_i}+N^{t_j}) = O(N)$ para a criação da matriz de sobreposições de $\zeta_{ref}^{t_j}$ com $\zeta_{evo}{}^{t_j}$. Se uma mudança relevante ocorrer, uma nova leitura é feita em $\zeta_{evo}{}^{t_j}$ para atualizar o agrupamento referencial, ou seja, é realizada mais uma operação $O(N^{t_j})$, que não altera a complexidade $O(N)$ da técnica.

Como para cada atualização do agrupamento evolutivo deve-se verificar a qual grupo pertence cada elemento, a tabela *hash* deve ser totalmente reconstruída a cada movimentação da janela.

4.3 Detectando mudanças relevantes

Como apresentado anteriormente, um agrupamento referencial ζ_{ref} é comparado com E agrupamentos evolutivos ζ_{evo}. Uma mudança significativa, que justifica a atualização do agrupamento referencial, pode ser definida pela detecção de transições externas e internas entre um agrupamento referencial e um agrupamento evolutivo.

Transições externas indicam uma mudança na configuração do agrupamento por meio de separações, uniões, desaparecimentos e surgimentos de grupos. Essas transições são verificadas por meio da matriz de sobreposições que é construída a cada atualização do agrupamento evolutivo, sendo que primeiramente busca-se uma equivalência entre os grupos. Uma equivalência é definida pela maior sobreposição entre um grupo $C_x^{t_i} \in \zeta^{t_i}$ e um grupo $C_y^{t_j} \in \zeta^{t_j}$ e que seja maior que um limiar $\tau = [0.5, 1.0]$, para enfatizar que um grupo evolutivo é equivalente somente se contém no mínimo metade do grupo referencial. A Equação 4.2 apresenta a função de equivalência.

$$match(C_x^{t_i}, \zeta^{t_j}) = max_{\{C_{p_j}^{t_j}\}_{k=0}^{p_j}} overlap(C_x^{t_i}, C_k^{t_j}) \geq \tau \tag{4.2}$$

Se um grupo referencial $C_x^{t_i}$ e um grupo evolutivo $C_y^{t_j}$ forem equivalentes, uma sobrevivência é detectada. Se mais de um grupo $C_x^{t_i}$ for equivalente a um mesmo grupo $C_y^{t_j}$, detecta-se uma união. Se nenhuma equivalência existe, então uma separação pode ter ocorrido, ou seja, o conteúdo de $C_x^{t_i}$ está em mais de um grupo em ζ^{t_j}. Então, para esse caso, as sobreposições devem ser não menos que um novo limiar τ_{split}, com $\tau_{split} < \tau$, para prevenir casos degenerativos. Em suma, todos esses grupos juntos devem formar uma equivalência para $C_x^{t_i}$. Se nenhum desses casos ocorrer, então $C_x^{t_i}$ desapareceu. Novos grupos são detectados após traçar todas as transições externas para cada grupo de ζ^{t_j}, pois assim sabe-se quais são os grupos que não são o resultado de mudanças externas, identificando um nascimento.

A taxonomia de transições externas definida para a técnica CETra é bastante similar às presentes na literatura e está apresentada na Tabela 6. No momento em que pelo menos uma detecção externa ocorrer, que não seja uma sobrevivência, entre um ζ_{ref} e qualquer ζ_{evo}, uma mudança significativa é identificada e um novo ζ_{ref} é armazenado.

Transições internas são mudanças das estatísticas de grupos que são identificados como sobreviventes ao se comparar dois agrupamentos distintos. Se a diferença de qualquer estatística de um grupo sobrevivente for significativa, uma transição interna para aquela estatística é identificada. Nos trabalhos da literatura, a transição interna é retratada como uma informação adicional sobre grupos sobreviventes, não tendo tanta relevância quanto a identificação de transições externas.

Para a tarefa de detecção de transições em agrupamento de fluxos de dados, onde pretende-se utilizar um agrupamento referencial, espera-se que pequenas mudanças ocorram na

Tabela 6 – Taxonomia de transições externas definida para a CETra.

Tipo de Transição	**Notação**	**Condição**
Sobrevivência	$C_x^{t_i} \rightarrow C_y^{t_j}$	$C_y^{t_j} = match_\tau(C_x^{t_i}, {\zeta_{evo}}^{t_j}) \quad \wedge$ $\nexists C_z^{t_i} \in {\zeta_{ref}}^{t_i} \setminus C_x^{t_i} : C_y^{t_j} = match_\tau(C_z^{t_i}, {\zeta_{evo}}^{t_j})$
Separação	$C_x^{t_i} \overset{\subseteq}{\rightarrow} \{C_1^{t_j}, ..., C_{p_y}^{t_j}\}$	$(\forall_y 1...p_j : overlap(C_x^{t_i}, C_y^{t_j}) \geq \tau_{split}) \quad \wedge$ $overlap(C_x^{t_i}, \bigcup_{y=1}^{p_j} C_y^{t_j}) \geq \tau \quad \wedge$ $(\nexists C_z^{t_j} \in {\zeta_{evo}}^{t_j} \setminus \{C_1^{t_j}, ..., C_{p_j}^{t_j}\} : overlap(C_x^{t_i}, C_z^{t_j}) \geq \tau_{split})$
União	$\{C_1^{t_i}, ..., C_{p_x}^{t_i}\} \overset{\subseteq}{\rightarrow} C_y^{t_j}$	$C_y^{t_j} = match_\tau(C_x^{t_i}, {\zeta_{evo}}^{t_j}) \quad \wedge$ $\exists C_z^{t_i} \in {\zeta_{ref}}^{t_i} \setminus C_x^{t_i} : C_y^{t_j} = match_\tau(C_z^{t_i}, {\zeta_{evo}}^{t_j})$
Morte	$C_x^{t_i} \rightarrow \varnothing$	Nenhum dos casos acima servem para $C_x^{t_i}$
Nascimento	$\varnothing \rightarrow C_y^{t_j}$	Nenhum dos casos acima servem para $C_y^{t_j}$

Fonte: Elaborada pelo autor.

configuração de grupos à medida que os dados são processados. Consequentemente, ao monitorar as transições a cada movimentação da janela, ocorrem muitas sobrevivências de grupos e muitas verificações de estatísticas internas são realizadas. Em geral, não é interessante esperar somente por uma transição externa para que seja definido um novo agrupamento referencial, pois se uma estatística interna muda significativamente, como média e desvio padrão, isso já é um indicativo de que o estado atual do agrupamento está suficientemente diferente para que o agrupamento referencial seja atualizado. Com isso, as transições internas passam a ter uma relevância maior no processo de detecção de transições em agrupamento de fluxos de dados.

Tabela 7 – Taxonomia de transições internas definida para a CETra.

Tipo de Transição	**Notação**	**Indicador**
Tamanho	$C_x^{t_i} \overset{Tam}{\rightarrow} C_y^{t_j}$	$\lvert Tam(C_y^{t_j})/Tam(C_x^{t_i}) - 1\rvert > \delta(Tam)$
Média	$C_x^{t_i} \overset{Media}{\rightarrow} C_y^{t_j}$	$\lvert Media(C_y^{t_j})/Media(C_x^{t_i}) - 1\rvert > \delta(Media)$
Desvio Padrão	$C_x^{t_i} \overset{DesPad}{\rightarrow} C_y^{t_j}$	$\lvert DesPad(C_y^{t_j})/DesPad(C_x^{t_i}) - 1\rvert > \delta(DesPad)$
...	...	...

Fonte: Elaborada pelo autor.

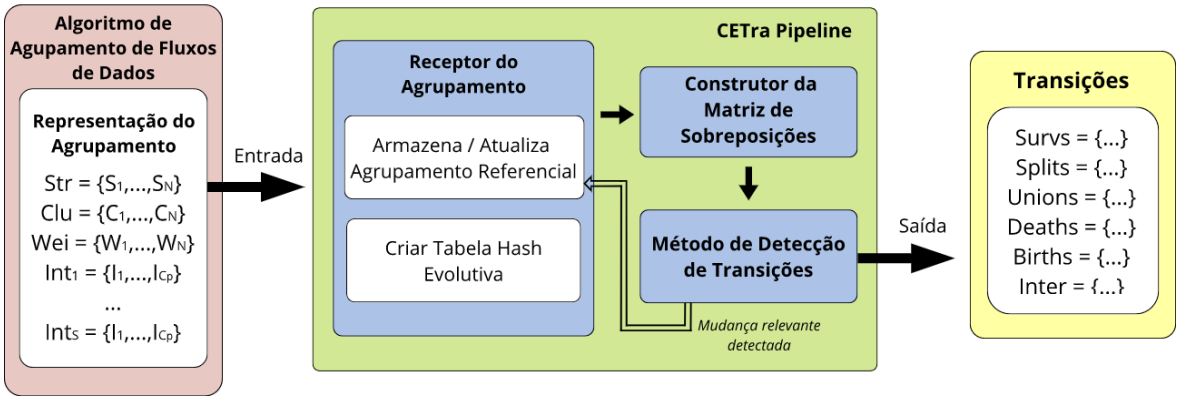
A taxonomia de transições internas depende das estatísticas de grupos que se deseja monitorar e que elas possam ser disponibilizadas pelo algoritmo de agrupamento para o mecanismo de detecção de transições. Para cada tarefa de agrupamento de fluxos de dados, é possível ter um conjunto $\{Int_i\}_{i=0}^{s}$ de estatísticas internas para serem monitoradas, definidas a partir das necessidades do domínio de aplicação em questão. Uma mudança relevante em uma estatística interna Int_i desse conjunto é definido por um limiar de entrada próprio $\delta(Int_i) = [0,1]$, que é comparado com a razão da estatística entre o grupo sobrevivente no agrupamento referencial com o seu respectivo par no agrupamento evolutivo. Um exemplo de taxonomia é apresentado na Tabela 7.

Utilizando a detecção de transições externas e internas para identificar mudanças relevantes no agrupamento e atualizar o agrupamento referencial, junto com as demais soluções apresentadas anteriormente, foi desenvolvida a técnica CETra para monitoramento de transições em agrupamento de fluxos de dados, apresentada a seguir.

4.4 *Cluster Evolution Tracker* - CETra

Cluster Evolution Tracker (CETra) é uma técnica de monitoramento e detecção de transições criada a partir do entendimento das características das tarefas de agrupamento de fluxos de dados. Recebe como entrada um conjunto de fluxos de dados agrupados e tem como saída um conjunto de transições externas e internas detectadas. A CETra é constituída por: um algoritmo que armazena um agrupamento referencial e que constrói uma tabela hash para um agrupamento evolutivo; um algoritmo para construção da matriz de sobreposições; um mecanismo de detecção de transições que retorna o conjunto de transições identificadas. O *pipeline* da CETra é apresentado na Figura 14.

Figura 14 – Arcabouço CETra



Fonte: Elaborada pelo autor.

A entrada para a CETra consiste no agrupamento gerado por um algoritmo de agrupamento de fluxos de dados. Essa entrada deve ser padronizada em no mínimo três vetores de tamanho N: o primeiro contém os identificadores de cada fonte geradora de dados S_i; o segundo contém, para cada fonte de dados S_i, o rótulo do grupo C_i ao que S_i pertence; o terceiro contém os pesos W_i atribuído a cada fonte de dados S_i. As estatísticas internas dos grupos são inseridas no seguinte formato: para cada estatística interna Int_i disponível, tem-se um vetor contento o valor I_{C_j} referente a cada um dos C_p grupos presentes no agrupamento. As estatísticas internas são entradas opcionais e a CETra sempre calcula uma estatística interna padrão Int_{tam}, referente ao tamanho dos grupos, possibilitando o monitoramento de pelo menos um tipo de transição interna para qualquer tarefa de agrupamento.

O agrupamento de entrada é tratado como apresentado no Algoritmo 1, que transforma

Algoritmo 1 – Receptor do Agrupamento

Entrada: Lista de fontes de dados S, Lista de rótulos de grupos C, Lista de pesos W, Conjunto de listas de estatísticas internas $\{Int_1 ... Int_s\}$

Saída: Nenhuma, armazena/atualiza um agrupamento referencial ζ_{ref}, o peso total dos grupos referenciais $W_{c_{p_i}}$ e as estatísticas internas referenciais Sta_{ref}, ou cria uma tabela hash do agrupamento evolutivo ζ_{evo} e as estatísticas internas evolutivas Sta_{evo}

Armazena ou atualiza o agrupamento referencial

Condição: Se o $\zeta_{ref} = \varnothing$ ou se o ζ_{ref} precisa ser atualizado.

1: **para** $i \leftarrow 1$ até N **faça**
2: $\quad \zeta_{ref}[C_i].inserir(S_i)$ ▷ Atrelando fontes de dados aos seus grupos
3: $\quad W_{C_i} = W_{C_i} + W_i$ ▷ Somando os pesos das fontes de dados e atrelando ao grupo
4: **fim para**
5: **para todo** $C_x \in \zeta_{ref}$ **faça** ▷ Para cada grupo no agrup. referencial
6: $\quad Sta_{ref}[C_x].Int_{tam} = |\zeta_{ref}[C_x]|$ ▷ Armazenando tamanho do grupo
7: $\quad$ **para todo** $I \in \{Int_1 ... Int_s\}$ **faça**
8: $\quad\quad Sta_{ref}[C_x].inserir(I_{C_x})$ ▷ Anexa a respectiva estatística ao respectivo grupo
9: $\quad$ **fim para**
10: **fim para**

Cria tabela hash do agrupamento evolutivo

Condição: Se o $\zeta_{ref} \neq \varnothing$.

11: **para** $i \leftarrow 1$ até N **faça**
12: $\quad \zeta_{evo}[S_i] = (C_i, W_i)$ ▷ Insere na tabela hash
13: $\quad Sta_{evo}[C_i].Int_{tam} + 1$ ▷ Atualiza a estatística interna de tamanho do grupo
14: **fim para**
15: **para todo** $C_y \in \zeta_{evo}$ **faça**
16: $\quad$ **para todo** $I \in \{Int_1 ... Int_s\}$ **faça**
17: $\quad\quad Sta_{evo}[C_y].inserir(I_{C_y})$ ▷ Anexa a respectiva estatística ao respectivo grupo
18: $\quad$ **fim para**
19: **fim para**

esse agrupamento, seja armazenando-o como um agrupamento referencial ou criando uma tabela *hash* para representar um agrupamento evolutivo. O primeiro agrupamento gerado pelo algoritmo de agrupamento de fluxos de dados é definido como o primeiro agrupamento referencial. Seus grupos, com as respectivas fontes de dados que os compõem, são armazenados juntamente com a somatória de seus pesos (Linhas 1 a 3). Posteriormente, as estatísticas internas são atreladas aos seus respectivos grupos $C_i = \{I_{C_i}^{Int_1}, ..., I_{C_i}^{Int_S}\}$ (Linha 7 a 8). Essas variáveis referentes ao agrupamento referencial persistem em memória principal até que ocorra uma mudança significativa, iniciando uma atualização do conteúdo dessas estruturas. A partir do agrupamento seguinte, é construída uma tabela *hash* onde o identificador S_i é utilizado como chave e o par de grupo e peso (C_i, W_i) (Linha 16) são utilizados como os valores indexados. As estatísticas internas são armazenadas do mesmo modo que para o agrupamento referencial (Linhas 19 a 21), inclusive a estatística interna Int_{tam} (Linha 17). Obtido tanto o agrupamento referencial quanto uma tabela *hash* do agrupamento evolutivo, ambos são encaminhados para a próxima etapa do

processo que consiste na construção da matriz de sobreposições.

Algoritmo 2 – Construtor da Matriz de Sobreposições

Entrada: Agrupamento referencial ζ_{ref}, Peso total dos grupos referenciais $W_{c_{p_i}}$, Tabela *hash* do agrupamento evolutivo $\zeta_{evo}(cluster, weight)$

Saída: Matriz de sobreposições M, Lista de fontes de dados desparecidas $FAIL$, Lista de fontes de dados novas NEW

1: $NEW = \zeta_{evo}$ ▷ Copiando a tabela *hash*
2: **para todo** $C_x \in \zeta_{ref}$ **faça**
3: **para todo** $elem \in C_x$ **faça**
4: **se** $\zeta_{evo}.find(elem)$ **is** $True$ **então**
5: $C_y \leftarrow \zeta_{evo}[elem].cluster$
6: $W_y \leftarrow \zeta_{evo}[elem].weight$
7: $M[C_x][C_y] \leftarrow M[C_x][C_y] + W_y$ ▷ Somando pesos para cálculo da intersecção
8: $NEW.erase(elem)$ ▷ Removendo elementos da tabela *hash* auxiliar
9: **senão**
10: $FAIL.insert(elem)$ ▷ Inserindo elementos não encontrados na lista de fontes de dados desaparecidas
11: **fim se**
12: **fim para**
13: **fim para**
14: **para** C_x **in** M **faça**
15: **para** C_y **in** M **faça**
16: $M[C_x][C_y] \leftarrow \frac{M[C_x][C_y]}{W_{C_x}}$ ▷ Sobreposição entre a intersecção e o peso total do grupo referencial
17: **fim para**
18: **fim para**

A construção da matriz de sobreposições, apresentada no Algoritmo 2, é realizada sempre entre o agrupamento referencial armazenado e a tabela *hash* criada para o agrupamento evolutivo atual. Cada fonte de dados S_i presente nos grupos do agrupamento referencial é utilizada como chave de busca na tabela *hash* do agrupamento evolutivo. Se a chave for encontrada, há uma correspondência de uma fonte de dados S_i presente em grupo referencial C_x que é encontrada em um grupo evolutivo C_y. Com isso, o peso W_i de S_i é somado e armazenado na célula da matriz $M[C_x][C_y]$, construindo as intersecções entre os grupos dos agrupamentos (Linhas 2 a 7). Posteriormente, essas intersecções são divididas pelos respectivos pesos totais do grupos referenciais que foram armazenados junto ao agrupamento referencial (Linhas 14 a 16). Após esse processo, obtém-se a matriz contendo as sobreposições entre os grupos dos agrupamentos referencial ζ_{ref} e evolutivo ζ_{evo}. Ademais, sendo esse o momento em que cada fonte de dados é buscada e localizada durante o monitoramento, procedimentos adicionais a respeito dessas fontes podem ser realizadas. Por exemplo, no caso do módulo CETra, fontes de dados do agrupamento referencial que não forem encontrados na busca (Linha 10) e fontes de dados do agrupamento evolutivo não presentes no agrupamento referencial (Linha 8) são contabilizadas, buscando assim obter a informação complementar de possíveis fontes de dados que falharam e fontes de dados

que surgiram posteriormente, respectivamente.

Algoritmo 3 – Mecanismo de detecção de transições externas

Entrada: Matriz de Sobreposições M
Saída: Conjunto de transições externas T_{ext}

1: $uni\_cands = \{\}$
2: **para** C_x **em** M **faça**
3: $sob\_cand \leftarrow NULL$
4: $sep\_cands = \{\}$ ▷ Candidatos de separação para C_x
5: **para** C_y **em** M **faça**
6: $mcel = M[C_x][C_y]$ ▷ Recupera a célula específica da matriz
7: **se** $mcel \geq \tau$ **então**
8: **se** $sob\_cand = NULL$ **então**
9: $sob\_cand \leftarrow C_y$
10: **senão se** $mcel > M[C_x][sob\_cand]$ **então**
11: $sob\_cand \leftarrow C_y$
12: **fim se**
13: **senão se** $mcel \geq \tau_{split}$ **então**
14: $sep\_cands.inserir(C_y)$ ▷ Atribui C_y nas possíveis separações de C_x
15: **fim se**
16: **fim para**
17: **se** $sob\_cand = NULL$ **E** $sep\_cands = \{\}$ **então**
18: $T_{ext}.Deaths.inserir(C_x)$
19: **senão se** $sep\_cands \neq \{\}$ **então**
20: $sep\_sob \leftarrow somar\_sob(M, sep\_cands)$ ▷ Somar sobrep. dos $\{C_y\}$
21: **se** $sep\_sob \geq \tau$ **então**
22: $T_{ext}.Splits.inserir(sep\_cands)$
23: **senão se** $sob\_cand \neq NULL$ **então**
24: $uni\_cands.inserir(C_x \rightarrow C_y)$ ▷ Atribui grupos ref. a um grupo evo.
25: **senão**
26: $T_{ext}.Deaths.inserir(C_x)$
27: **fim se**
28: **senão**
29: $uni\_cands.inserir(C_x \rightarrow C_y)$
30: **fim se**
31: **fim para**
32: **para** $\{C_x\} \rightarrow C_y$ **em** uni_cands **faça**
33: **se** $|\{C_x\}| \rightarrow C_y > 1$ **então** ▷ Se mais de um C_x está atrelado a um C_y, há união
34: $T_{ext}.Unions.inserir(\{C_x\} \rightarrow C_y)$
35: **senão** ▷ Se não, há sobrevivência
36: $T_{ext}.Survs.inserir(C_x \rightarrow C_y)$
37: **fim se**
38: **fim para**
39: $T_{ext}.Births.inserir(C_y \notin \{Survs, Splits, Unions\})$ ▷ C_y não presentes nas outras transições

A matriz de sobreposições construída é utilizada para detecção de transições. Primeiramente, é realizado o processo de detecção de transições externas, conforme Algoritmo 3. A matriz de sobreposições é percorrida e, para cada grupo do agrupamento referencial, é verificado

se algum dos grupos do agrupamento evolutivo tem uma sobreposição maior que um limiar τ (Linha 7), caracterizando assim uma sobreposição. Se não for superior a esse limiar, é verificado se existem sobreposições maiores que um limiar τ_{split} (Linha 13), para que posteriormente calcular a somatória dessas sobreposições e, se for superior ao limiar τ (Linha 21), identificar uma transição externa de separação. Posteriormente, se mais de um grupo referencial atender ao limiar τ para um mesmo grupo evolutivo, uma união é identificada (Linha 33). Se nenhum grupo evolutivo atender ao limiar τ para um grupo referencial, seja por meio de sobrevivência ou por meio de separação, é identificado uma morte (Linhas 18 e 26). Se um grupo evolutivo não está presente em nenhuma transição dos grupos referenciais, ele é considerado um nascimento (Linha 39). Posteriormente, para as mortes e nascimentos de grupos, é verificado se isso ocorreu em razão de fontes de dados desaparecidas e por surgimento de novas fontes de dados, respectivamente. Para tanto, são utilizados limiares definidos previamente τ_{MISS} e τ_{NEW} e as estruturas construídas para formação da matriz de sobreposições.

Algoritmo 4 – Mecanismo de detecção de transições internas

Entrada: Transições de sobrevivência $T_{ext}.Survs$, Estatísticas Internas do agrupamento referencial Sta_{ref} e do agrupamento evolutivo Sta_{evo}

Saída: Conjunto de transições internas T_{int}

1: **para** $C_x \rightarrow C_y$ **em** $T_{ext}.Survs$ **faça**
2: **para** I **em** $Int_1, ..., Int_s$ **faça**
3: $M_{int}[C_x][I] \leftarrow \frac{Sta_{evo}[C_y].I}{Sta_{ref}[C_x].I}$
4: **fim para**
5: **fim para**
6: **para** C_x **em** M_{int} **faça**
7: **para** I **em** M_{int} **faça**
8: **se** $abs(1 - M_{int}[C_x][I]) > \delta(I)$ **então**
9: $T_{int}.inserir(C_x, I)$
10: **fim se**
11: **fim para**
12: **fim para**

Com as transições externas, é possível identificar quais grupos sobreviveram e assim realizar a análise de transições internas, apresentada no Algoritmo 4. Uma matriz M_{int} é construída para relacionar as estatísticas internas dos grupos referenciais sobreviventes com as estatísticas internas de seu respectivos grupos evolutivos (Linhas 1 a 3). Em seguida, a matriz é percorrida e, se alguma das estatísticas tiver um crescimento ou diminuição superior ao seu respectivo limiar $\delta(Int)$, uma transição interna é identificada (Linhas 6 a 9). Vale ressaltar que essa matriz é construída para que as informações relacionadas às estatísticas internas possam ser persistidas e utilizadas posteriormente tanto pela CETra como por alguma necessidade do usuário. Um exemplo disso é a possibilidade de definir uma quantidade de vezes miníma que a mudança na estatística interna deve ocorrer à medida que sobrevivências se tornam comuns para um determinado domínio de dados. Isso evita que uma mudança súbita, inesperada e passageira seja identificada como uma mudança interna relevante. Por exemplo, se para definir uma transição

interna é preciso que a mudança se perpetue por dois janelamentos (i.e. agrupamentos evolutivos), uma variação do Algoritmo 4 pode ser executada para armazenar as matrizes de mudanças internas M_{int} correspondentes aos dois agrupamentos evolutivos mais recentes e verificar se em ambas matrizes se para cada estatística $\delta(Int_i)$ o respectivo limiar $\delta(Int_i)$ é atendido. Se em ambas matrizes o limiar for alcançado, então uma transição interna é detectada considerando essa tolerância.

Detectada uma transição externa, sem ser sobrevivência, ou uma transição interna, o agrupamento referencial é atualizado com o agrupamento evolutivo que gerou a tabela *hash*, tornando-se o mais novo agrupamento referencial. Finalmente, as transições obtidas nessa comparação de agrupamentos são retornadas pela CETra, e podem ser persistidas, visualizadas e utilizadas conforme a necessidade do usuário. Todas as informações referentes ao agrupamento evolutivo são descartadas ao longo da execução e assim a CETra está pronta para processar o próximo agrupamento.

4.5 Considerações Finais

Foram apresentados neste Capítulo aspectos essenciais para o entendimento e realização do monitoramento de transições para tarefas de agrupamento de fluxos de dados, resultando na concepção e implementação da técnica de monitoramento e detecção de transições CETra. Tal qual outros métodos da literatura que trabalham com agrupamentos não sumarizados, a CETra realiza a comparação de cada elemento em grupos de dois agrupamentos distintos para detectar transições ao longo da evolução dos dados.

No entanto, CETra é capaz de construir a principal estrutura utilizada para o processo de detectar transições, a matriz de sobreposições, com complexidade de tempo computacional inferior aos apresentados na literatura. A técnica também considera a característica de evolução gradual dos fluxos de dados para detectar transições que não podem ser obtidas com somente comparações de agrupamentos mais recentes. Por isso, uma taxonomia de transições também foi definida para representar as condições necessárias para identificar cada tipo de transição para essa lógica de monitoramento.

CAPÍTULO

5

EXPERIMENTOS

Os experimentos apresentados neste Capítulo foram elaborados e executados para comprovar a eficácia das soluções apresentadas no Capitulo 4 e avaliar a eficiência da técnica CETra tanto sobre o tempo de processamento quanto a sua capacidade de detectar transições, tanto em dados sintéticos quanto em dados reais.

Do melhor conhecimento do autor desta dissertação, não há na literatura da área nenhum trabalho que trate monitoramento de transições para agrupamento de fluxos de dados, com abordagem não sumarizada de grupos. Portanto, para análise comparativa foi aplicado o método MONIC (Seção 3.1), por ser voltado para grupos não sumarizados. Embora o MONIC tenha sido originalmente concebido para agrupamento de objetos em bases de dados convencionais, sua definição de grupo baseada em teoria de conjuntos permite utilizá-lo com algoritmos de agrupamento de fluxos de dados. O MONIC é um método bastante citado em trabalhos relacionados ao tema desta pesquisa e embasou o desenvolvimento de outros métodos correlatos. Foram realizados 3 experimentos, descritos em detalhes nas seções seguintes.

- **Experimento 01:** Identificação de transições graduais, utilizando um conjunto de dados sintéticos. Os resultados da CETra foram comparados aos obtidos pelo MONIC.
- **Experimento 02:** Avaliação do tempo de processamento da CETra, utilizando dados sintéticos. A escalabilidade da CETra é comparada à do MONIC.
- **Experimento 03:** Avaliação de desempenho da CETra e do MONIC quando aplicados a dados reais, em conjunto com um algoritmo de agrupamento de fluxos de dados.

A técnica CETra foi desenvolvida na linguagem C++, utilizando o compilador GNU Compiler Collection (GCC) Versão 9.3.0. Todos os experimentos foram realizados no sistema operacional Linux Mint 20.1, em uma máquina com um processador Intel i7-10510U e com 8.0 Gb de RAM. Os limiares da CETra e do MONIC foram determinados empiricamente,

Quadro 2 – Limiares utilizados nos experimentos.

Experimento	Limiares								
	τ	τ_{split}	$\delta(Tam)$	$\delta(Media)$	$\delta(DesPad)$	$\delta(Soma)$	$\delta(SQ)$	τ_{NEW}	τ_{MISS}
01	0.5	0.25	0.3	-	-	-	-	0.5	0.5
02	0.5	0.25	0.3	-	-	-	-	0.5	0.5
03	0.5	0.25	0.5	0.5	0.25	0.5	0.5	0.5	0.5

Fonte: Elaborada pelo autor.

considerando também os valores definidos nos estudos experimentais relatados pelos autores do MONIC. Os valores dos limiares são listados no Quadro 2. Para o cálculo das sobreposições, foi atribuído o mesmo peso (1.0) para todas a fontes geradoras de dados em todos os experimentos. Todos os testes utilizados e a implementação do CETra estão disponíveis em um repositório público[1].

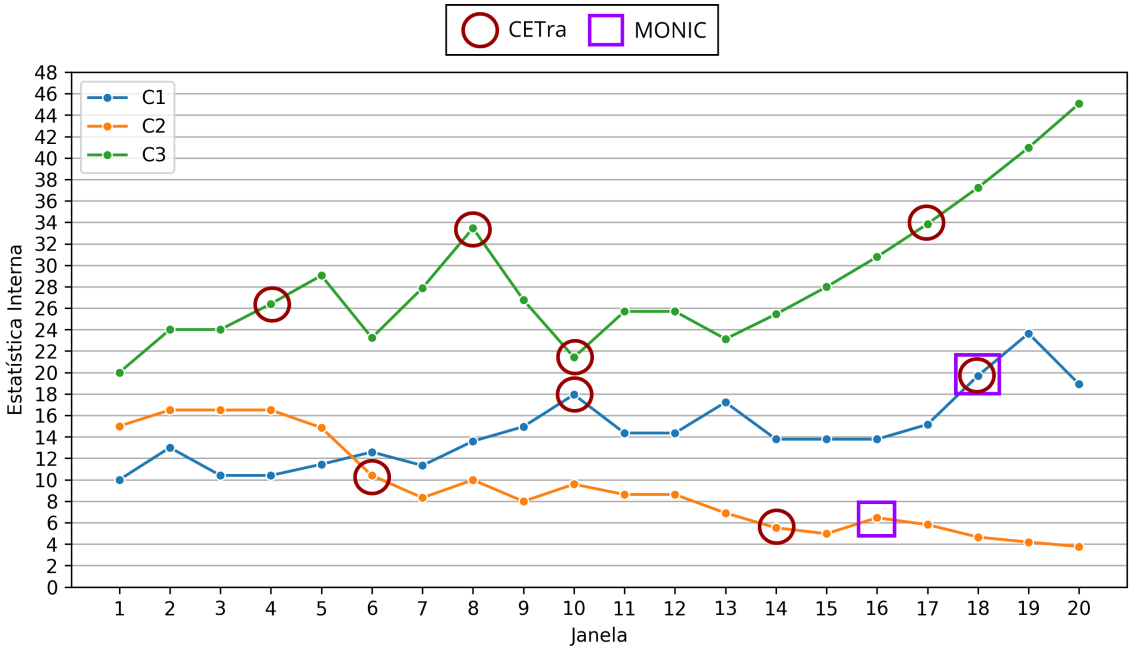
5.1 Experimento 01

O objetivo desse experimento é verificar a capacidade da CETra em detectar mudanças graduais entre agrupamentos, como discutido na Seção 4.1. Para isso, um conjunto com 20 agrupamentos sintéticos foi construído para simular os agrupamentos gerados por algoritmos de agrupamento em 20 janelas de dados consecutivas. Cada agrupamento possui 100 fontes de dados, com quantidade variável de grupos, dos quais 3 sobrevivem em todos eles. Esses 3 grupos têm uma estatística interna sintética que tem uma variação de até 30% entre cada um dos agrupamentos. O valor do limiar defino para essa estatística sintética é a mesma definida para a estatística de tamanho $\delta(Tam)$, neste experimento 0.3. O MONIC foi aplicado como originalmente concebido, ou seja, a detecção de transições foi realizada entre dois agrupamentos sequentes - $MONIC(\zeta^{t_{i-1}}, \zeta^{t_i})$. Além disso, é importante ressaltar que qualquer mudança em um dos grupos implica na atualização do agrupamento referencial no CETra, ou seja, os outros grupos sem mudanças internas detectadas também são atualizados para a sua configuração mais recente.

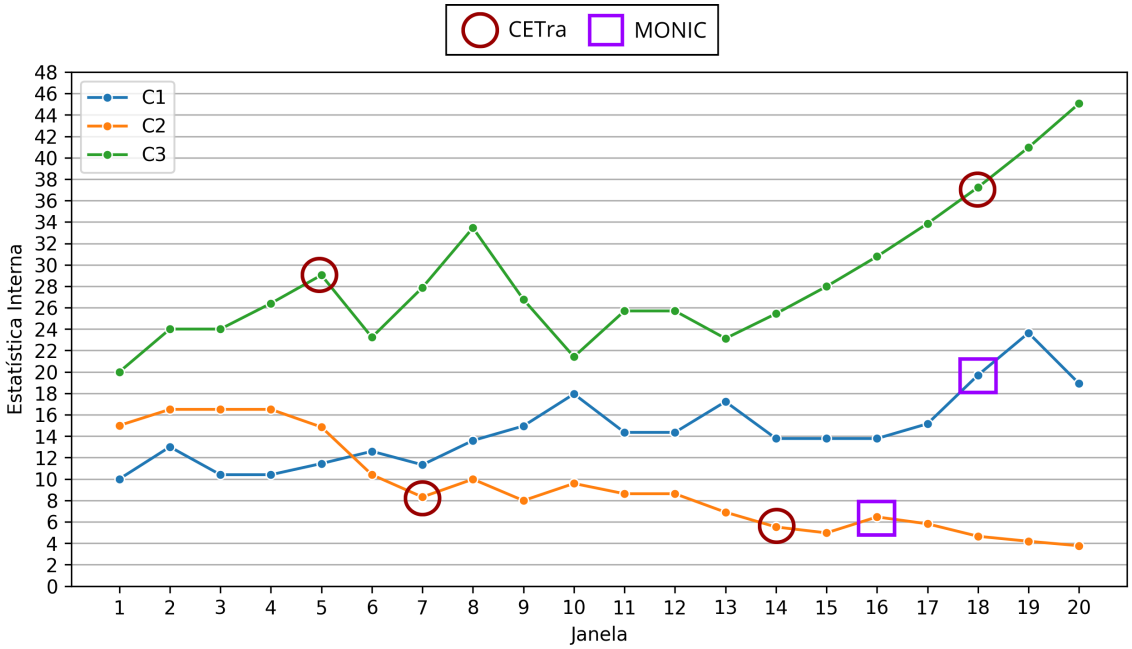
A Figura 15 mostra os valores da estatística interna para cada grupo sobrevivente nos 20 agrupamentos sucessivos, com destaque para as transições internas detectadas pela CETra (círculos vermelhos) e pelo MONIC (quadrados roxos). A Figura 15a mostra as transições detectadas pela CETra e pelo MONIC no momento em que elas ocorrem. Enquanto CETra é capaz de detectar 7 transições internas ao longo dos 20 agrupamentos, o MONIC é capaz de detectar somente duas. Isso é esperado, pois o MONIC só consegue detectar uma transição interna se houver a variação máxima de 30% entre dois agrupamentos sequentes, já que o limiar definido exige uma mudança mínima de 30%. Já a CETra monitora as transições comparando com o último agrupamento referencial, sendo então capaz de identificar mais mudanças possivelmente significativas uma vez que há pouca variação em espaços de tempos menores.

[1] Disponível em: https://github.com/afonsoMatheus/CETra

Figura 15 – Transições internas detectadas pela CETra e pelo MONIC.



(a) Sem tratamento de mudanças sequentes



(b) Com tratamento de duas mudanças sequentes

Fonte: Elaborada pelo autor.

A Figura 15b apresenta um cenário em que são necessárias duas mudanças significativas sequentes para a CETra considerar como uma transição interna. É possível observar que as mudanças na estatística interna do grupo C1 nos tempos 8 e 10 não são mais consideradas transições. Isso pode evitar que variações momentâneas nos dados dos fluxos impactem na detecção de transições internas, por exemplo causando atualizações desnecessárias do agrupamento referencial. Essa tolerância nas mudanças internas pode ser definida de acordo com a necessidade

do domínio do problema que está sendo trabalhado. Por exemplo, se toda mudança que ocorre em uma estatística interna que atinja o limiar pré-definido pelos especialistas do problema é importante para a tomada de uma decisão crítica, então é mais interessante não ter esse controle de mudanças sequenciais. Em contrapartida, se oscilações são previstas em certos domínios de problema, é interessante estabelecer uma quantidade de mudanças sequenciais necessárias para definir uma transição interna como significativa.

5.2 Experimento 02

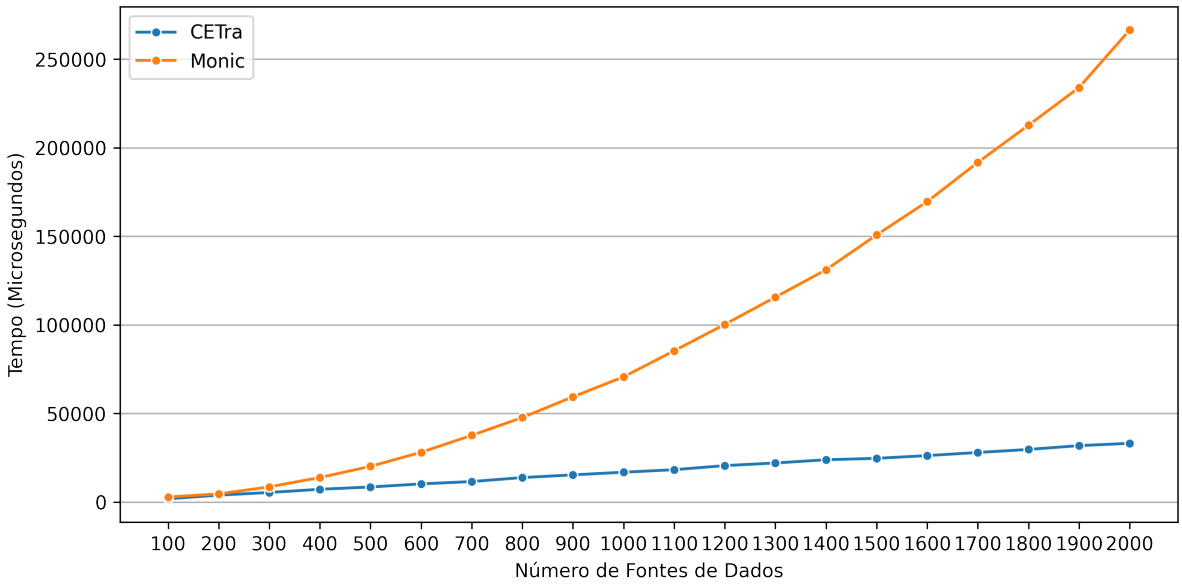
O objetivo desse experimento é avaliar o tempo de processamento da técnica CETra e verificar o impacto de sua complexidade de tempo computacional, como discutido na Seção 4.2. Para isso, foram gerados sequencialmente 20 agrupamentos sintéticos, cada qual contendo uma quantidade de fontes de dados N maior que a do agrupamento anterior. O primeiro agrupamento contém 100 fontes de dados, o segundo 200, até o último agrupamento que contém 2000 fontes de dados. Nesse experimento, é realizado o monitoramento de transições entre as N fontes de dados do mesmo agrupamento, por exemplo, o agrupamento 5 tem 500 fontes de dados e é comparado com uma cópia das mesmas 500 fontes de dados.

O experimento é realizado desse modo pois a quantidade de fontes de dados que compõem o agrupamento é o fator mais impactante para as técnicas de monitoramento de transições, uma vez que a operação mais custosa envolve a comparação entre todos os elementos de dois agrupamentos. A configuração do agrupamento tem um impacto pouco significativo no tempo de processamento, pois a quantidade de grupos que um agrupamento pode ter é geralmente muito inferior à quantidade de elementos do mesmo. O Capítulo 4 discute com mais detalhes a complexidade computacional dos algoritmos de monitoramento de transições.

Nesse experimento foi realizado o monitoramento de transições entre mesmos agrupamentos, para cada um dos 20 agrupamentos sintéticos gerados. Para evitar oscilações, o tempo de processamento médio foi obtido após 10 execuções de cada teste. Estatísticas internas de tamanho também são monitoradas. Por fim, o MONIC foi mais uma vez aplicado, com o objetivo de comparar a escalabilidade de ambos.

A Figura 16 apresenta os resultados obtidos. Para os primeiros agrupamentos, que contêm uma quantidade menor de fontes de dados, a CETra e o MONIC alcançam resultados similares. Porém, à medida que a quantidade de fontes de dados presentes no agrupamento amenta, a diferença no tempo de processamento entre a CETra e o MONIC fica cada vez mais perceptível, sendo a CETra 6 vezes mais rápida no agrupamento constituído por 2000 sensores. Esse comportamento é esperado, CETra calcula a sobreposição de conjuntos para a construção da matriz de sobreposições de modo linear em relação ao número de fontes de dados utilizando a lógica de buscas em tabela *hash*, enquanto o MONIC cresce de modo quadrático por realizar comparação entre todas as fontes de dados dos dois agrupamentos. A quantidade de transições

Figura 16 – Tempo de processamento da CETra e do MONIC



Fonte: Elaborada pelo autor.

externas e internas detectadas tem pouco impacto no tempo de processamento, uma vez que estão atreladas ao número de grupos em cada agrupamento, que é geralmente muito menor que o número total de fontes de dados.

A eficiência obtida pelo uso de uma técnica com complexidade linear é muito vantajosa para todos os casos, principalmente para domínios com quantidades maiores de fontes de dados. No entanto, considerando que a detecção de transições deve ser feita para qualquer granularidade de tempo, ou seja, para qualquer tamanho de movimentação da janela, um processamento mais eficiente, mesmo para quantidade menor de fontes de dados, é necessário em casos cenários com intervalos de tempo mais curtos entre gerações de novos agrupamentos.

5.3 Experimento 03

O terceiro experimento tem como objetivo testar a eficácia da técnica CETra em monitorar transições ao ser utilizada junto a um algoritmo de agrupamento de fluxos de dados. Para isso, foi utilizada uma base de dados real, constituída por 513 sensores meteorológicos localizados em diferentes cidades do Brasil[2]. Cada uma dessas fontes de dados contém dados diários de temperatura máxima referentes a cinco anos, de 01 de janeiro de 2012 até 31 de dezembro de 2016, totalizando 1827 objetos de dados para cada sensor. Com o objetivo de comparar performance e transições, o MONIC também foi utilizado. Para esse experimento, os limiares para as estatísticas internas tamanho, média, soma, soma dos quadrados foram definidas para 0.5 e o limiar para o desvio padrão foi definido para 0.25, para exemplificar a ocorrência de evolução

[2] Disponível em: https://www.cnpaf.embrapa.br/infoclima/

gradual em dados reais.

Quadro 3 – Parâmetros utilizados para o Pod-Clus

Caso de Teste	**Parâmetros**			
	Tamanho da janela	Atualização da janela	Quantidade de outliers	Desvio Padrão mínimo
Geral	60	15	10	0.60
União	60	15	5	0.65

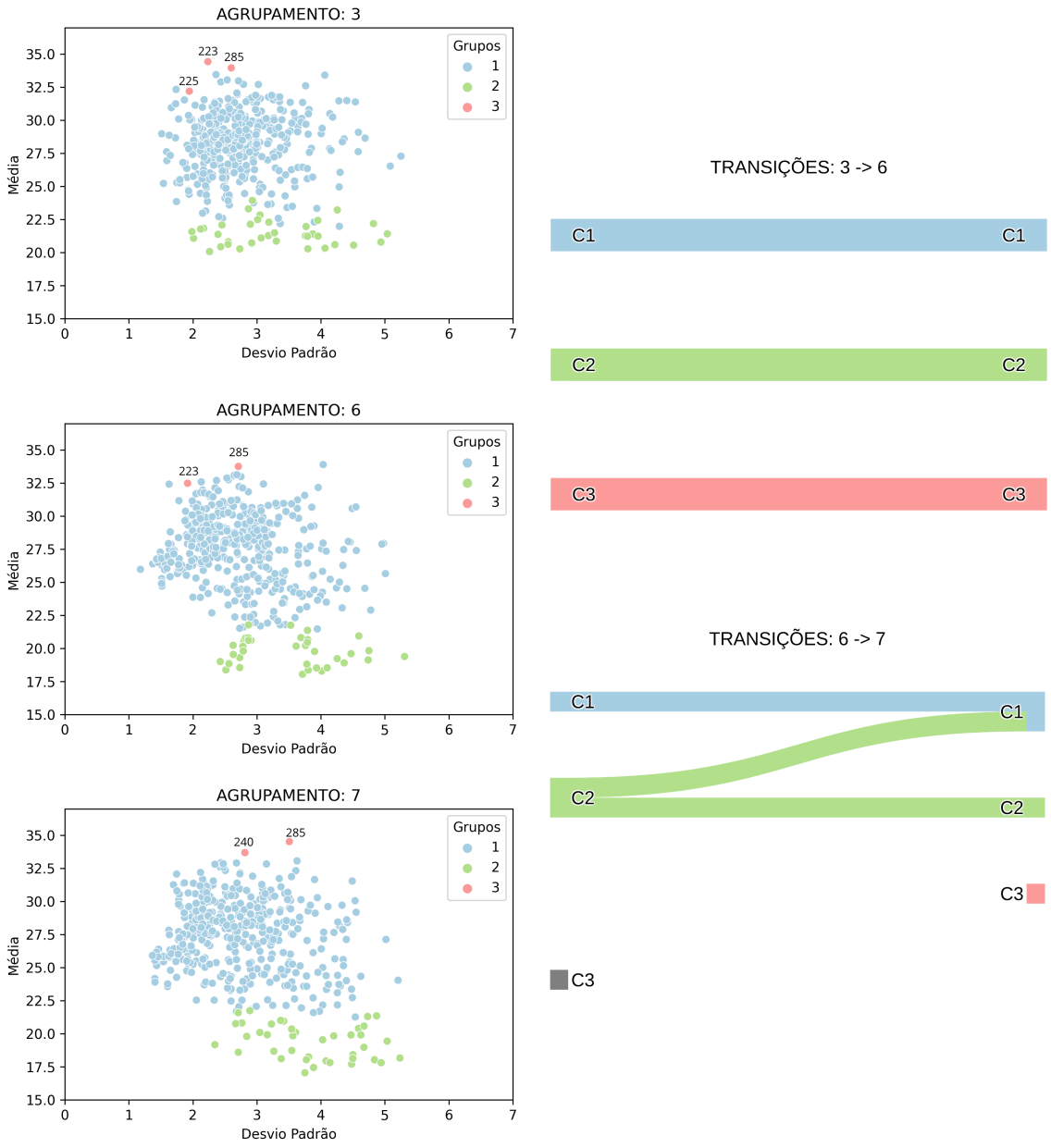
Fonte: Elaborada pelo autor.

O algoritmo escolhido para agrupamento foi o Pod-Clus (Seção 2.2.2), por ter sido um método concebido primariamente para as tarefas de agrupamento de fluxos e pela disponibilidade do código fonte do algoritmo, sendo implementado também em C++. Os parâmetros utilizados estão listados no Quadro 3. Foi definida uma janela de tamanho 60, englobando semanticamente dois meses de temperatura máxima, com um deslocamento de 15, semanticamente 2 semanas. Esses valores foram determinados empiricamente, de modo a ilustrar a ocorrência de evoluções graduais nos agrupamentos, além de transições tanto externas e como internas os grupos. O Pod-Clus também pede um parâmetro de número mínimo de elementos para definir quando um grupo de *outliers* se torna um grupo normal, definido como 10, e um parâmetro do desvio padrão máximo que um objeto pode ter ter um relação ao centroide de um grupo para ser alocado a ele, definido como 0.6. Uma variação desses parâmetros (número mínimo de elementos = 5 e desvio padrão máximo = 0.65, apresentados no Quadro 3 no caso de teste União) causou uma alteração nos agrupamento que resultou numa transição gradual de união, discutida nos resultados apresentados a seguir.

Dos 116 janelamentos realizados para processar todos os dados, o MONIC detectou 31 transições, sendo 19 externas e 12 internas, enquanto a CETra detectou 43 transições, sendo 21 externas e 22 internas. Dado o número elevado de transições, a seguir são apresentados alguns exemplos de transições detectadas pela CETra que o MONIC não detectou, destacando as diferenças entre os dois ao monitorar as transições. As transições são ilustradas por meio de diagramas de Sankey, criados a partir das mesmas. Neles, os grupos referenciais do lado esquerdo são conectados aos grupos evolutivos no lado direito, conexões essas que refletem a transição ocorrida entre os grupos associados.

A Figura 17 ilustra duas transições externas ocorridas entre o janelamento 6 e 7 que a CETra detectou enquanto o MONIC identificou os três grupos como sobreviventes. Essa diferença na detecção das transições está relacionada com a abordagem da CETra para acompanhar a evolução gradual do agrupamento. Observando o janelamento 3, referente ao agrupamento referencial definido pela CETra, percebe-se que o Grupo 3 é formado por três sensores (223, 225, 285). Após 3 janelamentos, ele passa a ser formado por somente 2 (223, 285). Até esse momento, o Grupo 3 ainda é formado por mais de 50% dos elementos presentes nesse grupo no agrupamento referencial. No janelamento seguinte, o Grupo 3 passa a ser formado por menos de

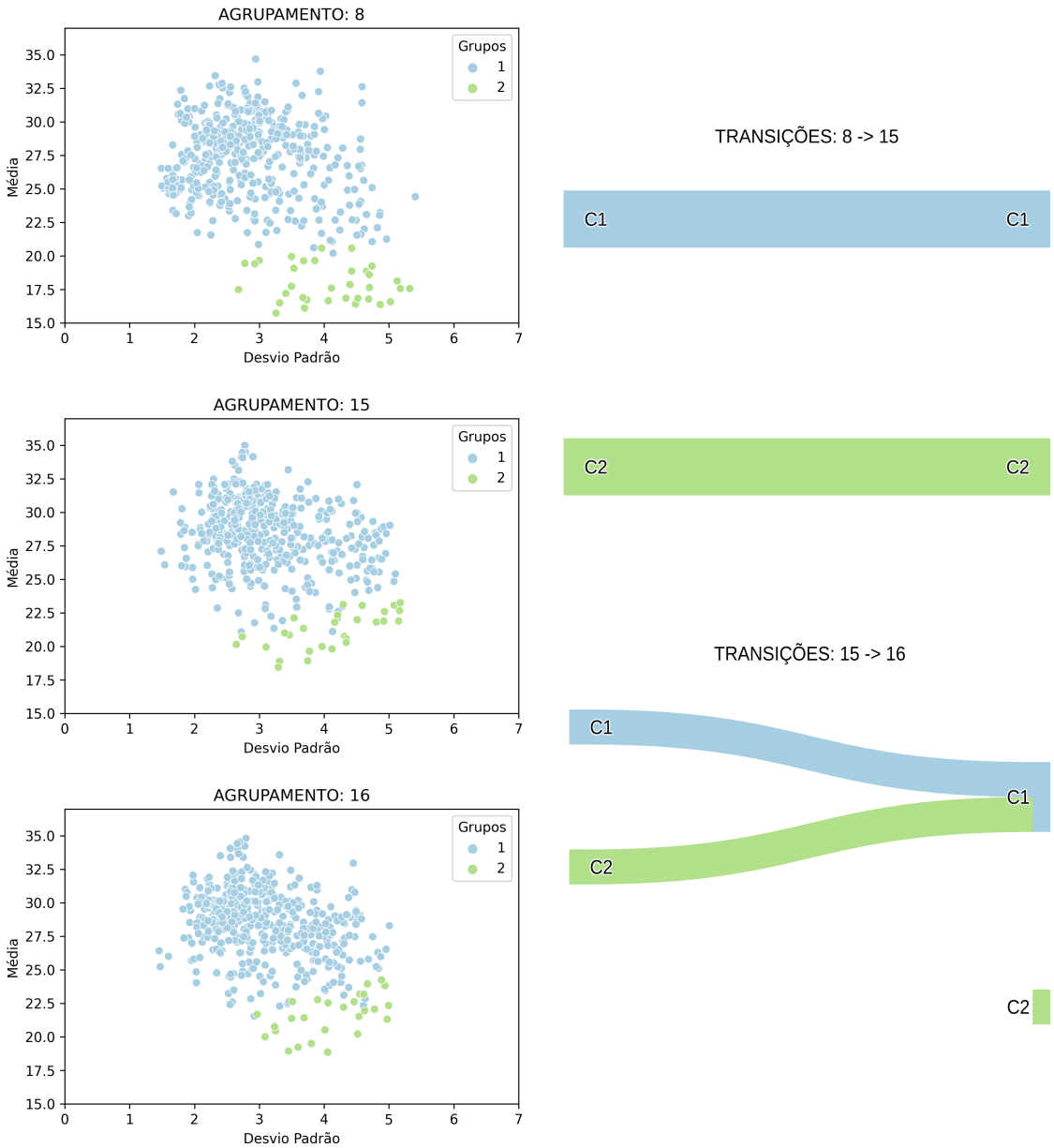
Figura 17 – Exemplo de transições de morte, nascimento e separação detectadas somente pela CETra.



Fonte: Elaborada pelo autor.

50% dos elementos do grupo referencial, enquanto os demais sensores que pertenciam ao Grupo 3 não estão presentes nos outros dois grupos, pois nesses janelamentos, eles foram agrupados nos grupos de *outlier* (constatação essa obtida pela detecção de fontes de dados desaparecidos do CETra), e portanto também não houve separação. Assim, o Grupo 3 é identificado como suficientemente diferente para que ele seja considerado um nascimento no agrupamento evolutivo, enquanto o do agrupamento referencial sofre, consequentemente, uma morte. O MONIC não consegue detectar essas transições, uma vez que a comparação feita é em relação ao janelamento 6, que ainda identifica o Grupo 3 com pelo menos 50% dos elementos. Essa mesma situação ocorreu com a separação do Grupo 2, o qual no janelamento 7 não tinha mais de 50% dos

Figura 18 – Exemplo de transição de união detectada somente pela CETra.

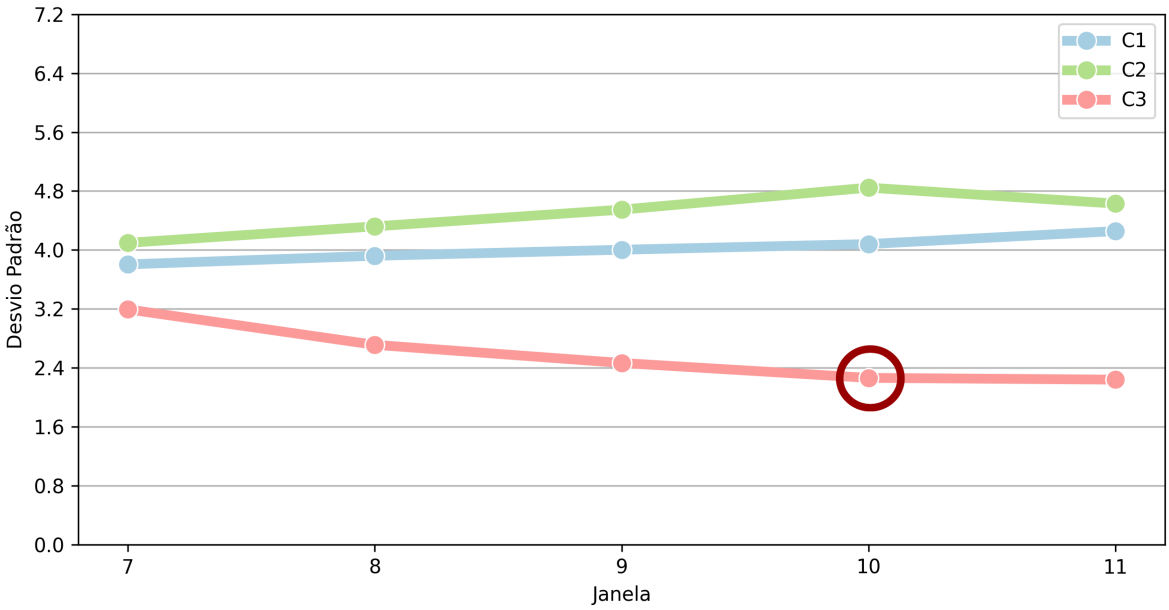


Fonte: Elaborada pelo autor.

elementos que o compunham no janelamento 3, identificando nesse caso uma separação nesse grupo.

A Figura 18 apresenta uma transição externa de união que somente a CETra é capaz de detectar, que se assemelha ao ocorrido na transição de separação. Nesse caso, os sensores que faziam parte do Grupo 2 foram gradualmente transitando para o Grupo 1, até o momento que mais de 50% dos elementos que compunham o Grupo 2 passaram a compor o Grupo 1 em uma janela posterior, gerando assim uma união. Portanto, o Grupo 2 do Agrupamento 8 não é o mesmo Grupo 2 do Agrupamento 16, sendo este considerado um nascimento nesse momento. Vale ressaltar que, ao observar somente os agrupamentos de ambas as Figuras 17 e 18, não fica

Figura 19 – Exemplo de transição interna detectada somente pela CETra.



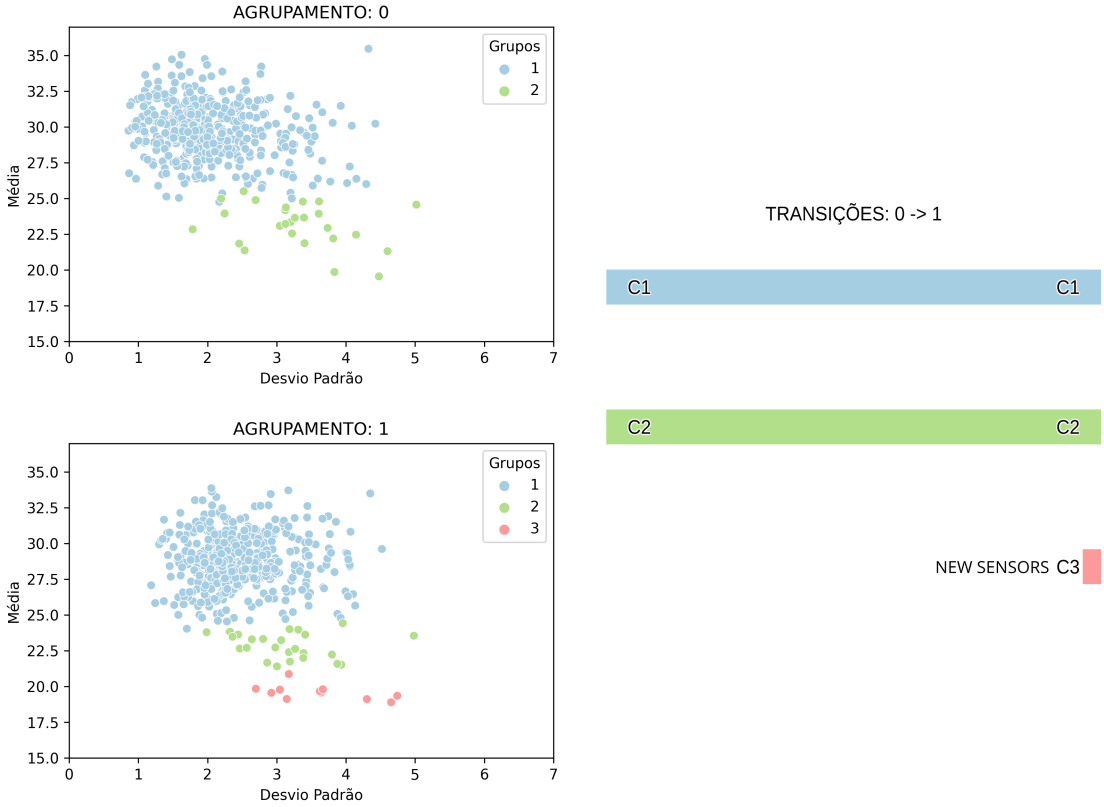
Fonte: Elaborada pelo autor.

perceptível que seus respectivos grupos são significativamente diferentes em termos de sensores que os compõem, mas esse fato se torna mais evidente a partir das transições detectadas.

A Figura 19 ilustra uma transição interna ocorrida entre os janelamentos 7 e 10, compostos por sobrevivências de três grupos, que a CETra detectou e o MONIC não. De modo muito similar ao ocorrido no Experimento 2, a CETra foi capaz de detectar uma mudança significativa na estatística de desvio padrão no Grupo 3 por adotar o uso do agrupamento referencial definido no janelamento 7. Assim, como há evolução gradual nas estatísticas internas, o MONIC não é capaz de detectar uma transição interna utilizando dois janelamentos consecutivos pois, em um único janelamento não é possível ter um aumento de 25% nesse estatística, que cresce numa grandeza menor a cada novo janelamento processado. Isso também ocorreu com as demais estatísticas internas, refletindo em uma quantidade maior de transições internas detectadas pela CETra.

A Figura 20 mostra uma transição de nascimento que é detectada tanto pela CETra como pelo MONIC, porém a CETra é capaz de identificar que o Grupo 3 nascido nesse agrupamento é composto de sensores que não estavam presentes no agrupamento referencial. Isso é possível em razão do mapeamento que a CETra faz para detectar sensores desaparecidos e sensores novos a medida que as transições são monitoradas, e por utilizar um limiar pré-definido de 50%, ou seja, se a composição de um grupo que nasceu é formada por mais de 50% de sensores novos, então a CETra detecta um nascimento por novos sensores. Com o Pod-Clus, isso ocorre porque esses sensores estavam presentes nos grupos de *outliers* inicialmente e, no momento da detecção da transição, esse grupo de *outlier* se tornou grande o suficiente para ser considerado um grupo normal. Essa é um informação adicional fornecida pela CETra que gera um entendimento mais

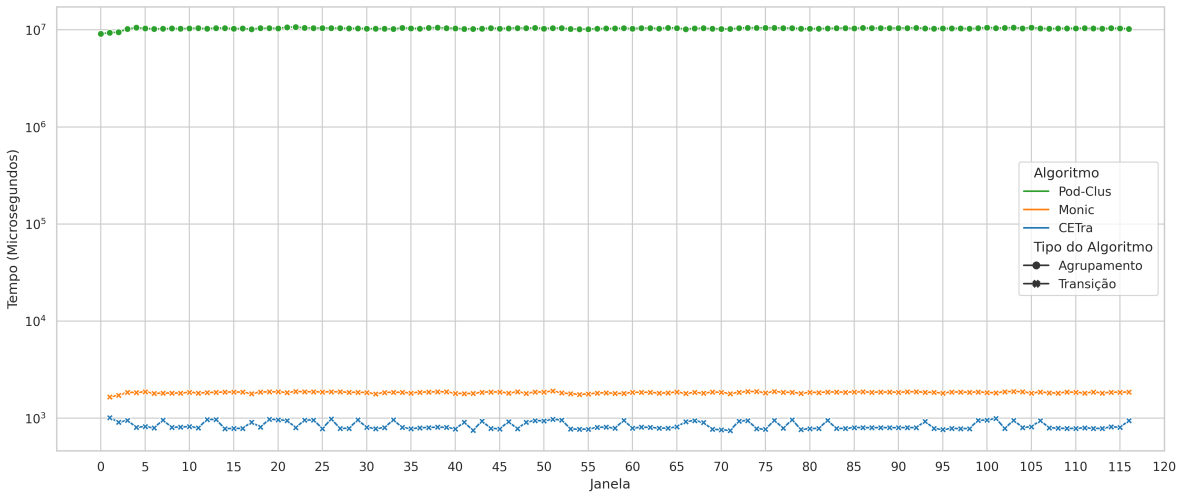
Figura 20 – Transição de nascimento por novos sensores detectada pela CETra.



Fonte: Elaborada pelo autor.

correto das mudanças no agrupamento.

Figura 21 – Tempos de processamento para processar cinco anos de dados de temperatura



Fonte: Elaborada pelo autor.

Por fim, o gráfico da Figura 21 mostra o tempo de processamento do algoritmo de agrupamento e dos algoritmos de monitoramento de transições para todos os janelamentos realizados

para processar todos os cinco anos de dados, excluindo o primeiro pois é o agrupamento inicial. A CETra teve um tempo muito menor em relação ao do Pod-Clus e é 2 vezes mais rápido que o do MONIC, mesmo sendo um experimento realizado com uma quantidade não tão grande de fontes de dados. Vale ressaltar que as oscilações em alguns tempos da CETra indicam janelamentos em que houve a detecção de alguma transição e por isso o agrupamento referencial foi atualizado, resultando em uma nova leitura nos dados do agrupamento atual.

5.4 Considerações Finais

Como apresentado neste Capítulo, tanto por meio dos experimentos realizados com agrupamentos gerados sinteticamente quanto pelos experimentos que utilizaram agrupamentos gerados por algoritmo de agrupamento de fluxos de dados a partir de dados reais, a técnica de monitoramento e detecção de transições CETra apresentou bons resultados comparados ao método correlato da literatura, tanto em questão da qualidade do conjunto de transições detectadas como da eficiência do processamento.

A partir das regras de decisão definidas, a técnica é capaz de detectar corretamente as transições que ocorrem ao longo da evolução dos conjuntos de dados, sendo capaz também de identificar transições tanto externas como internas próprias do contexto de fluxo de dados dado a sua característica de evolução gradual. Além disso, a CETra apresenta tempos de processamento menores tanto em relação ao algoritmo de agrupamento quanto ao método correlato, sendo até duas vezes mais rápido que o MONIC já para um caso com uma quantidade ainda pequena de fontes de dados, ou seja, a diferença tende a aumentar bastante conforme mais fontes de dados têm de ser processadas a cada janelamento.

A técnica também foi facilmente aplicada em conjunto com algoritmo de agrupamento, utilizando suas saídas à medida que novos dados foram processados. Esse é um forte indicativo da portabilidade da CETra, sendo rapidamente adicionada e utilizada junto às tarefas de agrupamento de fluxos de dados. Por fim, como o método de detecção de transições para agrupamentos não sumarizados monitora a localização de cada fonte de dado do conjunto de dados, a saída da CETra pode ser refinada para obter informações mais específicas sobre a evolução dos dados, como foi observado na identificação de fontes de dados que desaparecem e aparecem ao longo do processamento.

CAPÍTULO

6

CONCLUSÃO

As tarefas que visam adquirir conhecimento não óbvio a partir de fluxos de dados são de grande importância para os problemas da atualidade, pois uma quantidade imensurável de dados é gerada e disponibilizada por meio de fontes de dados dos mais diversos domínios de aplicação como em sensores industriais, sensores de segurança ou sensores meteorológicos. Nesse contexto, tarefas de agrupamento que têm como objetivo identificar conjuntos disjuntos de fontes de dados mais semelhantes entre si estão as atividades mais importantes para esse tipo de dado. Como apresentado por Silva *et al.* (2013), entender e aprofundar o entendimento da evolução dos agrupamentos gerados é de grande importância para o objetivo de descoberta de conhecimento do conjunto de dados processado. Esse entendimento pode ser obtido utilizando métodos que monitoram as transições dos grupos de agrupamentos de um mesmo contexto de dados que são gerados e atualizados continuamente. Todavia, são poucos os métodos da literatura que têm como objetivo monitorar transições, tanto para bases de dados convencionais como para fluxos de dados, especialmente para as tarefas de agrupamento de fluxos de dados. Tal lacuna na literatura motivou o desenvolvimento desta dissertação de mestrado.

As principais oportunidades de pesquisa relacionadas ao monitoramento de transições em agrupamento de fluxos de dados que promoveram a realização deste trabalho foram:

1. Conceber uma técnica totalmente baseada nos problemas de agrupamento de fluxos de dados.
2. Averiguar a possibilidade de utilizar uma abordagem de monitoramento com grupos não sumarizados.
3. Otimizar o tempo de processamento para uma abordagem não sumarizada própria para atender os requisitos de fluxos de dados.
4. Explorar o monitoramento de transições junto a um tipo de dado que evolui constantemente e gradualmente.

5. Possibilitar uma utilização simplificada desse método para diferentes algoritmos de agrupamento.

6. Avaliar o uso da técnica em uma situação real de processamento de dados junto a um algoritmo de agrupamento de fluxos de dados.

Sobre o primeiro item, não foi encontrado na literatura um método de monitoramento de transições criado especificamente para o contexto de agrupamento de fluxos de dados, o que pode ser atribuído à quantidade significativamente menor de trabalhos na literatura que abordem esse modo de agrupamento em fluxos de dados. O segundo item foi levantado pela falta da abordagem não sumarizada para o contexto de fluxos de dados, já que a abordagem sumarizada foi tratada em tarefas de agrupamento de objetos do fluxo. Pelas características do agrupamento de fluxos de dados, a abordagem não sumarizada é factível para ser aplicada e utilizada. O terceiro item nasce da necessidade de otimizar o processamento, até então quadrático, da abordagem não sumarizada, podendo ser um limitante para fluxos de dados. Sobre o quarto item, deve-se considerar que os agrupamentos tendem a ter pequenas mudanças incrementais ao longo de sua evolução, característica essa que deve ser levada em consideração para obter um conjunto de transições mais coerentes com esse tipo de dado. Além disso, o quinto item aponta a necessidade da técnica ser aplicável a diferentes algoritmos de agrupamento, evitando a necessidade de integrações mais complexas nos mesmos. Por fim, o sexto item está relacionado à aplicabilidade da técnica em uma situação mais próxima da real, visando eficácia e eficiência.

6.1 Principais Contribuições

Esta dissertação contém contribuições na área de mineração de fluxos de dados, apresentando a técnica CETra (*Cluster Evolution Tracker*) para monitoramento de transições em tarefas de agrupamento de fluxos de dados, abordando soluções para questões de pesquisa pertinentes a esse processo, em conformidade com o estado da arte.

CETra é baseada nas abordagens de monitoramento de transições da literatura para agrupamentos não sumarizados, utilizando sobreposições entre grupos de agrupamentos distintos para identificar equivalências entre eles e assim, por meio de um conjunto bem definido de regras de decisão, detectar as transições ocorridas. A técnica destaca-se por construir de modo mais otimizado a principal estrutura lógica utilizada para aplicar a detecção de transições e por considerar a evolução constante e gradual dos fluxos de dados.

A CETra foi comparada a um método correlato da literatura que utiliza uma abordagem não sumarizada de monitoramento. Além de ser capaz de detectar um conjunto de transições mais coerentes, CETra também obteve tempos de processamento duas vezes mais rápidos em um conjunto dados reais. Essa eficiência de processamento tende a ficar mais evidente conforme aumentam as fontes de dados no conjunto. Ademais, os testes com dados sintéticos demonstram

tanto essa eficiência de processamento quanto a detecção de transições que refletem a importância de considerar a evolução gradual dos dados, sendo que todas as transições em geral são mais precisas por serem obtidas a partir do monitoramento de cada elemento do agrupamento na abordagem não sumarizada. O módulo de software desenvolvido para a CETra também apresenta um formato de entrada acessível para que algoritmos de agrupamento utilizem-o sem alterações complexas em sua estrutura.

Sumarizando, as contribuições provenientes desta dissertação de mestrado são:

- A técnica CETra, para monitoramento de transições em agrupamentos de fluxos de dados.
- Um modo otimizado de se construir a principal estrutura lógica utilizada para detectar transições, a matriz de sobreposições.
- A lógica de armazenamento de agrupamentos referenciais e evolutivos para identificação de mudanças graduais.
- Um conjunto de regras de decisão para detecção de transições para o contexto de agrupamento de fluxos de dados.
- Avaliação experimental das principais características desenvolvidas para a CETra, onde mostrou-se que as questões de pesquisa puderam ser respondidas.

6.2 Propostas para Trabalhos Futuros

As propostas para trabalhos futuros são brevemente apresentadas a seguir.

- **Modelo mais complexo de pesos**: É interessante que sejam explorados modelos mais complexos de atribuição de pesos às fontes de dados, buscando refinar a detecção de transições junto ao algoritmo de agrupamento e ao domínio do problema abordado.
- **Utilização de transições**: Investigar como as transições detectadas pela técnica CETra podem ser utilizadas em conjunto com outros algoritmos de descoberta de conhecimento, por exemplo, para definir padrões e refinar novos grupos.
- **Conhecimento semântico**: Realizar análises exploratórias e aprofundar os conhecimentos semânticos que podem ser adquiridos junto a especialistas ao analisar as transições identificadas pela técnica CETra em um problema de um determinado domínio.
- **Aplicação em contextos reais**: Avaliar a performance de CETra em cenários que envolvem processamento em tempo real de sensores físicos e com os problemas inerentes a esse ambiente, como atrasos de rede, por exemplo.

REFERÊNCIAS

AGGARWAL, C. C. *et al.* **Data mining: the textbook**. [S.l.]: Springer, 2015. v. 1. Citado na página 29.

AGGARWAL, C. C.; PHILIP, S. Y.; HAN, J.; WANG, J. A framework for clustering evolving data streams. In: **Proceedings of the 29th Annual International Conference on Very Large Data Bases**. Berlim, Germany: VLDB Endowment, 2003. (VLDB '03, v. 29), p. 81–92. Citado na página 47.

ALVES, G.; BARIONI, M. C. N.; FARIA, E. R. A framework for online clustering based on evolving semi-supervision. In: SBC. **Proceedings of the 32th Brazilian Symposium on Databases**. Uberlândia, MG, Brazil, 2017. v. 32, p. 16–27. Citado nas páginas 32, 35 e 53.

ANDERSON, R.; KOH, Y. S.; DOBBIE, G. Predicting concept drift in data streams using metadata clustering. In: IEEE. **2018 International Joint Conference on Neural Networks**. Rio de Janeiro, RJ, Brazil, 2018. p. 1–8. Citado na página 29.

ARBELAITZ, O.; GURRUTXAGA, I.; MUGUERZA, J.; PÉREZ, J. M.; PERONA, I. An extensive comparative study of cluster validity indices. **Pattern Recognition**, Elsevier, v. 46, n. 1, p. 243–256, 2013. Citado nas páginas 44 e 45.

BARBARA, D. Requirements for clustering data streams. **The Association for Computing Machinery's Special Interest Group on Knowledge Discovery and Data Mining Explorations Newsletter**, ACM, New York, NY, USA, v. 3, n. 2, p. 23–27, jan. 2002. Citado nas páginas 40, 41, 44 e 46.

BERINGER, J.; HÜLLERMEIER, E. Online clustering of parallel data streams. **Data & Knowledge Engineering**, Elsevier, v. 58, n. 2, p. 180–204, 2006. Citado na página 48.

BERKHIN, P. A survey of clustering data mining techniques. In: **Grouping Multidimensional Data**. Berlin, Heidelberg: Springer, 2006. p. 25–71. Citado na página 45.

BIFET, A.; GAVALDA, R. Learning from time-changing data with adaptive windowing. In: **Proceedings of the 2007 Society for Industrial and Applied Mathematics International Conference on Data Mining**. Minneapolis, Minnesota, USA: SIAM, 2007. v. 7, p. 443–448. Citado na página 43.

BONES, C. C. **Agrupamento de fluxos de dados utilizando dimensão fractal**. Tese (Doutorado) — Universidade de São Paulo, 2018. Citado nas páginas 47 e 48.

BONES, C. C.; ROMANI, L. A.; SOUSA, E. P. de. Clustering multivariate data streams by correlating attributes using fractal dimension. **Journal of Information and Data Management**, v. 7, n. 3, p. 249–264, 2016. Citado nas páginas 32, 47, 48 e 49.

BRUN, M.; SIMA, C.; HUA, J.; LOWEY, J.; CARROLL, B.; SUH, E.; DOUGHERTY, E. R. Model-based evaluation of clustering validation measures. **Pattern Recognition**, Elsevier, v. 40, n. 3, p. 807–824, 2007. Citado nas páginas 44 e 45.

CAO, F.; ESTERT, M.; QIAN, W.; ZHOU, A. Density-based clustering over an evolving data stream with noise. In: SIAM. **Proceedings of the 2006 Society for Industrial and Applied Mathematics International Conference on Data Mining**. Bethesda, Maryland, USA, 2006. p. 328–339. Citado na página 47.

CHAOVALIT, P. **Clustering transient data streams by example and by variable**. Tese (Doutorado) — University of Maryland, 2009. Citado nas páginas 48 e 49.

CHEN, Y.; TU, L. Density-based clustering for real-time stream data. In: **Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. San Jose, California, USA: ACM, 2007. p. 133–142. Citado na página 47.

CORMODE, G.; GAROFALAKIS, M.; HAAS, P. J.; JERMAINE, C. *et al.* Synopses for massive data: Samples, histograms, wavelets, sketches. **Foundations and Trends in Databases**, Now Publishers, Inc., v. 4, n. 1–3, p. 1–294, 2011. Citado na página 43.

CYGANEK, B. Change detection in multidimensional data streams with efficient tensor subspace model. In: **International Conference on Hybrid Artificial Intelligence Systems**. Oviedo, Asturias, Spain: Springer, 2018. v. 10870, p. 694–705. Citado na página 40.

DATAR, M.; GIONIS, A.; INDYK, P.; MOTWANI, R. Maintaining stream statistics over sliding windows. **Journal on Computing**, SIAM, v. 31, n. 6, p. 1794–1813, 2002. Citado na página 43.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. **Artificial Intelligence Magazine**, v. 17, n. 3, p. 37–37, 1996. Citado nas páginas 30, 32, 44 e 45.

GABER, M. M.; ZASLAVSKY, A.; KRISHNASWAMY, S. Mining data streams: a review. **Association for Computing Machinery Sigmod Record**, ACM, v. 34, n. 2, p. 18–26, 2005. Citado na página 43.

____. A survey of classification methods in data streams. In: **Data Streams**. Boston, Massachusetts, USA: Springer, 2007, (Advances in Database Systems, v. 31). p. 39–59. Citado na página 44.

GAMA, J. **Knowledge discovery from data streams**. Boca Raton, Florida, USA: Chapman and Hall/CRC, 2010. 233 p. Citado nas páginas 31, 32, 33, 39, 40, 42, 43, 46, 47 e 52.

GAMA, J.; ŽLIOBAITĖ, I.; BIFET, A.; PECHENIZKIY, M.; BOUCHACHIA, A. A survey on concept drift adaptation. **Association for Computing Machinery Computing Surveys**, ACM, v. 46, n. 4, p. 44, 2014. Citado nas páginas 29, 41 e 42.

GANTZ, J.; REINSEL, D. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. **IDC iView: IDC Analyze the future**, v. 2007, n. 2012, p. 1–16, 2012. Citado na página 29.

GUHA, S.; MEYERSON, A.; MISHRA, N.; MOTWANI, R.; O'CALLAGHAN, L. Clustering data streams: Theory and practice. **Transactions on Knowledge and Data Engineering**, IEEE, v. 15, n. 3, p. 515–528, 2003. Citado na página 46.

HAWWASH, B.; NASRAOUI, O. Stream-dashboard: a framework for mining, tracking and validating clusters in a data stream. In: ACM. **Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications**. Beijing, China, 2012. p. 109–117. Citado nas páginas 35, 53, 62 e 63.

KRIEGEL, H.-P.; KRÖGER, P.; NTOUTSI, I.; ZIMEK, A. Density based subspace clustering over dynamic data. In: SPRINGER. **International Conference on Scientific and Statistical Database Management**. Berlin, Heidelberg, 2011. (Lecture Notes in Computer Science, v. 6809), p. 387–404. Citado na página 47.

LÜHR, S.; LAZARESCU, M. Incremental clustering of dynamic data streams using connectivity based representative points. **Data & Knowledge Engineering**, Elsevier, v. 68, n. 1, p. 1–27, 2009. Citado na página 46.

MANDELBROT, B. B. **The fractal geometry of nature**. New York: WH Freeman, 1983. v. 173. Citado na página 49.

MOULTON, R. H.; VIKTOR, H. L.; JAPKOWICZ, N.; GAMA, J. Clustering in the presence of concept drift. In: **European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases**. Dublin, Ireland: Springer, 2018. v. 11051, p. 339–355. Citado na página 29.

MUTHUKRISHNAN, S. Data streams: Algorithms and applications. **Foundations and Trends in Theoretical Computer Science**, Now Publishers, Inc., v. 1, n. 2, p. 117–236, 2005. Citado na página 39.

NAMITHA, K.; SAJU, N. K.; KUMAR, G. S. Tracking cluster transitions using summaries. In: IEEE. **2018 International Conference on Data Science and Engineering (ICDSE)**. Cochin, India, 2018. p. 1–5. Citado nas páginas 35, 53 e 64.

NGUYEN, H.-L.; WOON, Y.-K.; NG, W.-K. A survey on data stream clustering and classification. **Knowledge and Information Systems**, Springer, v. 45, n. 3, p. 535–569, 2015. Citado nas páginas 32, 39, 40, 41, 44 e 46.

NTOUTSI, E.; SPILIOPOULOU, M.; THEODORIDIS, Y. Fingerprint: Summarizing cluster evolution in dynamic environments. **International Journal of Data Warehousing and Mining**, IGI Global, v. 8, n. 3, p. 27–44, 2012. Citado nas páginas 35, 53 e 64.

NTOUTSI, I.; SPILIOPOULOU, M.; THEODORIDIS, Y. Tracing cluster transitions for different cluster types. **Control & Cybernetics**, v. 38, n. 1, p. 239–259, 2009. Citado nas páginas 35, 53 e 64.

NUNES, S. A.; ROMANI, L. A. S.; AVILA, A. M. H.; COLTRI, P. P.; TRAINA, A. J. M.; SOUSA, E. P. M. Finding spatio-temporal patterns in multidimensional data streams. **Journal of Information and Data Management**, v. 4, n. 3, p. 327–340, 2013. Citado na página 40.

OLIVEIRA, M.; GAMA, J. A framework to monitor clusters evolution applied to economy and finance problems. **Intelligent Data Analysis**, IOS Press, v. 16, n. 1, p. 93–111, 2012. Citado nas páginas 34 e 59.

OLIVEIRA, M. B.; GAMA, J. MEC - monitoring clusters' transitions. In: **Proceedings of the Fifth Starting Artificial Intelligence Researchers' Symposium**. Lisbon, Portugal: IOS Press, 2010. v. 222, p. 212–224. Citado nas páginas 31, 34, 52, 53, 58, 60 e 61.

PEREIRA, G.; MENDES, J. M. Monitoring clusters in the telecom industry. In: **New Advances in Information Systems and Technologies**. Germany: Springer, 2016, (Advances in Intelligent Systems and Computing, v. 445). p. 631–640. Citado nas páginas 31, 35, 53 e 64.

RODRIGUES, P. P.; GAMA, J. Distributed clustering of ubiquitous data streams. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, Wiley Online Library, v. 4, n. 1, p. 38–54, 2014. Citado nas páginas 32, 33 e 46.

RODRIGUES, P. P.; GAMA, J.; PEDROSO, J. Hierarchical clustering of time-series data streams. **IEEE transactions on knowledge and data engineering**, IEEE, v. 20, n. 5, p. 615–627, 2008. Citado na página 48.

RODRIGUES, P. P.; GAMA, J.; PEDROSO, J. P. ODAC: Hierarchical clustering of time series data streams. In: SIAM. **Proceedings of the 2006 Society for Industrial and Applied Mathematics International Conference on Data Mining**. Bethesda, Maryland, USA, 2006. p. 499–503. Citado na página 48.

SCHROEDER, M. **Fractals, chaos, power laws: Minutes from an infinite paradise**. New York, NY, USA: Courier Corporation, 2009. Citado na página 49.

SILVA, A. E. da; SANCHES, L. L.; FRAIDEINBERZE, A. C.; CORDEIRO, R. L. Haliteds: Fast and scalable subspace clustering for multidimensional data streams. In: SIAM. **Proceedings of the 2016 Society for Industrial and Applied Mathematics International Conference on Data Mining International Conference on Data Mining**. Miami, Florida, USA, 2016. p. 351–359. Citado na página 40.

SILVA, J. A.; FARIA, E. R.; BARROS, R. C.; HRUSCHKA, E. R.; CARVALHO, A. C. D.; GAMA, J. Data stream clustering: A survey. **Association for Computing Machinery Computing Surveys**, ACM, v. 46, n. 1, p. 13, 2013. Citado nas páginas 29, 30, 31, 33, 39, 40, 44, 46, 51, 52 e 93.

SONG, Q.; KASABOV, N. Ecm-a novel on-line, evolving clustering method and its applications. **Foundations of cognitive science**, Citeseer, p. 631–682, 2001. Citado na página 48.

SOUSA, E. P. de; TRAINA, A. J.; TRAINA, C.; FALOUTSOS, C. Measuring evolving data streams' behavior through their intrinsic dimension. **New Generation Computing**, Springer, v. 25, n. 1, p. 33–60, 2006. Citado nas páginas 40 e 49.

SPILIOPOULOU, M.; NTOUTSI, I.; THEODORIDIS, Y.; SCHULT, R. Monic: modeling and monitoring cluster transitions. In: ACM. **Proceedings of the 12th Association for Computing Machinery's Special Interest Group on Knowledge Discovery and Data Mining International Conference on Knowledge Discovery and Data Mining**. Philadelphia, Pennsylvania, USA, 2006. p. 706–711. Citado nas páginas 31, 34, 52, 53, 54, 55 e 56.

TASOULIS, D. K.; ROSS, G.; ADAMS, N. M. Visualising the cluster structure of data streams. In: SPRINGER. **International Symposium on Intelligent Data Analysis**. Ljubljana, Slovenia, 2007. p. 81–92. Citado na página 47.

TOYODA, M.; SAKURAI, Y.; ISHIKAWA, Y. Pattern discovery in data streams under the time warping distance. **The International Journal on Very Large Data Bases**, Springer, v. 22, n. 3, p. 295–318, 2013. Citado na página 44.

WAN, L.; NG, W. K.; DANG, X. H.; YU, P. S.; ZHANG, K. Density-based clustering of data streams at multiple resolutions. **ACM Transactions on Knowledge Discovery from Data**, ACM New York, NY, USA, v. 3, n. 3, p. 1–28, 2009. Citado na página 47.

WEBB, G. I.; HYDE, R.; CAO, H.; NGUYEN, H. L.; PETITJEAN, F. Characterizing concept drift. **Data Mining and Knowledge Discovery**, Springer, v. 30, n. 4, p. 964–994, 2016. Citado nas páginas 29, 41 e 46.

WIDIPUTRA, H.; PEARS, R.; KASABOV, N. Multiple time-series prediction through multiple time-series relationships profiling and clustered recurring trends. In: SPRINGER. **Pacific-asia conference on knowledge discovery and data mining**. Shenzhen, China, 2011. p. 161–172. Citado na página 48.

WITTEN, I. H.; FRANK, E. Data mining: practical machine learning tools and techniques with java implementations. **Association of Computing Machinery Sigmod Record**, ACM, v. 31, n. 1, p. 76–77, 2002. Citado na página 31.

WU, X.; ZHU, X.; WU, G.-Q.; DING, W. Data mining with big data. **Transactions on Knowledge and Data Engineering**, IEEE, v. 26, n. 1, p. 97–107, 2013. Citado nas páginas 29 e 39.

ZENG, H.-J.; HE, Q.-C.; CHEN, Z.; MA, W.-Y.; MA, J. Learning to cluster web search results. In: ACM. **Proceedings of the 27th Annual International Association for Computing Machinery's Special Interest Group on Information Retrieval Conference on Research and Development in Information Retrieval**. Sheffield, United Kingdom, 2004. p. 210–217. Citado na página 30.


ICMC
USP
SÃO CARLOS