UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE QUIXADÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

MESTRADO ACADÊMICO EM COMPUTAÇÃO

WALLESSON CAVALCANTE DA SILVA

INDUCTIVE MINER COM AGRUPAMENTO DE SUBLOGS DE NÓS FALL-THROUGH

QUIXADÁ

2024

WALLESSON CAVALCANTE DA SILVA

INDUCTIVE MINER COM AGRUPAMENTO DE SUBLOGS DE NÓS FALL-THROUGH

Dissertação apresentada ao Curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. Criston Pereira de Souza

QUIXADÁ

2024

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S584i Silva, Wallesson Cavalcante da.
Inductive miner com agrupamento de sublogs de nós fall-through / Wallesson Cavalcante da Silva. – 2024.
80 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Campus de Quixadá, Programa de Pós-Graduação em Computação, Quixadá, 2024.
Orientação: Prof. Dr. Criston Pereira de Souza.

1. Ciência de dados. 2. Mineração de processos. 3. Minerador indutivo. 4. Diagrama espaguete. I. Título.
CDD 005

WALLESSON CAVALCANTE DA SILVA

INDUCTIVE MINER COM AGRUPAMENTO DE SUBLOGS DE NÓS FALL-THROUGH

Dissertação apresentada ao Curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Ciência da Computação

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Criston Pereira de Souza (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Régis Pires Magalhães
Universidade Federal do Ceará (UFC)

Prof. Dr. Cesar Lincoln Cavalcante Mattos
Universidade Federal do Ceará (UFC)

À minha mãe, pelo amor incondicional, pela força e por sempre acreditar no meu potencial, mesmo nos momentos mais desafiadores. Ao meu pai, em memória, cuja sabedoria, coragem e princípios continuam a guiar meus passos e inspirar minhas conquistas. Aos familiares e amigos, pela amizade verdadeira, pelas risadas compartilhadas e pelo apoio constante.

AGRADECIMENTOS

Inicialmente, expresso profunda gratidão ao Grande Arquiteto do Universo. Encontrei constante amparo e inspiração na certeza de que todas as minhas conquistas são resultado dos planos divinos que Deus traçou para mim. Foi por meio dessa força inabalável que pude percorrer com determinação todos os caminhos necessários para concluir mais esta significativa etapa em minha vida.

Gostaria de expressar meu sincero agradecimento aos estimados professores Dr. Arthur de Castro Callado, Dra. Carla Ilane Moreira Bezerra, Dr. Emanuel Ferreira Coutinho, Dra. Maria Viviane de Menezes e Dr. Regis Pires Magalhães. O impacto incisivo de suas contribuições foi fundamental para a ampliação dos meus conhecimentos, tanto no contexto das disciplinas que ministraram quanto na compreensão abrangente das responsabilidades inerentes a um estudante de mestrado.

Ao Prof. Dr. Criston Pereira de Souza, expresso minha profunda gratidão por sua orientação ao longo de todo o percurso do mestrado. Enfrentamos inúmeros desafios, nos quais destaco a notável paciência, compreensão e amizade do professor. Além disso, ressalto sua significativa importância e contribuição para o aprimoramento dos meus pensamentos científicos, os quais foram fortalecidos e refinados ao longo de nosso período de estudos.

Agradeço também aos meus colegas de mestrado na Universidade Federal do Ceará (UFC). A jornada acadêmica foi enriquecida significativamente pela presença de cada um de vocês, e compartilhar este período de estudos e pesquisas foi verdadeiramente inspirador. Agradeço pela troca constante de conhecimentos, pelas discussões estimulantes e pelo apoio mútuo em momentos desafiadores. Cada um contribuiu de maneira única para o nosso ambiente acadêmico, criando uma comunidade de aprendizado dinâmica e colaborativa.

Gostaria também de expressar minha sincera gratidão à Universidade Federal do Ceará (UFC) por ser o cenário fundamental para a realização do meu mestrado. Especialmente à coordenação do programa de mestrado por proporcionar um ambiente propício ao aprendizado e à pesquisa. O comprometimento e apoio dos funcionários da UFC foram essenciais para a minha jornada acadêmica, desde o processo de inscrição até a defesa da dissertação. Cada indivíduo contribuiu de maneira única para o meu crescimento acadêmico, e estou profundamente grato pela dedicação e profissionalismo de todos os envolvidos. A experiência na UFC não apenas enriqueceu meu conhecimento, mas também moldou minha visão e paixão pela pesquisa. À Universidade Federal do Ceará e a todos os seus funcionários, meu mais sincero agradecimento

por serem parte integrante desta significativa etapa da minha vida acadêmica.

À minha mãe, que sempre foi sinônimo de força, perseverança e amor, dedico estes agradecimentos por todos os cuidados e atenções que a senhora teve comigo durante toda a minha vida, e ainda mais neste período em que me dediquei ao mestrado. Em cada um dos meus passos, a senhora sempre esteve comigo em pensamento ou em apoio, sendo a base fundamental para alcançar cada etapa e concluir cada um dos objetivos a mim propostos.

Ao meu pai (in memoriam), que sempre acreditou e comemorou cada uma das minhas conquistas. Esteve comigo durante o início do mestrado, mas infelizmente não pôde acompanhar fisicamente a conclusão. Agradeço e rogo para que continue a comemorar e a acompanhar todas as minhas vitórias do oriente eterno, onde se encontra ao lado do Grande Arquiteto do Universo.

Por último, quero expressar meu profundo agradecimento à minha família e amigos, pilares inabaláveis ao longo da minha jornada acadêmica. À minha família, que sempre foi meu alicerce, agradeço por seu amor incondicional, apoio constante e compreensão nos momentos desafiadores. Vocês foram minha fonte de força, incentivando-me a seguir em frente e alcançar cada objetivo, mesmo nos momentos mais difíceis. Cada conquista é compartilhada com vocês, e agradeço por fazerem parte da minha trajetória. Aos meus amigos, agradeço por estarem ao meu lado, proporcionando risos, conselhos valiosos e um apoio fundamental. Suas presenças tornaram essa jornada não apenas educacional, mas também rica em experiências significativas e momentos especiais.

“Aprender é a única coisa de que a mente nunca
se cansa, nunca tem medo e nunca se arrepende.”
(Leonardo da Vinci)

RESUMO

A crescente complexidade dos processos de negócios, aliada à coleta extensiva de dados por meio de logs de eventos do mundo real, tem gerado modelos de processos não estruturados, notoriamente referidos como modelos de processos do tipo “espaguete”. Esses modelos, caracterizados pela falta de clareza e estrutura, representam um desafio significativo na compreensão e otimização dos processos subjacentes. As abordagens convencionais frequentemente resultam em representações intrincadas e de difícil interpretação, impactando negativamente a eficiência operacional e a tomada de decisões. Diante desse cenário, este trabalho surge para abordar a lacuna existente na simplificação e compreensão desses modelos não estruturados através do algoritmo denominado IM_Cluster. O foco recai sobre o agrupamento de fragmentos de traços associados a nós *fall-throughs* como uma estratégia inovadora para extrair padrões e revelar estruturas subjacentes. Os resultados obtidos demonstram que a proposta do IM_Cluster produz resultados significativos ao ser comparada com alguns trabalhos da literatura, consolidando-se como uma abordagem eficaz na simplificação de modelos de processos. O problema central reside na necessidade de desenvolver uma abordagem eficaz que não apenas lide com a complexidade inerente desses modelos, mas também proporcione uma visão clara e coesa dos processos subjacentes, promovendo, assim, uma gestão mais eficiente e informada.

Palavras-chave: ciência de dados; mineração de processos; minerador indutivo; diagrama espaguete.

ABSTRACT

The increasing complexity of business processes, coupled with extensive data collection through real-world event logs, has given rise to unstructured process models commonly referred to as "spaghetti" process models. These models, characterized by a lack of clarity and structure, pose a significant challenge in understanding and optimizing the underlying processes. Conventional approaches often result in intricate and difficult-to-interpret representations, negatively impacting operational efficiency and decision-making. In response to this scenario, this work addresses the existing gap in the simplification and understanding of these unstructured models through the algorithm named IM_Cluster. The focus is on clustering trace fragments associated with *fall-throughs* as an innovative strategy to extract patterns and reveal underlying structures. The obtained results demonstrate that the IM_Cluster proposal yields significant outcomes when compared to some works in the literature, solidifying its effectiveness as an approach for process model simplification. The central problem lies in the need to develop an effective approach that not only deals with the inherent complexity of these models but also provides a clear and cohesive view of the underlying processes, thereby promoting more efficient and informed management.

Keywords: data science; process mining; inductive miner; spaghetti diagram.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Com o evidente crescimento da quantidade de dados, estudos na área de ciência de dados tornam-se cada vez mais necessários para a criação ou aprimoramento de ferramentas eficientes que lidem com tal volume de informações. A ciência de dados se dedica à extração de informações úteis a partir de imensas, complexas e dinâmicas bases de dados, onde os softwares são utilizados para a transformação de dados em informações que servem para apoiar a tomada de decisões (Rautenberg; Carmo, 2019). Os dados gerados durante a execução de processos são conhecidos como dados de eventos (ou log de eventos). Para lidar com dados de eventos, é empregado o campo da ciência de dados conhecido como mineração de processos. É dito em Aalst (2012) que a mineração de processos busca descobrir, monitorar e melhorar processos reais, extraindo conhecimento de logs de eventos disponíveis nos sistemas de informações. Um log de eventos consiste em um conjunto de dados formados por traços (ou casos), sendo cada traço uma instância do processo. Cada traço possui uma sequência bem definida de eventos (ou atividades) de algum processo, em que os eventos são ordenados de forma que possam ser vistos como uma "execução" do processo (Aalst, 2012).

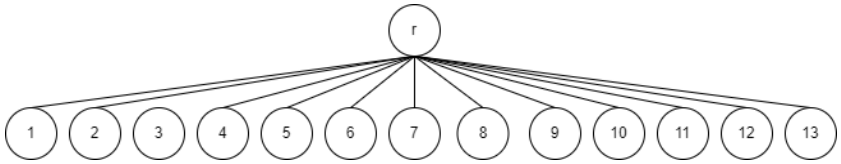
Uma das atividades de mineração de processos consiste na descoberta dos chamados modelos de processo a partir de um log de eventos. É dito em Weerdt *et al.* (2013) que um grande número de trabalhos ilustram que as técnicas de descoberta de processos enfrentam dificuldades em renderizar os modelos de processo de forma correta e interpretável a partir de logs de eventos formados com base em dados da vida real ou de ambientes extremamente flexíveis. Os processos podem ser bem estruturados e não estruturados, onde em Aalst (2011) é dito que a aplicação de mineração de processos em diversos domínios revelou a existências desses tipos de processos comumente chamados também de processos "lasanha" e processos "espaguete", respectivamente. Os processos estruturados (ou processos "lasanha") possuem a característica de que todas as atividades são repetíveis e tem as atividades que antecedem e precedem uma dada atividade bem definidas. Já nos processos não estruturados (ou processos "espaguete") é difícil definir as atividades que antecedem ou precedem uma dada atividade.

Aalst (2011) discute em seu trabalho as características dos processos "lasanha" e "espaguete". Para os processos "lasanha" é dito que o modelo de processo descoberto é menos relevante, porém, a relação entre logs de eventos e modelos de processo pode ser usada para detectar desvios, descobrir gargalos, sugerir mudanças em projetos, prever atrasos etc. Já para os processos "espaguete" é dito que o modelo descoberto fornece insights mais importantes, mesmo

com a mineração de processos sendo um desafio para este tipo de processo. Ainda em Aalst (2011) é dito que apesar das dificuldades encontradas nos processos “espaguete”, os benefícios potenciais são substanciais e a automação e a melhoria de processos só são possíveis após o entendimento e a agilização desses processos. Processos “espaguete” são bem mais interessantes que processos “lasanha” para mineração de processos (Aalst, 2016). Isso acontece devido às suas possibilidades de melhoria, seja no processo de descoberta de modelos (simplificação do modelo complexo gerado pelo processo “espaguete”), ou ainda em detecção de problemas relacionados ao domínio dos dados do log de eventos, como os citados anteriormente.

Um dos problemas encontrados em representações de processos “espaguete” é a formação de uma estrutura conhecida como “flor” (em inglês *“flower”*) gerada por conta de um evento conhecido como *fall-through*. O conceito de *fall-through* surge quando um evento ou atividade ocorre, mas não resulta na transição prevista para o próximo estado do processo. Em essência, ocorre uma situação em que um evento não segue o fluxo padrão do processo, não alinhando-se com o que era esperado com base na modelagem ou especificação original do processo. Os *fall-throughs* ocorrem com frequência em processos “espaguete”, pois o algoritmo não consegue definir bem qual operador escolher para aplicar no nó da árvore de processo. A Figura 1 é um exemplo representativo de uma árvore de processos (um dos diversos tipos de modelos de processos, a mesma é descrita posteriormente) que apresenta a estrutura descrita, onde o exemplo consiste em um nó raiz *r* que é um operador conectado a um número considerado de nós filhos que são atividades obtidas a partir de um log de eventos.

Figura 1 – Exemplo de *fall-through* em uma árvore de processos.



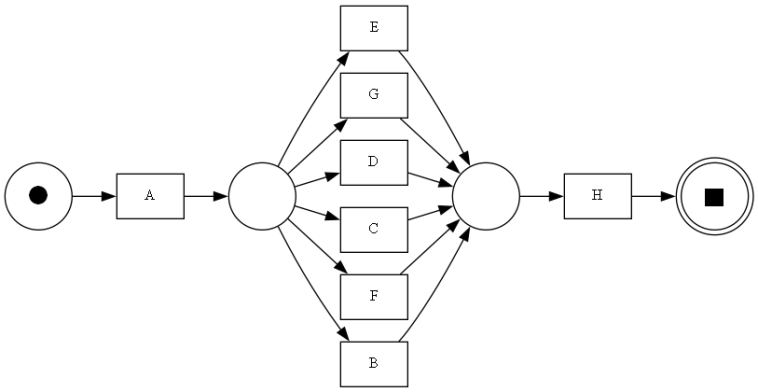
Fonte: Elaborada pelo autor (2024).

Para melhorar a compreensão de processos espaguete a técnica de agrupamento de traços é abordada na literatura (Song *et al.*, 2008; Weerdt *et al.*, 2013). A abordagem busca separar os traços em subgrupos de forma que simplifique o log de eventos original. Para realizar o agrupamento, as técnicas podem abordar conceitos como semelhança do comportamento dos traços, adequação do traço com um determinado grupo, baseada em processos genéticos, entre outros.

Em alguns modelos, como por exemplo, uma rede de Petri, construídos a partir de um

dado log de eventos em que existe a ocorrência de *fall-throughs*, produzem “empilhamentos” de atividades. A presença da estrutura descrita pode ocasionar diversos problemas no modelo obtido. Um dos problemas é o de tornar o modelo menos claro e ambíguo, pois fica difícil entender onde uma atividade termina e começa a próxima, o que pode levar a confusão e interpretações errôneas do processo. Outro problema pode ser a dificuldade de monitorar o progresso do processo, devido à falta de uma separação clara entre as atividades, o que piora a identificação de gargalos ou a análise de eficiência do modelo. Além disso, a presença de *fall-throughs* ainda acrescenta resistência à melhoria dos modelos, já que a falta de clareza dos modelos dificulta a identificação de áreas que precisam de aprimoramento. A Figura 2 apresenta um exemplo de um modelo de rede de Petri, no qual as atividades caracterizam a descrição do problema apresentado.

Figura 2 – Exemplo de empilhamento de atividades.



Fonte: Elaborada pelo autor (2024).

Neste trabalho, propomos uma adaptação ao algoritmo *Inductive Miner* (IM) (Aalst, 2016) que agrupa os traços do log de eventos associados a nós *fall-througs* denominada IM_Cluster. Desta forma, os modelos resultantes tendem a apresentar menos ocorrências de empilhamento de atividades. Também elaboramos uma nova métrica de simplicidade que pode capturar a ocorrência de empilhamento de atividades, onde o IM_Cluster se baseia nessa métrica para decidir onde na árvore de processos a mudança vale a pena. Além disso, uma avaliação empírica é conduzida para comparar o algoritmo proposto com o IM original e com o algoritmo *Active Trace Cluster* (ActiTraC) (Weerdt *et al.*, 2013).

A seguir, são apresentados os objetivos da pesquisa, delineando as principais metas e propósitos que guiaram o desenvolvimento deste estudo. Esses objetivos visam esclarecer as questões centrais que a pesquisa pretende abordar, além de definir o escopo e a direção das investigações realizadas.

1.1 Objetivos

Desenvolver e avaliar um algoritmo de mineração de processos que produza modelos mais "interpretáveis" do que as soluções fornecidas por algoritmos estado-da-arte (conforme uma métrica de simplicidade proposta), quando aplicados a processos "espaguete".

1.1.1 Objetivo geral

Desenvolver um algoritmo de descoberta para melhorar o entendimento e simplificar modelos de processos do tipo "espaguete" obtidos a partir de logs de eventos do mundo real, considerando suas estruturas frequentemente mal organizadas que apresentam *fall-throughs*.

1.1.2 Objetivos específicos

- Criar métrica de simplicidade de processos que capture a ocorrência de estruturas resultantes de nós *fall-through*;
- Desenvolver algoritmo de mineração de processos que produzam melhores modelos que o *Inductive Miner* (Aalst, 2016) e o *Active trace clustering* (Weerdt *et al.*, 2013), de acordo com a métrica proposta;
- Avaliar experimentalmente o algoritmo proposto utilizando logs de eventos públicos, e compará-lo com abordagens disponíveis na literatura. Esta avaliação leva em conta o tempo de execução, métricas de fitness e simplicidade utilizadas na literatura, e a métrica de simplicidade proposta neste trabalho

No Capítulo 2 é apresentada a fundamentação teórica, onde os principais conceitos que facilitam a compreensão deste trabalho são apresentados. O Capítulo 3 apresenta os trabalhos relacionados que descrevem abordagens propostas na literatura. No Capítulo 4 são apresentados os passos tomados para desenvolver o algoritmo $IM\_Cluster$. No Capítulo 5 são apresentados os resultados dos experimentos que foram obtidos e as discussões sobre os mesmos. No Capítulo 6 são apresentadas as conclusões do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentadas as definições dos conceitos utilizados neste trabalho. Os conceitos abordados de modelos de processos e mineração de processos contextualizam as definições do objeto de estudo deste trabalho, ou seja, o problema de representação de dados de eventos obtidos pela execução de processos. Já o conceitos de descoberta de agrupamentos ajuda a entender a motivação da solução escolhida para a implementação do algoritmo deste trabalho.

2.1 Modelos de processos

Um evento normalmente consiste em um "case id", que determina um identificador de caso específico de um processo, uma atividade, que corresponde ao evento que foi executado, e o carimbo de data/hora que determina o momento que a atividade ocorreu. Um log de eventos consiste em um conjunto de eventos, onde o termo "traço" é utilizado para referenciar a sequência de atividades ou eventos de um determinado case id respeitando a ordem definida pela data/hora dos eventos.

A Tabela 1 apresenta uma estrutura exemplo de um log de eventos. Pode-se observar que o traço referente ao caso 1101, por exemplo, consiste nas atividades A, B e C apresentadas na coluna "Atividade". A tabela ainda apresenta na coluna "Data/Hora" o momento que os eventos do caso 1101 ocorreram.

Tabela 1 – Exemplo de log de eventos

Case ID	Atividade	Data/Hora
1101	A	01-02-2022:10.02
1101	B	02-02-2022:09.35
1101	C	03-02-2022:11.14
1102	A	01-02-2022:08.06
1102	D	02-02-2022:10.32
1102	E	05-02-2022:11.55
1102	A	06-02-2022:07.22
1103	A	01-02-2022:10.32
1103	B	03-02-2022:08.17
1103	F	04-02-2022:11.12

Fonte: Elaborada pelo autor (2024).
Nota: Dados ilustrativos.

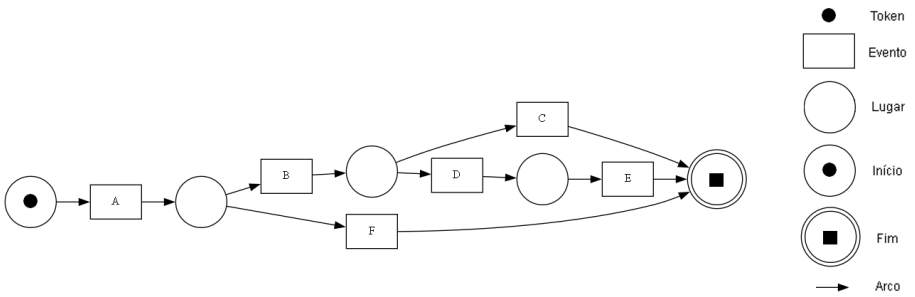
De acordo com Aalst (2012), modelos de processo eram tipicamente feitos à mão sem usar logs de eventos. Porém, modelos de processos podem ser obtidos a partir de técnicas de mineração de processos utilizando dados de eventos (Aalst, 2016). As notações comumente

utilizadas para modelos de processos são redes de Petri, EPCs, BPMN, diagramas de atividades UML, etc (Aalst, 2012). Os modelos de processos podem ser usados para descrever propriedades temporais, especificar a criação e o uso de dados (por exemplo, para modelar decisões) e estipular a maneira como os recursos integram com o processo (Aalst, 2016).

2.1.1 Rede de Petri

Para compreender melhor um modelo de processos, é ilustrado na Figura 3 uma representação do log de eventos da Tabela 1 em formato de rede de Petri. O token indica o estado de execução em que o processo se encontra. Os quadrados representam as atividades do modelo, onde o rótulo que identifica a atividade é apresentado. Os círculos caracterizam os estados de execução do processo, podendo um estado de execução ser de início, fim ou uma outra posição (estado intermediário) do modelo. Os arcos indicam as transições do modelo de processo.

Figura 3 – Rede de Petri do processo exemplo.



Fonte: Elaborada pelo autor (2024).

2.1.2 Árvore de processos

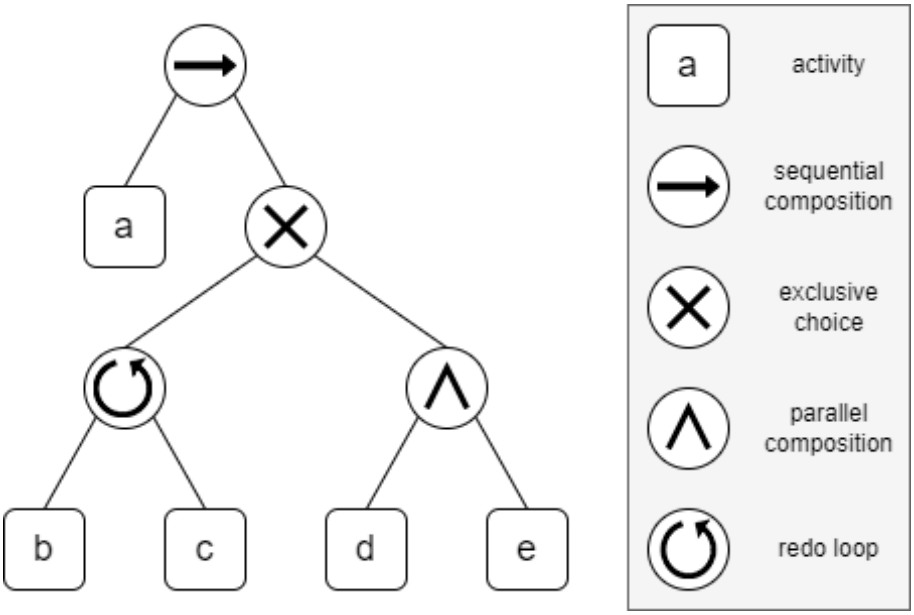
As árvores de processos (em inglês, *process trees*) são ferramentas utilizadas para modelar e analisar processos de negócio de forma visual. Elas fornecem uma representação gráfica hierárquica das atividades envolvidas em um processo, permitindo uma compreensão mais clara e estruturada do fluxo de trabalho. Uma árvore de processo é composta por nós e arestas, onde cada nó representa uma atividade específica e as arestas indicam a relação de dependência entre as atividades. A raiz da árvore representa o início do processo, enquanto os nós folha representam as atividades finais. Os nós intermediários representam as atividades intermediárias, que podem ser subdivididas em atividades mais detalhadas.

A Figura 4 apresenta um estrutura ilustrativa de uma árvore de processo. Aalst (2016)

define que os nós internos de uma árvore de processo representam operadores básicos e as folhas representam atividades. Os quatro operadores básicos de uma árvore de processo são:

- Composição sequêncial (em inglês *sequential composition*) $\rightarrow$: Representa a ordem em que as atividades ou processos devem ser executados. Cada atividade é executada em sequência, uma após a outra da esquerda para a direita na árvore;
- Escolha exclusiva (em inglês *exlusive choice*) $\times$: É usado para criar bifurcações em um fluxo de processos. Ele representa uma escolha ou condição que determina qual caminho será seguido a partir de um determinado ponto na árvore de processos. Dependendo da condição, diferentes atividades ou processos podem ser executados;
- Composição paralela (em inglês *parallel composition*) $\wedge$: Indica que as atividades ou processos podem ser executados simultaneamente, ou seja, em paralelo. Isso significa que várias atividades podem ser executadas ao mesmo tempo, sem depender uma da outra;
- Repetição (em inglês *redo loop*) $\circlearrowleft$: É usado para repetir uma sequência de atividades ou processos várias vezes. Ele permite que uma sequência de atividades seja executado repetidamente com base em uma condição específica, até que a condição não seja mais atendida.

Figura 4 – Exemplo de árvore de processos.



Fonte: Elaborada pelo autor (2024).

Uma das vantagens das árvores de processo é que a sua representação visual torna mais fácil para os envolvidos no processo entenderem e discutirem as etapas envolvidas. Também facilita a documentação e a transferência de conhecimento sobre o processo. Existem várias

técnicas e ferramentas disponíveis para a criação e análise de árvores de processo. Algumas delas incluem a utilização de diagramas de fluxo de trabalho, linguagens de modelagem de processos como o BPMN (em inglês, *Business Process Model and Notation*) e softwares específicos de modelagem de processos.

2.1.3 Processos espaguete

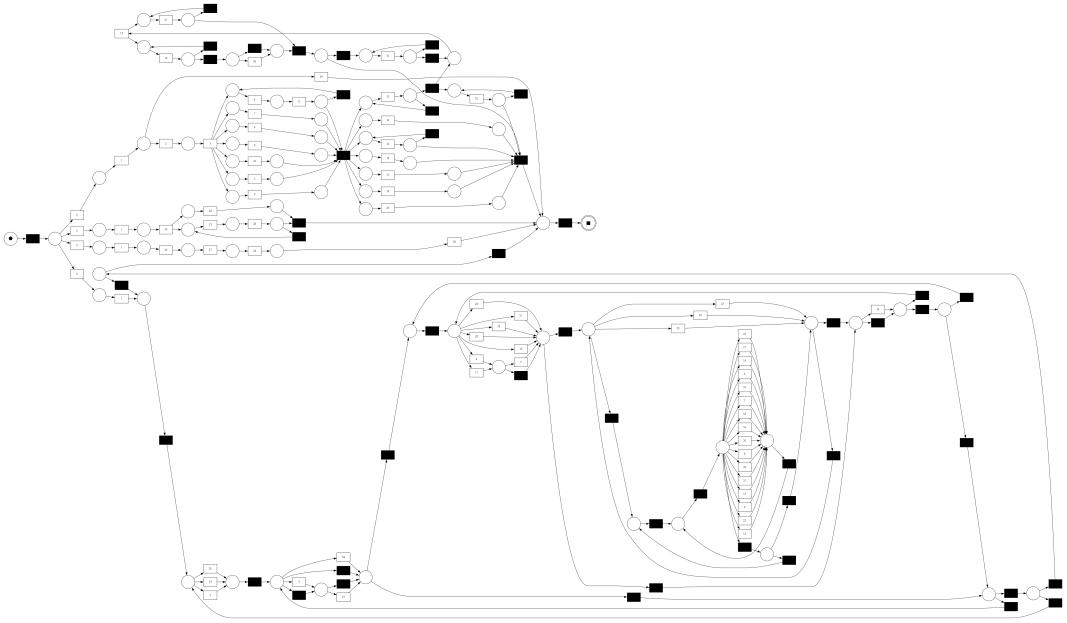
Em mineração de processos, o termo "processos espaguete" refere-se a fluxos de trabalho desorganizados e complexos, nos quais as atividades e os caminhos de execução se entrelaçam, assemelhando-se a um "prato de espaguete". Esse tipo de processo é caracterizado por ter múltiplas ramificações, loops e sobreposições, dificultando a compreensão e a análise do fluxo de trabalho. Os processos espaguete geralmente surgem em organizações onde a execução do fluxo de trabalho é menos estruturada ou padronizada. Eles podem ser resultado de exceções frequentes, variações significativas nos procedimentos ou falta de controle adequado dos processos. À medida que as atividades se acumulam ao longo do tempo, os fluxos de trabalho se tornam cada vez mais complexos e difíceis de serem visualizados de forma clara.

De acordo com Aalst (2016) os processos desse tipo são interessantes para mineração de processos, pois a mineração de processos pode ajudar a realizar melhorias dramáticas no processo, descobrindo os principais problemas. Também é dito que o mesmo não ocorre para processos bem organizados e estruturados, pois é fácil aplicar técnicas de mineração de processos e existe pouco potencial de melhoria, fazendo assim tais processos serem menos interessantes.

A Figura 5 mostra um exemplo de modelo de processo que pode ser chamado de "processo espaguete". O modelo conhecido como rede de Petri da figura foi obtido utilizando o algoritmo *Active Trace Cluster* (Weerdt *et al.*, 2013) da ferramenta ProM (Verbeek *et al.*, 2010). O modelo foi construído com base no log de eventos *International Declarations* do *BPI Challenge* 2020 (Dongen; Wynants, 2020) contendo 6.449 casos, 34 atividades e 72.151 eventos. Mesmo com a aplicação de pré-processamento para facilitar a compreensão dos dados, o modelo ainda é muito difícil de compreender. Entre os elementos da rede de Petri destacam-se algumas atividades pretas, isso se deve a ferramenta que foi utilizada para construir o modelo, onde essas atividades representam transições entre atividades no log de eventos que diferem das demais transições.

A presença de processos espaguete dificulta a eficiência, a qualidade e a conformidade dos processos de negócios. Eles podem levar a atrasos, erros, retrabalho e ineficiências

Figura 5 – Processo Espaguete BPI Challenge 2020.



Fonte: Elaborada pelo autor (2024).
Nota: Processo espaguete descrevendo as atividades da base de dados do *BPI Challenge* 2020. O modelo foi construído com base em um log de eventos contendo 6.449 casos, 34 atividades e 72.151 eventos.

operacionais. Além disso, a falta de visibilidade e compreensão dos fluxos de trabalho dificulta a identificação de gargalos, problemas e oportunidades de melhoria.

Em resumo, os processos espaguete são fluxos de trabalho desorganizados e complexos em que as atividades e os caminhos de execução se entrelaçam. A mineração de processos desempenha um papel crucial na identificação e análise desses processos, permitindo uma compreensão mais clara e visualização dos fluxos de trabalho existentes. Com essa compreensão, as organizações podem tomar medidas para simplificar e otimizar os processos, visando melhorias na eficiência, qualidade e conformidade dos seus fluxos de trabalho.

2.2 Mineração de processos

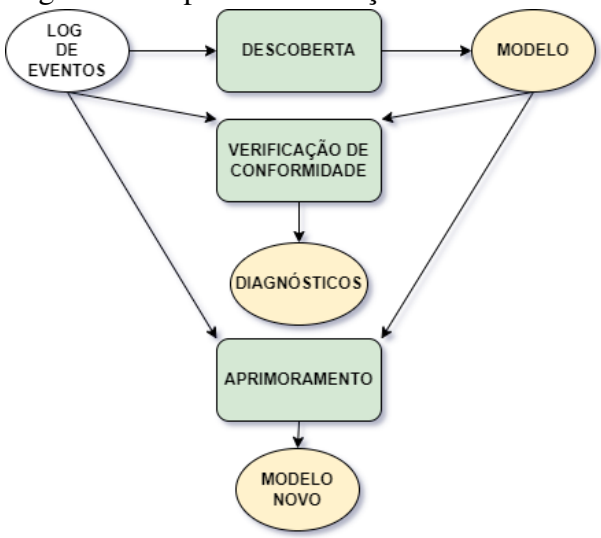
Para compreender mineração de processos é preciso entender o que é mineração de dados e aprendizado de máquina. De acordo com Aalst (2016) mineração de dados pode ser definida como “a análise de conjuntos de dados (muitas vezes grandes) para encontrar relacionamentos suspeitos e para resumir os dados de novas maneiras que são compreensíveis e úteis para o proprietário dos dados ”. Ainda é dito que os dados de entrada são normalmente fornecidos como uma tabela e a saída pode ser regras, grupos, estruturas de árvore, gráficos, equações, padrões, etc. Já para aprendizado de máquina, Aalst (2016) descreve que o conceito

está preocupado em como construir modelos ou programas que melhoram automaticamente com a experiência, onde para aprender a se adaptar, um modelo é construído a partir dos dados de entrada ao invés de usar rotinas de execuções fixas, permitindo que sua evolução seja usada para realizar previsões ou tomar decisões. Assim, Aalst (2016) fundamenta mineração de processo como a adição da perspectiva do processo ao aprendizado de máquina e mineração de dados.

O objetivo de mineração de processos é usar dados de eventos para extrair informações de processos de forma que permitam a descoberta automática do modelo de processo, onde busca realizar uma comparação entre os dados de eventos (comportamento observado) e os modelos de processos (feitos à mão ou descobertos automaticamente) (Aalst, 2016). Já Chen *et al.* (2021) descreve mineração de processos como uma tecnologia útil para compreensão de processos de negócios usando logs de eventos obtidos a partir de sistemas de informação. Ainda em Chen *et al.* (2021) é dito que mineração de processos tem apresentado um potencial promissor em perspectivas diferentes, destacando a descoberta de *insights* e a melhoria do desempenho de processos. Pesquisar mineração de processos, segundo Martin *et al.* (2019), tem foco predominante no desenvolvimento de novas técnicas ou na aplicação de técnicas existentes.

É descrito em Rudnitckaia (2016) que existem três tipos básicos de mineração de processos, sendo eles: a descoberta, a verificação de conformidade e a criação de novos modelos. A Figura 6 ilustra como se desenvolve cada um dos três processos. O primeiro tipo de mineração de processos descrito é a descoberta. A técnica requer um log de eventos como entrada, ao qual é possível gerar um modelo de processos sem usar nenhuma informação de suposição. O segundo tipo é a verificação de conformidade, onde busca comparar um log de eventos com um modelo existente para, por exemplo, verificar se a realidade do log de eventos corresponde à apresentada no modelo. Os dados fundamentais que possibilitam a realização da técnica descrita são o log de eventos e um modelo representativo. Já o terceiro tipo de mineração de processos consiste no aprimoramento ou melhoria de um dado modelo existente utilizando informações sobre o processo real obtidas a partir de algum log de eventos. Levando em consideração que a verificação de conformidade entre um modelo e o log de eventos busca medir o alinhamento entre eles, o aprimoramento busca estender o modelo inicial para um novo modelo melhorado.

Figura 6 – Tipos de Mineração de Processos



Fonte: Elaborada pelo autor (2024).

2.2.1 Inductive miner

O *Inductive Miner* (sigla em inglês IM) (em português, Minerador Indutivo) consiste em um algoritmo de descoberta de processos que faz parte da família de técnicas de mineração de processos, onde, segundo Aalst (2016), é altamente extensível e permite a realização de abordagens distintas. Ainda é dito que a técnica possui variações que permitem lidar com comportamentos infrequentes e com logs de eventos grandes, onde garante critérios formais de correção, como, por exemplo, a capacidade de redescobri o modelo original.

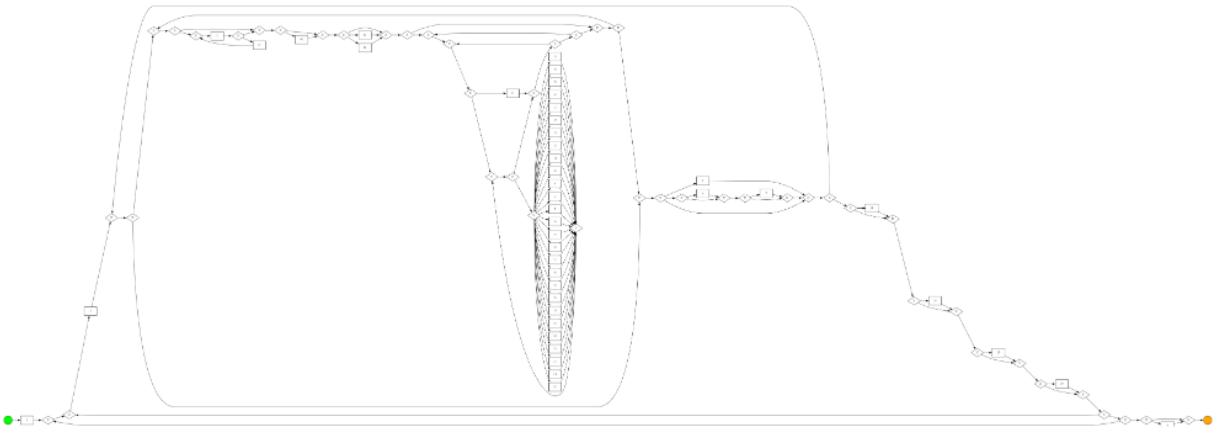
O algoritmo foi desenvolvido com o objetivo de extrair modelos de processos a partir de logs de eventos, utilizando abordagens indutivas. A abordagem indutiva do IM é baseada no princípio de generalização. Ele analisa os logs de eventos para identificar padrões frequentes de comportamento e, em seguida, gera um modelo de processo que representa esses padrões. O algoritmo é capaz de descobrir tanto o fluxo de trabalho principal quanto as variações e desvios do processo.

De acordo com Aalst (2016), o princípio básico do IM utiliza um *directly-follows graph* (em português, grafo de seguimento direto), onde o mesmo consiste em uma representação gráfica de um log de eventos que descreve a ordem em que as atividades ocorrem. Ele é usado para identificar e visualizar os relacionamentos sequenciais entre as atividades em um processo. O IM busca dividir o log de eventos, representado como *directly-follows graph*, em sublogs através de uma técnica de “corte”. Em Aalst (2016) é descrito que existem quatro tipos de corte, a

saber, *"exclusive-choice cut"*, *"sequence cut"*, *"parallel cut"* e ‘*'redo-loop cut"* (em português, corte de escolha exclusiva, corte de sequência, corte de paralelo e corte de repetição). Os cortes correspondem aos operadores de uma árvore de processos, assumindo que não há atividades duplicadas ou silenciosas.

Sendo assim, inicialmente o IM busca gerar o *directly-follows graph*. Depois ele aplica os cortes no *directly-follows graph* buscando dividir o log de eventos em sublogs menores, onde a maneira como o log de eventos é dividido depende do operador. No caso de nenhum dos quatro operadores triviais acontecer, então é escolhido o chamado *"fall-through"*. Caso o *"fall-through"* seja selecionado, então a parte do log de eventos que não pôde ser dividida é apresentada como um modelo chamado de "flor" ("tudo pode acontecer"). Após a realização dos procedimentos descritos, o IM retorna uma árvore de processos específica construída a partir dos sublogs gerados. A Figura 7 apresenta um exemplo de uma estrutura "flor" obtida a partir da conversão de uma árvore de processos gerada pelo algoritmo IM e convertida em um BPMN de um log de eventos artificial, onde pode-se observar que a partir de uma determinada atividade, o seu seguimento direto é uma "pilha" de atividades que não possuem uma sequência bem definida.

Figura 7 – Exemplo de modelo *fall-through*.



Fonte: Elaborada pelo autor (2024).

Uma das principais vantagens do IM é sua capacidade de lidar com logs de eventos incompletos, ou seja, quando algumas atividades estão faltando ou não são registradas. Ele é capaz de generalizar os padrões de comportamento com base nas informações disponíveis e criar um modelo coerente mesmo com dados incompletos. No entanto, o IM pode gerar modelos de processo com muitas variações e escolhas arbitrárias, especialmente quando há muitos desvios no comportamento registrado. O IM ainda possui outras variações para diferentes tipos de problemas, onde algumas das principais variações são as seguintes:

- *Indcutive Miner-infrequent* (IMf): É uma variação do IM que busca identificar padrões menos comuns ou infrequentes em um dado processo. O IMf pode ser útil para descobrir comportamentos anômalos ou processos alternativos em um conjunto de dados. Ele é particularmente útil quando se deseja analisar desvios em relação ao comportamento predominante em um processo.
- *Indcutive Miner-directly-follows based* (IMd): É uma variação que se concentra em identificar relações entre atividades em um processo. O IMd visa identificar as relações sequenciais diretas entre atividades, ou seja, quais atividades frequentemente ocorrem imediatamente após outras no fluxo de trabalho. Essa abordagem é útil para criar uma representação gráfica do processo que mostra as conexões diretas entre as atividades. Isso pode ajudar na visualização e análise do fluxo de trabalho, revelando padrões e relações de sequência importantes.
- *Indcutive Miner-infrequent-directly-follows based* (IMfD): É uma abordagem que busca padrões de sequência menos frequentes entre atividades em um processo. O objetivo do IMfD é identificar os relacionamentos sequenciais diretos menos comuns entre atividades em registros de eventos. Isso pode ser útil para encontrar caminhos de processos menos frequentes ou exceções em um conjunto de dados. Ao combinar o "Infrequent" com o "Directly Follows-Based", a ênfase está em identificar sequências de atividades que não são apenas sequenciais, mas também pouco frequentes.

2.2.2 Verificação de conformidade

As técnicas de verificação de conformidade buscam encontrar pontos em comum e discrepância entre o comportamento de um modelo de processo e o comportamento registrado por um log de eventos (Aalst, 2016). Simplificando, conformidade compara o comportamento observado com o comportamento modelado. Algumas utilidades desta verificação descritas por Aalst (2012) são as seguintes:

- verificar a qualidade dos processos documentados (avalia se descrevem a realidade com precisão);
- identificar casos desviantes e entender o que eles têm em comum;
- para fins de auditoria;
- como ponto de partida para o aprimoramento de modelos.

A partir das aplicações apresentadas pode-se notar que conformidade é utilizada por

vários motivos, seguindo do princípio de avaliação de um algoritmo de descoberta de processos até auditoria e monitoramento de conformidade. Aalst (2012) descreve que basicamente existem três abordagens para verificação de conformidade.

A primeira abordagem cria abstrações do comportamento no log de eventos e outra do comportamento permitido pelo modelo. A ideia mostra as dependências causais entre atividades. Por exemplo, considerando duas atividades *x* e *y*, pode-se observar que uma atividade *x* às vezes é seguida por uma atividade *y*, mas o contrário não acontece. Para mostrar tais dependências o conceito de *footprint* apresentado em Aalst (2012), onde um *footprint* consiste em uma matriz de dependências. Continuando com o exemplo das atividades *x* e *y* descrito, se o *footprint* do modelo mostra que *x* nunca é seguido por *y* ou que *y* as vezes é seguido por *x*, então os *footprints* do log de eventos e do modelo discordam sobre a relação de ordenação de *x* e *y*.

A segunda abordagem reproduz o log de eventos no modelo. Uma abordagem simples em relação à verificação de conformidade é contar a quantidade de casos que podem ser "analisados completamente", ou seja, essa quantidade corresponde a de casos em que as sequências de transições executadas levam do estado inicial ao final no modelo. Um problema desse tipo de abordagem é não distinguir entre um caso "quase adequado" e um caso que não tem correspondência nenhuma ao comportamento do modelo. A segunda abordagem descrita por Aalst (2012) continua reproduzindo o log de eventos no modelo mesmo quando as transições não estão habilitadas. Basicamente a ideia é "emprestar tokens", forçando a execução da transição de qualquer forma e registrando o problema. Terminando a computação, o número total de "tokens emprestados" e o de "tokens não consumidos" indicam o nível de aptidão do modelo.

A terceira abordagem calcula um valor de alinhamento ótimo entre cada traço no log de eventos e o comportamento mais semelhante no modelo. Verificando se o traço observado realiza um alinhamento perfeito, ou seja, se os movimentos do traço no log de eventos podem ser seguidos (representados) por movimentos no modelo, é possível calcular a precisão da representação do modelo em relação ao log de eventos. Em Aalst (2016) é dito que alinhamentos foram introduzidos para superar as limitações de outras técnicas de verificação de conformidade, como por exemplo a de reprodução, onde apresenta o problema de que se um caso não se encaixa, a abordagem não cria um caminho correspondente através do modelo. A descrição dada por Berti *et al.* (2019) é que a técnica baseada em alinhamento visa encontrar um dos melhores alinhamentos entre o traço e o modelo. Para cada traço, a saída de um alinhamento é uma lista de pares onde o primeiro elemento é um evento (do traço) e o segundo elemento é uma transição

(do modelo) ou ».

Para melhor compreensão da abordagem de alinhamento considere o modelo apresentado na Figura 3, onde, por exemplo, se verificarmos os alinhamentos dos traços {A, D, E, A, F} e {A, D, C, E, A, F} geramos às saídas representadas pelos alinhamentos Y_1 e Y_2 respectivamente. Para o alinhamento Y_1, representado na Tabela 2, podemos observar um alinhamento perfeito em todos os eventos e transições no traço e no modelo, onde nenhuma coluna da transição apresenta ». No entanto para o alinhamento Y_2, representado pela Tabela 3, encontramos uma não conformidade entre o traço e o modelo, pois o evento C do traço não ocorre na transição esperada do modelo (coluna 3), logo a transição é representada por ».

Tabela 2 – Alinhamento Y_1

Saída	1	2	3	4	5
Evento	A	D	E	A	F
Transição	A	D	E	A	F

Fonte: Elaborada pelo autor (2024).

Tabela 3 – Alinhamento Y_2

Saída	1	2	3	4	5	6
Evento	A	D	C	E	A	F
Transição	A	D	»	E	A	F

Fonte: Elaborada pelo autor (2024).

2.2.3 Precisão

No trabalho de Blum (2015), é introduzida a métrica de precisão. Essa métrica avalia o grau em que o modelo de processo captura o comportamento observado no log de eventos. Um modelo de processo com uma alta medida de precisão deve ser capaz de reproduzir a maioria do comportamento observado no log de eventos.

A métrica de precisão avalia a conformidade do modelo ao reproduzir cada tipo de traço. À medida que um token avança, ela registra o número de tokens que precisaram ser adicionadas artificialmente devido à falta de habilitação da transição associada ao evento registrado, impedindo sua execução bem sucedida, bem como o número de tokens restantes. Essa métrica retorna um valor compreendido entre 0 e 1. Um valor elevado indica que poucas fichas precisaram ser adicionadas artificialmente e que poucas fichas permaneceram após a repetição, o que sugere que o modelo se ajusta bem ao comportamento observado.

Para avaliar um modelo usando a métrica de precisão, o módulo PM4Py (Berti *et al.*, 2019) implementa o algoritmo *token-based replay*. Esse algoritmo combina um traço de evento com um modelo de rede de Petri, começando do lugar inicial, a fim de determinar quais transições são executadas e quais lugares possuem tokens restantes ou ausentes para a instância do processo em questão. Para verificar a conformidade, é considerado que um traço é adequado de acordo com o modelo se, durante sua execução, as transições podem ser executadas sem a necessidade de inserir tokens ausentes. Portanto, um traço é considerado adequado se alcançar a marcação final sem nenhum token faltando ou sobrando. Para cada traço, quatro valores devem ser determinados: tokens produzidos (p), tokens restantes (r), tokens ausentes (m) e tokens consumidos (c). Com base nisso, uma fórmula pode ser derivada para calcular o aptidão de acordo com a seguinte equação:

$$Precisao(n,t) = \frac{1}{2} \cdot (1 - \frac{r}{p}) + \frac{1}{2} \cdot (1 - \frac{m}{c}) \tag{2.1}$$

Nesta fórmula, n representa a rede de Petri e t representa o traço de evento. A precisão é calculado como a diferença entre os tokens produzidos e os tokens ausentes, dividida pela soma de todos os quatro valores mencionados anteriormente. Essa fórmula permite quantificar o grau de adequação do modelo em relação ao traço de evento, fornecendo um valor de aptidão que indica o quão bem o modelo se ajusta aos eventos observados. O valor resultante é um número entre 0 e 1, onde quanto mais próximo de 1 melhor é a precisão do modelo.

2.2.4 Simplicidade

A fim de que um modelo de processo seja útil para uma variedade de pessoas com diferentes origens, é necessário que ele seja interpretado e compreendido. Para facilitar esse processo, é desejável que o modelo descoberto seja o mais simples possível. Por exemplo, em muitos casos, existem várias formas sintáticas de expressar o mesmo comportamento, mas algumas podem ser mais preferíveis e representações menos adequadas. Portanto, em termos de qualidade, o melhor modelo é aquele que é capaz de explicar o comportamento registrado de maneira simples. A simplicidade é uma quarta dimensão considerada na análise de um modelo de processo. O trabalho de Blum (2015) apresenta o cálculo da simplicidade que leva em consideração apenas o modelo da rede de Petri, onde utiliza como critério para simplificar o modelo, é o grau do arco inverso.

É descrito em Blum (2015) que o grau do arco inverso está relacionado à contagem de arcos direcionados que conectam uma transição a um lugar em um modelo de processo. Em essência, ele representa o número de arcos que ligam uma atividade de volta ao estado anterior. Quanto menor for o grau do arco inverso, mais simples será considerado o modelo. Isso ocorre porque um baixo grau do arco inverso indica que as atividades ocorrem em uma sequência mais direta, sem loops desnecessários ou retrocessos. Ao utilizar o grau do arco inverso como um critério no cálculo da simplicidade, os modelos de processo com menos arcos inversos são geralmente preferidos, pois refletem um fluxo de atividades mais linear e intuitivo. No entanto, é importante ressaltar que o grau do arco inverso é apenas um dos critérios possíveis para avaliar a simplicidade de um modelo de processo. Outros fatores, como o número de tarefas, sequências, gateways de decisão e paralelismo, também podem ser levados em consideração, dependendo do contexto da análise do modelo.

A formula descrita em Blum (2015) primeiramente considera o grau médio (em inglês *mean_degree*) para um lugar/transição da rede de Petri, que é definido como a soma do número de arcos de entrada e arcos de saída. Se todos os lugares tiverem pelo menos um arco de entrada e um arco de saída, o número é pelo menos 2. Escolhendo um número k entre 0 e infinito, onde o valor de k define a partir de qual o grau médio começa a ter real importância na equação de simplicidade de um dado modelo de processos n. A simplicidade baseada no grau do arco inverso é então definida pela equação:

$$simplicidade(n) = \frac{1}{1 + max(grau\_medio - k, 0)} \tag{2.2}$$

Para calcular a simpliciade de um modelo de rede de Petri, usando o grau do arco inverso, o módulo PM4Py (Berti *et al.*, 2019) implementa o algoritmo de simplicidade que tem como valor resultante um número entre 0 e 1, onde quanto mais próximo de 1 mais simples é o modelo. Na implementação do módulo o valor de k padrão é igual a 2. Sendo assim, se a rede de Petri tiver as atividades bem distribuídas, o grau médio dos arcos seria igual a 2, logo a equação chegaria ao resultato de simplicidade perfeita com resultado 1 para a rede de Petri.

2.3 Descoberta de agrupamentos

Em Madhulatha (2012) é dito que os algoritmos de agrupamento de dados podem ser hierárquicos ou particionais, sendo que os hierárquicos encontram grupos sucessivos usando

grupos previamente estabelecidos, e os particionais determinam todos os grupos no momento. Também é dito em Madhulatha (2012) que os algoritmos hierárquicos podem ser aglomerativos ou divisivos. Algoritmos aglomerativos começam com cada elemento como um grupo separado e depois realiza a união dos elementos em grupos sucessivamente maiores. Algoritmos divisivos começam com todo o conjunto de elementos e procede dividindo-os em grupos menores.

Segundo Gaertler (2005), agrupamento é um sinônimo para a decomposição de um conjunto de entidades em grupos. Também é dito em Gaertler (2005) que existem dois aspectos principais na tarefa de decomposição: o primeiro envolve questões sobre como encontrar tais decomposições, enquanto o segundo diz respeito à atribuição de qualidade, ou seja, quão boa é a decomposição computada. Originalmente, o agrupamento foi introduzido à pesquisa de mineração de dados como a classificação não supervisionada de padrões em grupos, mas começou a evoluir em uma estrutura mais abrangente (Gaertler, 2005).

O trabalho de Ochi *et al.* (2004) apresenta detalhes sobre o que são problemas de agrupamento, onde consiste em a partir de uma base de dados X, agrupar os elementos de X de forma que os elementos similares fiquem no mesmo grupo e elementos com menor similaridade sejam colocados em grupos distintos. Ainda em Ochi *et al.* (2004) são apontadas algumas aplicações da técnica de agrupamento como: computação visual e gráfica, computação médica, biologia computacional, redes de comunicação, engenharia de transportes, redes de computadores, sistemas de manufatura, entre outras.

3 TRABALHOS RELACIONADOS

Neste Capítulo, são apresentados os trabalhos da literatura relacionados a este estudo. O trabalho de Aalst e Gunther (2007) é fundamentado nos princípios da cartografia para a simplificação e visualização dos processos em formato de espaguete. O trabalho de Song *et al.* (2008) utiliza o conceito de perfis para o agrupamento de traços semelhantes, simplificando, assim, o modelo do processo. O trabalho de Weerdt *et al.* (2013) apresenta uma técnica de agrupamento de traços inspirada em aprendizagem ativa, agrupando os traços de acordo com sua adequação a um determinado modelo do processo. Já em Wang *et al.* (2018), é apresentado o framework Workflow Management (WoMan), que busca melhorar a compreensão e a reutilização dos modelos de processos. O trabalho de Batista e Solanas (2019) introduz o algoritmo *Skip Miner*, que utiliza uma heurística buscando uma abordagem não determinística para ignorar eventos com base em uma probabilidade, a fim de simplificar o modelo de processos. O trabalho de Augusto *et al.* (2019) apresenta o método denominado *Split Miner*, que gera um modelo *Business Process Model and Notation* (BPMN) por meio de algumas etapas de simplificação, partindo de um dado log de eventos.

3.1 *Finding structure in unstructured processes: the case for process mining*

Em Aalst e Gunther (2007), os autores apresentam o algoritmo *Fuzzy Miner*. A proposta tem como objetivo simplificar modelos gerados a partir de logs de eventos da vida real. Tais modelos costumam ser complexos e desestruturados, gerando os "modelos espaguete". A proposta é levantada pelos autores devido ao fato de que mesmo as melhores técnicas de descoberta de processos tendem a produzir "modelos semelhantes ao espaguete".

Baseado em conceitos da cartografia como agregação, abstração, ênfase e customização, os autores exploram como eles podem ser usados para simplificar e visualizar adequadamente processos de alta complexidade e pouco estruturados. Para realizar os procedimentos, os autores identificaram dois fundamentos que apoiam tais decisões: significado e correlação. Com fundamentação nessas duas métricas, os autores esboçam a abordagem de simplificação de processos da seguinte forma:

- Altamente significativo: o comportamento é preservado, ou seja, contido no modelo simplificado;
- Menos significativo, porém altamente correlacionado: o comportamento é agregado, ou

seja, ocultos em grupos dentro do modelo simplificado;

– Menos significativo e pouco correlacionado: o comportamento é abstraído, ou seja, removido do modelo simplificado.

A ideia é que a abordagem proposta pode reduzir e focar o comportamento exibido, empregando os conceitos de agregação e abstração. Assim, com base no modelo simplificado pode-se empregar o conceito de ênfase, destacando o comportamento mais significativo.

Os autores mostram que abordagens clássicas não podem ser usadas para mineração de processos, pois as mesmas tendem a gerar "*overfitting*" (ou seja, gera adequação somente para os dados de treino do modelo). Assim, os autores descrevem que são necessárias técnicas de mineração de processos que equilibrem "*overfitting*" e "*underfitting*". A proposta dos autores, ou seja, o algoritmo *Fuzzy Miner*, é capaz de simplificar os modelos apresentados ao usuário. Por fim, os autores acrescentam que as técnicas do trabalho foram realizadas no contexto da ferramenta de código aberto ProM (Verbeek *et al.*, 2010).

3.2 *Trace clustering in process mining*

O trabalho de Song *et al.* (2008) apresenta uma metodologia de agrupamento de traços semelhantes em um log de eventos, através do conceito de perfis de traços, que são um meio adequado para caracterizar e comparar traços. Os autores caracterizam um perfil como um conjunto de itens que descrevem o traço de uma determinada perspectiva. Assim, considerando as informações típicas contidas nos logs de eventos pode-se derivar diversos perfis. Os exemplos destacados pelo trabalho são:

– **Transição:** os itens deste perfil são relações de segmento direto do traço. Por exemplo, para qualquer combinação de duas atividades <A, B>, o perfil contém um item que mede a frequência com que um evento A é seguido diretamente por outro evento B. A utilidade deste perfil é a de comparar o comportamento de traços.

– **Atributos de caso:** são atributos de dados de caso. Em muitas situações práticas, os traços são anotados com meta-informações, que podem ser comparadas por este perfil.

– **Atributos de evento:** os itens deste perfil são os atributos de dados de todos os eventos no log. Os valores dos itens são medidos de acordo com quantos eventos em um trace são anotados com o respectivo atributo. Uma utilidade deste perfil é a de capturar similaridades de traços através de comparação das meta-informações de seus eventos.

– **Desempenho:** diferente dos outros perfis, este possui um conjunto predefinido de itens.

O tamanho de um traço é definido como seu número de eventos. Quando as informações de carimbo de data/hora estão disponíveis, outros itens medem a duração do caso, e a diferença de tempo mínima, máxima, média, e mediana entre os eventos para cada rastreamento.

Para o agrupamento dos traços, os autores utilizam quatro algoritmos, a saber, K-Means, *Quality Threshold (QT)*, *Agglomerative Hierarchical Clustering (AHC)* e *Self-Organizing Maps (SOM)*. O K-Means divide os dados em k-grupos, e é um algoritmo clássico comumente utilizado nos métodos de partição. O QT foi desenvolvido para a bioinformática, mais especificamente para o agrupamento de genes co-expressos, onde o agrupamento é guiada por um limite de qualidade que determina o diâmetro máximo dos grupos. O AHC gera grupos gradualmente ao mesclar os traços mais próximos, ou seja, grupos menores são mesclados em maiores. O resultado do AHC é geralmente ilustrado como um dendrograma que mostra a hierarquia dos grupos. O SOM é uma técnica de Rede Neural usada para mapear dados de alta dimensão em espaços de baixa dimensão. O objetivo do SOM é agrupar casos semelhantes em determinadas áreas da faixa de valores, onde casos semelhantes são mapeados para o mesmo nó ou nós vizinhos.

Os autores apresentam a metodologia genérica de agrupamento dos traços e o conceito de perfis dos traços. A abordagem foi implementada no ProM e pode ser usada como uma operação genérica de pré-processamento do log de eventos. Como o grupo de rastreamento opera no nível do log de eventos, ele pode ser usado para melhorar os resultados de qualquer algoritmo de mineração de processos. Os autores também ressaltam que a abordagem e a implementação são fáceis de estender, por exemplo, adicionando perfis específicos de domínio ou algoritmos de agrupamento adicionais.

3.3 *Active trace clustering for improved process discovery*

O trabalho de Weerdt *et al.* (2013) busca tratar o problema de processos altamente incompreensíveis através de uma técnica de agrupamento de traços. Assim, buscam agrupar traços não porque exibem comportamento semelhante, mas sim porque se encaixam bem em um determinado modelo do processo. A proposta do trabalho difere significativamente das abordagens anteriores. O ponto observado pelos autores é que as técnicas anteriores disponíveis sofrem de grande divergência entre os vieses de agrupamento e de avaliação. Assim, assumem que ao empregar uma abordagem inspirada em aprendizado ativo as divergências entre os vieses

são resolvidas.

A abordagem inspirada em aprendizagem ativa é chamada pelos autores de ActiTraC. O ActiTraC é baseado em alguns princípios de aprendizado ativo no campo de aprendizado de máquina. A ideia do aprendizado ativo é que um algoritmo de aprendizado de máquina pode obter melhores resultados com menos dados de treinamento caso possa escolher os dados com os quais ele aprende. A técnica também é considerada gulosa, pois tenta maximizar um dado critério localmente, sem garantir uma solução ótima global.

O ActiTraC consiste em 3 fases distintas, a saber, seleção, antecipação e resolução dos traços residuais. A seleção busca selecionar traços iterativamente usando uma estratégia de amostragem seletiva. O objetivo é adicionar o traço ao conjunto de instâncias já selecionado para avaliação do modelo de processo descoberto a partir desse novo subconjunto do log de eventos. Se a precisão do modelo permanecer suficientemente alta, o traço é selecionado para o grupo atual e o processo é repetido. A etapa de antecipação busca adicionar instâncias restantes que se encaixem perfeitamente no modelo criado na etapa de seleção. E a etapa de resolução de traços residuais busca tratar os traços que podem ter sobrado por não se adequarem aos grupos modelados, agrupando eles em um novo grupo ou mantendo eles nos grupos atuais. Os autores, em uma avaliação utilizando quatro logs de eventos complexos da vida real, mostram que o ActiTraC supera significativamente as técnicas de agrupamento de traços anteriormente disponíveis.

3.4 *A novel trace clustering technique based on constrained trace alignment*

O trabalho de Wang *et al.* (2018) propõe uma técnica de agrupamento baseada no alinhamento restrito de traços, adaptando duas estratégias apropriadas na perspectiva de mineração de processos. O procedimento começa pela identificação de sequências causais típicas dos traços que representam a estrutura fundamental do processo. Estas sequências são então utilizadas como restrições para assegurar a prioridade de atividades importantes nos traços. Em seguida, são propostas duas estratégias de agrupamento alinhadas com a perspectiva de mineração de processos.

A primeira estratégia é o Agrupamento Hierárquico Aglomerativo (AHA), escolhido por sua flexibilidade no nível de abstração, proporcionando uma visão geral do processo complexo. O AHC permite uma organização hierárquica que reflete a estrutura do processo. O método organiza em um dendograma cujo seu topo representa o único grupo que contém todo o

processo minerado, onde, geralmente o mesmo é desestruturado quando obtido de ambientes com alta flexibilidade. Já as partes inferiores significam os grupos correspondentes a cada traço cujo processos minerados são certamente mais estruturados.

A segunda estratégia é o Agrupamento Espectral, que é recomendado pela sua capacidade de sugerir o número de grupos correspondentes a diferentes níveis de abstração genérica. Isso ajuda a capturar a complexidade do processo de uma maneira mais adaptável e eficiente. Os autores utilizam este método porque ele fornece uma boa recomendação sobre o número de grupos que podem conduzir para um nível de abstração genérico.

Portanto, o procedimento busca garantir a representação adequada das atividades essenciais nos rastreamentos, utilizando tanto a flexibilidade hierárquica do AHC quanto a capacidade adaptativa do Agrupamento Espectral para fornecer uma visão abrangente e precisa do processo em questão.

Os autores ao realizarem experimentos com logs de eventos reais mostram que a técnica tem desempenho superior em termos de construir modelos de processos com menor complexidade e mais compreensíveis. Dessa forma, concluem que a abordagem apresentada preserva de maneira mais efetiva as informações mais significativas do processo, resultando em medidas de similaridade mais precisas.

3.5 *Process model modularization by subprocess discovery*

O trabalho de Angelastro e Ferilli (2020) propõe uma abordagem inovadora para a descoberta automática de modelos de processos, representados como subprocessos, em processos não estruturados. A implementação dessa abordagem é realizada através do framework Workflow Management (WoMan), visando melhorar a compreensão e a reutilização dos modelos de processos. Segundo os autores o WoMan é uma estrutura de Mineração de Processo Declarativa baseada em técnicas e representações de Lógica de Primeira Ordem para proporcionar maior expressividade, permitindo a descrição de informações contextuais através de relacionamentos.

É descrito que o framework se destaca por ser totalmente incremental, possibilitando o refinamento contínuo de um modelo existente com a incorporação de novos casos e até aprendendo a partir de um modelo vazio e de um único caso. Essa característica diferencia a abordagem, pois outras técnicas exigem um número significativo de casos para iniciar a aprendizagem. O processo se divide em etapas sequencias de passos, ou seja, cada etapa recebe a saída da etapa anterior como entrada e fornece sua saída para próxima. As etapas são as

seguintes:

1. Transformação de um log de eventos L em um banco de dados D de objetos de dados gráficos.
2. Aplica mineração frequente de subgrafos em D, utilizando um derterminado limiar de suporte a fim de obter um conjunto S de subgrafos.
3. Repara cada s em S para um subgrafo s' que possui nós de origem e destino únicos. Resultando no S' que é o conjunto resultante de padrões reparados.
4. Análise de grupos de S', para particioná-lo em k clusters, cada um dos quais deve conter subgrafos semanticamente similares. Ou seja, dadas duas subgrafos, quanto mais diferentes forem, menos provável é que representem o mesmo subprocesso. Resultando na clusteruzação C.
5. Modularização, mediante a transformação de cada c em C em um módulo (ou seja, um modelo de subprocesso). Obtendo o conjunto de modelos de subprocessos M é escolhido um subconjunto M' de módulos não sobrepostos para modularização.
6. Abstração de M. Cada m em M' abstrai uma porção (diferente) de M ao substituir as tarefas e transições correspondentes em M por uma tarefa não atômica.

Na etapa de analise de grupos os autores adotaram dois algoritmos, a saber, Agrupamento Hierárquico Aglomerativo (AHA) e o DBSCAN (em português significa agrupamento especial baseado em densidade de aplicações com ruídos). Os resultados experimentais foram baseados em dados reais e modelos sintéticos, onde demonstram promissoras perspectivas para a eficácia da proposta. Os autores destacam que utilizando o AHA, com um método de agrupamento adequado, é possível fornecer um número de grupos que refletem melhor o número real de subprocessos padrão do que ao utilizar o algoritmo DBSCAN.

3.6 *Skip miner: towards the simplification of spaghetti-like business process models*

O trabalho de Batista e Solanas (2019) apresenta o algoritmo *Skip Miner* para simplificação de modelos de processo do tipo “espaguete”. O algoritmo implementa uma estratégia de simplificação integrada que tenta reduzir a complexidade dos modelos de processos. A estratégia utiliza uma heurística que busca uma abordagem não determinística para ignorar eventos com base em uma probabilidade. Dessa forma o *Skipe Miner* propõe não considerar todas as transições entre eventos consecutivos em modelo de processos. Em vez disso os autores introduzem um parâmetro que determina quantos eventos consecutivos podem ser ignorados

durante a fase de descoberta.

Os autores descrevem que cada atividade no log de eventos tem uma probabilidade de ser ignorada. Formalmente, a probabilidade de omissão de uma atividade a varia de 0 a 1. Quando $\varepsilon(a) = 0$, nenhum evento é ignorado após a atividade a, e quando $\varepsilon(a) = 1$, as próximas δ atividades são ignoradas após a atividade a, sendo o próximo evento considerado na posição δ + 1. A equação a seguir descreve o calculo da probabilidade $\varepsilon\ (a)$:

$$\varepsilon(a) = 1 - \frac{1}{x} \tag{3.1}$$

onde x é o número de atividades diferentes imediatamente após a. Dessa forma, uma atividade a tem maior probabilidade de ignorar eventos quando o número de atividades diferentes que seguem a é alto. Os autores destacam que é comum observar múltiplos caminhos nos modelos de processos espaguete após visitar um nó específico devido a diferentes instâncias de execução do processo. Os testes do *Skip Miner* foram realizados em um log de eventos médico real e comparado com outras duas estratégias simplificação da literatura. Os resultados apresentados no trabalho, segundo os autores, demonstram que o *Skip Miner* consegue equilibrar a simplificação e qualidade dos modelos descobertas.

3.7 *Split miner: automated discovery of accurate and simple business process models from event logs*

O trabalho de Augusto *et al.* (2019) apresenta o método denominado *Split Miner*. É dito que a técnica produz modelos de processo simples e com baixa complexidade de ramificação e possui aptidão (referente aos modelos de processos que se ajustam mal ao log de eventos) e precisão (referente à generalização excessiva dos modelos de processos) consistentemente altas e equilibradas. O *Split Miner* produz um *Business Process Model and Notation* (BPMN) partindo de um log de eventos e pelas seguintes etapas:

1. Descoberta de *self-loops* e *short-loops* (segundo os autores os mesmos são conhecidos por causar problemas em métodos de descoberta de processos baseados em DFG) e construção do *Directly-Follow Graph* (DFG).
2. Descoberta de simultaneidade entre duas atividades a e b, onde são consideradas simultâneas se:

$$|a \rightarrow b| > 0 \wedge |b \rightarrow a| > 0 \tag{3.2}$$

$$|a \leftrightarrow b| + |b \leftrightarrow a| = 0 \tag{3.3}$$

$$\frac{||a \rightarrow b| - |b \rightarrow a||}{|a \leftrightarrow b| + |b \leftrightarrow a|} < \varepsilon(\varepsilon \in [0,1]) \tag{3.4}$$

3. Filtragem das atividades que devem satisfazer três propriedades. A primeira deve garantir que cada atividade no DFG deve estar em um caminho desde a atividade inicial única até a atividade final única. A finalidade da propriedade descrita é de garantir um modelo sólido. Em segundo, para cada atividade, o seu caminho da origem ao destino deve ser aquele com capacidade máxima, onde a propriedade se refere à frequência da aresta menos frequente no caminho. Tal propriedade é descrita como destinada a maximizar a aptidão, uma vez que a capacidade de um caminho corresponde ao número de traços que podem ser reproduzidos no caminho. Por último, o número de arestas do DFG deve ser mínimo. Tal propriedade tem por finalidade maximizar a precisão.
4. Transformar o DFG filtrado para BPMN.
5. Descobrir os *Splits*. A tarefa é baseada na ideia que as tarefas que seguem diretamente o mesmo gateway (elo entre os caminhos) dividido são simultâneas ao mesmo conjunto de tarefas que não seguem diretamente tal gateway. Sabendo quais tarefas são sucessoras do mesmo gateway pode-se identificar o tipo de gateway verificando se seus sucessores são correspondentes ou mutuamente exclusivos.
6. Descobrir os *Joins*. Consiste em descobrir os gateways de junção. A tarefa é realizada a partir da *Refined Process Structure Tree* (RPST) do modelo BPMN atual.
7. Minimizar os *OR-joins*. A tarefa busca evitar a colocação de *OR-joins* triviais dentro de ligações homogêneas, porém não evita o abuso em casos heterogêneos.

Os autores realizaram testes em dados de 12 logs de eventos reais. Os resultados apresentados, segundo os autores, mostram que *Split Miner* é um avanço em direção a métodos escaláveis e robustos para descoberta automatizada de modelos de processos de negócios. Os dados mostram que a técnica produz modelos que são comparáveis em termos de tamanho e complexidade de fluxo de controle aos produzidos pelo IM e pelo *Evolutionary Tree Miner*, que nos testes produziram os melhores resultados ao longo dos testes na maioria dos logs de eventos.

3.8 *An event-level clustering framework for process mining using common sequential rules*

No trabalho de (Tariq *et al.*, 2021), é destacado que, embora técnicas de abstração de logs de eventos tenham sido desenvolvidas para agrupar atividades de baixo nível em atividades de nível superior, elas podem ignorar detalhes críticos do processo. Além disso, embora as técnicas de agrupamento de traços tenham sido amplamente utilizadas na literatura, o agrupamento de nível de evento ainda não foi considerado para mineração de processos. Os autores propõem um novo framework chamado Identificador de Eventos Comuns (IEC) para identificar grupos de nível de evento em um log de processo de negócios, dividindo-o em vários sublogs com base na similaridade das sequências entre eventos.

O framework IEC, segundo os autores, é dividido em 4 etapas. Na Etapa 1, trata-se da preparação do log do processo. Dado um log de eventos do processo, ele forma uma lista de eventos *L*. Na Etapa 2, *L* é percorrida com o incremento de um único passo, dado como tamanho da janela *n*, e regras sequenciais para os eventos correspondentes a *L* são extraídas. Na Etapa 3, são consideradas as regras sequenciais geradas em cada iteração para avaliação de uma correlação residual. A correlação é detectada entre diferentes grupos de regras geradas pelo algoritmo *Classification based on Association* (CBA). Um limiar fixo de 10% de correlação é considerado para distinguir a comunidade de regras entre os grupos. Os autores destacam que mantêm um valor baixo de limiar para garantir que a menor diferença nas regras entre os grupos seja detectada de forma eficaz. Finalmente, na Etapa 4, são identificadas as colunas na lista de eventos que fazem parte do Segmento de Eventos Comuns (SEC). O residual de discriminação representa a frequência de ocorrência de qualquer subsequência específica.

O IEC é aplicado a um log de eventos de telecomunicações do mundo real, e os resultados são comparados com duas técnicas de agrupamento de traços conhecidas da literatura. Os resultados, de acordo com os autores, mostram alta precisão de agrupamento e melhoria na qualidade dos modelos de processo resultantes, sem a necessidade de abstração de logs complexos. Além disso, os autores também descrevem que as técnicas propostas melhoraram os resultados de descoberta e conformidade de processos para o log de eventos fornecido.

3.9 *Event log preprocessing for process mining: a review*

O trabalho de (Marin-Castro; Tello-Leal, 2021) aborda a importância do pré-processamento de logs de eventos na mineração de processos, destacando a influência da qualidade dos dados

de entrada nos resultados finais. Ele apresenta uma revisão abrangente das técnicas de pré-processamento mais representativas, categorizando-as em técnicas de transformação e técnicas de detecção-visualização. As principais técnicas discutidas incluem:

- Técnicas de Transformação: Essas técnicas visam modificar a estrutura original do log de eventos para melhorar sua qualidade. Isso inclui técnicas como filtragem, baseadas em padrões, agrupamento de traços e técnicas baseadas em tempo.
- Técnicas de Detecção-Visualização: Estas técnicas têm como objetivo identificar, agrupar e isolar eventos ou traços que possam gerar problemas na qualidade do log de eventos. Isso inclui técnicas como agupamento e identificação de padrões.

O estudo examina os desafios enfrentados por essas técnicas, como a diversidade dos logs de eventos e a necessidade de lidar com dados ruidosos, ausentes, duplicados ou irrelevantes. Os autores também discutem a importância do pré-processamento na descoberta e conformidade de modelos de processo, destacando como técnicas adequadas podem reduzir a complexidade dos modelos descobertos e melhorar a correspondência entre o log de eventos e o modelo. Os resultados da revisão indicam que técnicas como agrupamento e filtragem de traços são as mais frequentemente utilizadas no pré-processamento de logs de eventos. Além disso, o estudo destaca a necessidade de desenvolver métricas para avaliar a qualidade do log de eventos e frameworks para avaliar o impacto do pré-processamento nos modelos de processo.

3.10 *Exploring decomposition for solving pattern mining problems*

No estudo de (Djenouri *et al.*, 2021) é apresentado o framework *Clustering-Based Pattern Mining* (CBPM) como uma solução promissora para lidar com grande conjunto de dados. O CBPM utiliza técnicas de agrupamento para particionar os traços em grupos homogêneos, buscando simplificar o processo de descoberta de padrões relevantes.

Para avaliar o desempenho do Framework CBPM, os autores realizam experimentos utilizando conjuntos de dados de diferentes tamanhos e características. São implementadas duas estratégias de mineração de padrões, uma abordagem aproximada e uma abordagem exata, e avaliadas suas eficácias em termos de tempo de execução e uso de memória. Além disso, os autores desenvolveram uma implementação paralela do framework na GPU e investigaram seu impacto no desempenho em grandes conjuntos de dados. Segundo os autores, os resultados experimentais demonstram que o Framework CBPM supera as soluções tradicionais de mineração de padrões em termos de tempo de execução e uso de memória. Além disso, a implementação paralela na

GPU apresenta um aumento significativo na velocidade de processamento, especialmente em grandes conjuntos de dados.

Os autores concluem que o Framework CBPM é uma abordagem eficaz e versátil para a mineração de padrões em grandes conjuntos de dados. Ainda destacam que o CBPM demonstra um grande potencial para aplicações em diversos domínios, incluindo análise de trajetórias, inteligência empresarial e mineração de dados financeiros.

3.11 *Expert-driven trace clustering with instance-level constraints*

No trabalho (Koninck *et al.*, 2021) são apresentadas duas técnicas de agrupamento de traços que buscam mitigar a dificuldade de avaliação e justificação das soluções por especialistas de domínio. Essas técnicas, de acordo com os autores, são capazes de incorporar conhecimento especializado na forma de restrições de nível de instância. Em uma avaliação experimental extensiva com dois conjuntos de dados do mundo real, os autores demonstram que suas técnicas são capazes de produzir soluções de agrupamento mais justificáveis sem um impacto substancial em sua qualidade.

A primeira técnica, chamada ConDriTraC, é uma extensão de uma abordagem anterior que permitia ao especialista fornecer um conjunto completo de restrições para o agrupamento de traços. No entanto, a ConDriTraC foi adaptada para aceitar restrições como entrada, em vez de uma solução completa de agrupamento, tornando-a mais flexível e prática.

A segunda técnica, ConDriTraC-MRA, é uma versão adaptada de uma técnica de modelagem de processo-agnóstica que utiliza características extraídas dos traços para realizar o agrupamento. Essa abordagem também foi modificada para aceitar restrições como entrada, tornando-a capaz de levar em consideração o conhecimento especializado durante o agrupamento.

Na avaliação experimental, os autores utilizaram dois conjuntos de dados do mundo real e compararam suas técnicas com outras abordagens de agrupamento de traços. Os resultados, segundo os autores, mostraram que suas técnicas superaram significativamente outras abordagens de agrupamento, especialmente quando há restrições de conhecimento especializado disponíveis.

Os autores destacam a importância de incorporar conhecimento especializado para melhorar a compreensão e a qualidade das soluções de agrupamento. Além disso, também discutiram as implicações práticas de suas técnicas e sugeriram possíveis direções para futuras pesquisas nessa área. Em conclusão, o estudo apresenta contribuições significativas para o campo de agrupamento de traços, oferecendo abordagens inovadoras que permitem a integração de

conhecimento especializado no processo de agrupamento.

3.12 *Ai-empowered process mining for complex application scenarios: survey and discussion*

O trabalho de (Folino; Pontieri, 2021) está fundamentado no problema de que os processos pouco estruturados apresentam diversos desafios. Os autores descrevem que, nesses casos, as chances de obter resultados de baixa qualidade são altas, sendo assim, é necessário um grande esforço para realizar projetos de mineração de processos (MP). É dito que a maior parte dos esforços é gasta tentando encontrar diferentes maneiras de selecionar e preparar os dados de entrada para as tarefas de MP. O trabalho então discute duas estratégias gerais baseadas em inteligência artificial (IA) que podem melhorar e facilitar a execução de tarefas de MP nesses ambientes. A primeira utiliza conhecimentos de domínio explícitos do usuário. Já a segunda realiza uma ou várias tarefas de aprendizado/inferência auxiliares em conjuntos de tarefas.

O estudo também apresenta uma revisão sistemática da literatura e uma classificação taxonômica das obras relevantes revisadas, concentrando-se em fortalecer as tarefas clássicas de PM por meio de capacidades complementares de representação do conhecimento, aprendizado e inferência. O estudo confirmou que ambos os tipos de estratégias constituem um meio valioso para melhorar a qualidade dos resultados que podem ser alcançados em registros incompletos, de baixo nível e heterogêneos, e para permitir algum nível de suporte operacional para processos pouco estruturados.

3.13 *Complex process modeling in process mining: A systematic review*

O estudo de (Imran *et al.*, 2022) busca realizar uma revisão sistemática da literatura sobre complexidade na mineração de processos em seis bases de dados acadêmicas proeminentes. Os autores descrevem que essa revisão teve como objetivo identificar uma definição explícita de complexidade, seus principais fatores contribuintes e seu impacto nos resultados da mineração de processos. Por meio dessa revisão, de acordo com os autores, foram identificadas diversas matrizes de complexidade de processos e seus contextos de aplicação. A revisão sistemática da literatura realizada no trabalho foi sobre complexidade no domínio da mineração de processos, focando em artigos publicados entre 2012 e 2022, onde foram pesquisados seis bancos de dados de pesquisa conhecidos. Os autores formularam uma taxonomia detalhando diferentes abordagens e sub-abordagens para lidar com a complexidade e identificar perspectivas para

pesquisas futuras.

Os resultados da revisão do trabalho indicam que a causa raiz do problema de complexidade são as características dos processos. É destacado que em algumas organizações, os processos são mantidos flexíveis ou não estruturados, o que resulta em logs de eventos ruidosos e comportamentos significativamente variados registrados no log. Os autores também descrevem que embora os resultados indiquem que a alta flexibilidade e variação no processo causem complexidade nos modelos de processo, às vezes, uma pequena variação na execução do processo também causa esse problema. Com base no contexto da execução do processo, é dito que a mudança de posição de uma única atividade também pode resultar em complexidade e imprecisão dos modelos de processo. Na perspectiva dos autores, a literatura negligenciou o tratamento de outra causa significativa de complexidade do processo, ou seja, a modelagem de múltiplos subprocessos em um único modelo, o que também pode se relacionar com o nível de execução detalhado do processo. Os resultados ainda indicam que o agrupamento de traços permaneceu a técnica mais popular para lidar com a complexidade. No entanto, foi observado pelos autores que, em vez de aplicar a abordagem de simplificação direta, as técnicas de agrupamento simplificam implicitamente os modelos de processo dividindo o log de eventos em grupos com base em certas características aleatórias. No entanto, foi inesperado ver que o agrupamento foi realizado principalmente a partir de uma perspectiva de dados em vez de uma perspectiva de processo, ou seja, se os grupos gerados apresentam comportamentos corretos.

Em resumo, os autores descrevem que foi identificado como o problema da complexidade do processo é percebido em diferentes estudos, quais fatores contribuem para isso e como a complexidade é analisada e prevenida. Também são propostas a identificação de lacunas na pesquisa e direções futuras de pesquisa. As descobertas revelam que a flexibilidade no processo, o nível detalhado e o ruído nos logs são os principais contribuintes para a complexidade do processo. Além disso, constatou-se que o problema da complexidade é resolvido usando quatro abordagens prospectivas: agrupamento, abstração, remoção de ruído e mineração de padrões. Diferentes métricas usadas para essas medidas foram identificadas. Também foi observado que a ênfase na análise de complexidade permaneceu nas medidas de complexidade comportamental. Ao mesmo tempo, menos importância é dada à complexidade estrutural, que se relaciona diretamente com a compreensibilidade do modelo de processo. Por fim, várias lacunas na pesquisa e direções futuras de pesquisa também são apresentadas.

3.14 *An approximate inductive miner*

No trabalho de (Detten *et al.*, 2023), é proposto o framework *Approximate Inductive Miner* (AIM) para oferecer uma maneira automatizada de descobrir modelos sólidos com complexidade de tempo polinomial, sem necessidade de pré-processamento ou entrada obrigatória de parâmetros. O AIM utiliza o IM em conjunto com uma técnica de agrupamento para identificar de forma recursiva estruturas no log de eventos. O framework também aplica uma otimização de parâmetro aproximada para indicar dinamicamente um parâmetro adequado.

O AIM utiliza uma estratégia recursiva para descobrir uma árvore de processos com base em um log de eventos fornecido e no espaço de parâmetros de filtro. Em cada passo de recursão, busca-se o melhor corte e parâmetro de filtro. Para selecionar tal corte, o AIM primeiro determina o espaço de estados de cortes potenciais em todo o espaço de parâmetros disponível e calcula uma estimativa de qualidade para cada um deles. Em segundo lugar, o AIM ajusta essas estimativas de qualidade de corte para levar em conta a perda de informação induzida durante a filtragem. Em terceiro lugar, o AIM poda esse espaço de estados de forma eficiente com a aplicação de técnicas de agrupamento para selecionar um corte.

Os autores comparam o AIM com outros algoritmos de descoberta existentes em relação a logs de eventos sintéticos e da vida real, onde também avaliam a qualidade da sugestão de parâmetro integrada. Foi constatado, segundo os autores, que o AIM, por si só, produz modelos sólidos com baixa complexidade de fluxo de controle e alta precisão, mesmo em registros de eventos complexos. Além disso, os autores também afirmam que o AIM é capaz de lidar com uma vasta gama de propriedades de registro de eventos, como comportamento infrequente e incompleto, sem exigir entrada de parâmetros humanos ou filtragem prévia.

3.15 *Interactive trace clustering to enhance incident completion time prediction in process mining*

O estudo de (Neubauer *et al.*, 2022) introduz o *Interactive Trace Clustering*, uma estratégia que visa melhorar a análise de processos de negócios ao combinar o conhecimento especializado dos profissionais com algoritmos de agrupamentos automatizados. A principal motivação para essa abordagem, segundo os autores, é a necessidade de produzir resultados de agrupamentos mais relevantes e alinhados com as necessidades do negócio, incorporando o conhecimento humano no processo.

Para testar a eficácia dessa abordagem, o estudo conduziu uma prova de conceito usando dados reais de um sistema de rastreamento de problemas, focando na previsão do tempo de conclusão de incidentes. Durante a prova de conceito, os especialistas interagiram com os resultados dos agrupamentos e forneceram feedbacks, resultando na identificação de regras de restrição que foram aplicadas ao algoritmo de agrupamento. Os resultados mostraram melhorias na precisão das previsões do tempo de conclusão dos incidentes, indicando que a incorporação do conhecimento humano no processo de agrupamento pode fornecer insights valiosos e aumentar a relevância dos resultados obtidos.

As contribuições do estudo, de acordo com os autores, incluem a introdução de uma abordagem inovadora para análise de processos de negócios que integra eficazmente o conhecimento humano com algoritmos de agrupamento automatizados. Além disso, o estudo destaca a importância da interação entre especialistas e algoritmos automatizados para melhorar a qualidade das análises de processos. Em conclusão, o estudo demonstra que o *Interactive Trace Clustering* pode ser implementado com sucesso sem comprometer significativamente a qualidade do agrupamentp, fornecendo resultados mais relevantes e úteis para as organizações.

3.16 Comparação dos trabalhos

No *Fuzzy Miner* (Aalst; Gunther, 2007) uma das semelhanças é a busca por simplificar processos de alta complexidade e pouco estruturados através de agrupamentos. A grande diferença é que a abordagem utiliza princípios da cartografia. A utilização destes princípios possibilita que parte das informações originais dos dados possam ser abstraídas. Já na proposta deste trabalho, todas as informações são mantidas, porém reorganizadas.

Song *et al.* (2008) agrupa os dados de acordo com seus perfis através de diversas técnicas conhecidas, buscando a simplificação dos modelos de processo. Uma das diferenças em relação a este trabalho são os critérios que a técnica utiliza para agrupar os dados de acordo com seus perfis, pois leva em consideração as transições, atributos de caso, atributos de eventos e desempenho dos traços para construir os grupos.

O ActiTraC (Weerdt *et al.*, 2013) apresenta uma semelhança com este trabalho ao buscar maximizar um determinado critério localmente para a escolha dos traços que comporão cada grupo. Uma das diferenças da abordagem é a utilização de aprendizagem ativa, comumente empregada em algoritmos de aprendizado de máquina, que visa obter melhores resultados com menos dados de treinamento.

Wang *et al.* (2018) busca também simplificar os modelos de processos, mas difere deste trabalho ao realizar os agrupamentos baseados no alinhamento restrito dos traços. Outra diferença é a utilização dos algoritmos de Agrupamento Hierárquico Aglomerativo e Agrupamento Espectral, que não são abordados na proposta deste trabalho.

Já o framework WoMan (Angelastro; Ferilli, 2020) busca simplificar os modelos de processos com base em técnicas e representações de lógica de primeira ordem, o que não é abordado neste trabalho. Uma das semelhanças é que o WoMan possibilita o refinamento contínuo em um modelo de processos existente por ser incremental. A similaridade com este trabalho se deve ao fato da melhoria do modelo de processos a partir da identificação de problemas observados nos grupos que são construídos.

O *Skip Miner* (Batista; Solanas, 2019) difere deste trabalho ao simplificar os modelos por meio de uma técnica que permite ignorar atividades de acordo com uma probabilidade na construção do modelo de processos. A proposta de não considerar todas as transições entre eventos consecutivos para a construção dos modelos de processos não é utilizada neste trabalho. Uma das semelhanças entre o *Skip Miner* e este trabalho é que ambos apresentam um único modelo de processos ao final de sua execução.

No *Split Miner* (Augusto *et al.*, 2019), uma das semelhanças com este trabalho também é o fato de produzir somente um modelo de processos ao final de sua execução. A principal diferença entre o *Split Miner* e o trabalho aqui apresentado são as etapas que o *Split Miner* utiliza diretamente na construção do BPMN a partir do log de eventos.

O estudo de (Tariq *et al.*, 2021)propõe o Identificador de Eventos Comuns (IEC), um framework para identificar grupos de eventos em logs de processos de negócios com base na similaridade de sequências entre eventos. Já neste trabalho os agrupamentos levam em consideração a distância de edição entre os traços de um log de eventos.

No estudo de Marin-Castro e Tello-Leal (2021) é feita uma revisão das técnicas de pré-processamento de logs de eventos na mineração de processos, categorizando-as em técnicas de transformação e detecção-visualização. O estudo reforça que as abordagens deste trabalho seguem os contextos mais estudados para resolução dos problemas que são encontrados ao agrupar traços em logs de eventos complexos, reforçando também a importância do pré-processamento dos dados de entrada.

(Djenouri *et al.*, 2021) apresenta o framework *Clustering-Based Pattern Mining* (CBPM) como uma solução para lidar com grandes conjuntos de dados. A ideia de particionar

os dados de acordo com padrões também se assemelha a estratégia de agrupamentos dos traços explorada neste trabalho.

No trabalho de Koninck *et al.* (2021), são apresentadas duas técnicas de agrupamento de traços, ConDriTraC e ConDriTraC-MRA, que integram conhecimento especializado na forma de restrições de nível de instância para facilitar a avaliação e justificação por especialistas de domínio. O estudo difere significativamente deste trabalho, tendo em vista que na proposta aqui apresentada não existe envolvimento de especialista para definir restrições no algoritmo.

Folino e Pontieri (2021) abordam os desafios de mineração de processos (MP) em cenários complexos e pouco estruturados, destacando a necessidade de selecionar e preparar dados de entrada adequados. Imran *et al.* (2022) também realiza uma revisão sistemática sobre a complexidade na mineração de processos, identificando fatores contribuintes e estratégias para lidar com ela. Os estudos reforçam a importância do problema abordado neste trabalho ainda de acordo com uma revisão sistemática da literatura de trabalhos similares.

Em (Detten *et al.*, 2023) é apresento o framework *Approximate Inductive Miner* (AIM), que automatiza a descoberta de modelos sólidos em processos complexos. O AIM e este trabalho tem grande similaridade, pois ambos utilizam o *Inductive Miner* em conjunto com técnicas de agrupamento para identificar estruturas recursivamente nos logs de eventos.

O estudo de Neubauer *et al.* (2022) introduz o *Interactive Trace Clustering*, uma estratégia que combina o conhecimento especializado dos profissionais com algoritmos de agrupamento automatizados para melhorar a análise de processos de negócios. Apesar de o estudo realizar processos automatizados assim como os deste trabalho, ambos diferem no quesito de utilização de conhecimento especializado, pois a técnica não é abordada neste estudo.

É notável que a maioria dos trabalhos utilizam técnicas ou princípios de agrupamento para gerar e buscar simplificar os modelos de processos dado um log de eventos. Os mesmos princípios são mantidos neste trabalho. Outro ponto fundamental é que parte dos trabalhos busca simplificar os modelos abstraindo parte das informações originais. Neste trabalho, todas as informações são levadas em consideração para a obtenção do resultado final. Outra grande diferença é que são poucos os trabalhos que apresentam um único modelo de processos após a execução de seus procedimentos. A maioria agrupa os traços em modelos de processos distintos. Os modelos de processos deste trabalho buscam manter uma mesma estrutura, agregando as partes agrupadas a uma única estrutura base.

A Tabela 4 apresenta quais trabalhos realizam agrupamentos para a construção dos

modelos de processos, descrevendo quais estratégias são utilizadas. Também são evidenciados os princípios de cada um dos trabalhos.

Tabela 4 – Comparação dos trabalhos.

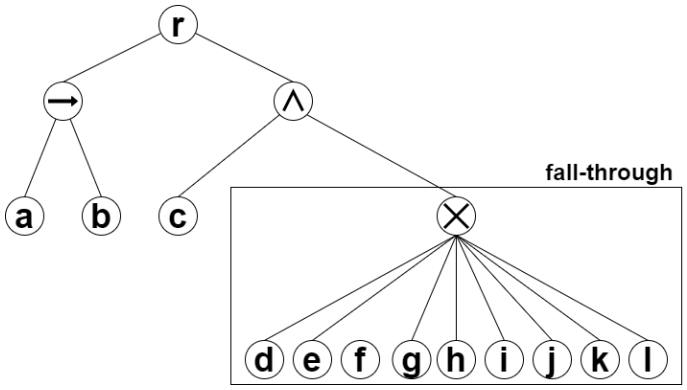
Trabalhos	**Agrupamento**	**Princípios**
Aalst e Gunther (2007)		Cartografia
Song *et al.* (2008)	Semelhança dos traços	Perfil
Weerdt *et al.* (2013)	Aprendizagem ativa	Aprendizado de máquina
Wang *et al.* (2018)	AHA ou Espectral	Sequencias causais
Angelastro e Ferilli (2020)	AHA ou DBSCAM	Lógica de primeira Ordem
Batista e Solanas (2019)		Probabilidade de ignorar eventos
Augusto *et al.* (2019)		Remover redundâncias
Tariq *et al.* (2021)	IEC	Similaridade de sequências
Marin-Castro e Tello-Leal (2021)		
Djenouri *et al.* (2021)	CBPM	Particionamento por padrões relevantes
Koninck *et al.* (2021)	ConDriTraC e ConDriTraC-MRA	Conhecimento especializado
Folino e Pontieri (2021)		
Imran *et al.* (2022)		
Detten *et al.* (2023)	AIM	Agrupamentos recursivos
Neubauer *et al.* (2022)	*Interactive Trace Clustering*	Conhecimento especializado
IM_Cluster	Distância de edição	Reconstrução de subárvores

Fonte: Elaborada pelo autor (2024).

4 *INDUCTIVE MINER COM AGRUPAMENTO DE SUBLOGS DE NÓS FALL-THROUGH*

Neste Capítulo é apresentada a solução proposta por este trabalho para o agrupamento de traços de logs de eventos de processos espaguete. O algoritmo busca identificar os nós que sejam *fall-through* em uma árvore de processos a partir do nó raiz *r*. Na Figura 8, é apresentado um exemplo de árvore de processos que a partir de um nó raiz *r* com 2 filhos, uma das ramificações deles é *fall-through* (destacado pelo retângulo).

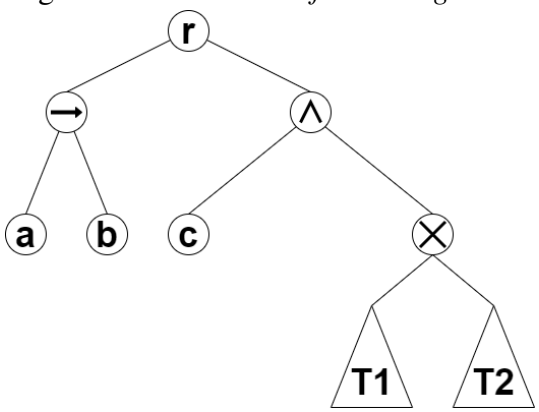
Figura 8 – Árvore com *fall-through*.



Fonte: Elaborada pelo autor (2024).

O procedimento adotado para remover o *fall-through* consiste em agrupar o sublog correspondente ao nó *fall-through*, produzindo duas subárvores $T1$ e $T2$, e em seguida conectá-las através de um operador de escolha exclusiva $\times$ e fazer com que a árvore gerada seja colocada como filha do nó *r* no lugar do nó original. Um sublog em um log de processos é uma parte específica do log original que contém um subconjunto de eventos ou casos do processo. O procedimento também é feito de forma recursiva para os filhos $T1$ e $T2$, para garantir que os mesmos não apresentem *fall-through* com um determinado tamanho. A estrutura final do exemplo pode ser observada na Figura 9.

Figura 9 – Árvore sem *fall-through*.



Fonte: Elaborada pelo autor (2024).

Este capítulo segue apresentando a métrica de simplicidade proposta para medir a qualidade dos modelos obtidos por um dado log de eventos. Em seguida, é apresentada uma descrição do algoritmo IM_Cluster.

4.1 Métrica de simplicidade proposta

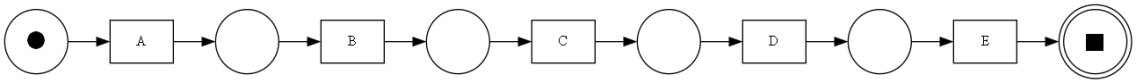
Para medir a complexidade dos modelos gerados, uma das formas é avaliar o grau dos lugares (*GP*) de uma dada rede de Petri gerada a partir de um log de eventos obtido por um dos algoritmos analisados. A justificativa para o uso dessa métrica deve-se ao fato da grande quantidade de arcos de entrada e saída observados nos lugares nos modelos que apresentam *fall-throughs*. Vale destacar que um dado lugar que possui uma diferença significativa entre os arcos de entrada e de saída possui maior penalidade de acordo com a equação proposta nessa seção. A formulação adapta a métrica do trabalho de Blum (2015), mas com a diferença de penalizar modelos que apresentem "empilhamento" em suas atividades. Outro ponto importante é que a métrica penaliza a diferença significativa entre os graus de entrada e saída doslugares. Para mensurar a complexidade dos modelos em relação à quantidade de arcos de entrada e saída de lugares em uma de rede de Petri N, utiliza-se a seguinte equação nos modelos analisados neste trabalho:

$$GP(N) = \frac{\sum_{i=1}^{n} (arcs\_in[i]^2 + arcs\_out[i]^2)}{n} \tag{4.1}$$

onde n representa o total de lugares que existem na rede de Petri N, e $arcs\_in[i]$ e $arcs\_out[i]$ são os totais de arcos de entrada e saída do i-ésimo lugar observado, respectivamente. Por exemplo, utilizando o modelo de rede de Petri da Figura 10, o grau dos lugares, levando em consideração os lugares de início e fim são desconsiderados, o *GP* da rede de Petri exemplo é igual a 2.

Levando em consideração o grau médio dos arcos, apresentado na métrica de simplicidade de (Blum, 2015), esse valor seria também igual a 2.

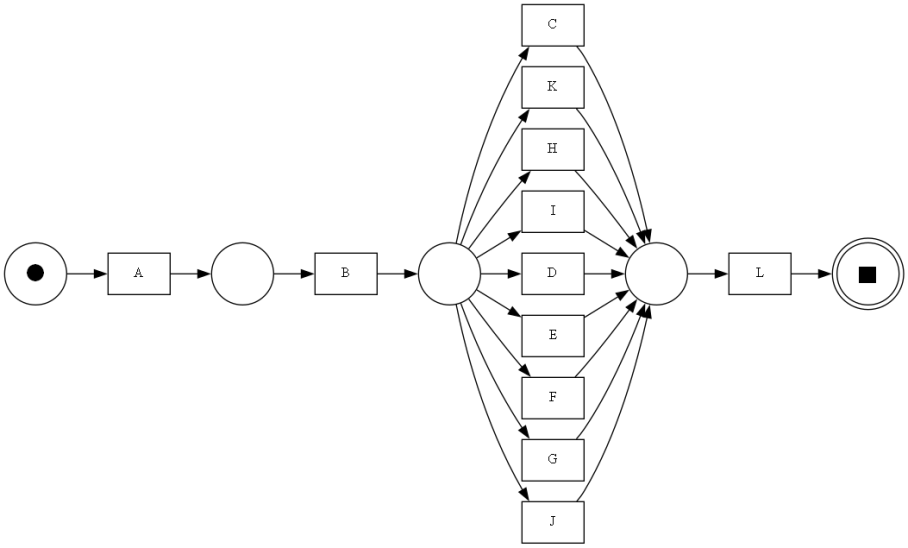
Figura 10 – Rede de Petri do processo exemplo.



Fonte: Elaborada pelo autor (2024).

Levando em consideração agora um exemplo em que aparece um empilhamento no modelo de processos observado na Figura 11, temos que o *GP* da rede de Petri exemplo é igual a 33,6. Já para a simplicidade de (Blum, 2015) seu grau médio seria igual a 4,4.

Figura 11 – Exemplo de empilhamento de atividades.



Fonte: Elaborada pelo autor (2024).

Com o *GP* calculado, a métrica deste trabalho que busca medir a simplicidade dos modelos construídos é dada pela seguinte equação:

$$Simplicidade\_GP(N) = \frac{1}{1 + max(\sqrt{GP} - k, 0)} \tag{4.2}$$

onde o valor de k pode variar entre 0 e o infinito, onde o valor de k define a partir de qual valor de *GP* começa a ter real importância na equação para a Simplicidade_GP de um dado modelo de processos N. Para a execução da métrica, o valor padrão de k é igual a 2, este valor foi escolhido pois mantém o padrão da definição da métrica de simplicidade de Blum (2015) apresentada na Seção 2.2.4. Aplicando agora a equação da Simplicidade_GP, com $k = 2$, nos dois modelos de processos das Figuras 10 e 11 teríamos então os seguintes resultados, respectivamente:

$$Simplicidade\_GP(N) = \frac{1}{1 + max((\sqrt{2} - 2, 0)} = 1 \tag{4.3}$$

$$Simplicidade\_GP(N) = \frac{1}{1 + max((\sqrt{33,6} - 2, 0)} = 0,2 \tag{4.4}$$

Agora levando em consideração a equação de simplicidade de (Blum, 2015) para os mesmos exemplos utilizando o grau médio dos arcos dos lugares temos então os seguintes resultados para os modelos de processos das Figuras 10 e 11, respecitvamente:

$$Simplicidade(N) = \frac{1}{1 + max((2 - 2, 0)} = 1 \tag{4.5}$$

$$Simplicidade(N) = \frac{1}{1 + max(4, 4 - 2, 0)} = 0,31 \tag{4.6}$$

Com os resultados é possível observar como a métrica Simplicidade_GP captura bem o problema dos empilhamentos, pois leva em consideração a diferença entre os arcos de entrada e saída de cada um dos lugares de um dado modelo de processos. A métrica de simplicidade de (Blum, 2015) também consegue penalizar os modelos, mas não tão substancialmente quanto a proposta por este trabalho.

4.2 Métrica de qualidade

Para calcular a qualidade de um modelo de processos a partir de uma rede de Petri, o algoritmo deste trabalho utiliza uma combinação das métricas de precisão (Blum, 2015) (subseção 2.2.3) e de Simplicidade_GP (Seção 4.1). A partir de uma rede Petri N obtida do log de eventos L, a seguinte equação é construída:

$$Qualidade(N, L, \alpha) = \alpha \cdot Simplicidade\_GP(N) + (1 - \alpha) \cdot \frac{\sum_{i=1}^{|T|} Precisao(N, t_i)}{|T|}. \tag{4.7}$$

Na equação, $|T|$ é o total de traços que o log de eventos L possui, logo a $Qualidade(N, L, \alpha)$ é obtida pelo resultado de $Simplicidade\_GP(N)$ mais a média aritmética da precisão de todos os traços de L em relação à rede de Petri N. O valor de α é um valor entre 0 e 1 que tem o objetivo de equilibrar a importância entre precisão e *Simplicidade_GP* da rede de Petri N.

4.3 Algoritmo *IM_Cluster*

O Algoritmo 1 descreve os passos utilizados para o agrupamento de nós *fall-throughs* em uma dada árvore de processos considerada espaguete. A entrada do algoritmo é um log de eventos L organizado em um conjunto de traços. A saída é uma árvore de processos T, onde cada nó r *fall-through*, que tenha uma determinada quantidade mínima de traços, é substituido por uma subárvore com duas ramificações conectadas por um operador $\times$ obtidas a partir do agrupamento do sublog de r.

Inicialmente, o algoritmo constrói a árvore de processos T a partir do algoritmo *Inductive Miner-infrequent* (IMf), descrito na Seção 2.2.1. A escolha da variação IMf se deve ao fato do algoritmo produzir uma representação de árvore de processo que permite identificar sublogs associados aos nós *fall-throughs*. Em seguida, para cada *fall-throughs* ft obtido pelo retorno da função *FindAllFTs* aplicada a T que atenda o requisito de possuir uma determinada quantidade mínima de traços, o algoritmo agrupa o log de eventos $ft.log$ em dois sublogs L_1 e L_2 através da função *LogClusters*. Para acelerar o processo de comparação e identificação dos traços, as atividades são convertidas em símbolos representativos. Por exemplo, dado o seguinte traço t de atividades relacionadas a um processo de compra, temos:

$t =$ {identificação da necessidade, pesquisa de produto/serviços, seleção de fornecedor, avaliação de opções, identificação da necessidade, negociação de termos e condições, realização da compra, entrega ou prestação de serviço, pesquisa de produto/serviços, negociação de termos e condições, avaliação pós-compra, suporte ao cliente, fidelização do cliente}

Seguindo ainda com o exemplo, ao aplicar a conversão em símbolos, como, por exemplo, letras do alfabeto, a sequência ficaria da seguinte forma:

$t =$ [A, B, C, D, A, E, F, G, B, E, H, I, J]

Com tal conversão, as atividades são unidas em uma string para ser utilizada nos processos de agrupamento da função *LogClusters* como será descrito a seguir. No exemplo existem atividades que se repetem, logo são preservados os símbolos que as representam. Ao converter t em string o resultado seria "A-B-C-D-A-E-F-G-B-E-H-I-J".

O próximo passo busca de forma recursiva lidar com os *fall-throughs*, onde, para isso, constrói as subárvores T_1 e T_2 chamando de forma recursiva, para cada subárvore, algoritmo $IM\_Cluster$ que tem como parâmetro os sublogs L_1 e L_2, respectivamente. O algoritmo segue criando um nó v que possui como filhos as subárvores T_1 e T_2 conectadas através de um operador XOR. O operador XOR é utilizado para justificar a escolha entre as subárvores T_1 e T_2 na árvore

de processos. Em outras palavras, o operador XOR indica que, no novo nó, existe uma decisão que direciona o fluxo para um dos dois caminhos possíveis. Apenas uma dessas opções pode ser seguida em cada instância do processo, garantindo que nunca mais de uma delas seja executada simultaneamente.Por último é utilizada a função *Qualidade* para verificar as qualidades dos nós ft e v, onde, caso o valor de ft seja menor que o de v, o nó ft é substituído pelo novo nó v na árvore de processos T. O algoritmo, após executar para todos os nós ft encontrados, retorna a árvore de processos T atualizada. As funções *Qualidade*, *LogClusters* e *FindAllFTs* são descritas nas Seções 4.2, 4.4 e 4.5, respectivamente.

Algoritmo 1: *IM_Cluster*

Input: L: log de eventos (conjunto de traços).
Output: T: árvore de processo de L com alguns nós *fall-throughs* substituídos por um XOR com subárvores de grupos de seus respectivos sublogs.

1 $T \leftarrow InductiveMiner(L)$
2 **Para** $ft \in FindAllFTs(T)$ **faça**
3 $L_1, L_2 \leftarrow LogClusters(ft.log)$
4 $T_1 \leftarrow IM\_Cluster(L_1)$
5 $T_2 \leftarrow IM\_Cluster(L_2)$
6 $v \leftarrow$ novo nó XOR com filhos T_1 e T_2
7 **Se** $Qualidade(ft) < Qualidade(v)$ substitua ft por v em T
8 **Retorne** T

Fonte: Elaborada pelo autor (2024).

4.4 Algoritmo *LogClusters*

A função *LogClusters* agrupa os traços através de uma combinação da distância de edição, também conhecida como distância de Levenshtein (Yujian; Bo, 2007), e do algoritmo k-medoids (Park; Jun, 2009).

A distância de edição é uma medida da similaridade entre duas strings, calculando o número mínimo de operações necessárias para transformar uma string na outra e armazenando os valores em uma matriz, onde as operações permitidas são inserção, remoção ou substituição de um caractere. A matriz utilizada é bidimensional, onde as linhas representam os caracteres da primeira string e as colunas representam os caracteres da segunda. Cada célula armazena o custo da transformação da substring correspondente da primeira string na substring correspondente da segunda. O preenchimento é iterativo, calculando o custo mínimo para chegar a cada célula. O algoritmo k-medoids é uma técnica de agrupamento de dados que utiliza pontos reais dos

dados, chamados de *medoids* (neste trabalho os medoids são representações dos traços), como representantes dos grupos. Inicia-se selecionando *k medoids* aleatórios (no caso deste trabalho o valor de k é 2), atribuindo cada ponto ao *medoid* mais próximo, atualizando os *medoids* minimizando a dissimilaridade total dentro dos grupos e iterando até que os *medoids* não mudem ou um critério de parada seja atingido.

O Algoritmo 2 descreve os passos da função *LogClusters*. A entrada consiste em um log de eventos L, e a saída são os sublogs $L1$ e $L2$, resultantes do agrupamento dos traços de L. Inicialmente, os traços de L são convertidos em strings e armazenados em S. Logo em seguida, é construída a matriz de distância de edição m (ou distância de Levenshtein) utilizando as strings de S. O próximo passo busca sortear de forma não determinística dois representantes $c1$ e $c2$, de forma que sejam distintos, para compor as bases dos agrupamentos. Com a matriz m e os dois representantes $c1$ e $c2$, as demais strings são agrupadas através do algoritmo k-medoids (executado pela função k-medoids). Por último, os grupos são convertidos novamente em traços (através da função *ConverteLog*) para serem retornados como os sublogs $L1$ e $L2$.

Algoritmo 2: *LogClusters*

Input: L: log de eventos (conjunto de traços).
Output: $L1$ e $L2$: sublogs originados de L.

1 $S \leftarrow ConverteString(L)$
2 $m \leftarrow MatrizDistancia(S)$
3 $c1, c2 \leftarrow Representantes(m)$
4 $L1\_string, L2\_string \leftarrow k\text{-}medoids(m, c1, c2)$
5 $L1 \leftarrow ConverteLog(L1\_string)$
6 $L2 \leftarrow ConverteLog(L2\_string)$
7 **Retorne** $L1$, $L2$

Fonte: Elaborada pelo autor (2024).

4.5 Algoritmo *FindAllFTs*

O Algoritmo 3 descreve os passos utilizados no algoritmo *FindAllFTs*. A entrada do algoritmo é um nó r que é raiz de uma árvore de processos. A saída consiste em um conjunto de nós *fall-through* de T, onde um nó r e seus ancestrais não satisfazem uma determinada condição de parada.

O algoritmo inicia verificando um caso base que consiste em avaliar se um nó r é nulo ou se ele atende uma condição de parada especificada, neste caso é retornado uma lista

vazia (linha 1). O próximo caso base consiste em verificar se o nó *r* é *fall-through*, neste caso o algoritmo retorna a lista $[r]$ (linha 2). Por último, de forma recursiva, o algoritmo retorna uma lista construída a partir da união de cada nó *v* filho de *r* que seja *fall-through* verificado pela função $FindAllFTs$ que recebe *v* como parâmetro (linha 3).

Algoritmo 3: *FindAllFTs*

Input: *r*: raiz de uma árvore de processos T.
Output: conjunto de nós *fall-throughs* de T, onde *r* e seus ancestrais não satisfazem uma condição de parada.

1 **Se** *r* **é nulo ou** $CondicaoParada(r)$ **Retorne** [ ]
2 **Se** *r* é um nó *fall-through* **Retorne** $[r]$
3 **Retorne** $\bigcup_{v \in r.filho} FindAllFTs(v)$

Fonte: Elaborada pelo autor (2024).

Neste trabalho, o criterio de parada *StopCondition* da função *FindAllFTs*, foi definido que se um nó subárvore é *fall-through*, mas o tamanho de seu sublog seja menor que um determindo valor *m*, o mesmo não é considerado adequado para ser tratado pelo algoritmo.

5 RESULTADOS DOS EXPERIMENTOS

Neste Capítulo são descritos os experimentos realizados e discutimos os resultados obtidos. Os experimentos comparam a abordagem proposta neste trabalho (descrita no Capítulo 4) com o ActiTrac proposto em (Weerdt *et al.*, 2013) e com a aplicação direta do IM (Aalst, 2016), de acordo com as métricas de Tempo, Precisão (Seção 2.2.3), Simplicidade (Seção 2.2.4), Simplicidade_GP (Seção 4.1) e Qualidade (Seção 4.4).

5.1 Ambiente computacional e instâncias

O ambiente computacional utilizado para a execução dos algoritmos consiste em uma máquina virtual com processador Intel(R) Core(TM) i7-10700, operando a 2.9GHz, e 64GB de memória RAM. Para os experimentos, foram utilizados dois logs de eventos distintos obtidos a partir dos *BPI Challenges* 2012 e 2020 (Dongen, 2012; Dongen; Wynants, 2020). A escolha desses logs de eventos se deve ao fato de serem públicos e apresentarem processos mal estruturados, ou seja, processos do tipo “espaguete”. O log de eventos da base de dados obtida do *BPI Challenge* 2012 compreende 13.087 traços, 36 atividades distintas e 262.200 eventos coletados no período de 01 de outubro de 2011 a 14 de março de 2012. O log de eventos foi obtido de um instituto financeiro holandês e contém eventos relacionados a pedidos de empréstimo/cheque especial de clientes. Já a base de dados obtida *BPI Challenges* 2020 consiste em informações coletadas sobre o processo de reembolso de despesas de viagens da Universidade de Tecnologia de Eindhoven (TU/e). Como a base do *BPI Challenge* 2020 possui várias categorias, neste trabalho utilizou-se a categoria das declarações internacionais, onde a mesma compreende 6.449 traços, 34 atividades e 72.151 eventos coletados entre 2017 e 2018.

Os testes relacionados ao algoritmo ActiTraC foram realizados levando em consideração que os parâmetros fossem os padrões do plugin ProM (Verbeek *et al.*, 2010) durante a execução. Os parâmetros assumem que o total de grupos para a execução é igual a 4 e que os traços remanescentes devem ser adicionados a um novo grupo. Como o algoritmo ActiTraC produz 4 grupos diferentes, o tratamento adotado foi gerar uma árvore de processo para cada grupo, onde depois foi realizada a união das mesmas em uma única árvore de processos através de um operador de escolha exclusiva. Para os testes com o algoritmo IM, é utilizada diretamente a implementação do algoritmo no módulo PM4Py para produzir uma única árvore de processos.

Para o log de eventos do *BPI Challenge* 2012 os valores testados para m no conjunto

{50, 100, 150, 200, 250 e 300}. O parâmetro *m* é definido na Seção 4.3, e fornece o número mínimo de traços no sublog do nó fall-through para que ele seja recursivamente agrupado. Estes valores foram escolhidos atráves de testes para identificar os valores de *m* em que se encontrava *fall-througs* nas árvores de processo. Para o log de eventos do *BPI Challenge* 2020 foi identificado somente 1 *fall-through* com *m* igual a 134. Como o log de eventos do *BPI Challenge* 2012 apresenta *fall-througs* para valores de *m* muito elevados, os valores testados de *m* foram espaçados em 50. Outro parâmetro é o valor de α (apresentado na Equação 4.4), que tem como objetivo equilibrar a importância da precisão e da Simplicidade_GP. O parâmetro α recebeu os valores no conjuto {0, 0,25, 0,5, 0,75, 1} para todas as execuções do IM_Cluster, a fim de verificar as métricas que o utilizam nas comparações entre os algoritmos. Vale destacar que os modelos de processos resultantes dos algoritmos comparados não dependem do parâmetro α, mas a métrica Qualidade (definida na Seção 4.2) depende.

5.2 Resultados do experimento

Nas Tabelas 5, 6, 7, 8 e 9 são apresentados os resultados obtidos pelo algoritmo IM_Cluster para o registro de eventos do *BPI Challenge* 2012, onde os valores de α são definidos como 0, 0,25, 0,5, 0,75 e 1. Os resultados são apresentados para cada valor de *m*. Em destaque são apresentados também os melhores resultados para cada métrica de acordo com o valor de α e os valores de *m* para cada tabela. Para cada valor de *m* em relação aos valores de α foram executadas 10 repetições para coletar os dados. Os resultados são apresentados de acordo com a média das execuções, onde é levado em consideração o desvio padrão representado pelo símbolo $\pm$ apresentado nas tabelas.

Tabela 5 – IM_Cluster aplicado ao log de eventos BPI Challenge 2012 e $\alpha = 0$.

m	Tempo (s)	Simplicidade	Simplicidade_GP	Precisão
50	751 ± 42	0,59 ± 0,01	**0,57** ± 0	**0,99** ± 0,02
100	626 ± 29	0,61 ± 0,02	0,56 ± 0	0,98 ± 0,01
150	586 ± 33	**0,63** ± 0,02	0,56 ± 0	0,98 ± 0,01
200	553 ± 31	0,58 ± 0,01	**0,57** ± 0	0,97 ± 0,02
250	525 ± 18	0,59 ± 0,02	0,56 ± 0	**0,99** ± 0,02
300	**503** ± 14	0,57 ± 0,03	0,56 ± 0	**0,99** ± 0,02

Fonte: Elaborada pelo autor (2024).

Tabela 6 – IM_Cluster aplicado ao log de eventos BPI Challenge 2012 e $\alpha = 0,25$.

m	Tempo (s)	Simplicidade	Simplicidade_GP	Precisão
50	853 ± 71	**0,64** ± 0,002	**0,86** ± 0,009	0,64 ± 0,02
100	709 ± 48	**0,64** ± 0,001	0,82 ± 0,01	0,68 ± 0,05
150	643 ± 35	0,63 ± 0,001	0,79 ± 0,008	0,67 ± 0,03
200	575 ± 37	0,63 ± 0,01	0,78 ± 0,01	**0,69** ± 0,04
250	556 ± 16	**0,64** ± 0,02	0,78 ± 0,03	**0,69** ± 0,04
300	**525** ± 6	**0,64** ± 0,01	0,76 ± 0,02	**0,69** ± 0,05

Fonte: Elaborada pelo autor (2024).

Tabela 7 – IM_Cluster aplicado ao log de eventos BPI Challenge 2012 e $\alpha = 0,5$.

m	Tempo (s)	Simplicidade	Simplicidade_GP	Precisão
50	920 ± 29	**0,64** ± 0,001	**0,85** ± 0,01	0,66 ± 0,03
100	721 ± 31	**0,64** ± 0,005	0,83 ± 0,009	0,66 ± 0,01
150	606 ± 39	0,63 ± 0,002	0,81 ± 0,009	0,66 ± 0,03
200	568 ± 35	0,63 ± 0,003	0,78 ± 0,02	**0,71** ± 0,03
250	542 ± 18	0,63 ± 0,02	0,78 ± 0,006	**0,71** ± 0,02
300	**520** ± 62	0,63 ± 0,02	0,76 ± 0,01	**0,71** ± 0,02

Fonte: Elaborada pelo autor (2024).

Tabela 8 – IM_Cluster aplicado ao log de eventos BPI Challenge 2012 e $\alpha = 0,75$.

m	Tempo (s)	Simplicidade	Simplicidade_GP	Precisão
50	859± 71	**0,64** ±0,001	**0,86** ± 0,01	0,67 ± 0,02
100	701± 24	**0,64** ±0,002	0,82 ± 0,01	0,67 ± 0,02
150	605± 47	**0,64** ±0,002	0,81 ± 0,007	0,65 ± 0,004
200	593± 33	0,63 ±0,002	0,78 ± 0,02	0,71 ± 0,02
250	527± 33	0,63 ±0,005	0,78±0,01	0,71 ± 0,03
300	517± 15	0,63 ± 0,002	0,78±0,01	0,69 ± 0,03

Fonte: Elaborada pelo autor (2024).

Tabela 9 – IM_Cluster aplicado ao log de eventos BPI Challenge 2012 e $\alpha = 1$.

m	Tempo (s)	Simplicidade	Simplicidade_GP	Precisão
50	894 ± 72	**0,64** ± 0,002	**0,85** ± 0,01	0,69 ±0,02
100	705 ± 17	**0,64** ± 0,001	0,83 ± 0,009	0,65 ± 0,01
150	626 ± 18	0,63 ± 0,003	0,81 ± 0,01	0,64 ± 0,01
200	578 ± 26	0,63 ± 0,004	0,79 ± 0,02	**0,72** ± 0,04
250	558 ± 22	0,63 ± 0,005	0,78 ± 0,01	0,70 ± 0,06
300	**532** ±28	0,63 ± 0,003	0,77 ± 0,007	**0,72** ± 0,004

Fonte: Elaborada pelo autor (2024).

A Tabela 10 apresenta os resultados obtidos pelo algoritmo IM_Cluster para o registro de eventos do *BPI Challenge* 2020 de acordo com os valores de α. Os resultados são apresentados para o valor de *m* fixado em 134. Em destaque são apresentados também os melhores resultados para cada métrica de acordo com o valor de α.

Tabela 10 – IM_Cluster aplicado ao log de eventos do BPI Challenge 2020.

α	**Tempo (s)**	**Simplicidade**	**Simplicidade_GP**	**Precisão**
0	6,35 ± 0,81	0,59 ± 0,01	**0,57** ± 0,001	**0,99** ± 0,001
0,25	6,58 ± 0,69	0,59 ± 0,01	0,56 ± 0,001	**0,99** ± 0,001
0,5	**6,32** ± 0,62	**0,61** ± 0,02	0,56 ± 0,001	0,98 ± 0,002
0,75	6,46 ± 0,49	0,6 ± 0,02	0,56 ± 0,001	0,98 ± 0,002
1	6,82 ± 0,44	0,59 ± 0,01	**0,57** ± 0,001	**0,99** ± 0,001

Fonte: Elaborada pelo autor (2024).

Na Tabela 11 são apresentas as qualidades dos modelos gerados pelo algoritmo IM_Cluster para cada um dos logs de eventos dos *BPI Challenge* 2012 e 2020 para cada valor de *m* e α, onde o valor de *m* igual a 134 corresponde aos resultados do log de eventos de 2020.

Tabela 11 – Qualidade dos modelos produzidos pelo IM_Cluster.

m	$\alpha = 0$	$\alpha = 0,25$	$\alpha = 0,5$	$\alpha = 0,75$	$\alpha = 1$
50	**0,99** ± 0,02	0,70 ± 0,02	**0,76** ± 0,02	**0,81** ± 0,008	0,85 ± 0,1
100	0,98 ± 0,01	**0,72** ± 0,03	0,75 ± 0,007	0,78 ± 0,01	0,83 ± 0,009
150	0,98 ± 0,01	0,70 ± 0,02	0,73 ± 0,01	0,77 ± 0,005	0,81 ± 0,01
200	0,97 ± 0,02	0,71 ± 0,03	0,75 ± 0,02	0,77 ± 0,02	0,79 ± 0,02
250	**0,99** ± 0,02	0,71 ± 0,03	0,75 ± 0,01	0,77 ± 0,005	0,78 ± 0,01
300	**0,99** ± 0,02	0,71 ± 0,03	0,74 ± 0,009	0,76 ± 0,01	0,77 ± 0,007
134	0,99 ± 0,001	0,88 ± 0,001	0,78 ± 0,001	0,68 ± 0,001	0,57 ± 0,001

Fonte: Elaborada pelo autor (2024).

A Tabela 12 apresenta os tamanhos dos clusters produzidos pelo ActiTraC, quando aplicado aos logs de eventos do *BPI Challenge* 2012 e *BPI Challenge* 2020. O cluster *"Other"* contém os traços que não foram bem ajustados aos outros clusters, sendo o maior cluster no *BPI Challenge* 2012, e o segundo maior cluster no *BPI Challenge* 2020, indicando que o algoritmo não conseguiu categorizar boa parte dos traços. Além disso, a metade dos outros clusters possuem menos de 10% do total de traços.

Tabela 12 – Tamanho dos clusters produzidos pelo ActiTraC.

Cluster	**BPI Challenge 2012**	**BPI Challenge 2020**
Cluster 1	3.430 (26,2%)	3.431 (53,2%)
Cluster 2	1.872 (14,3%)	502 (7,78%)
Cluster 3	271 (2,07%)	320 (4,96%)
Other	7.514 (57,41%)	2.196 (34,05%)

Fonte: Elaborada pelo autor (2024).

A Tabela 13 apresenta os resultados das métricas de Tempo (em segundos), Simplicidade, Simplicidade_GP e Precisão e as Qualidades de acordo com cada valor de α obtidas pelos algoritmos ActiTraC, IM e IM_Cluster para os logs de eventos dos *BPI Challenges* 2012 e

2020, respectivamente. Em relação ao algoritmo IM_Cluster foram colocados os resultados da variação de α e *m* em que foi obtido o melhor resultado da métrica de Simplicidade_GP. Para o log de eventos do *BPI Challenge* 2012 as métricas escolhidas foram para α igual 0,75 e *m* igual a 50. Para o log de eventos do *BPI Challenge* 2020 as métricas escolhidas foram para α igual 0 (os valores são os mesmos para $\alpha = 1$), o valor de *m* é 134.

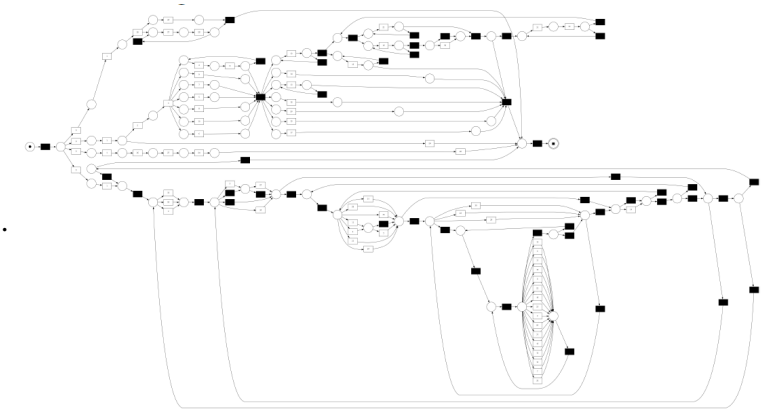
Tabela 13 – Métricas dos algoritmos para os logs de eventos dos *BPI's Challenges* 2012 e 2020.

Log de eventos	**BPI Challenge 2012**			**BPI Challenge 2020**		
Métrica	**ActiTraC**	**IM**	**IM_Cluster**	**ActiTraC**	**IM**	**IM_Cluster**
Tempo (s)	478	**66,16**	859	24	**3,25**	6,35
Simplicidade	0,6	0,54	**0,64**	0,55	0,45	**0,59**
Simplicidade_GP	0,39	0,28	**0,86**	0,18	0,09	**0,57**
Precisão	0,91	**1**	0,67	0,11	**1**	0,99
Qualidade α = 0	0,91	**1**	0,99	0,11	**1**	0,99
Qualidade α = 0,25	0,78	**0,82**	0,72	0,13	0,77	**0,88**
Qualidade α = 0,5	0,65	0,64	**0,76**	0,15	0,54	**0,78**
Qualidade α = 0,75	0,52	0,46	**0,81**	0,17	0,31	**0,68**
Qualidade α = 1	0,39	0,28	**0,85**	0,18	0,09	**0,57**

Fonte: Elaborada pelo autor (2024).

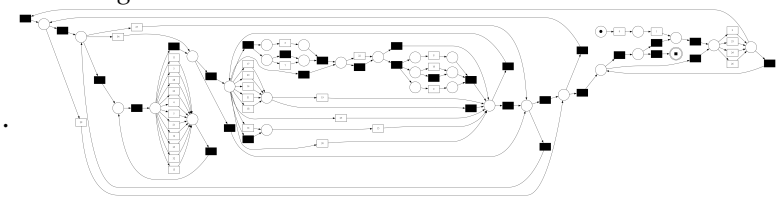
As Figuras 12, 13 e 14 apresentam os modelos de processos de rede de Petri obtidos a partir do log de eventos do *BPI Challenge* 2012 pelos algoritmos ActiTraC, IM e IM_Cluster, respectivamente. A rede de Petri do IM_Cluster foi gerada com $\alpha = 0,5$, que é o cenário com precisão média e Simplicidade GP equilibradas. É notório que o empilhamento de atividades aparece nos modelos do ActiTraC e do IM, mas não nos modelos do IM_Cluster. No entanto, o modelo do IM_Cluster para o log de eventos do *BPI Challenge* 2012 tem uma estrutura mais complexa, com muito mais transições e lugares.

Figura 12 – Rede de Petri produzida pelo ActiTraC para o *BPI Challenge* 2012



Fonte: Elaborada pelo autor (2024).

Figura 13 – Rede de Petri produzida pelo IM para o *BPI Challenge* 2012



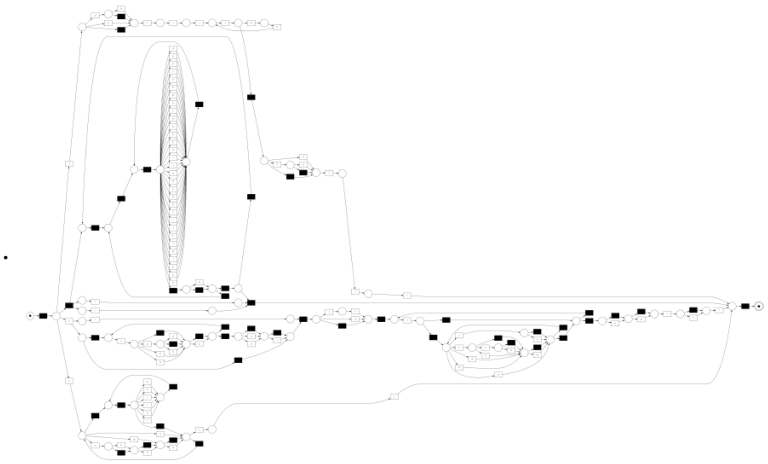
Fonte: Elaborada pelo autor (2024).

Figura 14 – Rede de Petri produzida pelo IM_Cluster para o *BPI Challenge* 2012.

Fonte: Elaborada pelo autor (2024).

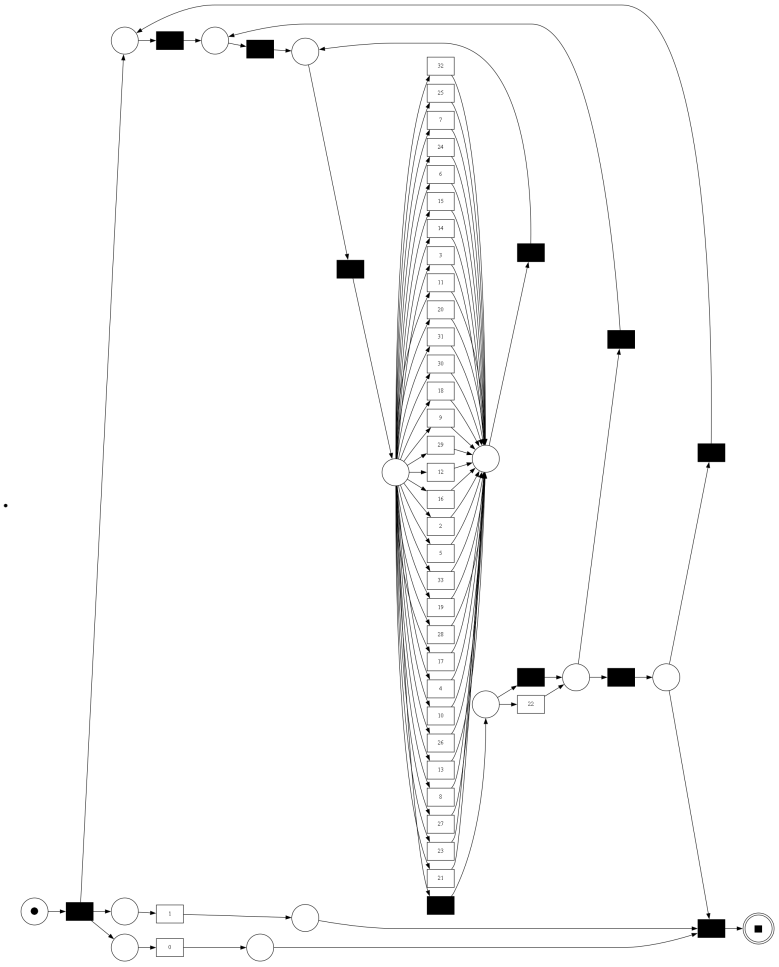
As Figuras 15, 16 e 17 apresentam os modelos de processos de rede de Petri obtidos a partir do log de eventos do *BPI Challenge* 2020 pelos algoritmos ActiTraC, IM e IM_Cluster, respectivamente.

Figura 15 – Rede de Petri produzida pelo ActiTraC para o *BPI Challenge* 2020



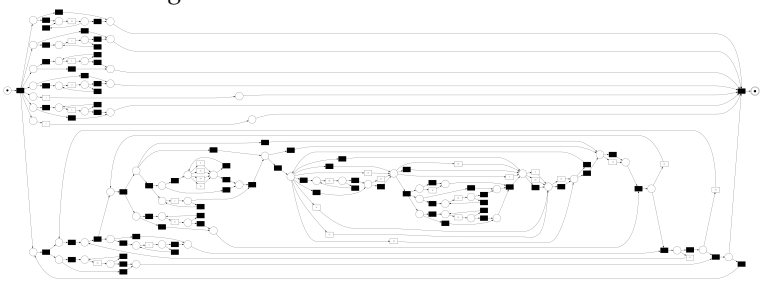
Fonte: Elaborada pelo autor (2024).

Figura 16 – Rede de Petri produzida pelo IM para o *BPI Challenge* 2020



Fonte: Elaborada pelo autor (2024).

Figura 17 – Rede de Petri produzida pelo IM_Cluster para o *BPI Challenge* 2020.



Fonte: Elaborada pelo autor (2024).

5.3 Discussão dos resultados

Em relação ao tempo de execução, levando em consideração os tempos do algoritmo IM_Cluster apresentados na Tabela 13, a diferença em relação ao algoritmo ActiTraC para o log de eventos do *BPI Challenge* 2012 foi pior em aproximadamente 55%. Já em relação ao log de eventos do *BPI Challenge* 2020, o tempo do IM_Cluster foi melhor que o do ActiTraC, com uma diferença de aproximadamente 73%. Quanto ao tempo do algoritmo IM em relação ao tempo do algoritmo IM_Cluster para os logs de eventos dos *BPI Challenges* 2012 e 2020, as diferenças foram piores em aproximadamente 92% e 48% respectivamente. Logo, o IM_Cluster apresentou maior tempo de execução que o algoritmo IM em ambos os logs de eventos. Em todos os cenários, o algoritmo IM também foi mais rápido que o ActiTraC.

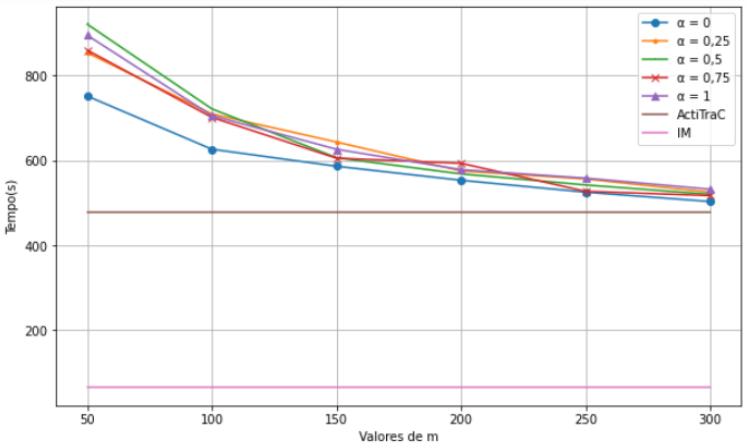
Também é importante destacar que, apesar do IM_Cluster ter um tempo elevado em relação aos trabalhos comparados, quando o valor de m é 300, a diferença em relação ao ActiTraC é inferior a 100 segundos. Portanto, dentro do desvio padrão considerado nas execuções, os algoritmos apresentam tempos próximos. A Figura 18 apresenta um gráfico com o tempo para cada execução do algoritmo IM_Cluster em relação ao tamanho de m e ao valor de α para o log de eventos do *BPI Challenge 2012*. Também apresenta linhas fixas com os tempos dos algoritmos IM e ActiTraC para comparação.

É possível observar que, quanto menor o valor de m e maior o valor de α, mais custoso é o tempo do IM_Cluster para execução. Uma explicação para isso é que, quanto menor o valor de m, mais *fall-throughs* são considerados convenientes para o tratamento do algoritmo. Em relação aos tempos do IM_Cluster para o log de eventos do *BPI Challenge* 2020, não houve variação significativa, já que só houve variação no velor de m, o que fortalece o entendimento de que o valor de m é a variável mais influente no tempo de execução. A Figura 19 apresenta os dados para o log de eventos do *BPI Challenge* 2020, onde o valor de m é 134 e os dados dos

tempos de execuções são apresentados para cada valor de α. Também apresenta linhas fixas com os tempos dos algoritmos IM e ActiTraC para comparação.

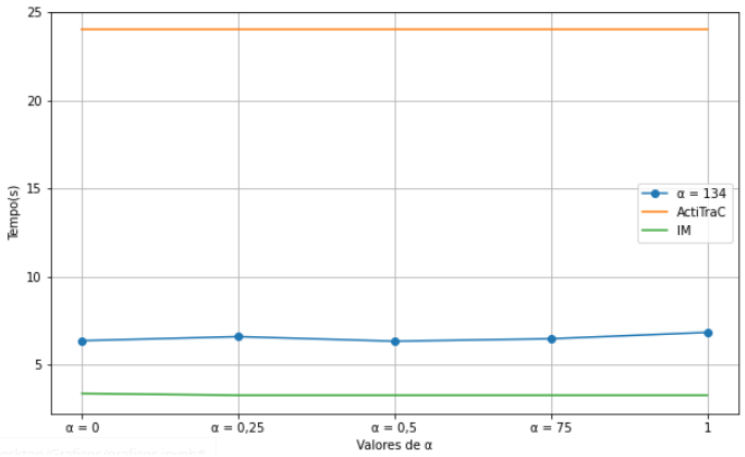
.

Figura 18 – Tempo de execução do IM_Cluster para o log de eventos *BPI Challenge* 2012.



Fonte: Elaborada pelo autor (2024).

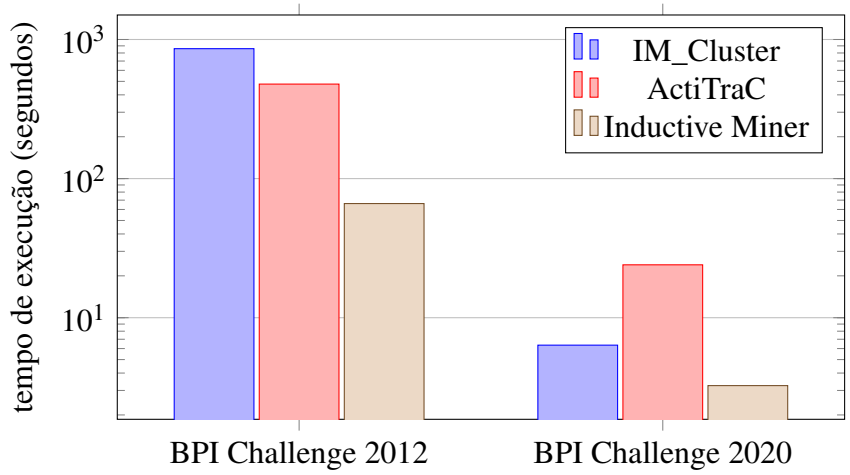
Figura 19 – Tempo de execução do IM_Cluster para o log de eventos *BPI Challenge* 2020.



Fonte: Elaborada pelo autor (2024).

A Figura 20 apresenta uma comparação mais geral dos tempos de execução dos algoritmos para os logs de eventos dos *BPI Challenges* 2012 e 2020. O log de eventos de 2012 possui aproximadamente o dobro do número de traços do log de eventos de 2020, portanto, o tempo de execução é muito maior para todos os algoritmos. Em ambos os logs de eventos, o IM tem o menor tempo de execução. É importante lembrar que o IM_Cluster executa o IM para cada chamada recursiva, incluindo uma chamada com o log de eventos completo como entrada. Para o log de eventos de 2012, o IM_Cluster tem o maior tempo de execução (80% superior ao

Figura 20 – Comparação dos tempos de execução.



Fonte: Elaborada pelo autor (2024).

do ActiTraC). Por outro lado, o IM_Cluster é quase quatro vezes mais rápido que o ActiTraC para o menor log de eventos de 2020.

Para a métrica de simplicidade (descrita na Seção 2.2.4), levando em consideração as simplicidades do algoritmo IM_Cluster apresentados na Tabela 13, para o log do *BPI Challenge* 2012 o algoritmo IM_Cluster foi melhor que os algoritmos ActiTraC e IM aproximadamente 4% e 10%, respectivamente. Já para o log de eventos do *BPI Challenge* 2020, o algoritmo IM_Cluster foi melhor que os algoritmos ActiTraC e IM aproximadamente 4% e 14%, respectivamente. Apesar da métrica não conseguir medir a qualidade dos modelos em relação ao grau dos arcos dos *places* de uma forma tão eficiente, o IM_Cluster apresentou melhores resultados em todos os cenários em relação ao ActiTraC e IM. Em todos os cenários o ActiTraC foi melhor que IM. A simplicidade não apresentou diferenças significativas nas variações dos valores de m e α apresentadas nas Tabelas 5, 6, 7, 8 e 9, onde descrevem os dados obtidos pelo IM_Cluster para o log de eventos de 2012, e a diferença entre os valores fica no intervalo de 7%. O mesmo se repete para o log de eventos de 2020, mas com uma variação de 2% como apresentado na Tabela 10.

Já para a métrica de Simplicidade_GP (descrita na Seção 4.1), levando em consideração as Simplicidades_GP do algoritmo IM_Cluster apresentados na Tabela 13, para o log do *BPI Challenge* 2012 o algoritmo IM_Cluster foi melhor que os algoritmos ActiTraC e IM aproximadamente 47% e 58%, respectivamente. Já para o log de eventos do *BPI Challenge*2020, o algoritmo IM_Cluster foi melhor que os algoritmos ActiTraC e IM aproximadamente 39% e 48%, respectivamente. O IM_Cluster apresentou melhores resultados em todos os cenários em relação ao ActiTraC e IM. Em todos os cenários o ActiTraC foi melhor que IM. Com os resultados é possível notar que a métrica consegue capturar de forma eficaz a discrepância que

os *fall-throughs* causam nos arcos de entrada e saída dos lugares nos diferentes modelos de processos.

Em relação à métrica precisão (descrita na Seção 2.2.3), levando em consideração as características do algoritmo IM_Cluster apresentadas na Tabela 13, o IM obteve o valor máximo de precisão nos logs de eventos do *BPI Challenge* 2012, superando o ActiTraC e o IM_Cluster em aproximadamente 9% e 33%, respectivamente. Para o log de eventos do *BPI Challenge* 2020, o IM também obteve o valor máximo, superando a ActiTraC e o IM_Cluster com uma diferença de aproximadamente 89% e 1%, respectivamente.

É possível observar nas Figuras 12 e 13, que apresentam os modelos de processos construídos pelos algoritmos ActiTraC e IM para o *BPI Challenge* 2012, respectivamente, que há *fall-throughs* com tamanho considerável em relação ao total de atividades. Ambos os casos destacam como a precisão é afetada em modelos de processos com tal estrutura, mostrando o quão prejudicial é para o modelo de processos em relação à representação fiel dos processos de negócios. O modelo de processos obtido pelo algoritmo IM_Cluster para o log de eventos do *BPI Challenge* 2012, apresentado na Figura 14, mesmo que sem apresentar *fall-throughs* consideráveis, o mesmo apresentou uma precisão mais realista. Vale destacar que, entretanto, o modelo de processos apresenta um tamanho muito grande, o que dificulta significativamente sua compreensão. Como consequência, a simplicidade visual do modelo também é prejudicada, tornando-o menos intuitivo e mais difícil de interpretar para os usuários.

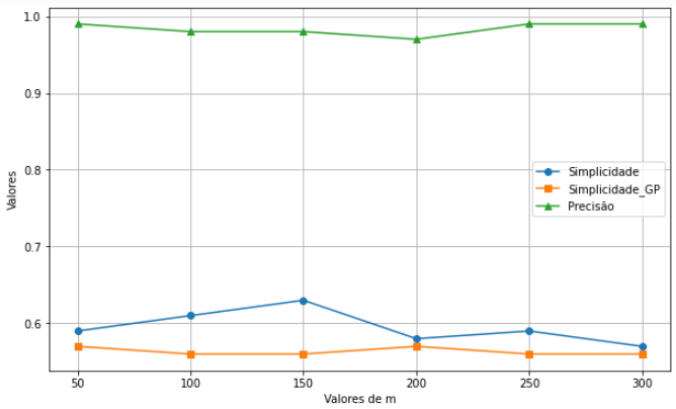
O problema dos empilhamentos causados por *fall-throughs* é ainda mais visível nos modelos de processos obtidos pelo ActiTraC e IM para o log de eventos do *BPI Challenge* 2020 mostrados nas Figuras 15 e 16, respectivamente. Os algoritmos IM e ActiTraC produziram modelos com empilhamentos significativos, o que resultou em precisão bastante variados. Já para o modelo obtido pelo IM_Cluster para o log de eventos do *BPI Challenge* 2020, uma precisão alta foi obtida; no entanto, o modelo de processos apresentado na Figura 17, ficou bem mais linear e menor. Sendo assim, vale destacar que um dos possíveis problemas para a obtenção de modelos de processos por algoritmos tradicionais, e até mesmo para a melhoria de modelos de processos, é o log de eventos que apresenta má construção, sendo um dos fatores principais para o problema.

A seguir são apresentados os gráficos dos valores de variações das métricas de Simplicidade, Simplicidade_GP e Precisão gerais dos modelos obtidos pelo IM_Cluster para o log de eventos do *BPI Challenge* 2012 nas Figuras 21, 22, 23, 24 e 25 para os valores de

α 0, 0,25, 0,5, 0,75 e 1, respectivamente. Em todos os gráficos é possível notar que não houve variação significativa na Simplicidade, mas podem-se observar relações importantes entre Simplicidade_GP e a Precisão.

Com o valor de $\alpha = 0$ a Simplicidade_GP teve pouca variação, mas vale destacar o alto valor de precisão. Um fator a ser levado em consideração é que nesse caso, a equação utilizada pelo algoritmo IM_Cluster para obter os modelos de processos não leva em consideração a métrica de Simplicidade_GP com tal valor de α.

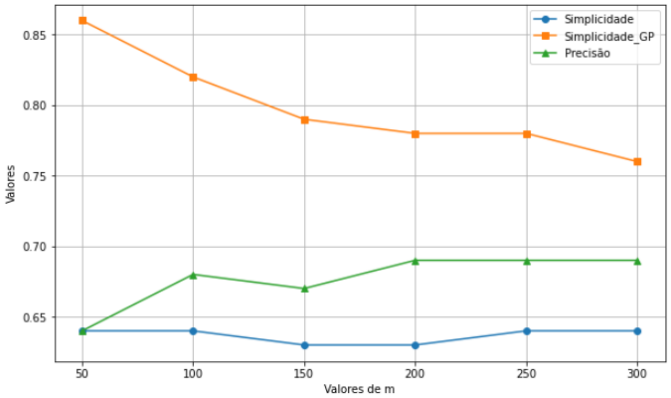
Figura 21 – Variação de Simplicidade, Simplicidade_GP e Precisão obtidas pelo IM_Cluster para o log de eventos *BPI Challenge* 2012 com $\alpha = 0$.



Fonte: Elaborada pelo autor (2024).

Para $\alpha = 0,25$ já é possível notar uma diminuição no valor da Precisão e um alto valor de Simplicidade_GP. Também se destaca que conforme o valor de *m* aumenta a Simplicidade_GP diminui, mas a Precisão faz o inverso, pois tende a crescer apesar de ser de forma sutil e estabilizar a partir de *m* = 200.

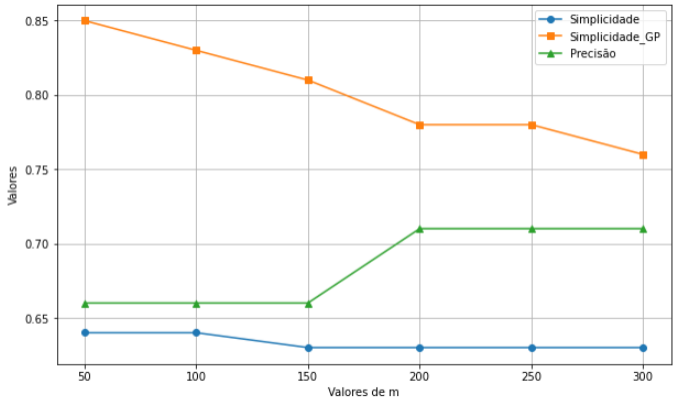
Figura 22 – Variação de Simplicidade, Simplicidade_GP e Precisão obtidas pelo IM_Cluster para o log de eventos *BPI Challenge* 2012 com $\alpha = 0,25$.



Fonte: Elaborada pelo autor (2024).

Com $alpha = 0,5$ o mesmo fator é observado nas variações de Simplicidade_GP e Precisão que o caso anterior.

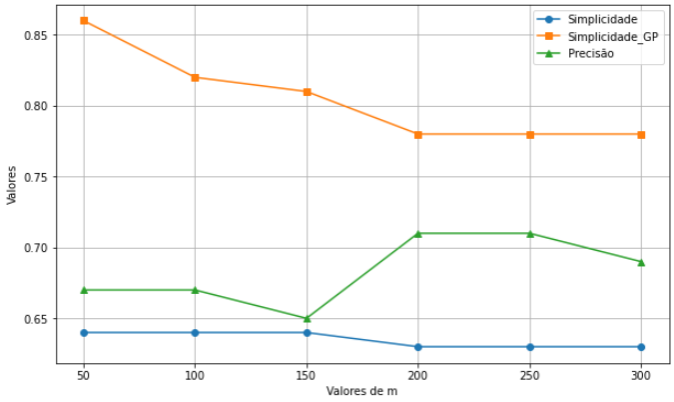
Figura 23 – Variação de Simplicidade, Simplicidade_GP e Precisão obtidas pelo IM_Cluster para o log de eventos *BPI Challenge* 2012 com $\alpha = 0,5$.



Fonte: Elaborada pelo autor (2024).

Já quando o valor de α é igual a 0,75, apesar da Simplicidade_GP apresentar também uma diminuição nos valores conforme o valor de m aumenta, ao atingir tamanho 200 é possível notar que a Simplicidade_GP estabiliza e a Precisão tem uma pequena queda após ter um aumento mais expressivo anteriormente.

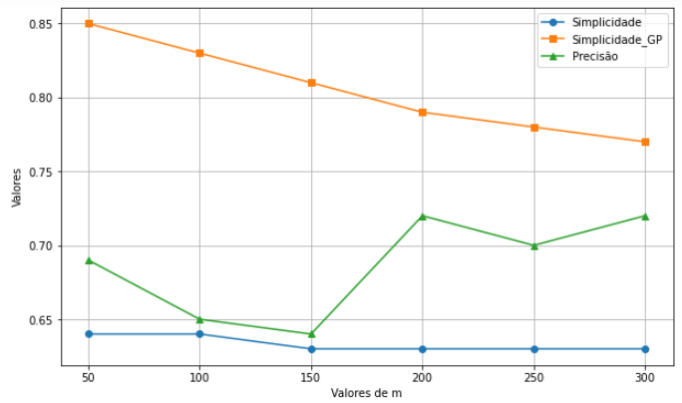
Figura 24 – Variação de Simplicidade, Simplicidade_GP e Precisão obtidas pelo IM_Cluster para o log de eventos *BPI Challenge* 2012 com $\alpha = 0,75$.



Fonte: Elaborada pelo autor (2024).

Por último $alpha = 1$ faz com que o IM_Cluster desconsidere na Precisão e leve em consideração completamente a Simplicidade_GP (o inverso do que acontece com $\alpha = 0$), nesse caso a Simplicidade_GP também tem uma diminuição conforme os valores de m aumentam, mas a Precisão apresenta uma variação significativa aparentando crescer conforme m também cresce.

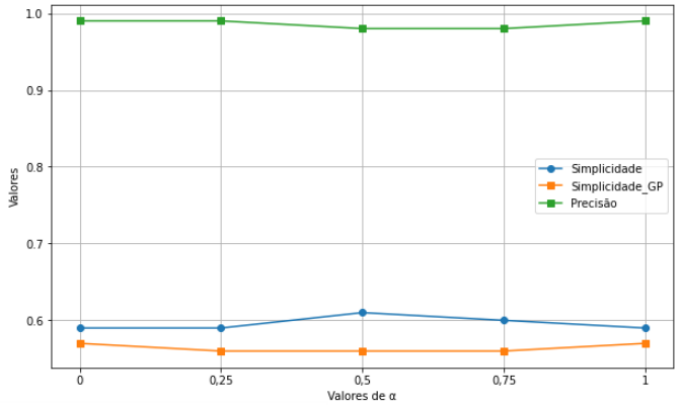
Figura 25 – Variação de Simplicidade, Simplicidade_GP e Precisão obtidas pelo IM_Cluster para o log de eventos *BPI Challenge* 2012 com $\alpha = 1$.



Fonte: Elaborada pelo autor (2024).

A Figura 26 apresenta o gráfico dos valores de variações das métricas de Simplicidade, Simplicidade_GP e Precisão gerais dos modelos obtidos pelo IM_Cluster para o log de eventos do *BPI Challenge* 2020 para os valores de α. Não houve variação significativa entre as métricas de Simplicidade, Simplicidade_GP e Precisão, mas é possível observar que as métricas Simplicidade e Simplicidade_GP foram razoáveis se comparadas à alta Precisão. Logo não houve diferenças nas relações de Simplicidade_GP e Precisão na influência dos modelos de processos gerados.

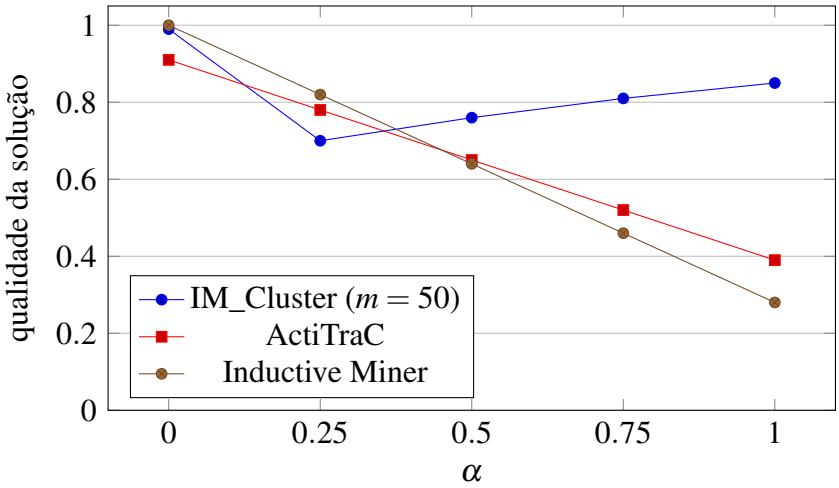
Figura 26 – Variação de Simplicidade, Simplicidade_GP e Precisão obtidas pelo IM_Cluster para o log de eventos *BPI Challenge* 2020.



Fonte: Elaborada pelo autor (2024).

Para a métrica qualidade (descrita na Seção 4.4), que tem como objetivo medir a qualidade de um nó na árvore de processos foi levado em consideração sua utilização para medir a qualidade da árvore de processos inteira. As Figuras 27 e 28 apresentam os resultados da métrica de Qualidade variando o parâmetro α para os *BPI Challenges* de 2012 e 2020, respectivamente.

Figura 27 – Qualidade da solução variando α para o BPI Challenge 2012.

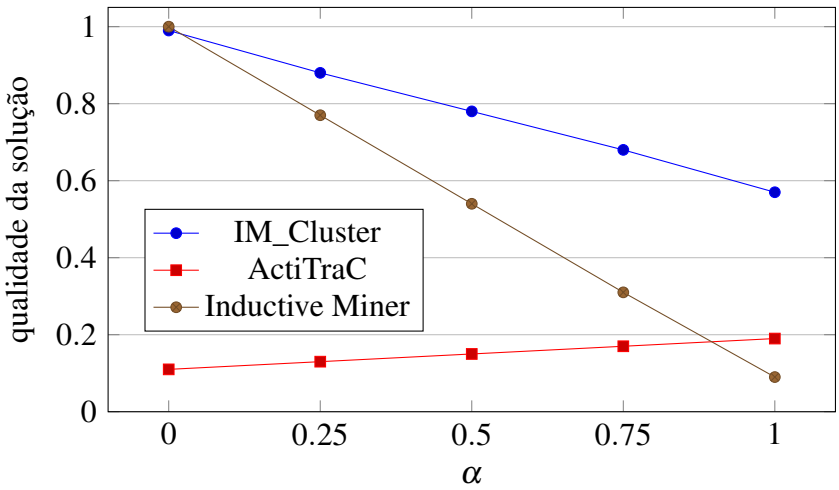


Fonte: Elaborada pelo autor (2024).

Note que a métrica de Qualidade é a precisão média quando $\alpha = 0$ e a Simplicidade_GP quando $\alpha = 1$. Como as soluções do ActiTraC e IM não são afetadas pelo parâmetro α, a métrica de Qualidade é uma função linear do parâmetro α, interpolando entre a precisão média e o Simplicidade_GP. Por outro lado, o comportamento do IM_Cluster depende do parâmetro α, uma vez que o algoritmo decide substituir um nó de *fall-through* com base na métrica de Qualidade, então a Qualidade do modelo resultante pode não ser uma função linear de α.

Para o log de eventos do *BPI Challenge* 2012, o IM_Cluster tem Qualidade similar ao ActiTraC e IM para $\alpha < 0{,}5$. No entanto, após este ponto, ele apresenta uma Simplicidade_GP muito melhor, resultando em melhor Qualidade. Os algoritmos ActiTraC e IM têm resultados semelhantes para o registro de eventos do *BPI Challenge* 2012, sendo que o IM tem uma precisão média ligeiramente maior e o ActiTraC tem um Simplicidade_GP ligeiramente maior. Para o registro de eventos do *BPI Challenge* 2020, o IM_Cluster tem Qualidade melhor do que os outros algoritmos para praticamente todos os valores de α. O IM apresenta uma baixa precisão média e baixa Simplicidade_GP para o log de eventos do *BPI Challenge* 2020, enquanto o ActiTraC tem uma Simplicidade_GP baixa, mas uma alta precisão média.

Figura 28 – Qualidade da solução variando α para o BPI Challenge 2020.



Fonte: Elaborada pelo autor (2024).

6 CONCLUSÕES E TRABALHOS FUTUROS

Processos mal estruturados (“espaguete”) são comuns em aplicações do mundo real e representam um desafio para os algoritmos de mineração de processos. O *Inductive Miner* (Aalst, 2016) é um algoritmo muito popular para mineração de processos, mas tende a empilhar atividades nos modelos resultantes quando aplicado a processos “espaguete”. O empilhamento de atividades resulta dos nós *fall-throughs*, que ocorrem quando o *Inductive Miner* não consegue determinar qual operador aplicar. Neste trabalho, abordamos o problema do empilhamento de atividades agrupando os traços no sublog dos nós *fall-throughs*. Chamamos esse algoritmo de IM_Cluster. Este trabalho também propõe uma métrica de simplicidade chamada Simplicidade_GP, que captura melhor a presença de nós *fall-throughs* em árvores de processos. Esta métrica é então empregada para guiar o algoritmo IM_Cluster.

Foi realizada uma avaliação empírica usando os logs de eventos do *BPI Challenge* 2012 e 2020, comparando o IM_Cluster com os algoritmos Inductive Miner e ActiTraC (Weerdt *et al.*, 2013). Os resultados mostram que o IM_Cluster foi capaz de reduzir os empilhamentos nos modelos construídos, mantendo uma precisão média que não foi pior do que a dos outros algoritmos.

Agora discutimos as limitações deste trabalho e sugerimos trabalhos futuros. O algoritmo IM_Cluster é mais lento para o maior log de eventos, demandando cerca de 80% mais tempo do que o ActiTraC. Portanto, pode ser interessante investigar o uso de algoritmos de agrupamento mais eficientes e variantes do *Inductive Miner*, levando em conta que apenas variantes do *Inductive Miner* que podem lidar com sublogs podem ser usadas, já que o IM_Cluster agrupa esses sublogs. Outra limitação deste trabalho é que o algoritmo IM_Cluster foi otimizado para o valor de m, e os parâmetros dos algoritmos alternativos não foram, tornando a comparação um tanto injusta. Além disso, o IM_Cluster está ciente da métrica de Qualidade empregada na comparação, enquanto os outros algoritmos não estão. Finalmente, o algoritmo IM_Cluster não pode controlar o tamanho do modelo resultante, o que pode tornar os modelos resultantes difíceis de interpretar. Portanto, vale a pena investigar maneiras de controlar o tamanho dos modelos fornecidos pelo IM Cluster, como empregar condições de parada alternativas.

REFERÊNCIAS

AALST, W. M. V. D. Process mining: discovering and improving spaghetti and lasagna processes. **2011 IEEE symposium on computational intelligence and data mining (CIDM)**, IEEE, p. 1–7, 2011.

AALST, W. M. V. D. Process mining: Overview and opportunities. **ACM Transactions on Management Information Systems (TMIS)**, v. 3, n. 2, p. 1–17, 2012.

AALST, W. M. V. D. **Process mining**: data science in action. [*S. l.*]: Springer Berlin Heidelberg, 2016. v. 2.

AALST, W. M. V. D.; GUNTHER, C. W. F. Finding structure in unstructured processes: The case for process mining. **Network analysis**, Spring, Berlin, Heidelberg, p. 3–12, 2007.

ANGELASTRO, S.; FERILLI, S. Process model modularization by subprocess discovery. In: IEEE. **2020 International Joint Conference on Neural Networks (IJCNN)**. [*S. l.*], 2020. p. 1–8.

AUGUSTO, A. *et al.* Split miner: Automated discovery of accurate and simple business process models from event logs. **Knowledge and Information Systems**, v. 59, p. 251–284, 2019.

BATISTA, E.; SOLANAS, A. Skip miner: Towards the simplification of spaghetti-like business process models. In: IEEE. **2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)**. [*S. l.*], 2019. p. 1–6.

BERTI, A.; ZELST, S. J. V.; AALST, W. M. V. D. Process mining for python (pm4py): Bridging the gap between process- and data science. **arXiv preprint arXiv:1905.06169**, 2019.

BLUM, F. R. **Metrics in process discovery, Tech. rep., Tech. Rep. TR/DCC.** [*S. l.*], 2015. 1–21 p.

CHEN, Q.; LU, Y.; TAM, C.; POON, S. A novel approach to detect redundant activity labels for more representative event logs. **arXiv preprint arXiv:2103.16061**, v. 16061, 2021.

DETTEN, J. N. V.; SCHUMACHER, P.; LEEMANS, S. J. An approximate inductive miner. In: IEEE. **2023 5th International Conference on Process Mining (ICPM)**. [*S. l.*], 2023. p. 129–136.

DJENOURI, Y. *et al.* Exploring decomposition for solving pattern mining problems. **ACM Transactions on Management Information Systems (TMIS)**, v. 12, n. 2, p. 1–36, 2021.

DONGEN, B. F. van. **BPI Challenge 2012**. 2012. Disponível em: **http://dx.doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f**. Acesso em: 12 jul. 2024.

DONGEN, B. F. van; WYNANTS, I. **BPI Challenge 2020**. 2020. Disponível em: **https://data.4tu.nl/articles/dataset/BPI_Challenge_2020/12696884**. Acesso em: 12 jul. 2024.

FOLINO, F.; PONTIERI, L. Ai-empowered process mining for complex application scenarios: survey and discussion. **Journal on Data Semantics**, v. 10, n. 1, p. 77–106, 2021.

GAERTLER, M. Clustering. **Seventh International Conference on Application of Concurrency to System Design (ACSD)**, IEEE, p. 178–215, 2005.

IMRAN, M. *et al.* Complex process modeling in process mining: A systematic review. **IEEE Access**, v. 10, p. 101515–101536, 2022.

KONINCK, P. D. *et al.* Expert-driven trace clustering with instance-level constraints. **Knowledge and Information Systems**, v. 63, n. 5, p. 1197–1220, 2021.

MADHULATHA, T. S. An overview on clustering methods. **arXiv preprint arXiv:1205.1117**, 2012.

MARIN-CASTRO, H. M.; TELLO-LEAL, E. Event log preprocessing for process mining: a review. **Applied Sciences**, v. 11, n. 22, p. 10556, 2021.

MARTIN, N.; MARTINEZ-MILLANA, A.; VALDIVIESO, B.; FERNáNDEZ-LLATAS, C. Interactive data cleaning for process mining: a case study of an outpatient clinic's appointment system. In: **Business Process Management Workshops**: Bpm 2019 international workshops, vienna, austria, september 1–6, 2019, revised selected papers. [*S. l.*]: Springer International Publishing, 2019. p. 532–544.

NEUBAUER, T. R. *et al.* Interactive trace clustering to enhance incident completion time prediction in process mining. In: **CI4PM/PAI@ WCCI**. [*S. l.*: *s. n.*], 2022. p. 8–20.

OCHI, L. S.; DIAS, C. R.; SOARES, S. S. F. Clusterização em mineração de dados. **Instituto de Computação-Universidade Federal Fluminense-Niterói**, v. 1, p. 46, 2004.

PARK, H.-S.; JUN, C.-H. A simple and fast algorithm for k-medoids clustering. **Expert systems with applications**, v. 36, n. 2, p. 3336–3341, 2009.

RAUTENBERG, S.; CARMO, P. R. V. D. Big data e ciência de dados: complementariedade conceitual no processo de tomada de decisão. **Brazilian Journal of Information Science: research trends**, v. 13, n. 1, p. 56–57, 2019.

RUDNITCKAIA, J. Process mining: Data science in action. **University of Technology, Faculty of Information Technology**, p. 1–11, 2016.

SONG, M.; GUNTHER CHRISTIAN, W.; AALST, W. M. V. D. Trace clustering in process mining. In: **Business Process Management Workshops: BPM 2008 International Workshops, Milano, Italy, September 1-4, 2008, Revised Papers**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 109–120.

TARIQ, Z. *et al.* An event-level clustering framework for process mining using common sequential rules. In: **International conference for emerging technologies in computing**. [*S. l.*]: Springer International Publishing, 2021. v. 4, p. 147–160.

VERBEEK, H. M. W.; BUIJS, J. C. A. M.; DONGEN, B. F. V.; AALST, W. M. Van der. Prom 6: The process mining toolkit. In: **Proceedings of BPM Demonstration Track**. [*S. l.*: *s. n.*], 2010. v. 615, p. 34–39.

WANG, P. *et al.* A novel trace clustering technique based on constrained trace alignment. In: **Human Centered Computing: Third International Conference, HCC 2017, Kazan, Russia, August 7–9, 2017, Revised Selected Papers**. [*S. l.*]: Springer International Publishing, 2018. v. 3, p. 53–63.

WEERDT, J. D.; BROUCKE, S. vander; VANTHIENEN, J.; BAESENS, B. Active trace clustering for improved process discovery. **Transactions on Knowledge and Data Engineering**, IEEE Transactions on Knowledge and Data Engineering, v. 25, n. 12, p. 2708–2720, 2013.

YUJIAN, L.; BO, L. A normalized levenshtein distance metric. **IEEE transactions on pattern analysis and machine intelligence**, v. 29, n. 6, p. 1091–1095, 2007.