Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciência da Computação

Aplicação de *Trace Clustering* em Ambientes Reais: um estudo de caso no judiciário brasileiro

José Lucas Correia Acioly

Trabalho de Graduação

Recife

Universidade Federal de Pernambuco
Centro de Informática

José Lucas Correia Acioly

Aplicação de *Trace Clustering* em Ambientes Reais: um estudo de caso no judiciário brasileiro

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: *Ricardo Massa Ferreira de Lima*

Recife
09 de abril de 2025

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Acioly, José Lucas Correia.
Aplicação de Trace Clustering em Ambientes Reais: um estudo de caso no judiciário brasileiro / José Lucas Correia Acioly. - Recife, 2025.
40 p : il., tab.

Orientador(a): Ricardo Massa Ferreira de Lima
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Ciências da Computação - Bacharelado, 2025.
Inclui referências.

1. Mineração de Processos. 2. Trace Clustering. 3. Análise de Desempenho. 4. Dados Reais. I. Lima, Ricardo Massa Ferreira de. (Orientação). II. Título.

000 CDD (22.ed.)

José Lucas Correia Acioly

Aplicação de Trace Clustering em Ambientes Reais: um estudo de caso no judiciário brasileiro

Monografia apresentada ao curso de Ciência da Computação, como requisito parcial para a obtenção do Título de Bacharel em Ciência da Computação, Universidade Federal de Pernambuco.

Aprovado em: 11/04/2025

BANCA EXAMINADORA

_____

Prof. Dr. Ricardo Massa Ferreira de Lima
Universidade Federal de Pernambuco

_____

Prof. Dr. Adriano Lorena Inacio de Oliveira
Universidade Federal de Pernambuco

Em memória das avós Tânia e Lúcia

Agradecimentos

A Deus, pela vida que me deu e por tudo de bom que me foi proporcionado. Na maioria das vezes, não sabemos o porquê da vida ser como ela é, por isso estudamos; para tentar entendê-la. Mas Ele alinhou tudo para que eu chegasse até onde cheguei.

Aos meus pais, Karla e Flávio, cujo apoio foi fundamental em todas as fases da minha vida, cujos conselhos e de cuja presença nos momentos mais importantes me incentivaram a perseguir meus sonhos independentemente do quão difíceis eles fossem de serem alcançados. Com garra, inteligência, sabedoria e criatividade, me guiaram por um caminho valoroso e virtuoso.

Ao meu irmão, Victor, pelo companheirismo e amizade durante minha trajetória até o presente momento; por ter enfrentado comigo vários dos desafios diários que a vida nos colocou, superando-os com bravura e determinação.

Aos demais membros de minha família, tios, tias e primos que se fizeram presentes na minha vida, estando perto em momentos de celebração ou momentos de dor.

Aos amigos João Campos, Eduardo Robalinho, Miguel Luna e Davi Mendes: grandes companheiros dos tempos de colégio e cuja amizade foi reforçada durante minha graduação. Hoje, mesmo que fisicamente distantes na maior parte do tempo, me ajudam a seguir em frente com conselhos, reflexões, ideias e motivações.

Aos amigos Artur, Matheus, Rodrigo, Gustavo, Lucas e Davi que me ajudaram a desbravar os desafios da graduação e a prosseguir em busca do futuro.

Aos colegas de pesquisa José Danilo e Sofia, ao meu co-orientador Thiago Araujo e ao meu orientador Ricardo Massa, que contribuíram para o desenvolvimento deste trabalho com várias reuniões, ideias, questionamentos e me ajudaram a colocar um ponto final na minha graduação.

Aos Colegas do Tribunal de Justiça de Pernambuco, especialmente a Raphael D'Castro, que me introduziu ao projeto de software sobre o qual este trabalho foi desenvolvido e a Juliana Neiva que me colocou ao lado de pessoas que abriram meus olhos para a imensidão de possibilidades dentro da área de tecnologia.

Aos professores que me passaram conhecimentos valiosos ou que de alguma forma me inspiraram.

À todos aqueles que, mesmo que por um instante infinitesimal, contribuíram com a pessoa que hoje sou.

À imaginação do leitor que transcenda os limites desses agradecimentos colocados em palavras.

Resumo

Trace clustering é uma técnica de pré-processamento da mineração de processos que evoluiu bastante nos últimos anos devido ao seu elevado potencial de aplicação na análise de logs de eventos complexos, cujo uso das técnicas tradicionais de mineração de processos leva à geração de modelos de processo "*spaghetti-like*" — ou seja, modelos cujo elevado número de nós e arestas compromete a interpretação do modelo. No entanto, a aplicação dessa técnica em áreas que enfrentam grande volume de dados e processos pouco estruturados, como o sistema judiciário brasileiro, carece de investigação e documentação. Assim, o objetivo deste trabalho consiste em identificar os algoritmos de *trace clustering* desenvolvidos nos últimos anos e quais algoritmos são mais eficientes para serem executados em ambientes como o do Sistema Judiciário Brasileiro.

Palavras-chave: Mineração de Processos, Trace Clustering, Análise de Desempenho, Dados Reais

Abstract

Trace clustering is a preprocessing technique in process mining that has significantly evolved in recent years due to its high potential for analyzing complex event logs. Traditional process mining techniques often lead to the generation of "spaghetti-like" process models — that is, models with a high number of nodes and edges, which hinders their interpretability. However, the application of this technique in domains characterized by large volumes of data and poorly structured processes, such as the Brazilian judicial system, still lacks sufficient investigation and documentation. Therefore, the objective of this work is to identify the trace clustering algorithms developed in recent years and to determine which ones are most efficient for execution in environments such as the Brazilian Judicial System.

Keywords: Process Mining, Trace Clustering, Performance Analysis, Real-World Data

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A mineração de processos (mineração de processos) tem se destacado como uma ferramenta poderosa para a análise de dados de processos, extraindo conhecimento valioso a partir de *event logs* [1]. No entanto, a aplicação de técnicas tradicionais de mineração de processos em ambientes complexos, como o sistema judiciário brasileiro, frequentemente resulta em modelos de processo difíceis de interpretar [2, 3]. A complexidade e a baixa estruturação dos processos, características marcantes do judiciário [2, 3, 4], contribuem para a geração de modelos *spaghetti-like*, como o da Figura 1.1, que se tornam um obstáculo para a compreensão e otimização dos fluxos de trabalho [4, 5].

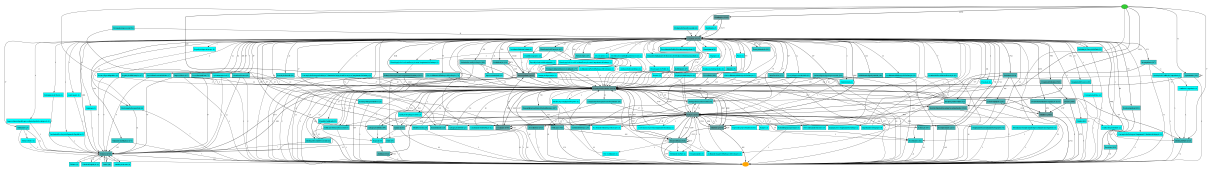

Figura 1.1 Exemplo de modelo *spaghetti* derivado de um *event log* de um tribunal brasileiro

Inicialmente, os estudos relacionados à aplicação de mineração de processos no contexto do judiciário concentraram-se na utilização de algoritmos clássicos como o *Heuristic Miner* e o *Alpha Miner* [1], que, embora eficientes na descoberta de modelos, frequentemente resultavam em representações densas e de difícil leitura [2]. Para mitigar esse problema, técnicas de simplificação de modelos foram incorporadas, como *conflict resolution*, *edge filtering*, remoção de conexões entre atividades e agregação/abstração, que permitiram não apenas a redução da complexidade visual dos modelos, mas também a identificação de agrupamentos de processos com padrões semelhantes [2].

Nesse cenário, o *trace clustering* — uma técnica de pré-processamento em mineração de processos — surge como uma solução promissora ainda pouco explorada na literatura, especialmente no contexto do judiciário. Essa técnica visa agrupar *traces* (sequências de eventos) em subconjuntos homogêneos com base em suas características, permitindo a criação de modelos de processo mais específicos e compreensíveis para cada *cluster* [5]. Assim, a aplicação do *trace clustering* possibilita a identificação de variantes de processo, descoberta de padrões de comportamento e análise de grupos de casos com características semelhantes [4, 5].

Ainda assim, a relevância da pesquisa sobre *trace clustering* no sistema judiciário brasileiro se justifica pela carência de investigação e de documentação sobre a aplicação dessa técnica em um contexto marcado por grande volume de dados e por processos

pouco estruturados [3]. Considerando que atualmente, no Brasil, existem mais de 40 mil unidades judiciárias em funcionamento, cada qual com seu fluxo de trabalho, e que o *log* de eventos de cada unidade é processado em um servidor centralizado, a análise de diferentes algoritmos de *trace clustering* e a comparação de seus desempenhos em termos de tempo de execução são cruciais para determinar a viabilidade da técnica neste domínio.

Assim, este trabalho visa suprir essa lacuna [4, 5], investigando a viabilidade de diferentes algoritmos de *trace clustering*, considerando a performance desses algoritmos para a aplicação em *event logs* do judiciário brasileiro. Adicionalmente, serão considerados os desafios específicos da aplicação de *trace clustering* nesse contexto, como a necessidade de algoritmos otimizados para serem executados em ambientes produtivos com processamento de múltiplos *event logs*, oriundos de diferentes unidades judiciais. Os resultados esperados incluem a identificação do estado da arte de algoritmos de *trace clustering* e algoritmos viáveis para a execução em ambientes como os do judiciário brasileiro.

Capítulo 2

Revisão Sistemática da Literatura

Para identificar os algoritmos candidatos que farão parte da análise de desempenho, foi realizada uma pesquisa prévia de Revisão Sistemática da Literatura, de acordo com a metodologia descrita por Carrera-Rivera, A. et al. [6], e apoiada por [7, 8]. Assim, para analisar e sintetizar os estudos existentes sobre os algoritmos de *trace clustering* no estado da arte da Mineração de Processos, a metodologia de Revisão Sistemática da Literatura proposta garante uma análise abrangente e rigorosa da literatura, minimizando vieses e aumentando a confiabilidade dos achados [7]. Conforme descrito, o processo de Revisão Sistemática da Literatura foi conduzido em duas etapas: Planejamento e Condução.

2.1 Metodologia

2.1.1 Planejamento

A primeira etapa da Revisão Sistemática da Literatura é o planejamento, que inclui a definição do PICOC (acrônimo em inglês para População, Intervenção, Comparação, Resultado e Contexto), a escolha de palavras-chave e seus sinônimos para a formulação da consulta de busca, além da definição das questões de pesquisa [6]. A População e a Intervenção são claramente representadas por *"process mining"* e *"trace clustering"*, respectivamente. Por outro lado, Petersen, K. et al. [6] argumenta que palavras-chave relacionadas à comparação, resultado e contexto podem limitar os resultados da pesquisa e devem ser utilizadas apenas caso a consulta retorne um grande número de resultados irrelevantes para o tema.

De fato, apenas palavras-chave relacionadas a "resultado"foram utilizadas no desenvolvimento da pesquisa, uma vez que foi identificado que, em primeira análise, este estudo não se propõe a comparar dois ou mais algoritmos, mas sim identificar os mais estudados e os recém-propostos. Além disso, o contexto desejado, como *"big data"* e termos correlatos, mostrou-se restritivo demais, reduzindo excessivamente o número de resultados. Dessa forma, o resultado esperado dos estudos consultados deveria ser um algoritmo ou qualquer sinônimo relacionado. A Tabela 2.1 sintetiza as palavras-chave identificadas e utilizadas durante a condução da pesquisa.

Após a definição das palavras-chave, e tendo em vista os objetivos da pesquisa, de identificar soluções eficientes baseadas em *trace clustering* que solucionem o problema dos modelos *spaghetti* derivados da aplicação direta de algoritmos clássicos da mineração de processos sobre os *event logs* relativos à processos ; as questões de pesquisa foram, então, elencadas:

- Quais são os algoritmos de *trace clustering* mais citados na literatura?
- Como as diferentes técnicas algorítmicas relacionadas ao *trace clustering* podem ser classificadas?
- Quais técnicas ou algoritmos podem apresentar um melhor desempenho em um ambiente real de *big data*?

Tabela 2.1 Palavras-chave PICOC

PICOC criteria	Word	Synonyms
Population	p*process mining*	–
Intervention	*trace clustering*	*event log clustering*
Comparison	–	–
Outcome	algorithm	pipeline, methodology, method, technique, approach, heuristic, framework, procedure
Context	*big data*	complex environment, high-volume data, massive dataset

2.1.2 Definição de Repositórios, Restrições e *Checklist* de Avaliação da Qualidade

Outro passo importante do planejamento é a definição dos repositórios de dados de onde os estudos serão extraídos. As bibliotecas eletrônicas ACM Digital Library, IEEE Xplore e Scopus foram escolhidas como repositórios de origem por serem amplamente utilizadas, possuírem um volume consolidado de estudos na área de pesquisa em Ciência da Computação e por oferecerem uma API robusta para consultas avançadas. Ainda assim, é importante mencionar que, por mais que o repositório digital da Scopus indexe artigos dos demais repositórios mencionados, nem todos os artigos desses repositórios foram completamente indexados na Scopus, de forma que a não inclusão desses repositórios no contexto de pesquisa torna-se importante para encontrar outros artigos relevantes publicados na área.

Em seguida, para evitar vieses na seleção dos estudos primários [6], foram definidos critérios de inclusão e exclusão, os quais foram refinados ao longo do processo de revisão.

2.1.2.1 Critérios de Inclusão

- Trabalhos publicados entre 2009 e 2024;
- Trabalhos escritos em inglês;
- Trabalhos de pesquisa que abordam diretamente o tema de *trace clustering*;
- Trabalhos de pesquisa publicados em periódicos, conferências ou teses.

2.1.2.2 Critérios de Exclusão

- Trabalhos que sejam revisões de literatura;
- Trabalhos que não discutam um ou mais algoritmos relacionados;

2.1.2.3 *Checklist* de Avaliação da Qualidade

Por fim, é necessário definir um *Checklist* de Avaliação da Qualidade para filtrar os estudos encontrados e manter apenas aqueles com maior relevância. As perguntas definidas para esse propósito incluem:

a) O objetivo principal da pesquisa está relacionado a *trace clustering*?

b) O artigo apresenta um problema de pesquisa bem definido?

c) Os algoritmos utilizados para *trace clustering* são claramente descritos?

d) A metodologia para aplicação do *trace clustering* está claramente definida?

e) O artigo compara o algoritmo proposto, ou estudado, com outros algoritmos de *trace clustering*?

f) O artigo fornece detalhes suficientes para a replicação dos experimentos?

g) O artigo passou por revisão por pares?

2.2 Condução da Revisão Sistemática

Conforme mostrado na Tabela 2.2, as *strings* de pesquisa foram utilizadas nas bibliotecas eletrônicas específicas, e os resultados primários foram coletados.

Os artigos identificados passaram, então, por um processo de remoção de duplicatas e posterior seleção baseado nos critérios de inclusão e exclusão previamente definidos. A primeira etapa dessa seleção foi a remoção de artigos duplicados entre as bases. Em seguida, foram realizadas as seguintes etapas:

a) Leitura dos títulos e resumos para eliminação de artigos irrelevantes;

b) Avaliação dos artigos restantes com base na introdução e conclusão;

c) Aplicação do *checklist* de qualidade para garantir a relevância e contribuição do estudo.

Após o processo de seleção, restou um conjunto de 93 artigos que foram analisados em profundidade para extrair as informações necessárias à resposta das questões de pesquisa. A Figura 2.1 mostra a distribuição temporal dos artigos selecionados de acordo com suas

datas de publicação, cuja curva evidencia um crescente interesse pelo tema. A Figura 2.2, por sua vez, retrata os artigos mais citados.

Tabela 2.2 Detalhes da consulta e número de artigos recuperados das bibliotecas digitais selecionadas

Digital Library	**Search String**	**Nº of results**
SCOPUS	TITLE-ABS-KEY ("process mining") AND TITLE-ABS-KEY (("*trace clustering*") OR ("event log *cluster*ing")) AND TITLE-ABS-KEY (algorithm OR pipeline OR methodology OR method OR "approach"OR heuristic OR framework OR procedure) AND SUBJAREA (comp) AND PUBYEAR > 2008 AND PUBYEAR < 2025 AND (LIMIT-TO (SUBJAREA , "COMP")) AND (EXCLUDE (DOCTYPE , "cr")) AND (LIMIT-TO (LANGUAGE , "English"))	88
IEEE Xplore	("Abstract":"process mining"OR "Index Terms":"process mining"OR "Document Title": "process mining") AND (Abstract": "*trace clustering*"OR "Abstract": "event log *cluster*ing"OR "Index Terms": "*trace clustering*"OR "Index Terms": "event log *cluster*ing"OR "Document Title":"*trace clustering*"OR "Document Title": "event log *cluster*ing") AND ("Abstract":"algorithm*"OR "Abstract":"pipeline"OR "Abstract":"method*"OR "Abstract": "technique"OR "Abstract": "approach"OR "Abstract":"heuristic*"OR "Abstract": "framework"OR "Abstract": "procedure"OR "Index Terms": "algorithm*"OR "Index Terms":"pipeline"OR "Index Terms": "method*"OR "Index Terms": "technique"OR "Index Terms":"approach"OR "Index Terms": "heuristic*"OR "Index Terms": "framework"OR "Index Terms": "procedure"OR "Document Title": "algorithm"OR "Document Title":"pipeline"OR "Document Title":"method*"OR "Document Title":"technique"OR "Document Title": "approach"OR "Document Title": "heuristic*"OR "Document Title": "framework"OR "Document Title": "procedure")	19
ACM Digital Library	"query": Abstract: ("*trace clustering*") "filter": E-Publication Date: (01/01/2009 TO 12/31/2024), ACM Content: DL	3

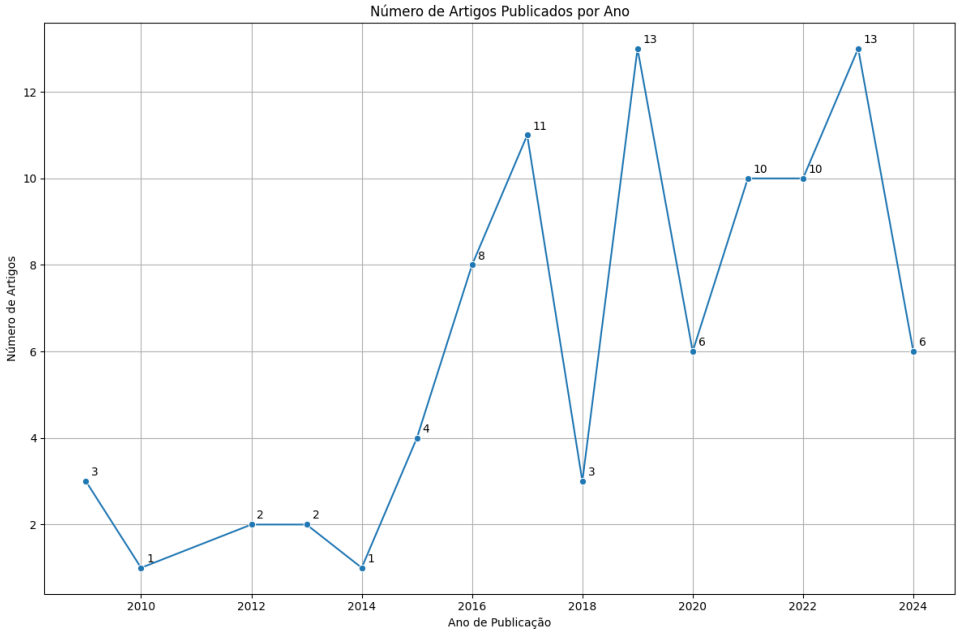

Figura 2.1 Distribuição Temporal dos Artigos Publicados

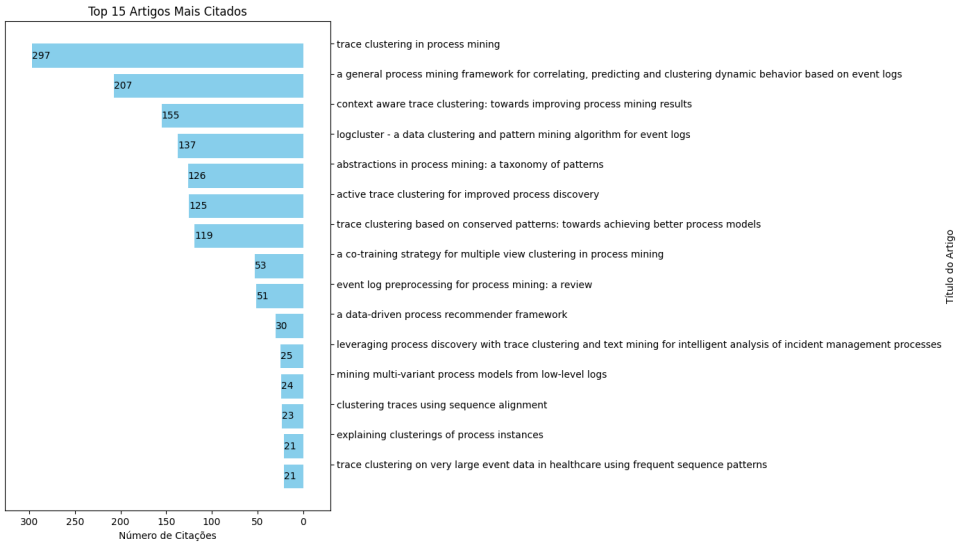

Figura 2.2 Artigos Mais Citados

2.3 Resultados

2.3.1 Desenvolvimento Histórico

Antes que qualquer palavra possa ser dita a respeito de como os algoritmos de *trace clustering* se diferenciam uns dos outros, faz-se necessária uma análise da evolução dos

algoritmos de *trace clustering*. Nessa perspectiva, a evolução dessa classe de algoritmos de mineração de processos pode ser analisada através das contribuições realizadas por diversos autores ao longo do tempo e que exploram diferentes abordagens algorítmicas para atingir diferentes propósitos.

Inicialmente, os trabalhos de Greco et al. foram pioneiros ao introduzir um modelo de espaço vetorial para representar *traces*, considerando atividades e transições, utilizando o algoritmo *k-means* para agrupamento e o conceito de *disjunctive workflow schemas* (DWS) para a descoberta de modelos de processos [9]. Eles também empregaram *n-grams* para mapear dados de eventos em vetores de características, estabelecendo as bases para futuras pesquisas.

Posteriormente, Song et al. expandiram este trabalho ao propor uma abordagem mais genérica, introduzindo o conceito de *trace profiles* [5]. Tais *profiles* representavam os *traces* sob diversas perspectivas, como atividades, transições, desempenho e atributos do caso, permitindo uma análise mais abrangente. Dessa forma, os estudos de Song et al. possibilitaram o uso de várias métricas de distância, algoritmos de *clustering*, e a exploração de técnicas de redução de dimensionalidade.

Bose e van der Aalst contribuíram significativamente ao introduzir o conceito de *context-aware trace clustering* [10], aprimorando abordagens contemporâneas e melhorando a forma como a informação de contexto de fluxo de controle era considerada. Também propuseram o uso de uma distância de edição genérica, baseada na distância de *Levenshtein*, para calcular a similaridade entre *traces*, levando em conta os custos de substituição, inserção e remoção [10], e refinaram a técnica de Greco et al. ao utilizar *conserved patterns*, como repetições máximas, supermáximas e quase supermáximas, como *features* para a projeção dos *traces* [11].

Outros autores e abordagens também enriqueceram o campo de *trace clustering*. Ferreira et al. (2007) aplicaram modelos de mistura de Markov de primeira ordem para agrupar sequências de eventos [12]. De Weerdt et al. (2013) propuseram o algoritmo ActiTraC, que busca otimizar a precisão dos modelos de processo subjacentes [13]. Appice e Malerba (2015) introduziram o Co-TraDic, uma abordagem *multi-view* baseada em *co-training* (uma técnica de aprendizado semi-supervisionado em que múltiplos classificadores são treinados de forma colaborativa a partir de diferentes visões dos dados) que utiliza *trace profiles* definidos por Song et al. [14]. Evermann et al. (2016) exploraram o alinhamento de sequências para *trace clustering*, buscando aproximar o *clustering* da avaliação [15, 16]. Delias et al. (2015) desenvolveram uma abordagem de *spectral learning* (técnica de clusterização baseada na decomposição de matrizes de similaridade em autovalores e autovetores) mais robusta, ajustando a métrica de similaridade com base no conhecimento de especialistas. Hompes et al. (2015) utilizaram *trace clustering* para identificar casos desviantes e variantes de processos [17]. Sun et al. (2017) apresentaram o *compound trace clustering*, técnica de *trace clustering* em duas etapas, que inicialmente agrupa os *traces* com base na redução da complexidade média dos modelos resultantes, e posteriormente refina os agrupamentos para maximizar a acurácia individual de cada submodelo [18]. Jablonski et al. (2019) sugeriram que considerar diferentes perspectivas é crucial para melhorar a qualidade e conformidade dos processos [19].

Fang et al. (2023) propuseram uma abordagem para descoberta de variantes de processo baseada em uma árvore de contexto de traces, estrutura inspirada em árvores de padrões frequentes, na qual cada nó representa uma atividade e suas relações de precedência. Essa técnica constrói uma árvore a partir de prefixos compartilhados entre os traces, preservando a semântica contextual e os relacionamentos entre atividades. A partir dessa árvore, agrupamentos de traces semanticamente similares são extraídos e utilizados para identificar variantes configuráveis de processos, mesmo quando apresentam comportamentos semelhantes com pequenas diferenças estruturais [20].

Nos estudos mais recentes, pode-se observar um interesse crescente em abordagens cada vez mais sofisticadas, especializadas e que consideram aspectos como inserção de conhecimento de especialistas de domínio e qualidade do modelo gerado. De Koninck et al. (2021) exploraram o *expert-driven trace clustering* com restrições a nível de instância, demonstrando como o conhecimento especializado pode influenciar o processo de *clustering* [21]. Vários autores trabalharam em técnicas para melhorar a qualidade e a precisão dos modelos de processo através do uso de *trace clustering*. Tang et al. (2022), por exemplo, desenvolveram um método de *genetic service mining* híbrido baseado em *trace clustering* e população, utilizando *alignment log* para guiar a operação de mutação e acelerar a convergência do algoritmo genético, visando melhorar a qualidade do modelo [22].

Outro ponto crucial, porém pouco explorado em comparação com os demais interesses de pesquisa, é a otimização da escolha de hiperparâmetros de um algoritmo de *trace clustering*. As pesquisas mais recentes exploram a combinação de *AutoML* (conjunto de técnicas e ferramentas projetadas para automatizar o processo de construção e otimização de modelos de aprendizado de máquina) com *trace clustering*, para automatizar a seleção de algoritmos de *clustering*, técnicas de *encoding* e seus hiperparâmetros. Tavares et al. (2022), por exemplo, apresentaram um estudo sobre a seleção de pipelines de *trace clustering* usando *meta-learning* [23]. Já, Grigore et al. (2024), usaram o *TPOT* (*Tree-based Pipeline Optimization Tool*) para otimizar pipelines de *trace clustering*, considerando diferentes codificações, pré-processamentos, algoritmos de *clustering* e seus hiperparâmetros [24].

2.3.2 Algoritmos e Classificações

Como a análise dos artigos revela, existem várias formas de classificar os algoritmos de *trace clustering* desenvolvidos até agora. Cada categoria explora um conjunto de características diferentes evidenciadas nos algoritmos. Cada categoria está intrinsecamente relacionada às capacidades algorítmicas, características dos dados, complexidade computacional e integração de informações adicionais [25]. Assim, os algoritmos podem ser classificados da seguinte forma:

2.3.2.1 Baseado na Representação dos *Traces*

a) ***Vector Space Model*** – Cada *trace* é representado como um vetor de suas características embutidas, que podem ser derivadas por meio de uma agregação dos valores de uma característica, por exemplo, a frequência de uma atividade. O vetor derivado é,

então, uma representação abstrata matemática do *trace*. Posteriormente, essa visão serve como base para calcular uma medida de distância entre os *traces*. A literatura consolidada sobre o tema se refere a essas representações matemáticas como *profiles* [5, 13, 14], e os nos artigos reunidos, os seguintes profiles foram mapeados:

- *Activity profile* - Representa cada *trace* como um vetor de frequência das atividades realizadas no *trace* [5].
- *Transition Profile* - Representa cada *trace* como um vetor de frequência das transições entre diferentes atividades do *trace* [5].
- *N-grams* - Representa cada *trace* como um vetor de frequência de N eventos consecutivos que ocorrem no *trace* [16].
- *MRA Profile* - Representa cada *trace* com base nos alfabetos de repetições máximas, ou seja, as combinações únicas de eventos que se repetem em todo o *log* de eventos, após a filtragem das repetições que são atividades em si [4].
- *Resource profile* - Representa cada *trace* como um vetor de frequência dos recursos (ou pessoas) envolvidos na execução das atividades de um *trace*, representando a distribuição de recursos ao longo do processo [18].
- *Performance Profile* - Representa cada *trace* com base em medidas de desempenho, como o tempo de execução ou a duração das atividades dentro de um processo. Essas informações ajudam a analisar a eficiência do processo em relação a tempo ou recursos [5].
- *Case Attributes Profile* – Representa os traces com base em atributos de dados associados ao caso, como informações adicionais ou meta-dados relacionados ao processo em questão. Em muitos cenários práticos, os *traces* são anotados com meta-informações que podem ser comparadas por meio deste perfil, permitindo analisar diferentes aspectos do caso em questão [5].
- *Event Attributes Profile* – Representa os traces com base em atributos de dados associados a todos os eventos presentes no *log*. Os valores dos itens são medidos de acordo com a quantidade de eventos em um *trace* que são anotados com o respectivo atributo. Este perfil pode capturar a similaridade entre os *traces* ao comparar a meta-informação dos eventos que os compõem [5].

b) ***Concrete Trace Representation*** – Não há transformação no espaço vetorial dos *traces*, ou seja, são utilizadas as sequências originais de cada *trace* para o cálculo da similaridade. A similaridade entre os *traces* é, então, derivada pela avaliação de métricas padrão de distância de strings, como:

- *Levenshtein edit distance*
- *Hamming distance*
- *Jaro-winkler-distance*
- *Damerau-levenshtein distance*

2.3.2.2 Baseado na Estratégia de *Clustering*

a) ***Distance Based Trace Clustering*** – Utilizam métricas de distância para calcular a similaridade entre os *traces* e agrupá-los com base nessa similaridade. Assim, diferentes algoritmos de *clustering* foram identificados como parte do processo de *trace clustering* [4, 5]. Exemplos incluem:

- *K-Means* – Algoritmo de clustering baseado em centroides, onde os dados são divididos em k clusters, minimizando a soma das distâncias quadráticas entre os pontos e os seus respectivos centroides. O número de clusters k deve ser especificado previamente [26].
- *K-Medoids* – Semelhante ao K-Means, mas, em vez de usar centroides, o K-Medoids escolhe um ponto real do conjunto de dados como o centro de cada cluster. Esse algoritmo é mais robusto a outliers, já que utiliza pontos reais como representantes dos clusters [26].
- *Agglomerative Hierarchical Clustering (AHC)* – Método hierárquico de clustering que começa com cada ponto como um cluster separado e, em seguida, agrupa os clusters mais próximos em cada etapa, formando uma árvore hierárquica (dendrograma) de clusters [26].
- *Spectral Clustering* – Técnica que utiliza a decomposição espectral de uma matriz de similaridade para reduzir a dimensionalidade do problema e, em seguida, aplica um algoritmo como o K-Means sobre os dados transformados. É eficaz para dados que possuem formas complexas de agrupamento [26].
- *DBSCAN* – Algoritmo de clustering baseado em densidade, que agrupa pontos em regiões de alta densidade e trata como outliers os pontos que estão em regiões de baixa densidade. Não requer o número de clusters ser especificado previamente e pode encontrar clusters de formas arbitrárias [26].
- *MR-DBSCAN* – Extensão do DBSCAN que combina a ideia de repetições máximas (Maximal Repeats) para melhorar o desempenho na identificação de clusters em logs de eventos, especialmente em dados temporais ou sequenciais [26].

As seguintes métricas de distância foram destacadas nos estudos coletados:

- *Euclidean distance*
- *Manhattan distance*
- *Minkowski Distance*
- *Jaccard distance*
- *Hamming distance*
- *Cosine Similarity*
- *Levenshtein Edit Distance*

b) ***Model Based Trace Clustering*** – Agrupa *traces* com base em modelos de processo subjacentes, como redes de Petri ou cadeias de Markov, em vez de calcular diretamente a distância entre os *traces* [27]. O objetivo é otimizar a qualidade dos modelos de processo descobertos para cada *cluster*, agrupando *traces* que levam a modelos mais precisos e compreensíveis. Os principais modelos explorados incluem:

- *First Order Markov Chains*
- *Markov Mixtures*
- *Predictive cluster Trees (PCTs)*
- *Petri Nets*
- *Dependency graphs*

c) ***Hybrid Trace Clustering*** – Esses algoritmos combinam abordagens baseadas em distância com técnicas de modelagem para orientar a formação dos *clusters*. Eles definem um modelo estrutural de como agrupar, mas ainda necessitam de uma métrica de distância.

2.3.2.3 Baseado na Incorporação de Conhecimento de Domínio

a) ***Supervised Clustering*** – Incorporam conhecimento de especialistas, na forma de condições de contorno ou modelos pré-definidos, para orientar o processo de *cluster*ização [21].

b) ***Unsupervised Clustering*** – Utilizam apenas os dados do *event log* para gerar os *clusters* [21].

2.3.2.4 Baseado no Objetivo do *Clustering*

a) **Redução da Complexidade do Modelo** – Visam dividir o *event log* em *sublogs* mais coesos, resultando em modelos de processo mais simples e fáceis de entender.

b) **Identificação de Variantes Processuais** – Buscam identificar grupos de *traces* que representam diferentes variantes do processo, ou seja, grupos de *traces* que representam diferentes maneiras de executar o processo.

c) **Melhoria da Qualidade do Modelo** – Concentram-se na otimização de métricas como *fitness* (quão bem o modelo pode reproduzir os traces no *log*) e *precision* (quão bem o modelo evita comportamentos não observados no *log*) dos modelos de processo descobertos a partir dos grupos de *traces*.

2.3.2.5 Outras Abordagens de Classificação

a) **Abordagem Compensatória vs. Não Compensatória** – Além das abordagens tradicionais baseadas em distância e orientadas por modelo, o *trace clustering* pode ser realizado considerando critérios adicionais que enriquecem a análise e permitem

a formação de *cluster*s mais informativos e relevantes para o contexto do processo. Essas abordagens se beneficiam da disponibilidade de dados de contexto nos *event logs*, indo além da simples sequência de atividades [10].

- *Compensatory Approach* – Permite que haja compensação de similaridade entre *traces* quando considerados diferentes aspectos (critérios). Por exemplo, um algoritmo com uma abordagem compensatória permite que uma alta similaridade na sequência de atividades possa compensar uma grande diferença na duração dos eventos. Isso pode levar a *cluster*s de *traces* que são considerados similares globalmente, mas diferem significativamente em critérios individuais [10].
- *Non-Compensatory Approach* – Avalia a similaridade considerando múltiplos critérios separadamente, utilizando conceitos como limiares de discriminação e "poder de veto". Não permite que uma alta similaridade em certos critérios compense completamente uma grande dissimilaridade em outros. O "poder de veto"garante que se *traces* forem significativamente diferentes em um critério importante, eles não serão considerados similares, mesmo que sejam semelhantes em outros aspectos. Essa abordagem busca agrupar *traces* que são consistentemente similares em vários critérios, evitando compensações que poderiam obscurecer diferenças importantes [10].

b) **Múltiplas Perspectivas** – Consideram diferentes perspectivas dos dados do *event log*. Atributos extras, que vão além da perspectiva de fluxo de controle (*case identifiers* e *sequence of activity identifiers*), como:

- **Aspectos organizacionais** – Podem ser informações sobre os recursos envolvidos em cada atividade, como o departamento, o usuário ou o papel.
  Exemplo: No processo de compra, saber qual departamento ou funcionário aprovou o pagamento adiciona uma perspectiva organizacional.
- **Dados de caso** – Atributos adicionais associados aos casos, como o tipo de pedido, a data de início ou o valor do pedido.
  Exemplo: No processo de compra, o valor total do pedido, o tipo de frete escolhido ou o *CEP* de entrega são dados de caso que podem ser utilizados.
- **Dados de desempenho** – Métricas de desempenho, como o tempo de execução de cada atividade, o tempo total de processamento do caso ou o custo total.
  Exemplo: No processo de compra, o tempo decorrido entre cada etapa do processo ("adicionar ao carrinho"– "efetuar pagamento"), ou o tempo total de processamento do pedido são métricas de desempenho relevantes.

A Tabela 2.3 sintetiza a classificação dos Algoritmos de *trace clustering* identificados durante a revisão da literatura.

Tabela 2.3 Comparação de Algoritmos de *trace clustering*

Algoritmo	Representação de Traços	Estratégia de *clustering*	Incorporação de Conhecimento de Domínio	Objetivos de Clustering	Abordagem Compensatória vs. Não Compensatória	Suporte a Múltiplas Perspectivas	Descrição Sintética	Ano Publicação	Artigo
Sequence clustering	Representação Concreta de Traços	Baseado em Modelo - Hidden Markov Model	Não supervisionado	Redução da Complexidade do Modelo, Identificação de Variantes	Compensatória	Pode considerar diferentes ordens de modelos de Markov	Agrupa traços com base na probabilidade de serem gerados por um modelo de Markov.	2007	[12]
3-gram	Modelo de Espaço Vetorial	Baseado em Distância	Não supervisionado	Redução da Complexidade do Modelo, Identificação de Variantes	Compensatória	Pode considerar incorporação de dados de contexto no *profile*	Utiliza sequências de 3 atividades (*trigrams*) como características para representar os traços.	2009	[5]
GED (Generic Edit Distance)	Representação Concreta de Traços	Baseado em Distância	Não supervisionado	Redução da Complexidade do Modelo, Identificação de Variantes	Compensatória	Pode incorporar custos de edição específicos do contexto	Calcula a distância entre os traços com base no número de operações de edição necessárias para transformá-los.	2009	[11]
MR (Maximal Repeat)	Modelo de Espaço Vetorial	Baseado em Distância	Não supervisionado	Redução da Complexidade do Modelo, Identificação de Variantes	Compensatória	Pode considerar incorporação de dados de contexto no *profile*	Concentra-se na deteção de padrões repetidos ("*repeats*") dentro das sequências de atividades nos traces de eventos.	2010	[4]
MRA (Maximal Repeat Alphabet)	Modelo de Espaço Vetorial	Baseado em Distância	Não supervisionado	Redução da Complexidade do Modelo, Identificação de Variantes	Compensatória	Pode considerar incorporação de dados de contexto no *profile*	Uma extensão do MR que considera o alfabeto completo de atividades.	2010	[4]
SemSup-MRA	Modelo de Espaço Vetorial	Híbrido	Semi-supervisionado	Melhoria da Qualidade do Modelo	Compensatória	Incorpora dados de contexto no *profile* e diferentes métricas de qualidade	Utiliza uma estratégia de aprendizado semi-supervisionado, combinando amostragem seletiva com otimização da qualidade do modelo para cada *cluster*. Baseia-se na técnica MRA para calcular a distância entre os traços.	2012	[28]
ActiTraC	Representação Concreta de Traços	Baseado em Modelo - Predictive *cluster* Trees (PCTs)	Não supervisionado	Melhoria da Qualidade do Modelo	Compensatória	Pode considerar diferentes métricas de qualidade do modelo	Utiliza uma abordagem de aprendizado ativo para otimizar a qualidade do modelo de processo.	2013	[13]
OT (Optimal Top-down)	Modelo de Espaço Vetorial	Baseado em Distância	Não supervisionado	Redução da Complexidade do Modelo, Melhoria da Qualidade do Modelo	Compensatória	Pode considerar incorporação de dados de contexto no *profile* e diferentes métricas de qualidade	Emprega uma estratégia gulosa para encontrar a melhor maneira de agrupar os traços com base em um esquema de avaliação de modelo específico que considera a qualidade e a complexidade dos modelos de subprocesso potenciais durante o tempo de execução.	2015	[29]
CoTraDiC	Modelo de Espaço Vetorial	Híbrido	Não supervisionado	Redução da Complexidade do Modelo	Compensatória	Integra múltiplas perspectivas por meio do *co-training*	Usa o *co-training* para integrar diferentes *profiles* de traços e gerar um *clustering* de consenso.	2016	[14]
Compound Trace clustering (CTC)	Modelo de Espaço Vetorial	Híbrido	Não supervisionado	Redução da Complexidade do Modelo, Melhoria da Qualidade do Modelo	Compensatória	Pode considerar incorporação de dados de contexto no profile e diferentes métricas de qualidade	Combina *clustering* baseado em distância e técnicas de otimização da qualidade do modelo.	2017	[18]
ActSemSup	Representação Concreta de Traços	Baseado em distância	Semi-supervisionado	Melhoria da Qualidade	Não Compensatória	Informação não disponível nas fontes	Utiliza *domain expert knowledge* para guiar o *clustering* semi-supervisionado, focando na melhoria da qualidade dos modelos de processo.	2017	[30]
NoHiC (Novel Hierarchical clustering)	Modelo de Espaço Vetorial (atributos relacionados à lógica de negócios)	Híbrido	Supervisionado	Redução da Complexidade do Modelo	Não Compensatória	Informação não disponível nas fontes	O algoritmo NoHiC é uma abordagem em múltiplas etapas que combina mineração baseada em regras com agrupamento hierárquico.	2020	[31]
Trace clustering with SGT	Modelo de Espaço Vetorial (Sequence Graph Transform - SGT)	Baseado em Distância	Não supervisionado	Redução da Complexidade do Modelo, Identificação de Variantes	Informação não disponível nas fontes	Informação não disponível nas fontes	Utiliza SGT para extrair características de sequências de eventos. Agrupa traces com base nessas características, com o objetivo de identificar variantes de processos.	2020	[23]
ConDriTraC	Representação Concreta de Traços	Baseado em Modelo - Petri Nets	Supervisionado	Melhoria da Qualidade do Modelo	Não Compensatória	Pode considerar diferentes métricas de qualidade do modelo	Incorpora conhecimento especializado por meio de restrições de nível de instância.	2021	[32]
SemS (Semantic alpha Splitting)	Árvore de Contexto	Baseado em Similaridade - *Weighted Frequency Cosine Similarity*	Não supervisionado	Identificação de Variantes	Compensatória na construção da árvore; Não compensatória na seleção de *clusters*	-	Constrói uma árvore de contexto a partir de um *log* de eventos e utiliza a similaridade de contexto para agrupar traços e descobrir variantes.	2023	[20]
ClusRepa (cluster-based Repair)	Modelo de Espaço Vetorial - Activity Profile	Baseado em Distância - Spectral Clustering	Não supervisionado	Reparação de Traços Faltantes	Compensatória na fase de *clustering*; Não compensatória na seleção do traço reparado (maior probabilidade)	Considera diferentes comprimentos de contexto	Agrupa traços completos usando *Spectral Clustering* e repara traços faltantes com base na probabilidade contextual do *cluster* mais similar.	2024	[17]

2.3.3 *Frameworks* e *Pipelines*

Como já foi mencionado, o objetivo principal da técnica de *trace clustering* é agrupar o *log* de eventos em sub-*logs* mais coesos [4, 25], permitindo a descoberta de modelos menos complexos e até mesmo variantes processuais desconhecidas. Como uma técnica de mineração de processos, não apenas algoritmos específicos foram propostos para resolver a questão, mas também *frameworks* que orientam o desenvolvimento de um algoritmo de *trace clustering*, estabelecendo as características principais que devem ser levadas em consideração [5, 25]. Assim, como subproduto da *SLR* realizada, foi identificado que alguns autores propuseram *frameworks* para descrever o *pipeline* de um algoritmo genérico de *trace clustering*. Dessa forma, conforme encontrado na literatura, um *Framework* de *trace clustering* fornece um guia passo a passo para aplicar efetivamente a técnica [25]. As etapas incluídas nesses *frameworks* geralmente incluem:

2.3.3.1 Pré-processamento do *Log* de Eventos

Uma etapa inicial que envolve a aplicação de transformações de dados no *log* de eventos para prepará-lo para as etapas seguintes. As transformações podem envolver limpeza de dados, seleção de características relevantes do *log* de eventos e até mesmo uma transformação de dados para um formato mais adequado [5, 25, 33]. Alguns *frameworks* podem incluir uma etapa de decomposição do *log* de eventos para lidar com um grande volume de dados [33].

2.3.3.2 Extração de Características

Esta é uma etapa opcional, pois os traços podem ser usados em sua representação concreta [13, 34, 32], onde os traços são convertidos em uma representação numérica, também conhecida como vetor de características, capturando o comportamento dos traços. [5] descreve como alcançar essas representações em espaço vetorial aplicando o que ele chama de *trace profiles*.

2.3.3.3 Seleção do Algoritmo de *Clusterização*

Aqui podemos considerar uma ampla variedade de técnicas de *clusterização*, dependendo de como o *log* de eventos foi pré-processado ou até mesmo dos objetivos do estudo. As abordagens mais específicas foram discutidas na seção anterior, dividindo as técnicas em três grupos: as baseadas em distância, as baseadas em modelos e as híbridas. Alguns algoritmos comuns que podem ser incluídos são *k-means*, *Hierarchical clustering*, *DBSCAN* e *Spectral clustering* [5, 35].

A escolha do algoritmo de *clusterização* também tem a ver com os objetivos da *clusterização*. Dependendo dos objetivos da *clusterização*, ou seja, descoberta de modelos de processos, identificação de variantes, redução de complexidade ou reparo de traços, o algoritmo de *clusterização* pode ser modificado para alcançar melhor esses objetivos. Além disso, a escolha também pode considerar se o conhecimento especializado do domínio pode ser adicionado à lógica do algoritmo [32].

Por fim, a escolha do algoritmo de *clusterização* pode também ser guiada por aspectos de qualidade dos dados, como a presença de *outliers* e ruído.

2.3.3.4 Aplicação do Algoritmo de *Clusterização*

Etapa em que o algoritmo de *clusterização* selecionado é aplicado aos vetores de características para agrupar os traços em *clusters*.

2.3.3.5 Avaliação dos *Clusters*

Etapa em que a qualidade da *clusterização* pode ser avaliada usando métricas como similaridade intra-*cluster* e dissimilaridade inter-*cluster* e a qualidade dos modelos de processos descobertos para cada *cluster* é avaliada.

2.3.3.6 Interpretação dos Resultados

Etapa em que os modelos processuais descobertos para cada *cluster* identificado são analisados e interpretados para identificar diferentes padrões de comportamento. Os modelos de processos descobertos para cada *cluster* podem ser usados para melhorar a compreensão dos processos.

2.3.3.7 Outras Considerações

Um *framework* genérico para *clusterização* de traços também pode incorporar conhecimento do domínio, permitindo que especialistas influenciem o processo de *clusterização* [25, 32]. Algumas abordagens permitem que o conhecimento do domínio seja usado para inicializar a *clusterização*, enquanto outras permitem que o conhecimento seja incorporado na forma de restrições [32]. Além disso, alguns *frameworks* podem incluir etapas para explicar os resultados da *clusterização* de traços, tornando-os mais acessíveis aos analistas de processos.

Alguns *frameworks* também consideram a otimização de hiperparâmetros, um processo iterativo que busca encontrar os parâmetros ótimos para um algoritmo específico [36], como uma extensão do problema de *clusterização* de traços, com o objetivo de melhorar não apenas a qualidade dos *clusters* gerados, mas também sua conformidade com os requisitos da mineração de processos (PM), como a qualidade do modelo e a complexidade do *log*, além de otimizar o *pipeline* de *clusterização* para alcançar um resultado ótimo em menos tempo [24]. *Frameworks* que abordam o problema da otimização de hiperparâmetros na *clusterização* de traços até agora consideraram a aplicação de algoritmos genéticos e *meta-learning* para otimizar métricas de qualidade de *clusterização*, como o índice de silhueta e a entropia da sequência [24, 37].

Capítulo 3

Estudo Experimental

Após analisar as dimensões dos algoritmos utilizados para *trace clustering*, foi possível avaliar quais deles podem ser viáveis para aplicação em um contexto de *Big Data*. No judiciário brasileiro, encontramos mais de 40 mil unidades judiciárias, cada qual com seu fluxo de trabalho, e um servidor centralizado para processar os dados de *event log* de cada unidade. Assim, para uma aplicação de descoberta de modelo de processo que pretenda utilizar a técnica de *trace clustering* seja viável nesse cenário, é preciso que o algoritmo de *trace clustering* adicione o mínimo possível de complexidade temporal, de forma que processar os *event logs* de cada unidade utilizando a técnica de *trace clustering* não implique em um tempo de execução proibitivo ou em uma sobrecarga excessiva no servidor centralizado.

Dado que cada unidade judiciária pode gerar um volume massivo de *traces*, a escalabilidade dos algoritmos de *trace clustering* torna-se um fator crítico. Se, por acaso, executar o algoritmo de *trace clustering* para várias instâncias de *event log* em sucessão necessitar de tempo excessivo (várias horas ou dias), tal fato se tornaria um impeditivo para a implantação do algoritmo em ambiente produtivo.

Outro fator importante que pode se tornar um fator impeditivo para a implantação do algoritmo em ambiente produtivo está relacionado à escalabilidade desses algoritmos considerando o aumento da complexidade dos eventos. Dessa forma, caso a execução de um algoritmo de *trace clustering* para uma instância de *event log* necessite de tempo excessivo (vários minutos ou horas), tal fato se tornaria um impeditivo para a implantação do algoritmo em ambiente produtivo.

3.1 Metodologia

Considerando os desafios relativos à performance de tempo de execução para implementação dos algoritmos de *trace clustering* no ambiente produtivo do Sistema Judiciário Brasileiro, foram extraídos para análise de performance *event logs* de diferentes unidades judiciárias do ambiente produtivo em questão.

Considerando que a heterogeneidade dos *event logs* entre diferentes unidades judiciárias pode impactar o desempenho do algoritmo de maneira desigual, torna-se necessário avaliar a performance dos algoritmos considerando diferentes critérios de complexidade dos *event logs*. Augusto et al. [9] definem três categorias de medidas de complexidade de processos baseadas puramente na análise do *event log*: **medidas de tamanho**, que incluem medidas que quantificam as dimensões básicas do *log* de eventos, como o **número**

de eventos, **número de tipos de eventos** (atividades distintas), **número de *traces*** e o **comprimento médio dos *traces***; **medidas de variação**, que abrangem medidas que capturam a diversidade do comportamento do processo registrado no *log de eventos*, como o **número de caminhos acíclicos na matriz de transição**, o **número e percentual de *traces* únicos** e o **número médio de eventos distintos por *trace***; e m**edidas de distância**, que incluem medidas que avaliam a dissimilaridade entre as sequências de eventos no *log*, como **afinidade média**, **desvio do aleatório** e **distância de edição média**.

Além dessas medidas, Augusto et al. [9] introduzem uma medida de complexidade do processo baseada em quatro medidas de entropia, calculadas a partir de um autômato de prefixo estendido derivado do *event log*, que visam capturar aspectos de tamanho, variação e distância presentes nos *logs de eventos* de uma forma integrada. Devido à complexidade computacional extra para derivar a complexidade baseada em medidas de distância, foram consideradas apenas as medidas de tamanho, variação e entropia na avaliação da complexidade dos *logs de eventos*.

Assim, com base nesses critérios de complexidade, foram selecionados para o estudo seis *event logs* contendo dados sobre a tramitação de processos que ocorrem no sistema judiciário brasileiro. Cada log de eventos reúne dados de processos oriundos de um tribunal pertencente a uma corte específica, registrando a sequência de atividades realizadas, o momento de sua ocorrência e os identificadores dos casos e agentes envolvidos. A partir desses logs, pretende-se extrair perfis de comportamento processual, identificar padrões de fluxo de atividades e, principalmente, aplicar técnicas de trace clustering para agrupar casos semelhantes com base na sequência de eventos, possibilitando uma análise mais estruturada e interpretável do funcionamento dos tribunais.

com diferentes complexidades estruturais, cujas características relativas à complexidade são descritas na Tabela 3.1

Como pode ser visualizado na Tabela 3.1, os *event logs* foram selecionados de forma que a principal variação de complexidade decorre da variação de medidas de tamanho como número de *traces*, com incremento médio de aproximadamente 1.271 unidades, entre instâncias com valores sucessivos de número de *traces*, e desvio padrão aproximado de 271; e número de eventos, com incremento médio aproximado de 42.800 unidades entre instâncias com valores sucessivos de número de eventos. Dessa forma, na seleção dos *event logs*, foi considerada uma relação que tende a linearidade entre o aumento do número de eventos e o aumento do número de *traces*. A escolha dos *event logs* também considerou valores de número de atividades, tamanho médio dos *traces*, entropia de variante, entropia normalizada de variante e entropia normalizada de sequência praticamente constante de forma a obter o menor impacto possível de parâmetros diferentes do número de *traces* e do número de eventos sobre o tempo de execução dos algoritmos. Já o aumento da entropia de sequência entre os *event logs* selecionados está em coerência com o lema proposto por [9] que afirma que a entropia de sequência é monótona a respeito ao aumento do número de eventos.

Após a seleção do conjunto de dados, foram selecionados os algoritmos de *Trace Clustering* mais apropriados para o experimento. Inicialmente, foram analisados estudos comparativos de escalabilidade realizados anteriormente entre diferentes algoritmos de

Tabela 3.1 Medidas de Complexidade dos *event logs* analisados

Event Log	Traces	Events	Activities	Avg Trace Length	Distinct Sequences	Avg Distinct Events per Sequence	Acyclic Paths	Variant Entropy	Norm. Variant Entropy	Sequence Entropy	Norm. Sequence Entropy
Log 1	2662	88944	97	33	1930	12	44764	2.31E+11	0.509	217460.00	0.2145
Log 2	4012	136735	118	34	3671	10	85348	8.39E+11	0.8084	360246.60	0.2228
Log 3	5005	199575	127	40	4731	10	116330	1.58E+12	0.8738	469932.05	0.1929
Log 4	6760	215689	135	32	5508	9	136554	1.53E+12	0.8265	502249.74	0.1896
Log 5	7788	270544	132	35	6905	9	159435	1.84E+12	0.813	670349.46	0.1981
Log 6	9038	303132	122	34	8192	9	124962	2.24E+12	0.8339	713335.72	0.1864

trace clustering permitindo-nos situar nossa análise no contexto da literatura existente e limitar quais algoritmos mereciam investimento de tempo na análise comparativa de performance.

Alguns desses estudos incluem:

- [28] comparou o tempo de execução entre os algoritmos *SemSup-MRA* e *MRA* padrão, de forma que o primeiro levou, em média, 33 segundos para gerar um resultado, enquanto o segundo levou apenas 2 segundos nas condições propostas pelo seu experimento.

- [13] avaliou a escalabilidade do algoritmo *ActiTraC* em relação aos algoritmos *MR*,*MRA*, *GED* (*Generic Edit Distance*), *Markovian Model Based*, *Levenshtein Edit Distance*, *Activity-K-Means* e *3-grams*. Os resultados mostraram que os algoritmos *MR*, *MRA*, *GED*, *LED* (*Levenshtein Edit Distance*), *BOA* (*Bag of Activities*) e *3-grams* superaram o *ActiTraC* e o *Markovian Model Based* em termos de performance e escalabilidade.

- [27] comparou o tempo de execução dos algoritmos *TraCluSi*, *ActiTraC*, *MRA* e *3-grams*, de forma que o *TraCluSi* obteve melhor desempenho que o *MRA* e o *3-grams* no pior caso.

Em seguida, a seleção dos algoritmos para análise também considerou que algoritmos supervisionados ou semi-supervisionados não serão incluídos na análise de tempo de execução, pois sua dependência de conhecimento de domínio não é o foco deste estudo. De forma análoga, os algoritmos cujo objetivo principal não consiste em resolver o problema de identificação de variantes processuais não foram selecionados, uma vez que não estão alinhados com o objetivo principal definido nos parâmetros do experimento.

Finalmente, para garantir uma análise comparativa relevante, selecionamos algoritmos que demonstraram bom desempenho em estudos anteriores ou que não foram encontradas avaliações de performance realizadas nos artigos encontrados. Assim, a avaliação de performance incluirá os seguintes algoritmos:

- *TraCluSi*
- *MRA_K-Means*
- *3-Gram_K-Means*
- *Activity-Profile_K-Means*
- *MRA_Agglomerative-clustering*
- *3-Gram_Agglomerative-clustering*
- *Activity-Profile_Agglomerative-clustering*
- *MRA_Spectral-clustering*

- *3-Gram_Spectral-clustering*
- *Activity-Profile_Spectral-clustering*

Por fim, os algoritmos selecionados foram executados repetidamente sobre os *event logs* selecionados e para valores incrementais de número de clusters em ambiente controlado com CPU octa-core AMD Ryzen 7 Mobile 4800H e 16 GB de memória RAM, e os valores de tempo de execução de cada algoritmo foram registrados para avaliação.

3.2 Resultados e Discussão

Após a execução dos algoritmos sobre os *logs* de eventos selecionados e a coleta dos valores de *runtime*, algumas observações podem ser feitas a respeito dos algoritmos em análise. Uma primeira análise das Figuras 3.1 e 3.2 mostram uma tendência de aumento no tempo de *runtime* para todos os algoritmos à medida que a complexidade estrutural dos *event logs* aumenta. Um resultado mais relevante decorre da análise das distâncias verticais das curvas de tempo de cada algoritmo. Nessa perspectiva, decorre que o algoritmo *activity_profile_kmeans* é o mais eficiente em termos de performance, com valores de *runtime* variando entre 0,06 e 0,25 segundos considerando os *logs* de eventos avaliados.

Um segundo grupo de algoritmos com performances semelhantes, constituído dos algoritmos *3_grams_kmeans*, *activity_profile_agglomerative*, *activity_profile_spectral*, *3_grams_spectral* e *TraCluSi* (em ordem crescente de performance), pode ser verificado. Por fim, um terceiro grupo, de algoritmos menos performáticos, composto dos algoritmos *3_grams_agglomerative*, *MRA_kmeans*, *MRA_spectral* e *MRA_agglomerative* (em ordem crescente de performance), pode ser identificado.

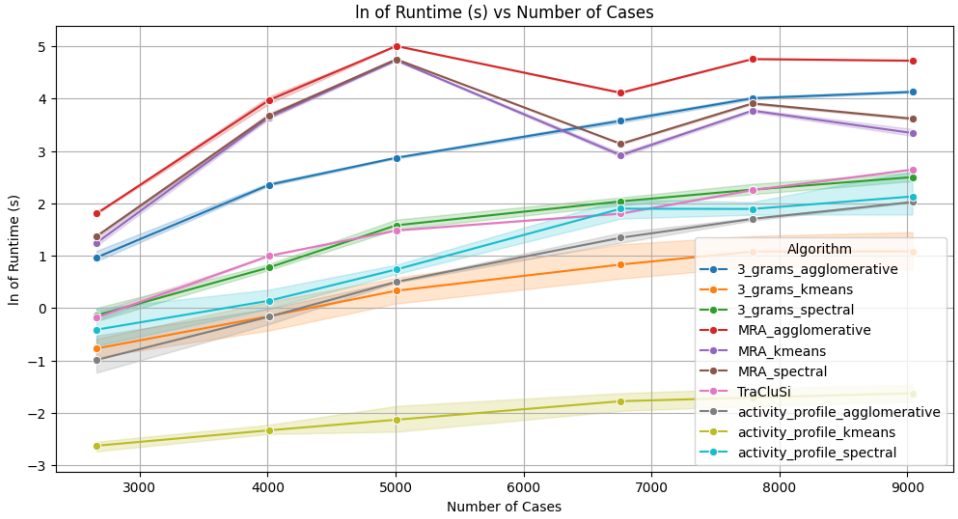

Figura 3.1 *Runtime* vs Número de Casos

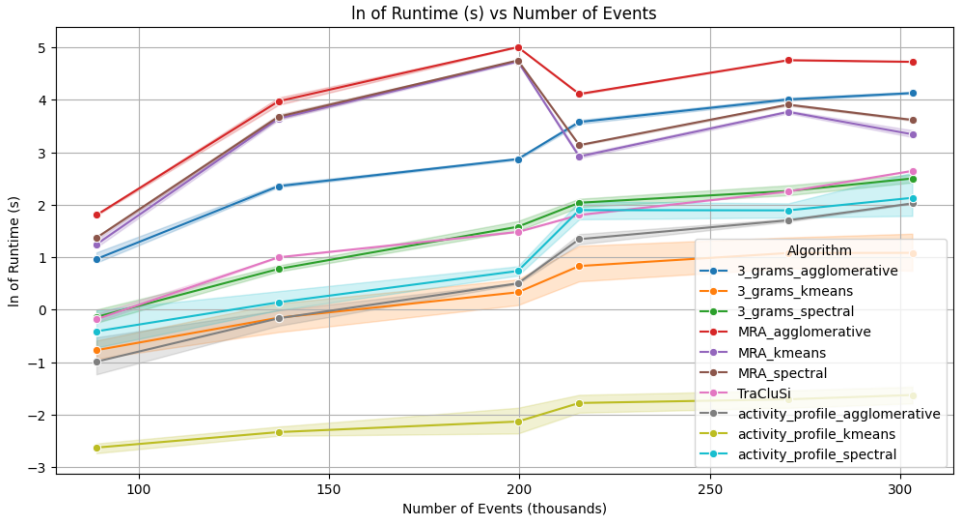

Figura 3.2 *Runtime* vs Número de Eventos

Tabela 3.2 Resultados de Tempo de Execução para os Algoritmos Analisados

Algorithm	**Min Runtime (s)**	**Max Runtime (s)**	**Avg Increment (s)**
3_grams_agglomerative	2,4317	64,4313	11,881
3_grams_kmeans	0,3798	5,0874	0,541
3_grams_spectral	0,7686	13,4875	2,268
MRA_agglomerative	5,9888	149,4619	28,581
MRA_kmeans	3,2447	115,3932	21,962
MRA_spectral	3,8735	115,8799	22,344
TraCluSi	0,7607	14,3335	2,647
activity_profile_agglomerative	0,289	7,6438	1,433
activity_profile_kmeans	**0,061**	**0,2443**	**0,025**
activity_profile_spectral	0,4843	16,7823	1,707

Um ponto importante a ser considerado na comparação da performance desses algoritmos é o comportamento das curvas de tempo versus número de casos e número de eventos dos algoritmos baseados em *MRA*. Uma análise cautelosa da Figura 3.3 permite inferir maior correlação entre o aumento do tamanho médio dos *traces* e o aumento do tempo de execução desses algoritmos. Assim, por mais que a escolha da base de dados considerasse valores muito próximos de tamanho médio dos *traces*, os algoritmos baseados em *MRA* mostraram uma variação relevante de tempo de execução mesmo para pequenas variações de tamanho médio dos *traces*.

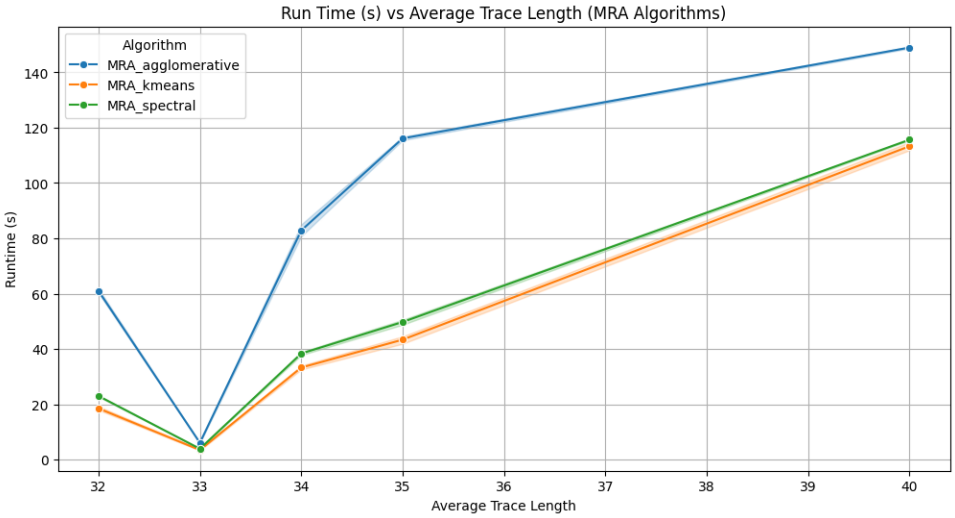

Figura 3.3 *Runtime MRA* vs Comprimento Médio dos Traces

Diante dos resultados obtidos, a viabilidade da aplicação desses algoritmos em ambientes produtivos depende não apenas da sua eficiência relativa, mas também do contexto em que serão utilizados. Dessa forma, dois cenários distintos podem ser imaginados: processamento em *batch* e execução *on-demand*. No primeiro, os *event logs* são processados periodicamente, permitindo o uso de algoritmos de maior custo computacional sem comprometer a disponibilidade do sistema. No segundo, os *logs* são analisados sob demanda, exigindo baixa latência para garantir uma boa experiência do usuário.

No cenário de processamento *batch*, os algoritmos mais eficientes, como *activity_profile_kmeans* e *3_grams_kmeans*, são altamente indicados, pois mantêm um tempo de execução reduzido mesmo para *event logs* complexos. Algoritmos de desempenho intermediário, como *TraCluSi* e *activity_profile_agglomerative*, podem ser utilizados sem impacto significativo na performance do servidor, desde que o processamento seja agendado de forma otimizada. Já os algoritmos com desempenho inferior, especialmente os baseados em *MRA*, requerem maior tempo de execução e podem gerar sobrecarga no servidor se executados em grande escala, tornando-se menos viáveis para operações frequentes.

No cenário *on-demand*, a necessidade de respostas rápidas restringe o conjunto de algoritmos viáveis. Métodos como *activity_profile_kmeans* e *3_grams_kmeans* são os mais adequados, pois apresentaram os menores tempos de execução. Por outro lado, algoritmos de maior complexidade, especialmente os baseados em *MRA* ou *agglomerative clustering*, podem comprometer a responsividade do sistema e degradar a experiência do usuário.

Ainda assim, para determinar quais algoritmos são os mais recomendados para se implementar em ambiente produtivo, outros critérios devem ser analisados em conjunto, como a qualidade dos *clusters* gerados e a qualidade do modelo processual derivado de cada *cluster*. Tais análises podem ser realizadas em estudos posteriores.

Capítulo 4

Conclusão

Este trabalho teve como objetivo analisar e comparar diferentes abordagens de *trace clustering*, considerando suas classificações, métricas de distância, eficiência computacional e viabilidade em cenários produtivos. A revisão sistemática da literatura revelou uma ampla diversidade de algoritmos e métodos, categorizados com base na representação dos *traces*, estratégia de *clustering*, incorporação de conhecimento de domínio e objetivo da *clusterização*. Dentre as abordagens identificadas, destacam-se os modelos baseados em distância, modelos baseados em representatividade estrutural e abordagens híbridas que combinam diferentes estratégias para melhorar a qualidade dos agrupamentos.

O estudo experimental demonstrou que a complexidade estrutural dos *event logs* tem impacto significativo na eficiência dos algoritmos analisados. Observou-se uma correlação direta entre o tamanho médio dos *traces* e o tempo de execução dos algoritmos, especialmente nos algoritmos baseados em *MRA*. Em termos de desempenho, o algoritmo *activity_profile_kmeans* apresentou os melhores resultados, com tempos de execução significativamente menores, seguido por um grupo intermediário composto por *3_grams_kmeans*, *activity_profile_agglomerative*, *activity_profile_spectral*, *3_grams_spectral* e *TraCluSi*. Por outro lado, os algoritmos baseados em *MRA* demonstraram maior variabilidade no tempo de execução, sugerindo que sua aplicação em cenários dinâmicos deve ser cuidadosamente avaliada.

A partir dos resultados, pode-se concluir que, se o tempo de execução for um fator determinístico para a escolha do algoritmo a ser utilizado na aplicação da técnica de *trace clustering*, o algoritmo mais adequado seria o *activity_profile_kmeans*. No entanto, se for observado que a qualidade dos *clusters* e dos modelos de processos gerados pelos algoritmos do segundo grupo identificado é superior à qualidade dos modelos gerados pelo *activity_profile_kmeans*, pode-se considerar a utilização desses algoritmos.

Em um cenário em que os *event logs* são processados em *batch*, por exemplo, e o resultado do processamento é armazenado de forma persistente para ser posteriormente consumido por outras aplicações é possível utilizar algoritmos de maior custo computacional sem comprometer a disponibilidade do sistema. Por outro lado, em aplicações *on-demand*, nas quais os *event logs* são processados em tempo de requisição para serem servidos ao usuário final do sistema, a necessidade de baixa latência impõe restrições mais rígidas, favorecendo algoritmos mais eficientes como o *activity_profile_kmeans* e o *3_grams_kmeans*.

Dessa forma, este estudo contribui para a compreensão das características e limitações dos algoritmos de *trace clustering*, fornecendo diretrizes para sua seleção em diferentes cenários. Futuras pesquisas podem explorar otimizações nos algoritmos analisados, bem

como investigar abordagens que conciliem alta eficiência computacional com qualidade aprimorada nos agrupamentos gerados.

Referências Bibliográficas

[1] W. M. P. van der Aalst, *Process Mining - Data Science in Action*. Springer, second ed., 2016.

[2] R. J. D'Castro, A. L. I. Oliveira, and A. H. Terra, "Process mining discovery techniques in a low-structured process works?," in *2018 7th Brazilian Conference on Intelligent Systems (BRACIS)*, pp. 200–205, IEEE, 2018.

[3] A. J. Unger, J. F. dos Santos Neto, M. Fantinato, S. M. Peres, J. Trecenti, and R. Hirota, "Process mining-enabled jurimetrics: Analysis of a brazilian court's judicial performance in the business law processing," in *Eighteenth International Conference for Artificial Intelligence and Law (ICAIL'21)*, (São Paulo, Brazil), pp. 240–251, ACM, 2021.

[4] R. J. C. Bose and W. M. van der Aalst, "Trace clustering based on conserved patterns: Towards achieving better process models," in *International Conference on Business Process Management (BPM 2009)*, vol. 5701 of *Lecture Notes in Computer Science*, (Ulm, Germany), pp. 170–187, Springer, 2009.

[5] M. Song, C. W. Günther, and W. M. P. van der Aalst, "Trace clustering in process mining," in *BPM 2008 Workshops*, vol. 17 of *Lecture Notes in Business Information Processing*, pp. 109–120, Berlin, Heidelberg: Springer-Verlag, 2009.

[6] A. Carrera-Rivera, W. Ochoa, F. Larrinaga, and G. Lasa, "How-to conduct a systematic literature review: A quick guide for computer science research," *MethodsX*, vol. 9, p. 101895, 2022.

[7] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Information and Software Technology*, vol. 64, pp. 1–18, 2015.

[8] A. Kofod-Petersen, "How to do a structured literature review in computer science," tech. rep., PiedBoeuf ApS, May 2015.

[9] A. Augusto, J. Mendling, M. Vidgof, and B. Wurm, "The connection between process complexity of event sequences and models discovered by process mining," *Information Sciences*, vol. 598, pp. 196–215, 2022.

[10] P. Delias, M. Doumpos, E. Grigoroudis, and N. Matsatsinis, "A non-compensatory approach for trace clustering," *International Transactions in Operational Research*, vol. 00, pp. 1–19, 2017.

[11] P. D. Koninck, K. Nelissen, B. Baesens, S. V. Broucke, M. Snoeck, and J. D. Weerdt, "An approach for incorporating expert knowledge in trace clustering," in *Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE)*, vol. 10253 of *Lecture Notes in Computer Science*, (Bozen-Bolzano, Italy), pp. 561–576, Springer International Publishing, June 2017.

[12] D. Ferreira, M. Zacarias, M. Malheiros, and P. Ferreira, "Approaching process mining with sequence clustering: Experiments and findings," in *Proceedings of the 5th International Conference on Business Process Management (BPM 2007)*, vol. 4714 of *Lecture Notes in Computer Science*, (Brisbane, Australia), pp. 360–374, Springer, 2007.

[13] J. D. Weerdt, S. vanden Broucke, J. Vanthienen, and B. Baesens, "Active trace clustering for improved process discovery," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 12, pp. 2708–2721, 2013.

[14] A. Appice and D. Malerba, "A co-training strategy for multiple view clustering in process mining," *IEEE Transactions on Services Computing*, vol. 9, no. 6, pp. 832–845, 2016.

[15] P. D. Koninck, J. D. Weerdt, and S. K. L. M. vanden Broucke, "Explaining clusterings of process instances," *Data Mining and Knowledge Discovery*, vol. 31, no. 3, pp. 774–808, 2017.

[16] R. P. J. C. Bose and W. M. P. van der Aalst, "Context aware trace clustering: Towards improving process mining results," in *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pp. 401–412, SIAM, 2011.

[17] H. Fang and W. Su, "Log clustering-based method for repairing missing traces with context probability information," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 15, no. 5, pp. 1445–1460, 2024.

[18] Y. Sun, B. Bauer, and M. Weidlich, "Compound trace clustering to generate accurate and simple sub-process models," in *International Conference on Service-Oriented Computing (ICSOC)*, vol. 10601 of *Lecture Notes in Computer Science*, pp. 175–190, Springer, 2017.

[19] S. Jablonski, M. Röglinger, S. Schönig, and K. M. Wyrtki, "Multi-perspective clustering of process execution traces," *Enterprise Modelling and Information Systems Architectures*, vol. 14, no. 2, 2019.

[20] H. Fang, W. Liu, W. Wang, and S. Zhang, "Discovery of process variants based on trace context tree," *Connection Science*, vol. 35, no. 1, p. 2190499, 2023.

[21] Z. Zhang, C. Guo, W. Peng, and S. Ren, "Using event log timing information to assist process scenario discoveries," in *2020 IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pp. 58–65, IEEE, 2020.

[22] Y. Tang, T. Li, R. Zhu, C. Liu, and S. Zhang, "A hybrid genetic service mining method based on trace clustering population," *IEICE Transactions on Information and Systems*, vol. E105-D, pp. 1443–1452, August 2022.

[23] Aksu and H. A. Reijers, "How business process benchmarks enable organizations to improve performance," in *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)*, pp. 197–206, IEEE, 2020.

[24] I. M. Grigore, G. M. Tavares, M. C. da Silva, P. Ceravolo, and S. B. Junior, "Automated trace clustering pipeline synthesis in process mining," *Information*, vol. 15, no. 4, p. 241, 2024.

[25] F. Zandkarimi, J.-R. Rehse, P. Soudmand, and H. Hoehle, "A generic framework for trace clustering in process mining," in *2020 2nd International Conference on Process Mining (ICPM)*, pp. 177–184, IEEE, 2020.

[26] C. C. Aggarwal and C. K. Reddy, *Data Clustering: Algorithms and Applications*. Boca Raton, FL, USA: CRC Press, Taylor & Francis Group, 2014.

[27] P. D. Koninck and J. D. Weerdt, "Scalable mixed-paradigm trace clustering using super-instances," in *2019 International Conference on Process Mining (ICPM)*, pp. 17–24, IEEE, 2019.

[28] J. D. Weerdt, S. K. vanden Broucke, J. Vanthienen, and B. Baesens, "Leveraging process discovery with trace clustering and text mining for intelligent analysis of incident management processes," in *IEEE World Congress on Computational Intelligence (WCCI)*, (Brisbane, Australia), IEEE, 2012.

[29] Y. Sun and B. Bauer, "A novel top-down approach for clustering traces," in *Business Process Management Workshops*, vol. 202 of *Lecture Notes in Business Information Processing*, pp. 333–347, Springer, 2014.

[30] L. N. N and J. V, "Trace clustering techniques for process mining," in *2023 Third International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, IEEE, 2023.

[31] Z. Tariq, N. Khan, D. Charles, S. McClean, I. McChesney, and P. Taylor, "Understanding contrail business processes through hierarchical clustering: A multi-stage framework," *Algorithms*, vol. 13, no. 10, p. 244, 2020.

[32] P. D. Koninck, K. Nelissen, S. vanden Broucke, B. Baesens, M. Snoeck, and J. D. Weerdt, "Expert-driven trace clustering with instance-level constraints," *Knowledge and Information Systems*, vol. 63, no. 1197, pp. 1197–1220, 2021.

[33] M. Imran, M. A. Ismail, and S. Hamid, "A trace clustering framework for improving the behavioral and structural quality of process models in process mining," *Malaysian Journal of Computer Science*, 2023.

[34] X. Lu, S. A. Tabatabaei, M. Hoogendoorn, and H. A. Reijers, "Trace clustering on very large event data in healthcare using frequent sequence patterns," in *Business Process Management - BPM 2019* (T. Slaats, H. A. Reijers, J. Mendling, and M. Weidlich, eds.), vol. 11675 of *Lecture Notes in Computer Science*, pp. 198–215, Springer, 2019.

[35] G. Greco, A. Guzzo, L. Pontieri, and D. Saccà, "Discovering expressive process models by clustering log traces," *Advances in Knowledge Discovery and Data Mining, PAKDD 2004*, vol. 3056, pp. 52–62, 2004.

[36] R. Andonie, "Hyperparameter optimization in learning systems," *Journal of Membrane Computing*, vol. 1, pp. 279–291, December 2019.

[37] G. M. Tavares, S. B. Junior, and E. Damiani, "Automating process discovery through meta-learning," in *Proceedings of the International Conference on Cooperative Information Systems*, (Berlin/Heidelberg, Germany), pp. 205–222, Springer, 2022.