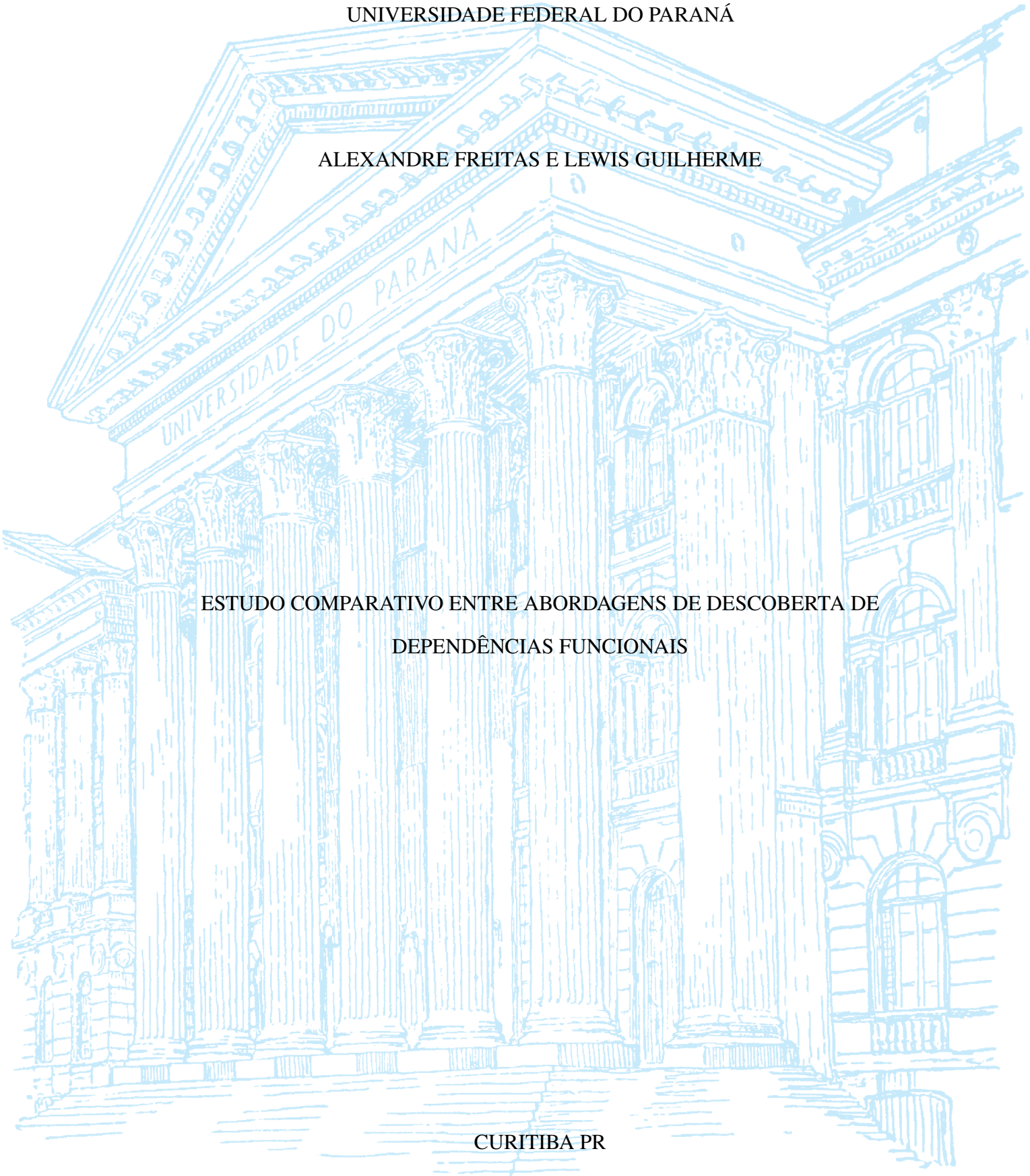
UNIVERSIDADE FEDERAL DO PARANÁ

ALEXANDRE FREITAS E LEWIS GUILHERME



ESTUDO COMPARATIVO ENTRE ABORDAGENS DE DESCOBERTA DE DEPENDÊNCIAS FUNCIONAIS

CURITIBA PR

2024

ALEXANDRE FREITAS E LEWIS GUILHERME

ESTUDO COMPARATIVO ENTRE ABORDAGENS DE DESCOBERTA DE DEPENDÊNCIAS FUNCIONAIS

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Bancos de dados*.

Orientador: Eduardo Cunha de Almeida.

CURITIBA PR

2024

Universidade Federal do Paraná
Setor de Ciências Exatas
Curso de Ciência da Computação

Ata de Apresentação de Trabalho de Conclusão de Curso 2

Título do Trabalho: ESTUDO COMPARATIVO ENTRE ABORDAGENS DE DESCOBERTA DE DEPENDÊNCIAS FUNCIONAIS

Autor(es):

GRR20205646 Nome: ALEXANDRE DE OLIVEIRA PLUGGE FREITAS

GRR20203893 Nome: LEWIS GUILHERME THEOPHILO GERALDO

Apresentação: Data: 3/7/2025 Hora: 10h Local: Sala Videoconferência

Documento assinado digitalmente
gov.br
EDUARDO CUNHA DE ALMEIDA
Data: 03/07/2025 15:58:50-0300
Verifique em https://validar.iti.gov.br

Orientador: Eduardo Cunha de Almeida _____

Co-Orientador: Eduardo Henrique Monteiro Pena _____

Membro 1: Simone Dominico _____

Membro 2: Sergio Luiz Marques Filho _____

(nome) (assinatura)

AVALIAÇÃO – Produto escrito		ORIENTADOR	MEMBRO 1	MEMBRO 2	MÉDIA
Conteúdo	(00-40)				40
Referência Bibliográfica	(00-10)				10
Formato	(00-05)				5
AVALIAÇÃO – Apresentação Oral					
Domínio do Assunto	(00-15)				15
Desenvolvimento do Assunto	(00-05)				5
Técnica de Apresentação	(00-03)				3
Uso do Tempo	(00-02)				2
AVALIAÇÃO – Desenvolvimento					
Nota do Orientador	(00-20)		**********	**********	20
NOTA FINAL		**********	**********	**********	**100**

Os pesos indicados são sugestões.

Conforme decisão do colegiado do curso de Ciência da Computação, a entrega dos documentos comprobatório de trabalho de Conclusão de Curso 2 deve deve respeitar os seguintes procedimentos: o orientador deve abrir um processo no Sistema Eletrônico de Informações (SEI – UFPR); Selecionar o tipo: *Graduação: Trabalho Conclusão de Curso*; informar os interessados: nome do aluno e o nome do orientador; anexar esta ata escaneada e a versão final do PDF da monografia do aluno; Tramitar o processo para CCOMP (Coordenação de Ciência da Computação).

Documento assinado digitalmente
gov.br
SIMONE DOMINICO
Data: 03/07/2025 17:39:15-0300
Verifique em https://validar.iti.gov.br

Documento assinado digitalmente
gov.br
SERGIO LUIZ MARQUES FILHO
Data: 03/07/2025 18:04:18-0300
Verifique em https://validar.iti.gov.br

Dedicamos este trabalho às pessoas que tornaram tudo possível: nossas famílias, amigos e namoradas. O vosso apoio incondicional, paciência e carinho foram a base que nos permitiu seguir em frente nos momentos mais desafiadores.

AGRADECIMENTOS

Gostaríamos de expressar nossa sincera gratidão ao Professor Dr. Eduardo Cunha de Almeida e ao Professor Dr. Eduardo H. M. Pena, pela orientação precisa e pelo suporte fundamental durante todo o processo de desenvolvimento desta pesquisa.

Agradecemos, igualmente, às nossas famílias, amigos e namoradas por todo o apoio incondicional. Sem a vossa presença e incentivo constantes, a realização deste trabalho não teria sido possível.

Por fim, agradecemos à Universidade Federal do Paraná e a todos os membros do curso de Ciência da Computação, do qual temos a honra de fazer parte, pela excelente formação e por toda a estrutura oferecida.

RESUMO

O perfilamento de dados é o processo de descoberta de metadados em conjuntos de dados, visando facilitar sua interpretação semântica [13]. As dependências funcionais (DFs) são um exemplo crucial de metadado, essenciais para a normalização de esquemas relacionais [14]. Contudo, a identificação manual de DFs é impraticável em grandes volumes de dados, o que impulsionou o desenvolvimento de algoritmos de automação, como o HyFD [14] para DFs exatas. Em contextos de dados ruidosos, as dependências funcionais aproximadas (DFAs) oferecem maior flexibilidade, com algoritmos proeminentes como Pyro [9] e FDX [22]. No entanto, esses algoritmos frequentemente geram um volume excessivo de dependências, dificultando a seleção dos bons candidatos. Para abordar essa questão, nosso estudo propõe um protocolo para a descoberta e o ranqueamento de DFs e DFAs, utilizando as métricas g_3, $RFI'+$, e $mu+$ [16] para pontuar e selecionar as dependências mais relevantes para a normalização, além de demonstrar como as características intrínsecas de DFs e DFAs afetam sua avaliação pelas métricas.

Palavras-chave: Perfilamento de dados. Dependências Funcionais. Normalização de esquemas.

ABSTRACT

Data profiling is the process of discovering metadata within datasets, aiming to facilitate their semantic interpretation [13]. Functional dependencies (FDs) are a crucial example of such metadata, essential for normalizing relational schemas [13]. However, manual identification of FDs is impractical for large volumes of data, which propelled the development of automation algorithms, such as HyFD [14] for exact FDs. In noisy data contexts, approximate functional dependencies (AFDs) offer greater flexibility, with prominent algorithms like Pyro [9] and FDX [22]. Nevertheless, these algorithms often generate an excessive volume of dependencies, making it difficult to select promising candidates. To address this issue, our study proposes a protocol for the discovery and ranking of FDs and AFDs, utilizing the metrics g_3, $RFI'+$, and $mu+$ [16] to score and select the most relevant dependencies for normalization. Furthermore, we demonstrate how intrinsic characteristics of FDs and AFDs affect their evaluation by these metrics.

Keywords: Data Profiling . Functional Dependencies. Schema Normalization.

LISTA DE FIGURAS

3.1 Fluxo de execução dos componentes do HyFD. Extraído de [15] 25
3.2 Etapas de execução do FDX. Extraído de Zhang et al. [22] 26

4.1 Extraído de Parciak et al. [16]. Comparação das métricas com relação aos parâmetros de erro, LHS-uniqueness e RHS-skew. Dentre as 3 melhores, vemos que $g3'$ possui uma sensibilidade considerável ao parâmetro RHS-skew Parciak et al. [16] 29
4.2 Diagrama do fluxo de experimentos 30

5.1 Número de DFs encontradas em cada *dataset* com cada algoritmo 35
5.2 Densidade das métricas por *dataset*. 37
5.3 Mapas de calor de correlação (Spearman) entre as métricas por *dataset* e algoritmo 39
5.4 Comportamento de métricas conforme o LHS das DFs aumenta 41
5.5 Comportamento de métricas conforme o a unicidade do LHS das DFs aumenta . 42
5.6 Número de DFAs encontradas pelo Pyro variando e_{max} 49

LISTA DE TABELAS

2.1 Exemplo de relação 15

5.1 Dimensões dos conjuntos de dados 34
5.2 Estatísticas básicas referentes ao LHS 40
5.3 Primeiras 30 DFS encontradas pelo algoritmo Pyro para o *dataset* 'hate_crimes' ranqueadas a partir da métrica μ^+. 43
5.4 Todas as DFS encontradas pelo algoritmo FDX para o *dataset* 'hate_crimes' ranqueadas a partir da métrica μ^+. 44
5.5 Primeiras 30 DFS encontradas pelo algoritmo Pyro para o *dataset* 'ncvoter' ranqueadas a partir da métrica μ^+. 44
5.6 Todas as DFS encontradas pelo algoritmo FDX para o *dataset* 'ncvoter' ranqueadas a partir da métrica μ^+.. 44
5.7 Primeiras 30 DFS encontradas pelo algoritmo Pyro para o *dataset* 'wifi_hotspot_location' ranqueadas a partir da métrica μ^+.. 45
5.8 Todas DFS encontradas pelo algoritmo FDX para o *dataset* 'wifi_hotspot_location' ranqueadas a partir da métrica μ^+. 45
5.9 Primeiras 30 DFS encontradas pelo algoritmo Pyro para o *dataset* 'sisu_ufpr_curitiba_politecnico' ranqueadas a partir da métrica μ^+. 46
5.10 Todas DFS encontradas pelo algoritmo FDX para o *dataset* 'sisu_ufpr_curitiba_politecnico' ranqueadas a partir da métrica μ^+. 46
5.11 Primeiras 30 DFS encontradas pelo algoritmo Pyro para o *dataset* 'adult' ranqueadas a partir da métrica μ^+. 47
5.12 Todas as DFS encontradas pelo algoritmo FDX para o *dataset* 'adult' ranqueadas a partir da métrica μ^+.. 47
5.13 DFs descobertas pelo FDX no *dataset* 'ncvoter' com parâmetros ajustados (`tol=0.001`, `eps=0.05`, `sparsity=0.01`), ranqueadas pela métrica μ^+.. . 50

LISTA DE ACRÔNIMOS

DINF	Departamento de Informática
PPGINF	Programa de Pós-Graduação em Informática
UFPR	Universidade Federal do Paraná
DF	Dependência Funcional
DFs	Dependências Funcionais
DFA	Dependência Funcional Aproximada
DFAs	Dependências Funcionais Aproximadas
LHS	Left Hand Side (Lado esquerdo de uma dependência funcional)
RHS	Right Hand Side (Lado direito de uma dependência funcional)

SUMÁRIO

1 INTRODUÇÃO 12
1.1 CONTRIBUIÇÕES 13
1.2 ORGANIZAÇÃO DO DOCUMENTO 13
2 FUNDAMENTAÇÃO TEÓRICA 14
2.1 NOTAÇÕES BÁSICAS E CONVENÇÕES 14
2.2 DEPENDÊNCIAS 15
2.3 DEPENDÊNCIA FUNCIONAL 15
2.4 DEPENDÊNCIAS RELAXADAS 16
2.5 INTERPRETAÇÃO PROBABILÍSTICA DE DF. 16
2.6 PERFILAMENTO DE DADOS 16
2.6.1 Descoberta de dependências 17
2.7 DESCOBERTA DE DEPENDÊNCIAS FUNCIONAIS 17
2.7.1 Definição formal 18
2.7.2 Resultados não confiáveis 18
2.7.3 Descoberta de Dependências Funcionais Aproximadas 18
2.8 MÉTRICAS E RANQUEAMENTO DE DEPENDÊNCIAS 19
2.8.1 Métricas para Dependências Funcionais Aproximadas 19
2.8.2 Parâmetros 19
2.8.3 Medidas de Erro. 20
2.8.4 Fração de Informação 20
2.8.5 Medidas de probabilidade 21
3 ABORDAGENS DE DESCOBERTA DE DEPENDÊNCIAS FUNCIONAIS . 23
3.1 HYFD 24
3.2 PYRO 25
3.3 FDX 26
4 CONTRIBUIÇÃO 28
4.1 COMPARANDO ABORDAGENS DE DESCOBERTA 29
4.2 EXECUÇÃO DOS ALGORITMOS DE DESCOBERTA 30
4.3 CÁLCULO DAS MÉTRICAS 31
4.3.1 Arquitetura do sistema de métricas 31
4.4 AGREGAÇÃO DOS DADOS PARA VISUALIZAÇÃO E COMPARAÇÃO . . . 32
5 EXPERIMENTOS. 33
5.1 CONJUNTOS DE DADOS 33

5.2 COMPARATIVOS GERAIS 35
5.2.1 Estatísticas gerais 35
5.2.2 Densidades 36
5.2.3 Correlação entre métricas 38
5.2.4 Análise das Características do Determinante (LHS) 40
5.2.5 Ranqueamentos 42
5.3 PARÂMETROS DOS ALGORITMOS 48
5.3.1 Pyro 48
5.3.2 FDX 50
6 CONCLUSÃO 51
6.1 TRABALHOS FUTUROS 52
REFERÊNCIAS 53

1 INTRODUÇÃO

A área de perfilamento de dados tem como objetivo a descoberta de metadados em conjuntos de dados, visando facilitar sua interpretação semântica [13]. Essa prática revela diversos metadados, como restrições, erros, regras de negócios e dependências. As dependências, por sua vez, são particularmente relevantes, pois descrevem a semântica dos bancos de dados e são incorporadas ao modelo relacional [14], evidenciando propriedades de instâncias específicas.

Existem diferentes tipos de dependências. Uma combinação única de colunas ocorre quando há valores exclusivos em uma coluna. Já as dependências funcionais (DFs), foco do presente estudo, são observadas quando os valores de uma coluna - ou conjunto de colunas - determinam funcionalmente os valores de outra. Além disso, a definição de dependência funcional pode ser flexibilizada para as dependências funcionais aproximadas (DFAs), que também serão abordadas. As DFAs permitem que a determinação entre colunas não seja verdadeira para todo o conjunto de dados, mas para uma porcentagem de registros, isso é extremamente útil ao trabalhar com dados que contêm ruídos e erros. Essas propriedades fazem as DFs e DFAs serem fundamentais em atividades de normalização de esquemas relacionais.

Entretanto, a identificação manual de Dependências Funcionais (DFs) e Dependências Funcionais Aproximadas (DFAs) se torna impraticável em cenários com grandes volumes de dados ou com bases de dados que sofrem alterações frequentes. Esse processo exigiria a alocação de inúmeros profissionais com vasto conhecimento tanto em dependências funcionais quanto na semântica específica dos conjuntos de dados, o que seria pouco eficiente. Assim, surge a necessidade de descobrir DFs e DFAs de maneira automática.

Dada a relevância dessa atividade, diversos estudos foram propostos, cada um com uma abordagem distinta para otimizar os resultados. Para a descoberta automática de DFs, o algoritmo HyFD se destaca [15]. Ele introduziu um método inovador para identificar DFs mínimas, não triviais e exatas, ou seja, aquelas que abrangem todo o conjunto de dados. No entanto, em conjuntos de dados do mundo real, que frequentemente contêm ruídos, erros e outras imperfeições, algoritmos exatos como o HyFD podem não ser eficazes na captura de dependências que seriam valiosas para a normalização. Para superar essa limitação, surgem as DFAs. Graças à sua flexibilidade, as DFAs conseguem lidar com esses problemas, mantendo um alto valor para a normalização de esquemas. Para a descoberta automática de DFAs, destacam-se os algoritmos Pyro [9] e FDX [22].

Apesar de sua eficácia em alcançar resultados que seriam dificilmente alcançados manualmente, os algoritmos de descoberta automática de DFs e DFAs frequentemente geram um volume excessivo de dependências. Essa vasta quantidade pode dificultar a seleção e identificação daquelas que seriam mais relevantes para um processo de normalização de dados. Assim, surge a dúvida de como filtrar e escolher as melhores dependências dentro de um grande conjunto de resultados. Uma das contribuições de nosso trabalho é a adoção de métricas para resolver esse problema, ranqueando os resultados a partir delas. Cada métrica foca em um aspecto distinto de uma DF ou DFA, atribuindo uma pontuação a partir de fórmulas matemáticas que consideram a relação em que o algoritmo de descoberta operou. Um estudo abrangente realizado por Parciak et al. [16] apresenta uma análise detalhada de diversas métricas para DFAs, comparando-as em vários cenários. Desse estudo, as métricas g_3, RFI'^{+} e mu^{+} foram destacadas como as mais robustas, e são exatamente essas que utilizaremos em nosso estudo.

1.1 CONTRIBUIÇÕES

O presente estudo apresenta um protocolo para a descoberta e o ranqueamento de dependências funcionais (DFs) e dependências funcionais aproximadas (DFAs). Além disso, a metodologia apresentada no trabalho respondeu às seguintes perguntas:

1. **Qual a eficácia das métricas desenvolvidas para DFAs quando aplicadas a DFs?** Demonstramos que, na maioria dos casos, essas métricas fornecem poucas informações úteis para o ranqueamento de DFs exatas. No entanto, é possível extrair algumas percepções valiosas dessa análise.

2. **Como o tamanho e a unicidade do determinante de uma DF ou DFA afetam a pontuação dada pelas métricas?** Mostramos que a métrica μ^+ é sensível a esses parâmetros, o que a torna uma boa indicadora de chaves para um esquema.

3. **As métricas atribuem pontuações altas a DFs e DFAs de projeto, ou seja, àquelas que capturam o verdadeiro valor semântico dos conjuntos de dados?** Realizamos uma análise semântica do ranqueamento das dependências, exibindo que, em alguns casos, as métricas obtêm sucesso ao atribuir valores altos a DFs e DFAs de projeto.

4. **Em quais contextos cada abordagem é mais eficaz?** Em nossas análises, comparamos os resultados de diferentes abordagens para os mesmos conjuntos de dados. Essa comparação oferece ao leitor discussões que o ajudarão a escolher a abordagem mais adequada ao seu contexto.

1.2 ORGANIZAÇÃO DO DOCUMENTO

O trabalho está organizado da seguinte maneira: o Capítulo 2 fornece a base teórica dos assuntos abordados e uma padronização de termos e expressões. O Capítulo 3 apresenta algoritmos com abordagens distintas para a descoberta de dependências. No Capítulo 4, detalhamos o protocolo criado para desenvolver a metodologia do trabalho. O Capítulo 5 mostra os experimentos realizados juntamente com a análise desses resultados e, por fim, no Capítulo 6, apresentamos a conclusão do estudo.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentadas notações e definições formais de termos e expressões que serão utilizados no restante do trabalho.

2.1 NOTAÇÕES BÁSICAS E CONVENÇÕES

Seguindo a definição de Liu et al. denota-se $R = \{A_1, \ldots, A_m\}$ um **esquema relacional** ou **relação** de banco de dados, e r uma instância de relação (ou tabela) sobre R. Os elementos $A_1, \ldots, A_m$ são denominados **atributos** e formam as colunas da tabela; estes possuem um **domínio** de valores denotado por $dom(A_i)$. Assim, r é um subconjunto do produto cartesiano $dom(A_1) \times \cdots \times dom(A_m)$. Cada linha da tabela é chamada de **tupla** t (ou registro), onde $t \in r$. Para referenciarmos uma tupla, consideramos a posição da tupla como seu identificador, como mostrado na Tabela 2.1. Usamos X e Y para denotar conjunto de colunas, e usamos A para referenciar cada coluna em X, e B para referenciar cada coluna em Y, isto é, $A \in X$ e $B \in Y$. A projeção de uma tupla $t \in r$ sobre uma coluna ou conjunto de colunas é denotada por $t[A]$ ou $t[X]$.

Exemplo 1. A Tabela 2.1 representa a relação $R = \{\text{Nome, Sobrenome, Gênero, CEP, Cidade}\}$. Suponha que $dom(\text{Nome}) = \{\text{Ana, Ariel, João, Maria}\}$, e que a tupla $t_1 = (1, \text{Ana, Souza}, f, 01001, \text{São Paulo})$. Se $X = \{\text{Nome, Sobrenome}\} \subseteq R$, então a projeção $t_1[X] = (\text{Ana, Souza})$.

Uma **superchave** é um subconjunto de atributos $S \subseteq R$ que identifica unicamente cada tupla em uma instância da relação r (Elmasri e Navathe [4]). Ou seja, não podem existir duas tuplas distintas $t_1, t_2 \in r$ tais que $t_1[S] = t_2[S]$.

Exemplo 2. Na Tabela 2.1, o atributo {Id} é uma superchave, pois seu valor é único para cada tupla. O conjunto {Id, Nome} também é uma superchave. Nesta instância específica, o conjunto {Nome, Sobrenome} também atua como uma superchave, pois nenhuma combinação de nome e sobrenome se repete.

Uma **chave candidata** é uma superchave *minimal*. Isso significa que, se qualquer atributo for removido do conjunto da chave candidata, ela deixa de ser uma superchave (Elmasri e Navathe [4]). Uma relação pode ter várias chaves candidatas, e a **chave primária** é a chave candidata escolhida para ser o identificador principal da tabela.

Exemplo 3. O conjunto {Id} é uma chave candidata para a relação na Tabela 2.1, pois é uma superchave e é minimal (não se pode remover nenhum atributo). O conjunto {Nome, Sobrenome} também é uma chave candidata nesta instância, pois é uma superchave e nem {Nome} nem {Sobrenome} isoladamente são superchaves. Por outro lado, o conjunto {Id, Nome} **não** é uma chave candidata, pois embora seja uma superchave, não é minimal, dado que seu subconjunto {Id} já é uma superchave. Por convenção, {Id} seria escolhido como a chave primária.

Tabela 2.1: Exemplo de relação

	Id	**Nome**	**Sobrenome**	**Gênero**	**CEP**	**Cidade**
t_1	1	Ana	Souza	f	01001	São Paulo
t_2	2	Ariel	Lima	m	03003	Belo Horizonte
t_3	3	João	Pereira	m	02002	Rio de Janeiro
t_4	4	Maria	Oliveira	f	01001	São Paulo
t_5	5	João	Lima	m	03003	Belo Horizonte
t_6	6	Ana	Mendes	f	02002	Rio de Janeiro
t_7	7	Ariel	Ferreira	m	04004	Curitiba
t_8	8	Ariel	Almeida	f	04004	Curitiba

2.2 DEPENDÊNCIAS

Conforme mostrado por Pena e De Almeida [17], dependências de dados (ou, simplesmente, *dependências*) constituem uma categoria relevante de restrições que descrevem a semântica dos bancos de dados, incorporando-a ao modelo relacional. Enquanto o termo *restrição* geralmente se refere a propriedades relacionadas ao projeto do banco de dados, o termo *dependência* refere-se a propriedades de instâncias específicas do banco de dados, as quais podem — ou não — demandar aplicação. Por exemplo, valores únicos em uma coluna podem configurar uma dependência conhecida como *combinação única de colunas*.

As dependências presentes em tabelas únicas, denominadas *dependências intra-relação*, definem propriedades semânticas entre uma coluna ou conjunto de colunas. Sua definição é menos rígida do que a de restrições, sendo aplicável a instâncias específicas do banco de dados.

Dependências funcionais são um exemplo clássico de dependências intra-relação. No entanto, outros tipos também são comuns, como as combinações únicas de colunas e as dependências de ordem.

2.3 DEPENDÊNCIA FUNCIONAL

Uma dependência funcional $X \rightarrow Y$ é uma declaração sobre uma instância relacional r de R, onde $X \subseteq R$ e $Y \subseteq R$. Uma dependência funcional, ou DF, é dita válida para em r se:

$$\forall t_x, t_y \in r : t_x[X] = t_y[X] \Rightarrow t_x[Y] = t_y[Y].$$

Em outras palavras, os valores em X determinam funcionalmente os valores em Y [17]. Um exemplo de dependência funcional, considerando a relação da tabela 2.1 seria $CEP \rightarrow Cidade$. Essa dependência funcional sugere que dado o atributo *CEP* é possível determinar de forma única o atributo *Cidade*.

O determinante X é chamado de lado esquerdo ou *left hand side (lhs)*, e o dependente Y de lado direito ou *right hand side (rhs)*. Uma DF é não trivial se não possui nenhum atributo redundante ($X \nsubseteq Y$), e é mínima se não existe conjunto Z tal que $(X - Z) \rightarrow Y$ é uma DF válida. Podemos decompor uma DF em múltiplas DFs usando cada coluna em Y, por exemplo, considere a DF $X \rightarrow Y$, onde $Y = \{B_1, B_2\}$, as duas DFs $X \rightarrow B_1$ e $X \rightarrow B_2$ são equivalentes à única DF $X \rightarrow Y$. Uma DF $X \rightarrow Y$ é uma generalização de uma outra DF $Z \rightarrow Y$ se $X \subset Z$, e é uma especialização se $X \supset Z$ [15].

2.4 DEPENDÊNCIAS RELAXADAS

Dependências relaxadas estendem sua definição com o objetivo de lidar com erros, exceções ou ambiguidades em conjuntos de dados que não são cobertos por dependências exatas [9]. Para isso, há duas abordagens para definir uma dependência relaxada: uma baseada em um *critério de satisfazibilidade* e outra baseada na *comparação de colunas* [1]. Neste trabalho, será abordada apenas a primeira definição.

Um exemplo de dependência baseada no *critério de satisfazibilidade* é a *dependência funcional aproximada* (DFA), na qual o critério se refere ao número de violações, dentro de uma tabela, que não satisfazem a dependência [1]. Se esse número estiver abaixo de um *valor limite*, a DFA é considerada válida para a instância [9]. Considerando a Tabela 2.1, a dependência funcional *Nome* $\rightarrow$ *Gênero* é um exemplo de *dependência funcional aproximada*, assumindo um valor limite de uma tupla violada (tuplas t_2, t_7 e t_8).

2.5 INTERPRETAÇÃO PROBABILÍSTICA DE DF

A notação $P(Y = y \mid X = x)$ representa a probabilidade condicional de uma variável aleatória Y assumir o valor y, dado que a variável X assume o valor x, segundo uma distribuição de probabilidade P.

Segundo Zhang et al. [22], uma interpretação probabilística de dependências funcionais considera que, dada uma distribuição de probabilidade P_R sobre um esquema relacional R, uma FD $X \rightarrow Y$ é verdadeira se existe uma função $\phi : V(X) \rightarrow V(Y)$ tal que, para todo $x \in V(X)$:

$$P_R(Y = y|X = x) = \begin{cases} 1 - \epsilon, & \text{quando } y = \phi(x) \\ \epsilon, & \text{caso contrário} \end{cases}$$

onde ϵ é uma pequena constante. Esta condição permite que uma FD seja válida para a maioria das tuplas, admitindo algumas violações. Essa equação captura a essência das FDs aproximadas usadas em múltiplos trabalhos.

Existe ainda outra interpretação mostrada por Zhang et al. [22]: Em vez de modelar diretamente a estrutura da distribuição P_R, considera-se uma distribuição diferente com estrutura equivalente com respeito às FDs presentes em P_R. Para qualquer par de tuplas t_i e t_j amostradas de P_R, considera-se a variável aleatória $I_{ij}[Y] = 1(t_i[Y] = t_j[Y])$, onde $1(\cdot)$ é a função indicadora. Uma FD $X \rightarrow Y$ é verdadeira para P_R se, para todos os pares de tuplas t_i, t_j em R, temos a seguinte condição para a distribuição sobre as variáveis aleatórias $I_{ij}[Y]$: $Pr(I_{ij}[Y] = 1|t_i[X] = t_j[X]) = 1 - \epsilon$ onde ϵ é uma pequena constante para garantir robustez contra ruído. Esta condição estabelece que os eventos aleatórios $(\wedge_{A \in X}(t_i[A] = t_j[A]))$ e $(1(t_i[Y] = t_j[Y]))$ estão deterministicamente correlacionados, o que é equivalente à FD $X \rightarrow Y$.

2.6 PERFILAMENTO DE DADOS

Um esquema relacional não possui por si só, dados semânticos sobre dependências, restrições e regras entre seus atributos. O perfilamento de dados, ou *data profiling*, consiste em um conjunto de tarefas que ajudam na descoberta e extração de metadados sobre um conjunto de dados. Existem vários tipos de metadados que podem ser extraídos, como restrições, dependências, erros, regras de negócio etc. Todas com o objetivo de facilitar a interpretação semântica sobre os dados, tanto por humanos como para programas e algoritmos. Neste trabalho, focaremos em um tipo de metadado chamado de dependência funcional (DF), definido nas seções acima, pelo fato

de ser amplamente utilizado na normalização de esquemas e redução de redundância de bancos de dados. Ainda assim, é interessante citar alguns outros tipos de metadados, como:

- Restrições de integridade - As restrições de integridade são condições que os dados devem satisfazer. Estas são simples e muitas vezes enforçadas pelo próprio sistema de gerenciamento de bancos de dados. Exemplos de restrições tradicionais de integridade são restrições de chaves, restrições de chaves externas e restrições de domínio [17].

- Restrições de negação ou *Denial constraints* - Definem conjuntos predicados para os quais não pode existir par de tupla em R que seja verdade para todos os predicados da restrição. Sendo assim, seja a restrição de negação φ_1, com predicados $p_1, p_2, \cdots, p_n \in \varphi 1$:

  $$\varphi 1 = \neg(p_1 \wedge p_2 \wedge \cdots \wedge p_n)$$

  Vemos em Chu et al. [2] que caso exista um par de tuplas em que todos os predicados de φ_1 sejam verdade, trata-se de uma violação de φ_1.

- Dependências funcionais condicionais - Segundo Fan et al. [5], uma dependência funcional condicional (CFD) σ sobre uma relação R é um par (X $\rightarrow$ A, t_p). Este par é composto por:

  1. X é um conjunto de atributos em R, e A é um único atributo em R.
  2. X $\rightarrow$ A é uma DF padrão, referida como a DF embutida em σ.
  3. t_p é uma tupla padrão com atributos em X e A, onde para cada atributo B em $X \cup A$, $t_p[B]$ é ou uma constante 'a' no domínio de B ($dom(B)$), ou uma variável sem nome '_' que corresponde a um valor arbitrário [5].

2.6.1 Descoberta de dependências

A descoberta de dependências é o processo de encontrar um conjunto de dependências, formuladas em uma linguagem específica, que se aplicam a uma dada relação. A abordagem padrão para essa tarefa envolve a enumeração e validação de um conjunto de dependências candidatas. A principal dificuldade computacional desse processo decorre do fato de que o espaço de busca por candidatos é exponencial em relação ao número de atributos do esquema relacional, podendo ser ainda maior a depender da classe de dependência investigada. Consequentemente, várias soluções foram desenvolvidas para otimizar essa busca e garantir um desempenho satisfatório [17, 10].

2.7 DESCOBERTA DE DEPENDÊNCIAS FUNCIONAIS

A descoberta automática de dependências funcionais é uma etapa fundamental em rotinas de perfilamento e engenharia de dados, pois, a partir delas, pode-se normalizar esquemas, melhorar consultas, evitar duplicação de dados e diversas outras utilidades [1, 14]. Por este motivo, além do fato de possuir um alto custo computacional — mais precisamente, ser um problema NP-Completo [3] — esta tarefa é tópico de muitos estudos que serão citados neste trabalho, abrindo espaços para otimizações, melhorias e diferentes abordagens [10]. As mais clássicas, alvos de estudos como Papenbrock et al. [14], são aquelas que enumeram todas as DFs mínimas e não triviais dado um conjunto de dados de entrada, sendo alguns algoritmos capazes de encontrar dependências aproximadas, como em Kruse e Naumann [9]. Existem também

abordagens alternativas, como é mostrado em Zhang et al. [22], que traz uma solução estatística para o problema de descoberta.

2.7.1 Definição formal

Podemos definir formalmente a descoberta de dependências funcionais como o problema computacional que recebe como entrada um esquema relacional R e uma instância de dados r sobre R e tem como saída um conjunto de dependências funcionais DF contendo dependências da forma $X \rightarrow Y$, sendo X um conjunto de atributos $X \subseteq R$ e Y um único atributo $Y \in R$.

2.7.2 Resultados não confiáveis

Na descoberta automática de dependências funcionais, especialmente em bases de dados reais com ruídos, valores nulos ou inconsistências, é comum a identificação de dependências que não refletem corretamente o relacionamento entre os atributos, principalmente quando se trata de DFs exatas. Em muitos casos, buscando uma maior robustez às inconsistências dos dados de entrada, usa-se algoritmos de abordagens não exatas para a descoberta de DFs, como é visto em [9, 22] e outros trabalhos. Consideramos então como dependências *não confiáveis* ou *de baixo valor semântico* aquelas que são enumeradas por algoritmos devido a coincidências ou à presença de padrões falsos induzidos por registros nulos ou anomalias.

Esse tipo de erro está diretamente relacionado ao sobreajuste (*overfitting*), que ocorre quando o algoritmo se ajusta excessivamente aos dados de entrada. Isso acontece porque, em uma amostra finita de dados, à medida que o número de atributos no conjunto determinante (o lado esquerdo da DF, X) aumenta, torna-se empiricamente mais provável que a condição para que uma DF $X \rightarrow Y$ seja satisfeita pareça verdadeira por acaso [22]. Esse comportamento leva à descoberta de dependências não confiáveis e à identificação de estruturas de dependência complexas (densas) entre os atributos. A identificação de um grande número de DFs errôneas ou complexas torna difícil a interpretação e a validação dessas dependências, fazendo com que sejam necessários métodos de pós-processamento, medição e ranqueamento dos resultados encontrados [16].

Uma alternativa ao pós-processamento é a utilização de heurísticas para descartar candidatos durante a descoberta de DFAs, como é visto em [21]. No entanto, uma possível consequência disso é o oposto do sobreajuste: o subajuste ou *underfitting*, em que candidatos que deveriam ser enumerados acabam sendo descartados por uma heurística muito conservadora ou rígida.

2.7.3 Descoberta de Dependências Funcionais Aproximadas

De forma geral, a descoberta de dependências funcionais aproximadas é mais difícil que a descoberta de dependências funcionais exatas. Isso ocorre porque o principal desafio para a descoberta de DFs exatas é o grande espaço de busca que cresce exponencialmente com o número de atributos de um conjunto de dados. Isso é mitigado com podas intensas empregadas pelos algoritmos, de maneira que logo que é descoberta apenas uma violação de um candidato a dependência ele é podado do espaço de busca. Esse princípio diminui a quantidade de processamento necessário, porém não funciona para as dependências funcionais aproximadas, pois elas permitem um certo nível de violações [9].

Algoritmos de descoberta de DFs exatas podem ser adaptados para a descoberta de DFAs, como é o caso do algoritmo *TANE*. Por outro lado, o algoritmo *Pyro*, que foi projetado

especificamente para DFAs — e não como adaptação de um algoritmo para DFs — destaca-se como uma das soluções mais rápidas atualmente [9].

2.8 MÉTRICAS E RANQUEAMENTO DE DEPENDÊNCIAS

O número de dependências encontradas por algoritmos de enumeração de dependências funcionais pode ser muito grande, o que dificulta o trabalho de quem analisa e utiliza estes resultados para melhorar o esquema de banco de dados ou realizar rotinas de limpeza de dados. No contexto de dependências funcionais aproximadas, ou seja, aquelas que são verdadeiras para uma porcentagem dos pares de tuplas, mas não para todos, é útil ter uma maneira de avaliar e ordenar esses resultados para analisar dependências mais relevantes, além de descartar dependências potencialmente espúrias.

Para isso, existem métricas que ajudam neste processo. As métricas são calculadas tendo como entrada um esquema R, uma instância r e uma dependência funcional; sua saída é um número, geralmente entre 0 e 1, onde valores mais altos indicam uma DF de maior satisfação. Segundo Parciak et al. [16], uma boa métrica de DF é aquela que retorna valores altos em DFs que podem ser usadas em projeto de esquema, e baixos em DFs que não podem.

Assim, uma métrica de DFAs fornece uma maneira de ranquear o espaço de busca de todas as DFs, onde DFs com pontuação alta são ranqueadas antes de DFs com pontuação baixa. Com isso, uma boa métrica de DFA é aquela que ranqueia DFs que estão no conjunto alvo acima das que não estão.

2.8.1 Métricas para Dependências Funcionais Aproximadas

Uma métrica para uma dependência funcional aproximada é uma função que mapeia os pares (φ, R) em um número no intervalo $[0, 1]$, onde φ representa uma DF e R uma relação. O valor atribuído indica o grau em que a DF φ é satisfeita na relação R: quanto mais próximo de 1, menor o número de violações; se R satisfaz completamente φ, então $f(\varphi, R) = 1$ [16].

Parciak et al. [16] classifica as métricas em três categorias: (i) métricas que quantificam a taxa de violações, denominadas *VIOLATION*; (ii) métricas baseadas na entropia de Shannon (*SHANNON*); e (iii) métricas baseadas na entropia lógica (*LOGICAL*). O artigo também analisa como os parâmetros taxa de erro, *LHS-uniqueness* e *RHS-skew* impactam o desempenho das métricas. A conclusão é que as métricas *RFI'+*, $\mu+$ e g'_3 são as menos afetadas por esses parâmetros, com a ressalva de que g'_3 é sensível ao aumento do *RHS-skew*, conforme ilustrado na Figura 4.1.

2.8.2 Parâmetros

Parciak et al. [16] aponta que diversas propriedades da relação de entrada R podem influenciar o comportamento das métricas. Entre as mais relevantes, destacam-se:

(1) **Taxa de erros**: um requisito desejável para uma boa métrica de DFA é que ela seja inversamente proporcional à taxa de erros, ou seja, um aumento no número de erros deve resultar em uma diminuição no valor da métrica.

(2) **Estatísticas do lado esquerdo (LHS) e lado direito (RHS)** de uma DF $X \rightarrow Y$: mais especificamente, a *unicidade do LHS* (*LHS-uniqueness*) quantifica a proporção de valores distintos de X em R, e pode ser definida como $|dom_R(X)|/|R|$. Já a *assimetria do RHS* (*RHS-skew*) é definida como a distribuição de probabilidade conjunta de Y [16].

2.8.3 Medidas de Erro

As métricas-g (*g-measures*), definidas por Kivinen e Mannila [8] — às quais nos referiremos como **medidas de erro** — têm como objetivo quantificar o número de violações de uma dependência funcional em uma relação R. Essas medidas podem ser convertidas em métricas, sendo elas: g_1, g'_1, g_2 e g_3 [16]. Devido à sua natureza, essas métricas são classificadas na categoria *VIOLATION*.

Conforme apresentado por Parciak et al. [16], as métricas g_1, g'_1, g_2 e g_3 são sensíveis aos parâmetros *error-rate*, *LHS-uniqueness* e *RHS-skew*, razão pela qual não serão utilizadas na presente avaliação. Apenas a métrica g'_3 [7] demonstrou ser pouco influenciada por esses parâmetros, com uma ressalva em relação ao parâmetro *RHS-skew*, como ilustrado na Figura 4.1.

A métrica g'_3 é uma versão normalizada da métrica g_3, que calcula o tamanho relativo da maior sub-relação de R para a qual uma dependência funcional $X \to Y$ é satisfeita. Mais especificamente, definimos $R'(W)$ como uma sub-relação de $R(W)$, denotada por $R' \subseteq R(W)$, tal que $R'(w) \leq R(w)$ para todo $w \in W$. Sendo $G_3(X \to Y, R)$ o conjunto de todas as sub-relações de R que satisfazem $X \to Y$, temos:

$$G_3(X \to Y, R) := \{R' \mid R' \subseteq R,\ R' \models X \to Y\}.$$

Assim, g_3 é definido como o tamanho relativo máximo dessas sub-relações:

$$g_3(X \to Y, R) := \max_{R' \in G_3(X \to Y, R)} \frac{|R'|}{|R|}.$$

Por fim, Giannella e Robertson [7] propuseram a versão normalizada da métrica g_3, denotada por g'_3, definida como:

$$g'_3(X \to Y, R) := \max_{R' \in G_3(X \to Y, R)} \frac{|R'| - |\mathrm{dom}_R(\mathbf{X})|}{|R| - |\mathrm{dom}_R(\mathbf{X})|}.$$

2.8.4 Fração de Informação

Como definido por Roulston [19], a *fraction of information* (FI), ou **fração de informação**, é uma forma de generalizar dependências funcionais (DFs) de conjuntos de dados determinísticos para probabilísticos. Posteriormente, seu uso como métrica foi estudado por Giannella e Robertson [7], sendo definida da seguinte forma:

$$\mathrm{FI}(X \to Y, R) := \frac{H_R(Y) - H_R(Y \mid X)}{H_R(Y)}.$$

O numerador, $H_R(Y) - H_R(Y \mid X)$, é conhecido como **informação mútua**, denotada por $I_R(X;Y)$ [16].

Dada uma DF $\varphi : X \to Y$, a FI representa a proporção da incerteza sobre Y que é reduzida ao se conhecer X. Aqui, $H_R(Y)$ mede a incerteza de Y, enquanto $H_R(Y \mid X)$ representa a incerteza de Y após se conhecer X. Assim, quando R satisfaz φ, não há incerteza sobre Y dado X, logo $H_R(Y \mid X) = 0$, e, portanto, FI = 1. Por outro lado, se X e Y são variáveis independentes em R, não há redução de incerteza, então $H_R(Y \mid X) = H_R(Y)$, e FI = 0. A métrica FI e todas as suas variações são da classe *SHANNON* [16].

Um refinamento da FI, desenvolvido por Mandros et al. [11, 12], é a *reliable fraction of information* (RFI), ou **fração de informação confiável**, que corrige o viés de superestimação presente na FI [16]. A RFI é definida como:

$$\mathrm{RFI}(\varphi, R) := \mathrm{FI}(\varphi, R) - \mathbb{E}_R[\mathrm{FI}(\varphi, R)]$$

em que $\mathbb{E}_R[\mathrm{FI}(\varphi, R)]$ representa o valor esperado da FI sob todas as permutações da relação R, considerando as colunas X e Y fixas. Essa subtração tem o objetivo de compensar o viés observado por Mandros et al. [11].

No entanto, essa correção pode gerar valores negativos, o que impede a RFI de ser considerada uma métrica válida. Para contornar esse problema, foi proposta a métrica:

$$\mathrm{RFI}^+(\varphi, R) := \max(\mathrm{RFI}(\varphi, R), 0).$$

Contudo, como a RFI corrige o viés subtraindo a expectativa $\mathbb{E}_R[\mathrm{FI}]$ da FI de maneira absoluta, ela é uma medida não normalizada. Assim, sua variante normalizada é definida como:

$$\mathrm{RFI}'^+(\varphi, R) := \max\left(\frac{\mathrm{FI}(\varphi, R) - \mathbb{E}_R[\mathrm{FI}(\varphi, R)]}{1 - \mathbb{E}_R[\mathrm{FI}(\varphi, R)]},\ 0\right).$$

Conforme mostrado por Parciak et al. [16], apenas a métrica RFI'^+, dentre aquelas pertencentes à classe *SHANNON*, apresenta influência mínima dos parâmetros *error-rate*, *LHS-uniqueness* e *RHS-skew*. Por esse motivo, apenas essa métrica foi considerada em nossa avaliação.

2.8.5 Medidas de probabilidade

A **probabilidade determinística** de Y dado X em R, também conhecida como *probabilistic dependency* (pdep), foi proposta por Piatetsky-Shapiro e Matheus [18] e é denotada por $pdep(X \to Y, R)$. Essa medida representa a probabilidade condicional de que duas tuplas, retiradas aleatoriamente com reposição da relação R, sejam iguais em Y, dado que são iguais em X. Formalmente:

$$\mathrm{pdep}(X \to Y, R) := \sum_{\mathbf{x}} p_R(\mathbf{x})\, \mathrm{pdep}(Y \mid \mathbf{x}, R),$$

onde $\mathrm{pdep}(Y \mid \mathbf{x}, R)$ é a probabilidade de que duas tuplas com valor $\mathbf{x}$ em X, extraídas com reposição da distribuição condicional $p_R(Y \mid \mathbf{x})$, sejam iguais em Y. Essa probabilidade é definida como:

$$\mathrm{pdep}(Y \mid \mathbf{x}, R) := \sum_{y} p_R(y \mid \mathbf{x})^2 = 1 - h_R(Y \mid \mathbf{x}).$$

A **probabilidade determinística** é, portanto, uma medida baseada na entropia lógica e é classificada na categoria *LOGICAL*. Ela pode ser interpretada da seguinte forma: dada uma condição em que duas tuplas possuem o mesmo valor $\mathbf{x}$ em X, $pdep(Y \mid \mathbf{x}, R)$ expressa a probabilidade de que essas tuplas também sejam iguais em Y, e $pdep(X \to Y, R)$ é o valor esperado dessa probabilidade sobre todos os possíveis valores de $\mathbf{x}$.

Duas outras medidas foram derivadas de $pdep$: τ e μ [18]. A primeira é uma versão normalizada de $pdep$, enquanto a segunda corrige o viés que τ apresenta em relação ao tamanho do domínio dos atributos de R [16].

Conforme demonstrado por Parciak et al. [16], as medidas $pdep$, τ e μ são fortemente influenciadas pelos parâmetros *error-rate*, *LHS-uniqueness* e *RHS-skew*, motivo pelo qual não as utilizamos em nossa avaliação. Apenas a métrica μ^+ demonstrou ser pouco sensível a esses parâmetros.

A métrica μ é definida como:

$$\mu(X \to Y, R) := \frac{\text{pdep}(X \to Y, R) - \mathbb{E}_R[\text{pdep}(X \to Y, R)]}{1 - \mathbb{E}_R[\text{pdep}(X \to Y, R)]}.$$

Considerando uma relação aleatória R de tamanho $N \geq 2$ contendo os atributos X e Y, com $K = |\text{dom}_R(X)|$, o valor esperado de $pdep$ sob permutações aleatórias de R é dado por:

$$\mathbb{E}_R[\text{pdep}(X \to Y, R)] := \text{pdep}(X \to Y, R) + \frac{K-1}{N-1}\left(1 - \text{pdep}(Y, R)\right).$$

No entanto, como μ pode assumir valores negativos, ela não é considerada uma métrica. Para contornar isso, foi proposta a métrica μ^+ [16], definida como:

$$\mu^+(X \to Y, R) := \max\left(\mu(X \to Y, R), 0\right).$$

3 ABORDAGENS DE DESCOBERTA DE DEPENDÊNCIAS FUNCIONAIS

Como foi discutido em 2.6, o perfilamento de dados trata-se da extração de metadados de um conjunto de dados de entrada, já que essa entrada não possui informações sobre como seus atributos se relacionam, suas dependências, regras de negócio e restrições de integridade.

Cada tipo de metadado extraído possui uma função dentro do contexto de gerenciamento de dados, seja para limpeza, normalização, redução de anomalias ou erros, ou até mesmo melhor entendimento humano sobre aquele conjunto. A descoberta desses metadados a partir dos próprios dados de entrada de maneira automática é fundamental para essas rotinas, já que realizar esse processo manualmente torna-se muito difícil conforme o tamanho e a complexidade dos dados de entrada [10]. Este trabalho terá como foco o estudo e comparação de algoritmos de descoberta de dependências funcionais, um dos metadados mais importantes no projeto de bancos de dados, já que DFs são utilizadas no processo de normalização de esquemas e, portanto, na redução de redundância.

Este capítulo explorará três algoritmos que representam três abordagens distintas para a descoberta de DFs. A primeira delas trata-se da descoberta de DFs exatas, ou seja, aquelas que são verdadeiras para todos os pares de tuplas do conjunto de dados de entrada, sendo essa a mais tradicional e que possui diversos outros trabalhos relacionados. Papenbrock et al. [14] traz um estudo comparativo de 7 algoritmos de descoberta de DFs exatas. Essa abordagem utiliza o conjunto potência de todas as combinações de atributos para encontrar dependências funcionais mínimas e não triviais. Liu et al. [10] mostra que a complexidade dessa abordagem, dado uma relação R e uma instância r é de $O\left(n^2 \left(\frac{m}{2}\right)^2 2^m\right)$, onde m é o número de atributos de R e n é o número de linhas de r. Por mais que existam otimizações e diferentes tipos de busca nessa descoberta de DFs exatas, a saída esperada de qualquer algoritmo que siga esse formato deve ser o mesmo: todas as dependências funcionais mínimas e não triviais que são respeitadas por 100% dos pares de tuplas de r. Como vemos em Papenbrock e Naumann [15], existem algoritmos que otimizam a descoberta para ganhar escalabilidade no número de colunas, outros, no número de tuplas de r. Já o algoritmo que escolhido como representante desta abordagem, o *HyFD*, utiliza uma forma híbrida, de modo a ganhar escalabilidade nas duas dimensões.

Entretanto, conjuntos de dados reais podem conter vários tipos de anomalias, inconsistências e dados nulos. Sendo assim, a descoberta de DFs exatas se torna insuficiente nesse contexto, uma vez que grande parte das dependências encontradas pode se tratar de coincidências ou não representar de maneira correta a forma como os atributos dependem uns dos outros. Desta forma, buscando mais robustez a esse tipo de entrada, apresentamos a segunda abordagem a ser estudada neste trabalho: algoritmos que encontram dependências funcionais aproximadas, ou seja, aquelas que não precisam ser verdadeiras para todos os pares de tuplas, mas para uma porcentagem definida deles. A descoberta de dependências aproximadas é considerada uma tarefa ainda mais desafiadora do que a já complexa descoberta de dependências exatas [9]. A principal dificuldade para ambas as tarefas reside no enorme espaço de busca que cresce exponencialmente com o número de atributos em um conjunto de dados. No entanto, algoritmos de descoberta exata podem empregar podas agressivas: ao encontrar um único contraexemplo para um candidato a dependência, eles podem podá-lo imediatamente, o que não funciona quando violações são permitidas. Diante desse desafio, foi proposto o algoritmo *Pyro* para a descoberta de DFAs, que utiliza a técnica de divisão e conquista e amostragem focada para encontrar e validar de maneira eficiente os candidatos a DFAs. O Pyro consegue mitigar, mas não evitar completamente, os efeitos dessa complexidade exponencial [9].

Encontrar DFs aproximadas é uma solução que traz robustez a conjuntos de dados reais com anomalias. No entanto, aumenta ainda mais o número de dependências encontradas, já que seu critério elimina menos candidatos, fazendo com que a tarefa de selecionar dependências para projetos e normalização seja ainda mais difícil. Tentando resolver esse problema, surgem abordagens alternativas à simples enumeração de todos os candidatos que respeitem a condição de minimalidade e não trivialidade de DFs, como a solução proposta em Zhang et al. [22], que se trata de uma mudança completa de paradigma. Nessa abordagem, usa-se uma perspectiva estatística e probabilística para a descoberta de DFs, utilizando-se da técnica de aprendizado de estrutura. A descoberta de DFs, nessa abordagem, é equivalente ao aprendizado de estrutura de um modelo sobre variáveis binárias aleatórias, onde cada variável corresponde a uma funcionalidade dos atributos do conjunto de dados [22]. Seu resultado é um conjunto muito menor de dependências funcionais, chamadas por Zhang et al. [22] de dependências parcimoniosas.

Nas próximas seções, entraremos em mais detalhes de implementação dos algoritmos que representam cada uma das abordagens mencionadas acima, sendo eles o *HyFD* o representante dos algoritmos clássicos de descoberta de dependências funcionais exatas, o *Pyro* na descoberta de DFs aproximadas, e o *FDX* trazendo uma solução estatística para o problema.

3.1 HYFD

Devido a limitações de eficiência, a maioria dos algoritmos atuais de descoberta de dependências funcionais (DFs) não consegue processar conjuntos de dados que ultrapassem 50 colunas e 1 milhão de linhas — volumes comuns em aplicações práticas [15]. Isso ocorre porque tais algoritmos normalmente otimizam apenas um entre dois fatores: o número de atributos ou o número de registros, mas não ambos simultaneamente [15]. Como a complexidade do problema é $O\left(n^2 \left(\frac{m}{2}\right)^2 2^m\right)$ [15], a falta de uma abordagem que lide com essas duas dimensões compromete sua escalabilidade [15].

Para superar esse desafio, foi desenvolvido o *HyFD*, um algoritmo híbrido que otimiza tanto a análise dos registros quanto dos atributos. Ele opera em duas fases interdependentes: na primeira, extrai-se uma amostra estratégica do conjunto de dados, sobre a qual são geradas as DFs candidatas; na segunda, essas DFs são validadas no conjunto completo, sendo refinadas ou descartadas conforme necessário. Se a validação se tornar ineficiente, o algoritmo retorna à primeira fase, utilizando os resultados obtidos até o momento [15].

A alternância entre as fases permite reduzir a complexidade associada a ambas as dimensões do problema. A primeira fase minimiza o impacto do número de colunas ao trabalhar com menos registros, enquanto a segunda reduz o custo dos dados ao restringir o espaço de busca com base nas DFs descobertas. Essa estratégia, aliada às otimizações implementadas, permite ao *HyFD* superar outros algoritmos em termos de tempo de execução e escalabilidade, mantendo a geração de DFs mínimas [15].

O *HyFD* é estruturado em componentes especializados que operam de forma interligada e iterativa, alternando entre geração de candidatos, validação e controle de recursos, como representado na Figura 3.1. Os componentes *Sampler* e *Inductor* integram a primeira fase do algoritmo, responsável pela geração de candidatos. O *Validator* compõe a segunda fase, onde ocorre a verificação em todo o conjunto de dados. Já o *Guardian*, opcional, atua no controle de memória durante a execução.

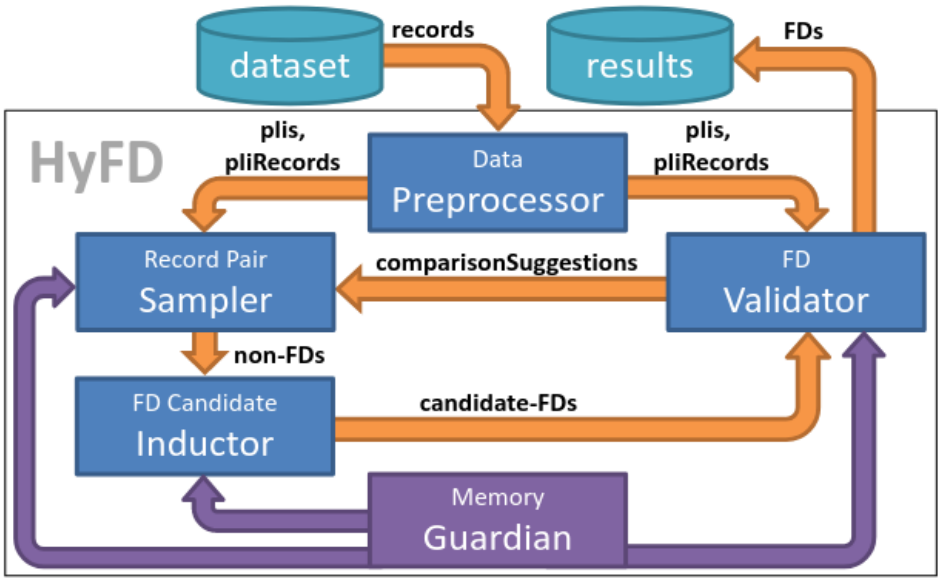


Figura 3.1: Fluxo de execução dos componentes do HyFD. Extraído de [15]

3.2 PYRO

Algoritmos de descoberta de DFs exatas podem ignorar potenciais candidatos devido a erros, ambiguidades ou exceções presentes em conjuntos de dados do mundo real. Diante disso, surge a necessidade de identificar dependências funcionais aproximadas (DFAs), nas quais violações são toleradas dentro de um limite aceitável, a fim de contornar os problemas mencionados [9].

O algoritmo *Pyro* [9] é voltado para a descoberta de Dependências Funcionais Aproximadas e Combinações de Colunas Únicas Aproximadas. No entanto, o foco deste trabalho é apenas nas DFAs. Para isso, o *Pyro* considera dependências que podem ser violadas por uma certa proporção de tuplas ou pares de tuplas.

Para a descoberta de todas as DFAs mínimas com base em um erro limite, o *Pyro* adota uma abordagem unificada fundamentada em duas ideias principais: (i) utilizar uma amostragem direcionada para gerar candidatos a DFAs e (ii) explorar o espaço de busca de forma a validar os candidatos com o menor custo computacional possível.

Segundo Kruse e Naumann [9], o algoritmo quantifica o grau de aproximação de uma DFA com base em uma adaptação da métrica de erro g_1. A fórmula para o cálculo do erro de um candidato a DFA $X \rightarrow A$ em um conjunto de dados r é definida como:

$$e(X \rightarrow A, r) = \frac{|\{(t_1, t_2) \in r^2 \mid t_1[X] = t_2[X] \wedge t_1[A] \neq t_2[A]\}|}{|r|^2 - |r|}$$

Exemplo 1. Para a Tabela 2.1, podemos calcular $e(first\_name \rightarrow gender, r) = \frac{4}{8^2-8} = 0{,}07$, considerando as violações pelas tuplas (t_2, t_8), (t_7, t_8) e seus inversos.

Assim, dada uma relação r e um erro limite e_ϕ, o algoritmo *Pyro* busca determinar todas as DFAs *mínimas* com apenas um atributo no lado direito (*RHS*). Segundo Kruse e Naumann [9], uma DFA é considerada mínima se apresenta um erro inferior ou igual a e_ϕ, e todas as suas generalizações possuem erro superior a esse limite. Seja uma DFA $\varphi' = XA \rightarrow Y$, ao removermos atributos do LHS de φ, obtendo $\varphi' = X \rightarrow Y$, φ' é uma generalização de φ (Kruse e Naumann [9]).

Com isso, dado um conjunto de dados com n atributos, o algoritmo *Pyro* inicia $n + 1$ espaços de busca: um para cada atributo, com o objetivo de descobrir as DFAs mínimas que possuem esse atributo como lado direito (*RHS*). Cada espaço de busca é modelado como um reticulado do conjunto potência (*powerset lattice*) dos atributos da relação, composto por todos os subconjuntos dos atributos, exceto o atributo A, que será o *RHS*. Nesse contexto, cada conjunto de atributos X no espaço de busca representa uma DFA candidata $X \rightarrow A$, de modo que cada nó da estrutura forma exatamente um candidato.

O *Pyro* percorre o espaço de busca utilizando uma estratégia de divisão e conquista para identificar DFAs mínimas. Essa abordagem faz uso de estruturas de dados otimizadas, que permitem calcular estimativas de erro de forma eficiente. A partir dessas estimativas, o algoritmo localiza dependências promissoras e as valida poucos cálculos de erros completos.

3.3 FDX

Os algoritmos propostos até o momento trazem em sua saída todas as DFs encontradas que são respeitadas por todos os pares de tuplas ou por uma porcentagem definida deles. Estas abordagens de força bruta possuem alguns problemas, como o tamanho de sua saída, em que pode-se obter um número de DFs algumas vezes maior que o próprio conjunto de dados. Além disso, muitas dessas DFs são espúrias, devido ao sobreajuste causado por esse tipo de enumeração combinatória de dependências funcionais. Como consequência, muitas vezes é necessário realizar algum tipo de pós-processamento para limpar a saída desses algoritmos e selecionar DFs mais relevantes.

O FDX tenta solucionar este problema propondo uma abordagem estatística para a descoberta de DFs. Segundo Zhang et al. [22], descobrir DFs é equivalente a aprender a estrutura de um modelo sobre variáveis aleatórias binárias, onde cada variável vem de uma função baseada nos atributos do conjunto de dados.

Assim como os outros algoritmos de descoberta de DFs, o **FDX** possui como entrada um conjunto de dados D que segue um modelo Relacional R, podendo D conter dados com anomalias, como dados nulos ou errôneos. A saída dele é um conjunto de dependências funcionais da forma $X \rightarrow Y$, onde X pode ser um conjunto de atributos, enquanto Y corresponde a um único atributo.

A execução do FDX passa por três etapas:

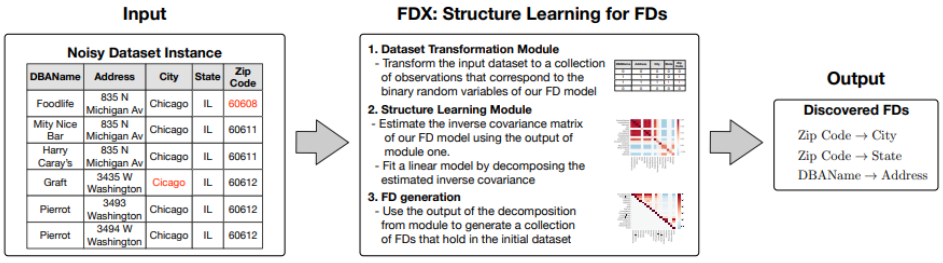


Figura 3.2: Etapas de execução do FDX. Extraído de Zhang et al. [22]

1. **Transformação do conjunto de dados:** Dado o conjunto de dados de entrada, é gerado um novo conjunto de dados D_t, contendo os mesmos atributos de D, porém seus valores são todos binários, ou seja, seu domínio se limita a $\{0, 1\}$. Os valores correspondem a amostras da distribuição do evento:

$$\bigwedge_{A \in X} (t_i[A] = t_j[A]) = \text{True} \quad \text{e} \quad t_i[Y] = t_j[Y],$$

ou seja, dado um par de tuplas aleatório t_i e t_j, um conjunto de atributos X e um atributo Y, caso para cada atributo $A \in X$, $t_i[A] = t_j[A]$ e $t_i[Y] = t_j[Y]$, a amostra corresponde ao valor 1, caso contrário, 0. Essa etapa trata-se do mapeamento do conjunto de dados de entrada para uma interpretação probabilística de dependências funcionais.

2. **Aprendizado de estrutura:** A saída da primeira etapa é a entrada desta segunda, em que a estrutura do modelo obtido na etapa anterior (amostras binárias do evento descrito acima) é aprendida fazendo uma estimativa da matriz de covariância inversa Θ, que pode ser descrita como: $\Theta = \Sigma^{-1} = (I - B)\Omega^{-1}(I - B)^T$, onde I é a matriz identidade, B é a matriz de autorregressão do modelo e Ω é a matriz de covariância.

   Dado um conjunto de N observações (as linhas de D_t) e seja S a matriz de covariância empírica dessas observações, a matriz de covariância inversa esparsa Θ é obtida resolvendo um problema de otimização:

   $$\min_{\Theta \succ 0} f(\Theta) := -\log\det(\Theta) + \mathrm{tr}(S\Theta) + \lambda|\Theta|_1$$

   onde tr denota o traço da matriz e $|\Theta|_1$ é a norma L_1 de Θ, que promove a esparsidade na matriz inversa. A constante λ controla o nível de esparsidade. Para resolver este problema, o FDX utiliza o algoritmo Graphical Lasso [6], conhecido por sua escalabilidade para conjuntos de dados com um grande número de variáveis (atributos) [22].

3. **Enumeração das dependências funcionais:** A matriz encontrada na etapa anterior Θ é fatorada é então fatorada como $\Theta = UDU^T$, onde U é uma matriz triangular superior, obtendo $\hat{B}$. Finalmente, a matriz de autorregressão estimada $\hat{B}$ é utilizada para gerar as DFs. Para cada atributo A_j (correspondente à coluna j de $\hat{B}$), o FDX examina os elementos não nulos na coluna j, excluindo o elemento diagonal. Se o elemento $\hat{B}i, j$ (onde $i < j$) for não nulo, isso indica uma dependência linear de A_j em relação a A_i (considerando a ordenação global). O conjunto de atributos X que corresponde às linhas i com valores não nulos em $\hat{B}\cdot, j$ forma o lado esquerdo de uma DF que determina o atributo A_j (o lado direito). Assim, uma DF descoberta tem a forma $X \rightarrow A_j$.

4 CONTRIBUIÇÃO

Nos capítulos anteriores, foi visto que a descoberta de dependências funcionais possui diversos desafios e diferentes abordagens. A escolha de abordagem depende muito do cenário e não se trata de uma escolha fácil. Em casos em que se sabe que o conjunto de dados analisados é livre de erros, dados nulos e anomalias, pode fazer sentido buscar um algoritmo de descoberta de DFs exatas; no entanto, a partir do momento em que os dados possuem anomalias, mesmo que poucas, essa abordagem exata já passa a ser inviável, pois é provável que existam boas DFs de normalização que não são encontradas por algoritmos exatos. Sendo assim, na maioria dos casos, é mais interessante buscar pelas abordagens de DFAs, que apresentam robustez a conjuntos de dados reais.

Um segundo desafio seria selecionar em um grande conjunto de dependências aproximadas, aquelas que realmente serão usadas para guiar uma normalização de esquema relacional. Uma solução conhecida na literatura é a utilização de métricas para atribuir pontuações às DFAs encontradas, o que permite uma ordenação no formato de ranqueamento destes resultados. Parciak et al. [16] mostra diversas métricas que podem ser utilizadas para isso, além de comparar métricas entre elas. Uma métrica de DFA formalmente quantifica o grau em que uma DF se mantém aproximadamente em uma dada relação, atribuindo uma pontuação no intervalo $[0, 1]$, onde valores mais altos indicam um maior grau de satisfação da DF [16]. Essa pontuação permite ranquear o espaço de busca de todas as DFs possíveis, com DFs de maior pontuação classificadas antes das de menor pontuação.

Parciak et al. [16] realiza uma análise de sensibilidade em relação a propriedades estruturais das DFs de entrada usando dados sintéticos, controlando parâmetros como nível de erro, unicidade do lado esquerdo (LHS-uniqueness) e assimetria do lado direito (RHS-skew). Além disso, o estudo quantifica a eficácia das métricas de DFA para ranquear DFAs em dados do mundo real, utilizando um novo benchmark chamado RWD (*Real World Data*), obtido pela criação manual de DFs de projeto para benchmarks existentes. Dentre as métricas analisadas, é visto em [16] que as que tem melhor desempenho em dados do mundo real são as métricas $\mu+$, $RFI'+$ e $g3'$, pois são as métricas que selecionam corretamente as DFs esperadas possuindo menos sensibilidade aos parâmetros apresentados.

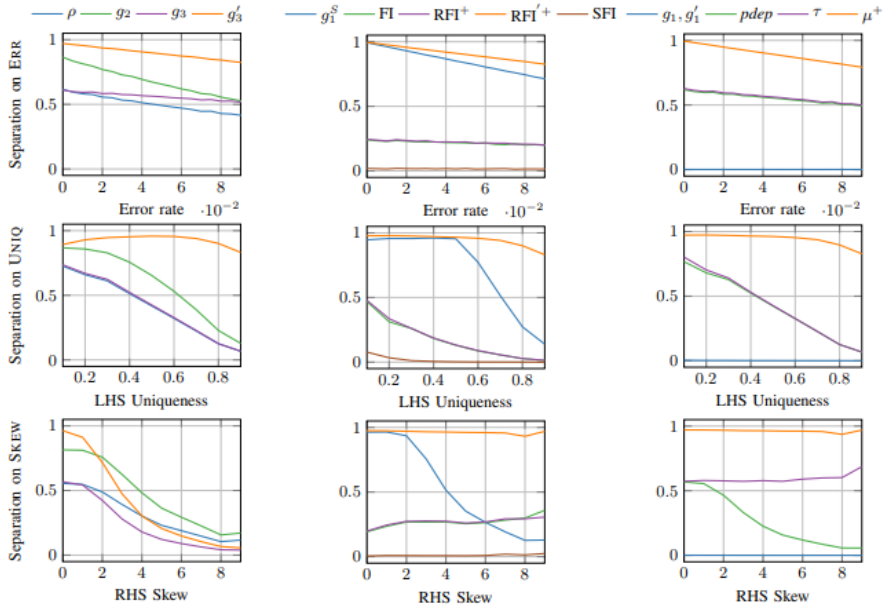


Figura 4.1: Extraído de Parciak et al. [16]. Comparação das métricas com relação aos parâmetros de erro, LHS-uniqueness e RHS-skew. Dentre as 3 melhores, vemos que $g3'$ possui uma sensibilidade considerável ao parâmetro RHS-skew Parciak et al. [16]

.

4.1 COMPARANDO ABORDAGENS DE DESCOBERTA

A contribuição deste trabalho é um estudo comparativo entre as 3 abordagens de descoberta de DFs por meio das métricas apresentadas. Desta forma, o leitor pode selecionar, de acordo com seu cenário específico e de maneira mais informada, qual abordagem quer adotar para realizar a descoberta de DFs, além de como as métricas avaliam as DFs.

Para realizar as comparações, foi desenvolvido um protocolo de experimentos, composto por 3 principais etapas: (1) Execução dos algoritmos de descoberta, (2) Cálculo das métricas para as DFs encontradas e (3) Agregação dos dados para visualização e comparação.

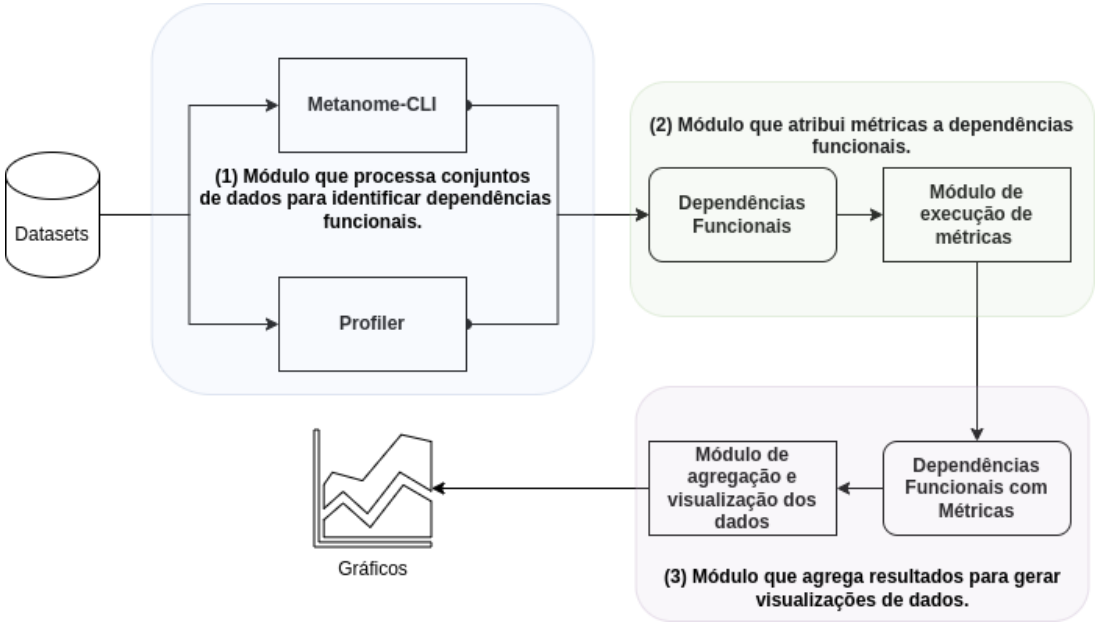


Figura 4.2: Diagrama do fluxo de experimentos

4.2 EXECUÇÃO DOS ALGORITMOS DE DESCOBERTA

Nesta etapa, os algoritmos de descoberta de dependências funcionais são executados sobre os *datasets* preparados, gerando como saída conjuntos de dependências funcionais, como visto na etapa (1) da figura 4.2. A execução foi automatizada através de *scripts* para garantir a consistência e a reprodutibilidade dos experimentos.

Para a execução do *HyFD* e do *Pyro*, foi utilizado o Metanome CLI[1], uma interface de linha de comando derivada da plataforma Metanome [13]. O Metanome é amplamente reconhecido na área de perfilamento de dados, sendo compatível com diversos algoritmos da literatura para descoberta de metadados. O script de execução automatiza o processamento de todos os *datasets*, configurando parâmetros específicos para cada algoritmo e organizando os resultados em diretórios estruturados. O Metanome CLI permitiu, portanto, a execução do *Pyro* com alguns limites de erro diferentes, ou seja, a taxa de violações que um candidato a DF pode ter sem ser descartado pelo algoritmo. Isso é interessante pois, desta forma, pode-se ajustar este limite dependendo do *dataset* de entrada, tornando-o robusto para dados com diferentes taxas de anomalias.

Já o *FDX* foi executado através de sua implementação em Python *(Profiler)*, que utiliza técnicas de estatística e aprendizado de máquina para inferir dependências funcionais aproximadas. A execução do *FDX* envolve algumas etapas:

1. Carregamento e pré-processamento dos dados: O *FDX* realiza uma análise inicial dos tipos de dados, categorizando-os como numéricos, categóricos ou textuais, e aplicando normalizações quando necessário. o algoritmo permite, no entanto, que esses tipos sejam alterados manualmente, caso os tipos inferidos não representem de maneira adequada os atributos da entrada.

2. Representação vetorial para dados textuais: Para atributos textuais, o algoritmo utiliza modelos de linguagem para representar os valores em espaços vetoriais, permitindo a análise de similaridade semântica.

[1] https://github.com/HPI-Information-Systems/Metanome/tree/metanome_cli/metanome-cli

3. Aprendizado de estrutura: O algoritmo utiliza técnicas de regressão para identificar relações entre atributos, aplicando parâmetros de esparsidade para obter estruturas parcimoniosas em conjuntos de dados maiores. É visto em [22] que o parâmetro de esparsidade diferente de 0 apenas melhora os resultados em muito grandes, com muitas tuplas ou muitos atributos, caso contrário, [22] recomenda manter este parâmetro fixado em zero.

4. Extração de dependências: As dependências são extraídas com base em métricas de erro de ajuste, permitindo identificar relações aproximadas entre os atributos.

5. Vale citar que o *FDX* permite execução paralela através do parâmetro *workers*.

Os resultados de todos os algoritmos foram armazenados em um formato padronizado em um diretório, facilitando a análise comparativa posterior.

4.3 CÁLCULO DAS MÉTRICAS

Após a descoberta das dependências funcionais pelos algoritmos, foi implementado um processo automatizado para calcular as métricas para cada DF encontrada, representado na etapa (2) da figura 4.2. O cálculo das métricas é executado através de um script Python que processa os resultados dos algoritmos de descoberta de DFs (Metanome e FDX) e aplica as métricas implementadas.

4.3.1 Arquitetura do sistema de métricas

O sistema de métricas é composto por três componentes principais:

- Driver de execução: Um *script shell* que itera sobre os arquivos de resultados dos algoritmos de descoberta de DFs, extraindo informações sobre o algoritmo e o conjunto de dados, e invocando o *script* Python principal.

- *Parser* de resultados: Módulo Python responsável por carregar e interpretar os resultados dos diferentes algoritmos (Metanome e FDX), convertendo-os para um formato unificado de pares (LHS, RHS).

- Implementação das métricas: Módulo Python que implementa as métricas baseadas na literatura [16], adaptadas para trabalhar com listas de colunas.

- Módulo Python principal: Responsável por invocar os *parsers* de DFs e, posteriormente, o módulo de cálculo de métricas. Além disso, monta os arquivos de saída contendo as DFs com suas respectivas métricas, além de dados de tempo de execução.

Esta estrutura de saída facilita a análise posterior e a comparação entre diferentes algoritmos e conjuntos de dados, permitindo identificar as DFs mais relevantes de acordo com as métricas implementadas.

4.4 AGREGAÇÃO DOS DADOS PARA VISUALIZAÇÃO E COMPARAÇÃO

Os resultados finais dos experimentos tratam-se de arquivos CSV que contêm as DFs e suas pontuações para cada métrica. Para realizar análises mais profundas e comparativas sobre esses resultados, utilizamos bibliotecas da linguagem Python para agregar e visualizar os dados, como *Pandas* [2], *matplotlib* [3], *seaborn* [4] e *pandastable* [5].

Como foram realizados experimentos em múltiplos *datasets*, todos os resultados foram agregados em uma única estrutura de dados (*Dataframe* [6]), permitindo assim filtrar, agrupar, ordenar e plotar gráficos contendo todos os resultados. Isso foi feito por meio de um módulo Python que gera esse agregado global dos dados. Depois disso, foram criados alguns *Jupyter notebooks* [7], cada um com um propósito, mas todos tendo a mesma fonte de dados.

Esse formato permite tanto visualizações específicas de um determinado *dataset* como outras globais, comparando diferentes abordagens e como se comportam em diferentes *datasets*. As bibliotecas citadas acima permitem diversas análises gráficas. Dentre as mais utilizadas neste trabalho estão as distribuições por densidade, ranqueamentos (ordenação por uma métrica), gráficos lineares e outros.

[2] https://pandas.pydata.org/
[3] https://matplotlib.org/
[4] https://seaborn.pydata.org/
[5] https://pandastable.readthedocs.io/
[6] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html
[7] https://jupyter.org/

5 EXPERIMENTOS

Nesse capítulo, serão discutidos os experimentos realizados seguindo o protocolo descrito no capítulo 4 e alguns resultados encontrados. Os experimentos realizados tiveram como objetivo central comparar diferentes abordagens para descoberta de DFs, avaliando seu desempenho através de métricas estabelecidas na literatura. Esta análise comparativa permite identificar as vantagens e limitações de cada algoritmo em diferentes contextos e características de conjuntos de dados.

Inicialmente, são apresentados os conjuntos de dados utilizados nos experimentos, destacando suas principais características e eventuais adaptações realizadas para viabilizar as análises. Em seguida, são explorados comparativos gerais entre os algoritmos estudados, oferecendo uma visão abrangente de seu desempenho através de diferentes métricas e visualizações. Por fim, discutimos particularidades de cada abordagem, analisando aspectos específicos que influenciam seus resultados e comportamentos.

Os resultados aqui apresentados contribuem para uma compreensão mais aprofundada das diferentes técnicas de descoberta de DFs, permitindo análises para pesquisadores e profissionais que buscam selecionar o método mais adequado para seus contextos específicos de aplicação.

Além disso, as perguntas levantadas na seção 1.1 serão abordadas nas seguintes seções:

1. **Qual a eficácia das métricas desenvolvidas para DFAs quando aplicadas a DFs?** seção 5.2.2;
2. **Como o tamanho e a unicidade do determinante de uma DF ou DFA afetam a pontuação dada pelas métricas?** seção 5.2.4;
3. **As métricas atribuem pontuações altas a DFs e DFAs de projeto, ou seja, àquelas que capturam o verdadeiro valor semântico dos conjuntos de dados?** seção 5.2.5;
4. **Em quais contextos cada abordagem é mais eficaz?** seção **??**.

5.1 CONJUNTOS DE DADOS

Foram selecionados sete conjuntos de dados. Abaixo estão disponíveis as fontes dos dados, algumas informações sobre eles, adaptações para os experimentos e as datas de acesso, para facilitar futuras reproduções de experimentos.

Três dos *datasets* selecionados são amplamente conhecidos na literatura da área de banco de dados. São eles:

- NCVoter: Registra eleitores no estado da Carolina do norte, nos Estados Unidos. Retirado de: https://hpi.de/naumann/projects/repeatability/data-profiling/fds.html; Acessado em: 12/12/24.
- Adult: Predição de quando o salário anual de um indivíduo excede $50000 no ano baseado em dados do censo. Retirado de: https://hpi.de/naumann/projects/repeatability/data-profiling/fds.html; Acessado em: 12/12/24 - *Obs*: Para esse *dataset* foram consideradas as primeiras 1800 linhas, por questões de viabilidade dos experimentos.

- Iris: *dataset* principalmente utilizado para realização de treinamentos de modelos de aprendizado de máquina. É interessante como exemplo de *dataset* sem dados nulos e com perfil não relacional. Retirado de: https://hpi.de/naumann/projects/repeatability/data-profiling/fds.html; Acessado em: 12/12/24.

Além desses, selecionamos três conjuntos de dados do repositório público de dados da cidade de Nova Iorque, nos Estados Unidos. Estes são interessantes pois são dados do mundo real, com dados nulos e faltantes, além de serem atualizados periodicamente. São eles:

- Wi-Fi Hotspot Location: Análise dos registros *OpenData* do *NYC Wi-Fi Hotspot Locations* (Locais de pontos de acesso Wi-Fi em Nova Iorque) com foco no uso de Wi-Fi em espaços abertos (em parques públicos e na Governors Island).. Retirado de: https://data.cityofnewyork.us/City-Government/Wi-Fi-in-Public-Space-Open-Space-/npnk-wrj8/about_data; Acessado em: 12/12/24.

- Leading Causes of Death: Principais causas de morte por sexo e etnia na cidade de Nova Iorque desde 2007. A causa da morte é obtida a partir da certidão de óbito da cidade de Nova Iorque, emitida para cada morte que ocorre na cidade.. Retirado de: https://data.cityofnewyork.us/Health/New-York-City-Leading-Causes-of-Death/jb7j-dtam/about_data; Acessado em: 12/12/24.

- Hate Crimes: Conjunto de dados contendo incidentes confirmados de crimes de ódio em Nova Iorque. Retirado de: https://data.cityofnewyork.us/Public-Safety/NYPD-Hate-Crimes/bqiq-cu78/about_data; Acessado em: 14/04/25.

Também foi utilizado um *dataset* retirado e adaptado do portal de dados abertos do Ministério da Educação brasileiro:

- Lista de espera sisu 2020: Para viabilizar os experimentos, foram filtradas apenas as linhas contendo dados de candidatos para cursos no Centro Politécnico da Universidade Federal do Paraná. Além disso, foram removidas algumas colunas usadas para cálculo de notas e outros metadados. Retirado de https://dadosabertos.mec.gov.br/sisu/item/163-2020-relatorio-sisu-lista-de-espera-1-2020. Acessado em: 02/06/2025.

O número de tuplas e de atributos de cada conjunto de dados são informações importantes, já que estão diretamente relacionadas com o tempo de execução dos algoritmos [10], do cálculo das métricas e também aos resultados encontrados. A tabela abaixo mostra a relação dos tamanhos de cada *dataset* nessas duas dimensões:

Tabela 5.1: Dimensões dos conjuntos de dados

dataset	Linhas	Colunas
ncvoter	1000	19
hate_crimes	3255	14
wifi_hotspot_location	3319	29
adult	1799	15
leading_causes_of_death	1094	7
sisu_ufpr_curitiba_politecnico	2339	29
iris	150	5

5.2 COMPARATIVOS GERAIS

Os algoritmos de descoberta de dependências funcionais apresentam comportamentos distintos quando aplicados a diferentes conjuntos de dados. Nesta seção, foi realizada uma análise comparativa abrangente entre as abordagens estudadas, evidenciando suas características em diversos cenários. O objetivo é fornecer uma visão holística que permita identificar padrões de comportamento que possam orientar a escolha da técnica mais adequada para contextos específicos. Foi explorado desde estatísticas básicas sobre os conjuntos de dados e o volume de DFs descobertas, até análises mais sofisticadas sobre a distribuição das métricas de qualidade e a estrutura das dependências identificadas. Esta abordagem multifacetada visa proporcionar um entendimento mais profundo sobre o comportamento dos algoritmos estudados. Além disso as perguntas de pesquisa serão respondidas respectivamente:

5.2.1 Estatísticas gerais

O número de DFs encontradas por cada algoritmo em cada conjunto de dados é interessante para entender o comportamento básico dos algoritmos. A figura 5.1 mostra essa estatística simples, mas que já rende algumas discussões valiosas.

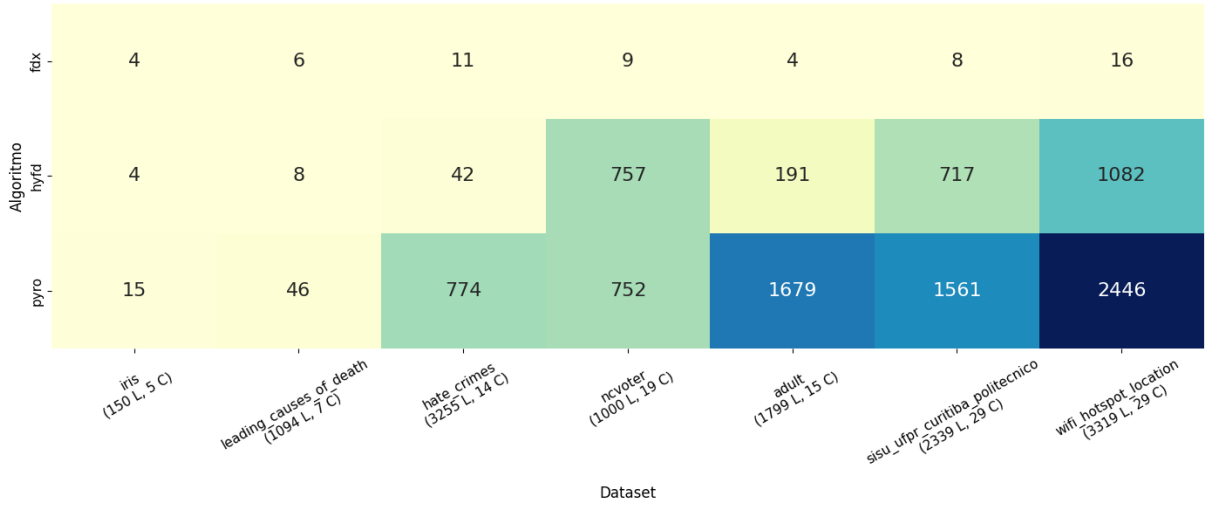


Figura 5.1: Número de DFs encontradas em cada *dataset* com cada algoritmo

Existe uma diferença significativa no número de dependências funcionais descobertas por cada algoritmo. O FDX apresenta consistentemente a menor quantidade de DFs identificadas em todos os conjuntos de dados analisados. Essa característica não representa uma deficiência do algoritmo, mas reflete sua proposta fundamental, que não visa enumerar exaustivamente todas as DFs mínimas e não triviais, mas sim identificar um conjunto reduzido de DFs parcimoniosas [22] com relevância estatística. Um diferencial chave é que o número de DFs encontradas pelo FDX independe do tamanho do conjunto de dados, ao contrário dos demais.

O HyFD, por sua vez, ocupa uma posição intermediária no volume de DFs descobertas. Por ser um algoritmo exato, ele realiza podas mais rigorosas durante o processo de busca, eliminando candidatos que não satisfazem completamente a definição formal de dependência funcional. Essa abordagem mais restritiva, naturalmente, resulta em um conjunto menor de DFs quando comparado a métodos aproximados.

Já o Pyro destaca-se pelo maior volume de DFs identificadas em quase todos os conjuntos de dados. Ao relaxar o critério de validação das DFs, o Pyro consegue capturar relações que

seriam descartadas pelos algoritmos exatos, resultando em um conjunto significativamente maior de dependências.

5.2.2 Densidades

Partindo para uma análise levando em consideração o critério de avaliação utilizado neste trabalho, ou seja, as métricas de DFAs aproximadas para comparar DFs encontradas pelos três algoritmos propostos, iniciamos com a análise da distribuição das pontuações dadas pelas métricas. A Figura 5.2 apresenta gráficos de distribuição das métricas RFI'^{+}, μ^{+} e g'_3 para as DFs encontradas por cada algoritmo (FDX, Pyro e HyFD) em diferentes conjuntos de dados. Cada boxplot no gráfico representa a distribuição das pontuações de uma métrica para as DFs descobertas por um algoritmo específico em um dado conjunto de dados. O boxplot mostra a mediana (linha central), o primeiro e terceiro quartis (limites da caixa), além dos valores mínimos e máximos (extremidades das linhas verticais), permitindo identificar também possíveis *outliers*. Esses gráficos de caixa (boxplots) são úteis para visualizar estatísticas resumidas da distribuição das pontuações das métricas, incluindo a tendência central, a dispersão dos valores e a presença de valores atípicos.

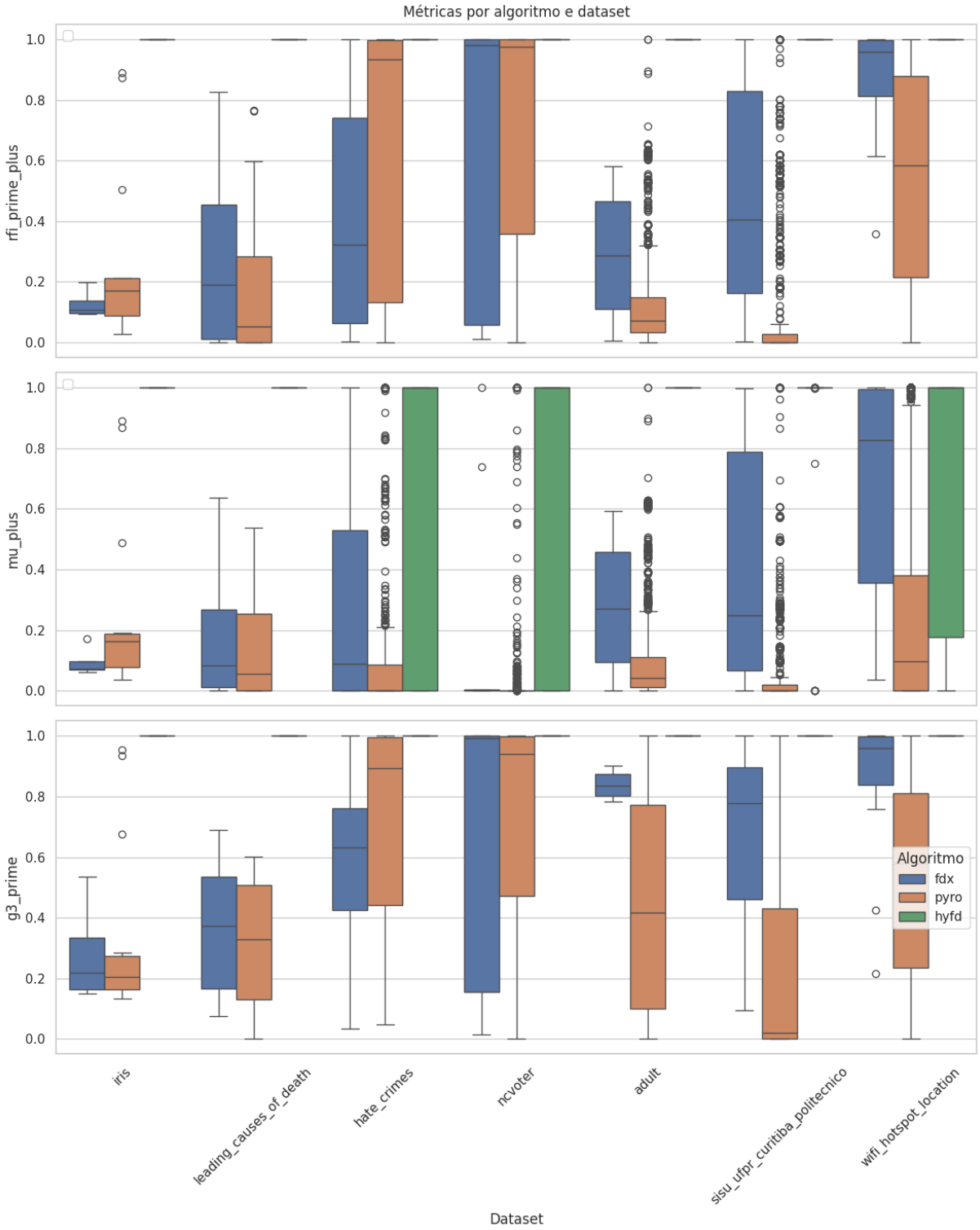


Figura 5.2: Densidade das métricas por *dataset*

Nessa análise, vale levar em consideração o número de DFs encontradas em cada algoritmo, que pode ser visto na Figura 5.1. Isso pois, por mais que algumas distribuições na Figura 5.2 sejam parecidas, o número de dependências funcionais presentes na distribuição é muito distinto, principalmente quando comparando resultados entre FDX e Pyro. O HyFD, por se tratar de um algoritmo de abordagem exata, ou seja, apenas retorna DFs verdadeiras em todos os pares de tuplas da entrada, possui pontuação 1 (valor máximo para métricas de

DFAs) para as métricas RFI'^{+} e g'_3 em todas as DFs. Já no caso da μ^{+}, apesar de as pontuações ficarem quase todas iguais ou muito próximas de 1, nos *datasets* 'hate_crimes', 'ncvoter' e 'wifi_hotspot_location', existem alguns casos diferentes de 1. Este ponto é importante pois fica claro que utilizar métricas de DFs aproximadas para avaliar DFs exatas não é o ideal, ao menos não as métricas apresentadas neste trabalho, já que trará pouca ou nenhuma informação relevante que auxilie na escolha de DFs.

Outro ponto importante é que mais nem sempre é melhor. Um dos principais motivos que se faz esse pós-processamento de ranquear as dependências funcionais encontradas é justamente facilitar a escolha de DFs de projeto, ou seja, aquelas que guiarão uma normalização de esquema ou um melhor projeto de relação. Sendo assim, o fato de um algoritmo ter poucas DFs com notas altas dadas pelas métricas pode, na verdade, ser positivo por esse ponto de vista.

Tendo em vista essa grande diferença de volumes de DFs encontradas por cada algoritmo, é difícil estabelecer padrões entre as distribuições, mas é possível notar que, na maioria dos *datasets*, foram encontradas ao menos algumas DFs com notas altas em todas as métricas para o Pyro e FDX. Um caso que se destaca é o *dataset* 'iris', que possui um número muito baixo de DFs (15 para o Pyro, 4 para o FDX) e as notas das métricas são baixas para a grande maioria delas.

5.2.3 Correlação entre métricas

Analisando a figura 5.2, é possível notar que, para um mesmo *dataset* e algoritmo, algumas distribuições entre métricas diferentes se assemelham. Por exemplo, no 'ncvoter', as distribuições das métricas g'_3 e RFI'^{+} são muito semelhantes para todos os algoritmos. Outro exemplo é o *dataset* 'sisu_ufpr_curitiba_politecnico', em que as distribuições da μ^{+} e RFI'^{+} têm maiores similaridades. No entanto, essa análise entre métricas pode ser melhor visualizada em um gráfico de correlação como o da Figura 5.3 abaixo:

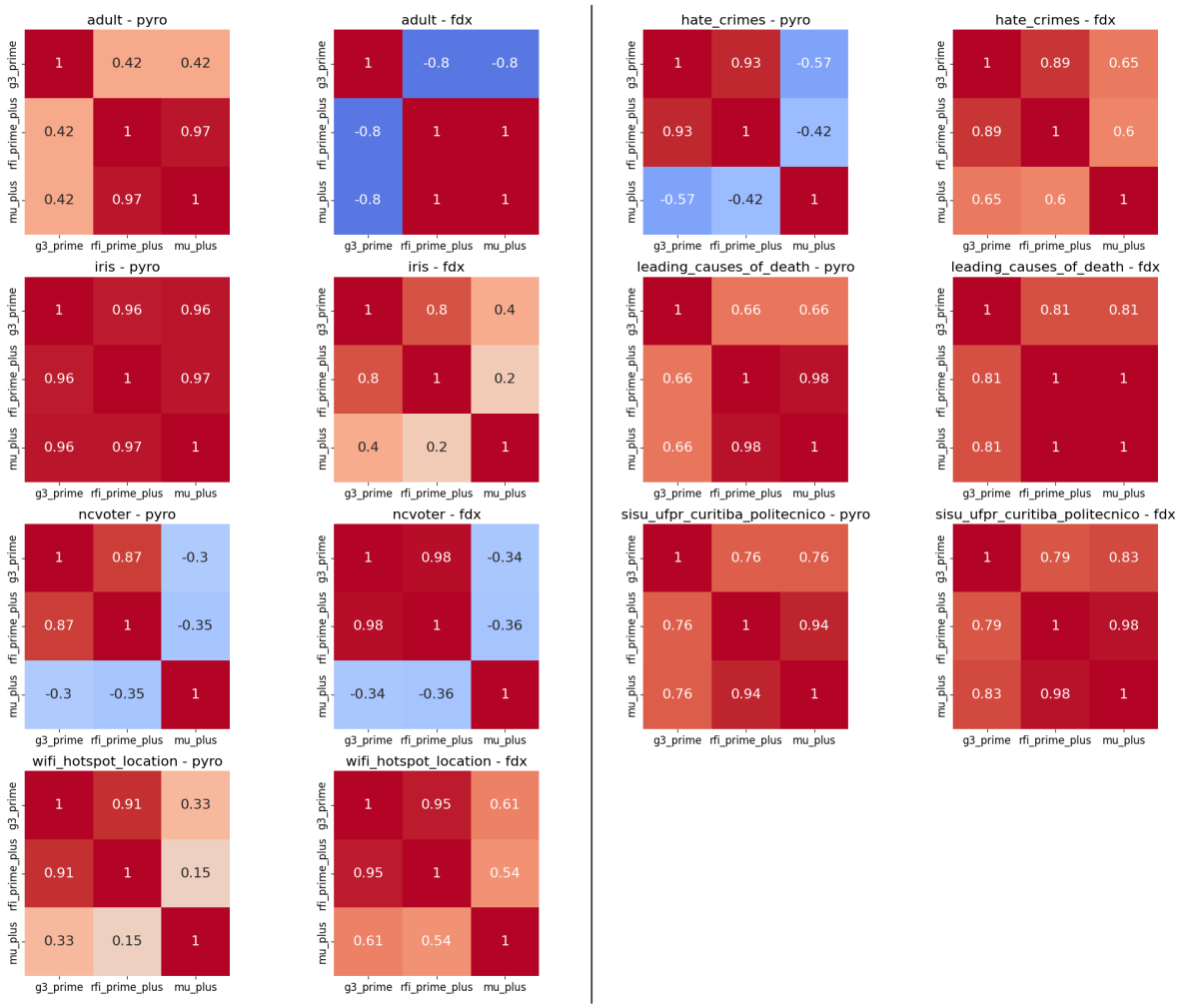


Figura 5.3: Mapas de calor de correlação (Spearman) entre as métricas por *dataset* e algoritmo

A análise de correlação entre as métricas de avaliação de dependências funcionais pode ser feita utilizando métodos distintos, como de Pearson, Spearman ou de Kendall, dependendo da natureza dos dados analisados. O resultado da correlação trata-se de um número no intervalo entre -1 e 1, onde valores altos significam uma alta correlação e baixos o contrário. Optamos pela correlação de Spearman devido às características das distribuições observadas nos *datasets*, que apresentam forte assimetria, agrupamentos em valores extremos (0 ou 1) e relações não estritamente lineares. Segundo [20], o método de Spearman, baseado em ranqueamentos ao invés de valores brutos, oferece maior robustez a dados atípicos e captura melhor relações monotônicas não lineares entre as métricas avaliadas. Isso significa que, havendo uma forte correlação entre duas métricas, elas resultam em ranqueamentos similares para aquele conjunto de dependências funcionais, mesmo que muitas vezes as pontuações absolutas não sejam parecidas.

Os mapas de calor da Figura 5.3 demonstram que, na maioria dos casos, existe uma forte correlação entre pares específicos de métricas, mas raramente entre as três simultaneamente. Observa-se que as correlações mais intensas ocorrem entre RFI'^{+} e uma das outras métricas, enquanto a correlação entre g'_3 e μ^+ tende a ser mais fraca. O *dataset* 'sisu_ufpr_curitiba_politecnico' constitui uma exceção notável, apresentando correlações acima de 0.75 entre todas as métricas, embora ainda preserve o padrão de pares com correlações mais fortes; o *dataset* 'iris' tem um caso ainda mais extremo, em que a correlação entre todas as métricas é acima de 0.95 entre todas as métricas para o algoritmo Pyro. Outro ponto interessante é que os *datasets* 'ncvoter', 'adult' e

'hate_crimes' mostram correlações negativas entre pares de métricas, sugerindo comportamentos distintos das métricas conforme a natureza dos dados analisados.

5.2.4 Análise das Características do Determinante (LHS)

O determinante de uma dependência funcional, conhecido como *Left-Hand Side* (LHS), é um componente central para a avaliação de sua qualidade e relevância semântica. Conforme discutido na Seção 2.7.2, um dos principais desafios na descoberta de DFs é o sobreajuste, onde DFs sintaticamente válidas para uma instância de dados específica carecem de significado geral. Frequentemente, tais DFs são caracterizadas por um LHS excessivamente grande.

Nesta seção, analisamos duas propriedades fundamentais do LHS: sua cardinalidade (tamanho) e sua unicidade (uniqueness). O objetivo é entender como essas propriedades variam entre os algoritmos de descoberta e como as métricas de qualidade respondem a elas.

5.2.4.1 Cardinalidade do LHS

A cardinalidade do LHS é um indicador primário de complexidade. À medida que o número de atributos no LHS aumenta, cresce a probabilidade de que o conjunto de atributos se torne uma superchave ou chave candidata, determinando unicamente cada tupla. Embora isso resulte em uma DF tecnicamente válida, ela pode não representar uma regra de negócio útil, sendo apenas um artefato dos dados.

A Tabela 5.2 apresenta estatísticas descritivas sobre o LHS das DFs descobertas por cada algoritmo nos *datasets* de nossos experimentos.

Tabela 5.2: Estatísticas básicas referentes ao LHS

Algoritmo	Mín LHS	Máx LHS	LHS médio	Unicidade média do LHS
fdx	1	4	1.534	0.189
hyfd	1	10	3.262	0.887
pyro	1	18	3.465	0.352

Os dados da Tabela 5.2 revelam perfis distintos. O FDX consistentemente encontra DFs com LHS de baixa cardinalidade (máximo de 4), enquanto Pyro e HyFD exploram combinações maiores, o que é esperado por terem uma natureza combinatória e incluem DFs com LHS maiores em seus resultados. É importante destacar que os algoritmos Pyro e HyFD garantem minimalidade e não trivialidade das DFs descobertas, enquanto o FDX não oferece tais garantias.

Para investigar como as métricas de qualidade reagem a essa característica, a Figura 5.4 ilustra o comportamento das métricas conforme a cardinalidade do LHS aumenta. A análise exclui o HyFD, pois, ao descobrir apenas DFs exatas, suas métricas se aproximam de 1 em quase todas as dependências funcionais, inviabilizando uma comparação de sensibilidade.

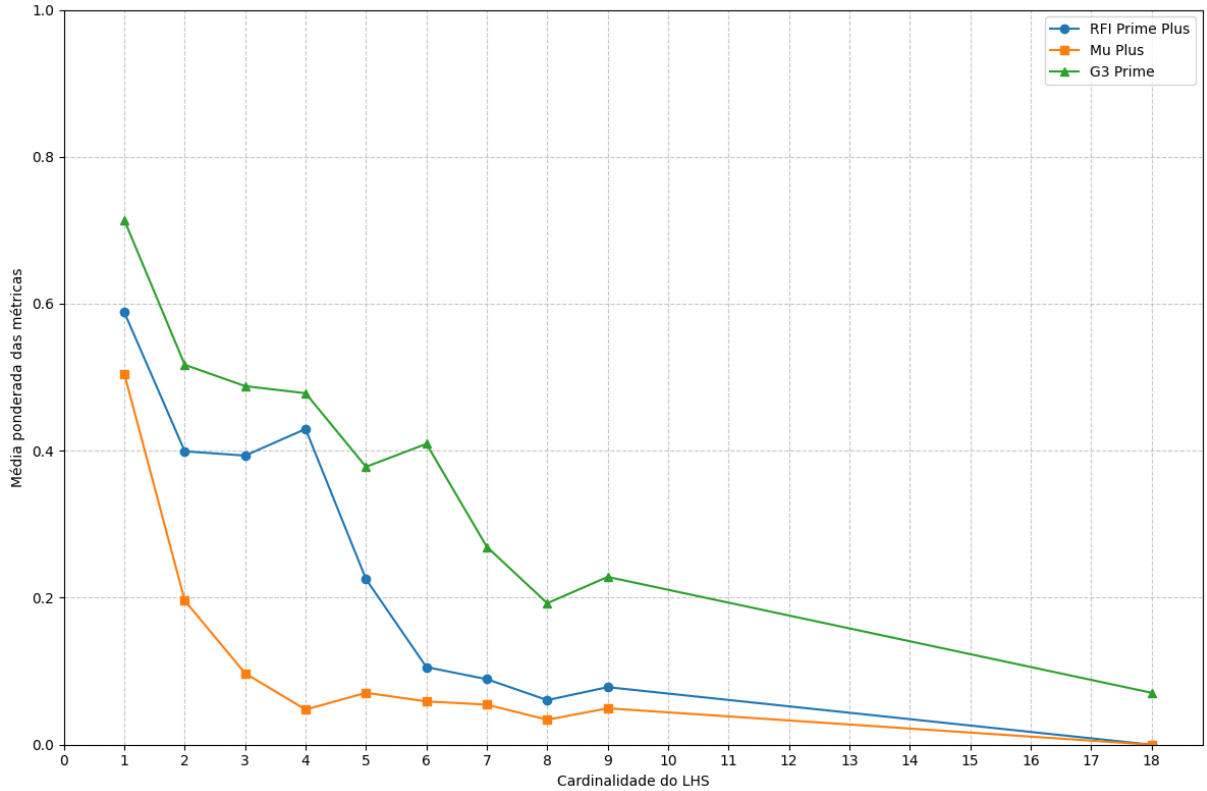


Figura 5.4: Comportamento de métricas conforme o LHS das DFs aumenta

É importante levar em consideração que o gráfico da Figura 5.4 utiliza a média ponderada de cada métrica no LHS do eixo X para **todos** os conjuntos de dados analisados, portanto as médias acabam sendo muito influenciadas por *datasets* com grande volume de DFs. Ainda assim, é interessante observar a tendência de queda na avaliação média de todas as métricas à medida que o LHS cresce, especialmente para |LHS| > 5. Este resultado é significativo, pois sugere que as métricas são eficazes em penalizar DFs com determinantes grandes, que são mais propensas ao sobreajuste.

5.2.4.2 Análise da Unicidade do LHS

Embora a cardinalidade seja informativa, a unicidade do LHS (LHS uniqueness) oferece uma visão mais refinada. Segundo Parciak et al. [16], para uma relação R e uma dependência funcional $X \rightarrow Y$, temos $LHS\_uniqueness = |dom(X)|/|dom(R)|$. Valores próximos a 1 indicam que os valores do conjunto X raramente se repetem, sugerindo uma aproximação a uma chave candidata, enquanto valores próximos a zero indicam alta repetição.

A Tabela 5.2 já apontava para diferenças drásticas: a unicidade média do hyfd (0.89) é extremamente alta, confirmando que o algoritmo exato tende a identificar DFs que são quase chaves. Em contrapartida, a baixa unicidade do fdx (0.19) sugere que ele captura dependências onde o determinante é mais genérico e menos óbvio. A Figura 5.5 detalha como as métricas se comportam em relação à unicidade do LHS.

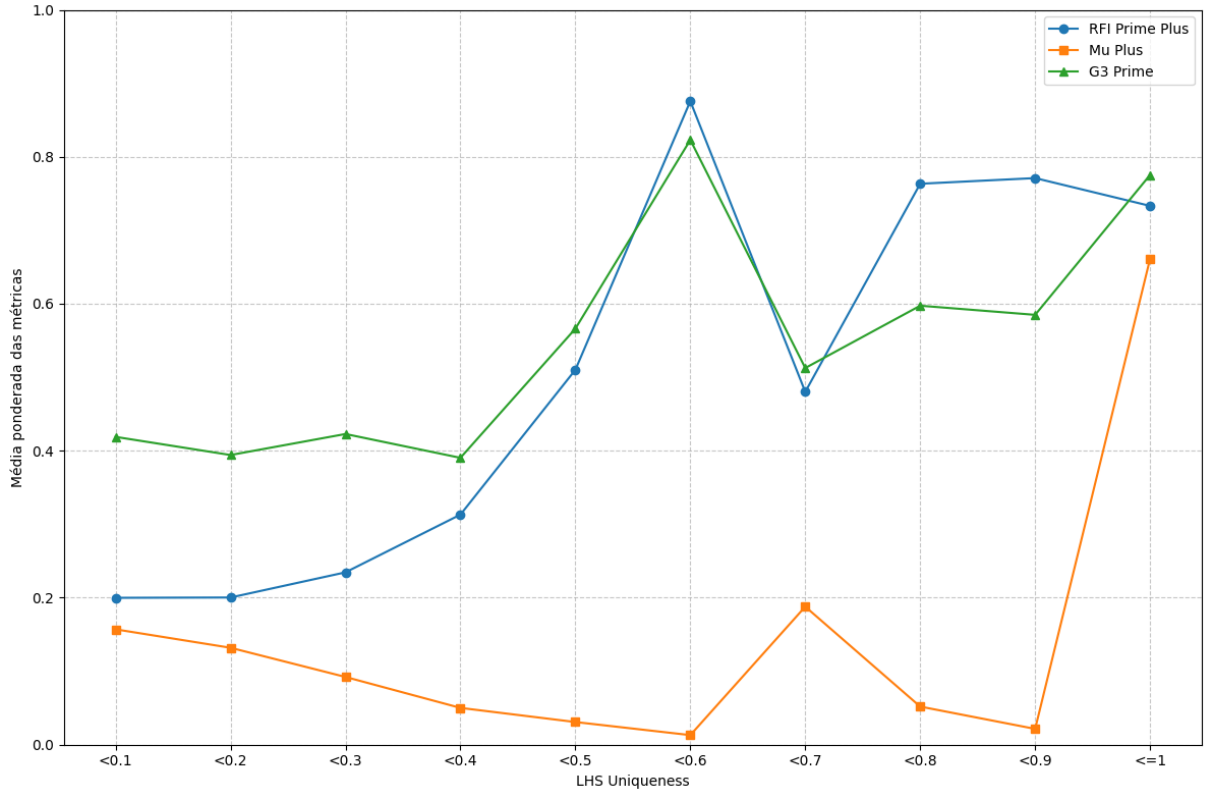


Figura 5.5: Comportamento de métricas conforme o a unicidade do LHS das DFs aumenta

A análise da Figura 5.5 revela padrões de avaliação distintos:

- A métrica μ^+ demonstra alta sensibilidade, atribuindo notas elevadas apenas a DFs com unicidade superior a 0.9. Isso a posiciona como um detector de DFs cujos determinantes são quase-chaves.
- Já g'_3 e RFI'^+ apresentam um comportamento mais distribuído, avaliando com pontuações melhores DFs em faixas de unicidade intermediárias (a partir de 0.5). Seus comportamentos são notavelmente similares, sugerindo que capturam aspectos parecidos das DF em relação a essa propriedade.

Em síntese, a análise da unicidade do LHS se mostra mais discriminante que a da cardinalidade. Ela permite uma avaliação mais precisa da natureza de uma DF, ajudando a distinguir entre dependências semanticamente relevantes e aquelas que são meros artefatos estatísticos da instância de dados.

5.2.5 Ranqueamentos

Até agora, vimos nas seções acima resultados estatísticos e numéricos sobre as DFs encontradas nos experimentos, o que é muito útil para entender comportamentos dos algoritmos e das métricas utilizadas. No entanto, um dos principais objetivos de utilizar esses algoritmos e métricas é selecionar algumas dependências funcionais para guiar o processo de normalização de esquemas ou para entender como os atributos se relacionam entre si. Nesta seção, analisamos qualitativamente os ranqueamentos gerados para avaliar sua utilidade prática.

Existem casos em que ranquear as DFs encontradas não se mostra útil. Para o HyFD, por exemplo, quase todas as DFs enumeradas recebem pontuação 1 por parte das métricas (conforme

a Figura 5.2), o que ocorre porque ele foi projetado para encontrar DFs exatas. Isso demonstra que as métricas deste trabalho, propostas para avaliar DFs **aproximadas**, não são eficazes para diferenciar a importância entre DFs exatas.

Mesmo com um algoritmo de DFs aproximadas como o Pyro, as métricas RFI'^{+} e g'_3 frequentemente atribuem notas muito altas a um grande volume de DFs, poluindo o topo do ranqueamento e dificultando a seleção. Em contrapartida, a métrica μ^{+} mostrou-se mais criteriosa. Como visto na Figura 5.5, ela reserva suas maiores notas para DFs cujo lado esquerdo (LHS) se aproxima de uma superchave. Assim, a estratégia de ranqueamento mais eficaz para o grande volume de DFs do Pyro foi ordenar pela μ^{+} (decrescente). Para RFI'^{+} e g'_3, uma ordenação secundária pela cardinalidade do LHS (crescente) é recomendada para priorizar DFs mais simples.

Já o FDX, por padrão, retorna um conjunto muito menor de DFs, tornando a análise individual factível e o ranqueamento menos crucial. Contudo, a qualidade de seus resultados varia drasticamente com a natureza dos dados, como será discutido a seguir.

5.2.5.1 Tabelas com ranqueamentos

As tabelas abaixo mostram alguns ranqueamentos gerados nos experimentos, separados por algoritmo.

Tabela 5.3: Primeiras 30 DFS encontradas pelo algoritmo Pyro para o *dataset* 'hate_crimes' ranqueadas a partir da métrica μ^{+}.

DF	mu_plus	rfi_prime_plus	g3_prime
['Complaint Precinct Code']->Patrol Borough Name	1.000	1.000	1.000
['Patrol Borough Name']->County	1.000	1.000	1.000
['Full Complaint ID']->County	1.000	1.000	1.000
['Full Complaint ID']->Record Create Date	1.000	1.000	1.000
['Full Complaint ID']->Month Number	1.000	1.000	1.000
['Full Complaint ID']->PD Code Description	1.000	1.000	1.000
['Full Complaint ID']->Offense Description	1.000	1.000	1.000
['Full Complaint ID']->Patrol Borough Name	1.000	1.000	1.000
['Full Complaint ID']->Complaint Precinct Code	1.000	1.000	1.000
['PD Code Description']->Offense Description	1.000	1.000	1.000
['Full Complaint ID']->Complaint Year Number	1.000	1.000	1.000
['Full Complaint ID']->Law Code Category Description	1.000	1.000	1.000
['Full Complaint ID']->Offense Category	1.000	1.000	1.000
['PD Code Description']->Law Code Category Description	1.000	1.000	1.000
['Complaint Precinct Code']->County	1.000	1.000	1.000
['Record Create Date']->Complaint Year Number	0.998	0.999	0.999
['Full Complaint ID']->Bias Motive Description	0.995	0.995	0.996
['Bias Motive Description']->Offense Category	0.990	0.982	0.997
['Record Create Date']->Month Number	0.917	0.928	0.939
['Bias Motive Description', 'Offense Description']->Law Code Category Description	0.843	0.837	0.944
['Complaint Year Number', 'Offense Description']->Law Code Category Description	0.837	0.833	0.938
['Offense Description', 'Patrol Borough Name']->Law Code Category Description	0.832	0.828	0.936
['Offense Category', 'Offense Description']->Law Code Category Description	0.830	0.828	0.937
['County', 'Offense Description']->Law Code Category Description	0.830	0.827	0.937
['Month Number', 'Offense Description']->Law Code Category Description	0.827	0.824	0.935
['Bias Motive Description', 'County', 'Offense Description']->PD Code Description	0.700	0.763	0.822
['Complaint Precinct Code', 'Offense Description']->PD Code Description	0.698	0.752	0.809
['County', 'Offense Category', 'Offense Description']->PD Code Description	0.680	0.751	0.804
['Month Number', 'Offense Description']->PD Code Description	0.672	0.746	0.799
['Complaint Year Number', 'Offense Description']->PD Code Description	0.671	0.748	0.799

Tabela 5.4: Todas as DFS encontradas pelo algoritmo FDX para o *dataset* 'hate_crimes' ranqueadas a partir da métrica μ^+.

DF	mu_plus	rfi_prime_plus	g3_prime
['Complaint Precinct Code', 'Full Complaint ID']->Complaint Year Number	1.000	1.000	1.000
['Law Code Category Description', 'Offense Description']->PD Code Description	0.706	0.791	0.832
['County', 'Offense Category']->Bias Motive Description	0.569	0.661	0.783
['County']->Patrol Borough Name	0.488	0.688	0.639
['Law Code Category Description']->Offense Description	0.193	0.321	0.465
['Complaint Precinct Code', 'County']->Offense Category	0.088	0.095	0.544
['Complaint Precinct Code']->Full Complaint ID	0.005	0.032	0.034
['Law Code Category Description']->County	0.002	0.001	0.388
['Arrest Date']->Arrest Id	0.001	0.820	0.739
['Law Code Category Description']->Complaint Precinct Code	0.001	0.003	0.043
['County', 'Bias Motive Description']->Arrest Date	0.000	0.147	0.632

Tabela 5.5: Primeiras 30 DFS encontradas pelo algoritmo Pyro para o *dataset* 'ncvoter' ranqueadas a partir da métrica μ^+.

DF	mu_plus	rfi_prime_plus	g3_prime
['voter_id']->last_name	1.000	1.000	1.000
['voter_reg_num']->city	1.000	1.000	1.000
['street_address']->zip_code	1.000	1.000	1.000
['voter_reg_num']->voter_id	1.000	1.000	1.000
['street_address']->city	1.000	1.000	1.000
['voter_id']->register_date	1.000	1.000	1.000
['voter_reg_num']->ethnic	1.000	1.000	1.000
['voter_reg_num']->gender	1.000	1.000	1.000
['voter_reg_num']->last_name	1.000	1.000	1.000
['voter_reg_num']->zip_code	1.000	1.000	1.000
['voter_reg_num']->race	1.000	1.000	1.000
['voter_reg_num']->age	1.000	1.000	1.000
['voter_id']->age	1.000	1.000	1.000
['voter_id']->race	1.000	1.000	1.000
['voter_id']->gender	1.000	1.000	1.000
['voter_id']->zip_code	1.000	1.000	1.000
['voter_id']->ethnic	1.000	1.000	1.000
['voter_id']->voter_reg_num	1.000	1.000	1.000
['voter_reg_num']->first_name	1.000	1.000	1.000
['voter_id']->city	1.000	1.000	1.000
['voter_id']->first_name	1.000	1.000	1.000
['voter_reg_num']->register_date	1.000	1.000	1.000
['zip_code']->city	0.993	0.995	0.996
['first_name']->gender	0.859	0.851	0.950
['city', 'ethnic']->zip_code	0.795	0.866	0.830
['city', 'race']->zip_code	0.790	0.864	0.822
['city', 'gender']->zip_code	0.785	0.843	0.821
['city', 'download_month']->zip_code	0.775	0.836	0.806
['city', 'register_date']->zip_code	0.760	0.804	0.790
['street_address']->last_name	0.738	0.714	0.738

Tabela 5.6: Todas as DFS encontradas pelo algoritmo FDX para o *dataset* 'ncvoter' ranqueadas a partir da métrica μ^+.

DF	mu_plus	rfi_prime_plus	g3_prime
['street_address']->city	1.000	1.000	1.000
['street_address', 'city']->last_name	0.738	0.714	0.738
['race', 'download_month']->register_date	0.004	0.030	0.154
['gender']->first_name	0.004	0.059	0.042
['register_date']->street_address	0.003	0.009	0.015
['name_prefix']->name_suffix	0.000	1.000	1.000
['name_prefix', 'name_suffix']->race	0.000	1.000	1.000
['name_prefix', 'name_suffix', 'race']->ethnic	0.000	1.000	1.000
['name_prefix', 'race']->download_month	0.000	0.981	0.994

Tabela 5.7: Primeiras 30 DFS encontradas pelo algoritmo Pyro para o *dataset* 'wifi_hotspot_location' ranqueadas a partir da métrica μ^+.

DF	mu_plus	rfi_prime_plus	g3_prime
['OBJECTID']->Borough	1.000	1.000	1.000
['X']->Neighborhood Tabulation Area Code (NTACODE)	1.000	1.000	1.000
['Latitude']->BoroCode	1.000	1.000	1.000
['Latitude']->Type	1.000	1.000	1.000
['OBJECTID']->BoroCode	1.000	1.000	1.000
['OBJECTID']->Type	1.000	1.000	1.000
['Neighborhood Tabulation Area (NTA)']->Borough Name	1.000	1.000	1.000
['Neighborhood Tabulation Area (NTA)']->Neighborhood Tabulation Area Code (NTACODE)	1.000	1.000	1.000
['OBJECTID']->DOITT_ID	1.000	1.000	1.000
['OBJECTID']->City	1.000	1.000	1.000
['Neighborhood Tabulation Area Code (NTACODE)']->Borough	1.000	1.000	1.000
['Latitude']->City	1.000	1.000	1.000
['OBJECTID']->BCTCB2010	1.000	1.000	1.000
['BCTCB2010']->Census Tract	1.000	1.000	1.000
['DOITT_ID']->Census Tract	1.000	1.000	1.000
['OBJECTID']->Census Tract	1.000	1.000	1.000
['Longitude']->City	1.000	1.000	1.000
['Longitude']->Type	1.000	1.000	1.000
['Longitude']->BoroCode	1.000	1.000	1.000
['X']->Type	1.000	1.000	1.000
['Y']->Neighborhood Tabulation Area Code (NTACODE)	1.000	1.000	1.000
['OBJECTID']->BBL	1.000	1.000	1.000
['DOITT_ID']->BBL	1.000	1.000	1.000
['DOITT_ID']->Type	1.000	1.000	1.000
['Y']->Postcode	1.000	1.000	1.000
['Location (Lat, Long)']->Borough	1.000	1.000	1.000
['DOITT_ID']->Postcode	1.000	1.000	1.000
['Y']->BoroCode	1.000	1.000	1.000
['Location (Lat, Long)']->Council Distrcit	1.000	1.000	1.000
['Y']->Type	1.000	1.000	1.000

Tabela 5.8: Todas DFS encontradas pelo algoritmo FDX para o *dataset* 'wifi_hotspot_location' ranqueadas a partir da métrica μ^+.

DF	mu_plus	rfi_prime_plus	g3_prime
['Census Tract']->BCTCB2010	1.000	1.000	1.000
['Borough']->BoroCode	1.000	1.000	1.000
['City']->Borough Name	0.998	0.996	0.999
['Longitude']->X	0.994	0.996	0.994
['Latitude']->Y	0.994	0.996	0.994
['Neighborhood Tabulation Area Code (NTACODE)']->Neighborhood Tabulation Area (NTA)	0.994	0.997	0.996
['SSID']->Provider	0.864	0.924	0.925
['SSID', 'Provider']->Location_T	0.854	0.880	0.921
['Borough', 'BoroCode', 'Council Distrcit', 'Postcode']->BoroCD	0.801	0.891	0.858
['SSID', 'Location_T']->Remarks	0.616	0.615	0.916
['BIN']->BBL	0.596	0.673	0.757
['Remarks']->Type	0.378	1.000	1.000
['Borough', 'BoroCode', 'Council Distrcit']->Postcode	0.294	0.625	0.426
['Borough', 'BoroCode']->Council Distrcit	0.095	0.359	0.216
['SourceID']->Activated	0.072	0.998	0.997
['SourceID']->Name	0.035	0.860	0.782

Tabela 5.9: Primeiras 30 DFS encontradas pelo algoritmo Pyro para o *dataset* 'sisu_ufpr_curitiba_politecnico' ranqueadas a partir da métrica μ^+.

DF	mu_plus	rfi_prime_plus	g3_prime
['CO_IES_CURSO']->NO_CURSO	1.000	1.000	1.000
['NO_INSCRITO']->SG_UF_CANDIDATO	1.000	1.000	1.000
['NU_CPF']->TP_MOD_CONCORRENCIA	1.000	1.000	1.000
['NO_INSCRITO']->ST_OPCAO	1.000	1.000	1.000
['NU_CPF']->SG_UF_CANDIDATO	1.000	1.000	1.000
['NU_CPF']->MUNICIPIO_CANDIDATO	1.000	1.000	1.000
['CO_IES_CURSO']->DS_GRAU	1.000	1.000	1.000
['NO_INSCRITO']->TP_SEXO	1.000	1.000	1.000
['NU_NOTACORTE']->CO_IES_CURSO	1.000	1.000	1.000
['NU_NOTACORTE']->NO_CURSO	1.000	1.000	1.000
['NU_NOTACORTE']->DS_TURNO	0.999	1.000	1.000
['NU_NOTACORTE']->DS_GRAU	0.999	1.000	1.000
['NU_NOTACORTE']->TP_MOD_CONCORRENCIA	0.999	1.000	1.000
['MUNICIPIO_CANDIDATO']->SG_UF_CANDIDATO	0.998	0.998	1.000
['NO_CURSO']->DS_GRAU	0.961	0.970	0.978
['CO_IES_CURSO']->DS_TURNO	0.904	0.923	0.947
['NO_CURSO']->CO_IES_CURSO	0.865	0.940	0.899
['NO_CURSO']->DS_TURNO	0.696	0.754	0.846
['CO_IES_CURSO', 'DS_GRAU', 'DS_TURNO', 'TP_MOD_CONCORRENCIA']->NU_NOTACORTE	0.608	0.801	0.676
['CO_IES_CURSO', 'DS_TURNO', 'NO_CURSO', 'TP_MOD_CONCORRENCIA']->NU_NOTACORTE	0.608	0.801	0.676
['DS_GRAU', 'DS_TURNO', 'NO_CURSO', 'TP_MOD_CONCORRENCIA']->NU_NOTACORTE	0.608	0.801	0.676
['DS_TURNO', 'NO_CURSO', 'TP_MOD_CONCORRENCIA', 'TP_SEXO']->NU_NOTACORTE	0.605	0.779	0.677
['CO_IES_CURSO', 'DS_TURNO', 'TP_MOD_CONCORRENCIA', 'TP_SEXO']->NU_NOTACORTE	0.605	0.779	0.677
['CO_IES_CURSO', 'SG_UF_CANDIDATO', 'TP_MOD_CONCORRENCIA']->NU_NOTACORTE	0.578	0.769	0.652
['CO_IES_CURSO', 'ST_OPCAO', 'TP_MOD_CONCORRENCIA']->NU_NOTACORTE	0.575	0.757	0.645
['CO_ETAPA', 'CO_IES_CURSO', 'TP_MOD_CONCORRENCIA']->NU_NOTACORTE	0.574	0.780	0.645
['CO_CAMPUS', 'CO_IES_CURSO', 'TP_MOD_CONCORRENCIA']->NU_NOTACORTE	0.574	0.780	0.645
['CO_IES_CURSO', 'NU_ANO', 'TP_MOD_CONCORRENCIA']->NU_NOTACORTE	0.574	0.780	0.645
['CO_IES_CURSO', 'NO_MUNUCIPIO_CAMPUS', 'TP_MOD_CONCORRENCIA']->NU_NOTACORTE	0.574	0.780	0.645
['CO_IES_CURSO', 'DS_GRAU', 'NO_CURSO', 'TP_MOD_CONCORRENCIA']->NU_NOTACORTE	0.574	0.780	0.645

Tabela 5.10: Todas DFS encontradas pelo algoritmo FDX para o *dataset* 'sisu_ufpr_curitiba_politecnico' ranqueadas a partir da métrica μ^+.

DF	mu_plus	rfi_prime_plus	g3_prime
['NU_NOTACORTE']->TP_MOD_CONCORRENCIA	0.999	1.000	1.000
['NO_CURSO', 'TP_MOD_CONCORRENCIA']->DS_GRAU	0.961	0.970	0.979
['NO_CURSO', 'DS_GRAU']->DS_TURNO	0.732	0.783	0.868
['CO_IES_CURSO', 'SG_UF_CANDIDATO']->TP_SEXO	0.269	0.216	0.730
['NO_CURSO']->NU_NOTACORTE	0.229	0.594	0.371
['SG_UF_CANDIDATO']->MUNICIPIO_CANDIDATO	0.077	0.206	0.493
['TP_MOD_CONCORRENCIA', 'DS_GRAU', 'CO_IES_CURSO']->SG_UF_CANDIDATO	0.031	0.034	0.823
['TP_MOD_CONCORRENCIA']->CO_IES_CURSO	0.001	0.002	0.095

Tabela 5.11: Primeiras 30 DFS encontradas pelo algoritmo Pyro para o *dataset* 'adult' ranqueadas a partir da métrica μ^+.

DF	mu_plus	rfi_prime_plus	g3_prime
['education']->Education-num	1.000	1.000	1.000
['Education-num']->education	1.000	1.000	1.000
['fnlwgt']->race	0.899	0.896	0.980
['fnlwgt']->sex	0.890	0.888	0.959
['age', 'relationship']->Marital-status	0.702	0.714	0.851
['income', 'race', 'relationship', 'sex', 'workclass']->Marital-status	0.629	0.632	0.819
['Education-num', 'Hours-per-week', 'relationship']->Marital-status	0.628	0.636	0.812
['Hours-per-week', 'education', 'relationship']->Marital-status	0.628	0.636	0.812
['Education-num', 'relationship', 'workclass']->Marital-status	0.626	0.631	0.816
['education', 'relationship', 'workclass']->Marital-status	0.626	0.631	0.816
['Capital-gain', 'race', 'relationship', 'sex', 'workclass']->Marital-status	0.625	0.626	0.818
['Native-country', 'race', 'relationship', 'sex', 'workclass']->Marital-status	0.625	0.634	0.816
['Capital-loss', 'race', 'relationship', 'sex', 'workclass']->Marital-status	0.624	0.628	0.816
['education', 'relationship', 'sex']->Marital-status	0.622	0.630	0.816
['Education-num', 'relationship', 'sex']->Marital-status	0.622	0.630	0.816
['Capital-loss', 'Native-country', 'income', 'relationship', 'sex', 'workclass']->Marital-status	0.621	0.627	0.814
['education', 'race', 'relationship']->Marital-status	0.620	0.624	0.816
['Education-num', 'race', 'relationship']->Marital-status	0.620	0.624	0.816
['occupation', 'relationship']->Marital-status	0.620	0.622	0.814
['Hours-per-week', 'Marital-status', 'sex']->relationship	0.618	0.655	0.784
['Hours-per-week', 'race', 'relationship']->Marital-status	0.616	0.621	0.810
['Capital-gain', 'Capital-loss', 'Native-country', 'relationship', 'sex', 'workclass']->Marital-status	0.616	0.619	0.814
['Hours-per-week', 'relationship', 'sex']->Marital-status	0.616	0.623	0.808
['Capital-gain', 'Native-country', 'income', 'relationship', 'sex', 'workclass']->Marital-status	0.615	0.619	0.812
['Education-num', 'income', 'relationship']->Marital-status	0.615	0.613	0.812
['education', 'income', 'relationship']->Marital-status	0.615	0.613	0.812
['Education-num', 'Native-country', 'relationship']->Marital-status	0.613	0.616	0.813
['Native-country', 'education', 'relationship']->Marital-status	0.613	0.616	0.813
['Hours-per-week', 'relationship', 'workclass']->Marital-status	0.611	0.618	0.805
['Capital-loss', 'Education-num', 'relationship']->Marital-status	0.611	0.610	0.811

Tabela 5.12: Todas as DFS encontradas pelo algoritmo FDX para o *dataset* 'adult' ranqueadas a partir da métrica μ^+.

DF	mu_plus	rfi_prime_plus	g3_prime
['relationship']->Marital-status	0.592	0.580	0.808
['relationship']->sex	0.415	0.426	0.785
['Native-country']->race	0.123	0.145	0.866
['income']->Native-country	0.000	0.004	0.902

5.2.5.2 Análise Prática dos Ranqueamentos

Ao analisar os ranqueamentos gerados, observamos padrões claros que revelam as forças e fraquezas de cada abordagem.

O algoritmo Pyro, quando ranqueado pela métrica μ^+, consistentemente identifica DFs que representam relações de negócio ou estruturais óbvias e úteis. No *dataset* 'hate_crimes' (Tabela 5.3), as primeiras posições são dominadas por DFs com 'Full Complaint ID' no LHS, todas com μ^+ = 1,000. Isso corretamente aponta 'Full Complaint ID' como uma chave primária, pois determina de forma única quase todos os outros atributos do registro (como County, Record Create Date, Offense Description, etc.). Da mesma forma, no *dataset* ncvoter (Tabela 5.5), Pyro identifica 'voter_id' e 'voter_reg_num' como chaves candidatas, novamente com pontuações perfeitas.

Além de chaves, essa combinação é hábil em detectar DFs aproximadas com forte sentido semântico. Por exemplo, no ranqueamento do ncvoter, a DF 'zip_code' -> 'city' aparece com μ^+ = 0,993. Esta é uma regra de negócio clássica do mundo real: um código postal quase sempre determina uma cidade, mas pequenas exceções podem existir. Outro exemplo interessante

é 'first_name' -> 'gender' (μ^+ = 0,859), que captura uma relação probabilística socialmente construída, útil para análise de dados e enriquecimento.

O desempenho do FDX é visivelmente mais instável. No *dataset* ncvoter (Tabela 5.6), ele falha em identificar as chaves candidatas óbvias. Além disso, ele apresenta DFs com pouco ou nenhum sentido semântico, como 'gender' -> 'first_name' com um μ^+ de apenas 0,004. Embora possa existir uma correlação, um gênero obviamente não determina um primeiro nome. Para essa DF, as outras métricas também deram notas baixas, sendo RFI'^+ = 0,059 e g'_3 = 0,042. Outro exemplo é 'Arrest Date' -> 'Arrest Id' no *dataset* 'hate_crimes' (Tabela 5.4) com μ^+ = 0,001 mas RFI'^+ = 0,820. A data de prisão não determina a ID da prisão; o oposto seria mais provável. Isso mostra que FDX, sem ajuste de parâmetros, pode capturar correlações espúrias ou na direção errada.

O FDX pode, em certos casos, encontrar DFs válidas e úteis. No *dataset* 'wifi_hotspot_location' (Tabela 5.8), o FDX teve um bom desempenho. Ele identificou DFs fortes e corretas como 'Borough' -> 'BoroCode' (μ^+ = 1.000) e a relação quase perfeita entre coordenadas 'Latitude' -> 'Y' (μ^+ = 0,994). Ele também encontrou 'SSID' -> 'Provider' (μ^+ = 0,864), uma DF aproximada útil que o Pyro não reportou em seu Top 30. Este sucesso no 'wifi_hotspot_location' em contraste com o fracasso no 'ncvoter' e 'hate_crimes' reforça a conclusão de que a utilidade do FDX (com seus parâmetros padrão) é altamente dependente das características intrínsecas do *dataset*.

5.3 PARÂMETROS DOS ALGORITMOS

Os algoritmos Pyro e FDX possuem alguns parâmetros que alteram seus comportamentos. Nessa seção, abordaremos brevemente algumas dessas particularidades, já que podem ter implicações na performance dos algoritmos, além de resultados distintos.

5.3.1 Pyro

O algoritmo Pyro utiliza um parâmetro de limiar de erro (e_{max}) para definir o que constitui uma Dependência Funcional Aproximada (DFA) válida e minimal. Uma DFA $X \rightarrow A$ é considerada uma generalização de $XY \rightarrow A$ (ou qualquer DFA com mais atributos em seu lado esquerdo que a contenham). A medida de erro utilizada pelo Pyro é monótona, o que significa que adicionar atributos ao lado esquerdo de uma DFA nunca aumenta o erro; ele pode apenas diminuir ou permanecer o mesmo (Kruse e Naumann [9]). Uma DFA é definida como minimal se seu erro for no máximo e_{max}, e todas as suas generalizações (com menos atributos no LHS) tiverem um erro maior que e_{max} [9]. Isso significa que o Pyro busca a DFA mais geral que ainda satisfaz o critério de erro.

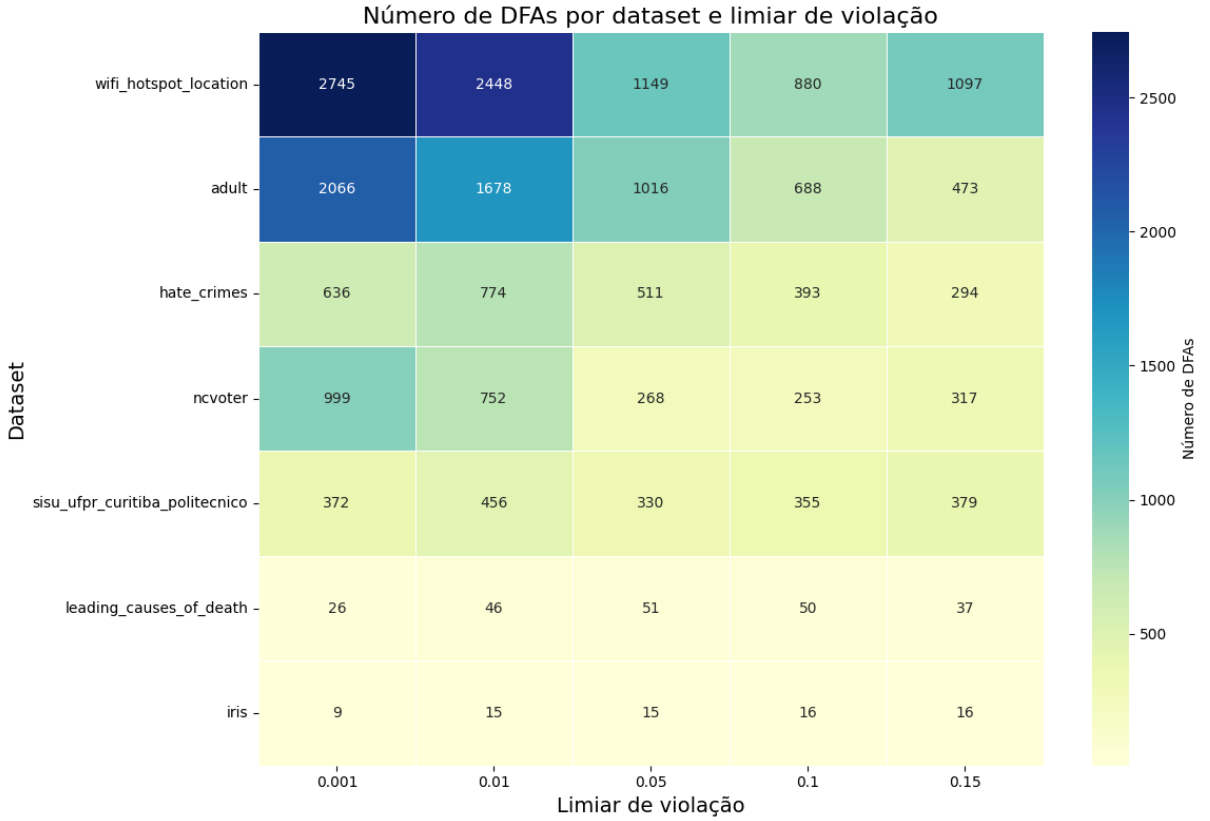


Figura 5.6: Número de DFAs encontradas pelo Pyro variando e_{max}

O comportamento observado de encontrar menos DFAs quando o valor de e_{max} é maior é contra-intuitivo à primeira vista, mas é uma consequência direta da definição de minimalidade no Pyro. Aumentando o e_{max}, mais dependências (incluindo generalizações) podem agora ser consideradas válidas porque seu erro está abaixo do novo, mais alto, e_{max}. Consequentemente, uma DFA $X \rightarrow A$ que antes era minimal (porque suas generalizações tinham erro acima do e_{max} anterior) pode deixar de ser minimal se uma de suas generalizações $Y \rightarrow A$ (onde $Y \subseteq X$) agora também tiver um erro abaixo do novo e_{max}. Nesse cenário, $Y \rightarrow A$ seria a dependência minimal reportada, e $X \rightarrow A$ (sendo uma especialização de $Y \rightarrow A$) não seria considerada minimal e, portanto, não seria explicitamente descoberta ou reportada como tal.

Um exemplo prático desse comportamento pode ser observado na relação da Tabela 2.1, onde o conjunto de DFAs descobertas varia conforme o limiar de erro e_{max}:

- Com um limiar de erro baixo ($e_{max} = 0,01$), a dependência $\{\texttt{Nome}, \texttt{Sobrenome}\} \rightarrow \texttt{Id}$ é identificada.
- Ao aumentar o limiar para $e_{max} = 0,1$, o resultado é alterado: a DFA mais geral $\{\texttt{Sobrenome}\} \rightarrow \texttt{Id}$ é descoberta, enquanto a dependência anterior, mais específica, é removida do conjunto final.

O *threshold* e_{max} ideal dependerá das características específicas do *dataset* e do caso de uso, influenciando tanto a quantidade de dependências descobertas quanto a eficiência do tempo de execução do algoritmo, com um e_{max} mais alto geralmente resultando em tempos de execução mais rápidos, já que o algoritmo precisa verificar menos candidatos (Kruse e Naumann [9]). O ajuste de e_{max} também é importante para diminuir o sobreajuste, já que caso esteja muito baixo, é possível que sejam enumeradas muitas DFAs com |LHS| maiores, que tendem ao sobreajuste.

Já para valores muito baixos de e_{max}, é possível que sejam enumeradas DFAs incorretas para uma proporção exagerada de pares de tuplas, o que também não é ideal.

5.3.2 FDX

O uso do FDX para a descoberta de DFs vai além de simplesmente executá-lo, exigindo do usuário um entendimento aprofundado tanto de seus parâmetros operacionais quanto da natureza dos dados de entrada. Isso porque sua eficácia está diretamente atrelada à configuração de parâmetros, além da inferência e o ajuste dos tipos de dados das colunas. Embora o FDX possa inferir se uma coluna é numérica, categórica ou textual, essa inferência pode não ser ótima e o algoritmo permite que ela seja ajustada. Essa decisão é crítica, pois o tipo de dado determina como as comparações entre tuplas são realizadas durante a etapa de transformação dos dados. Por exemplo, tratar um identificador categórico como um valor numérico levaria a comparações sem sentido e, consequentemente, a resultados falhos.

A necessidade de intervenção do usuário estende-se aos demais parâmetros centrais que governam o comportamento do algoritmo. O $sparsity$, por exemplo, corresponde diretamente ao termo de regularização L1 (λ) na otimização da matriz de covariância inversa e controla a complexidade do modelo de dependência resultante ([22]). A escolha de um valor adequado é um balanço que o usuário deve fazer entre a descoberta de um conjunto mais abrangente de DFs (menor esparsidade) e a obtenção de um modelo mais simples e parcimonioso, com menos DFs resultantes (maior esparsidade) [22]. Da mesma forma, os parâmetros eps e tol ajustam a robustez do FDX a ruídos e imperfeições. O eps implementa a constante de erro (ϵ) da definição probabilística de uma DF (2.5), sendo o limiar de violações que o usuário define com base em sua expectativa sobre o nível de ruído nos dados. Já a tol define o limiar para a *igualdade aproximada* na fase de transformação de dados, e seu significado prático depende intrinsecamente do tipo de dado da coluna em questão, reforçando a tese de que o conhecimento do domínio e dos dados é indispensável para parametrizar o FDX e obter resultados melhores.

Para contextualizar a análise, os resultados de ranqueamento apresentados na Seção 5.2.5.1 foram gerados utilizando uma configuração de parâmetros base para o FDX, definida como `tol = 0.000001`, `eps = 0.05` e `sparsity = 0`. A fim de demonstrar a sensibilidade do algoritmo à sua parametrização, uma nova execução foi realizada sobre o *dataset* `ncvoter`, com os valores ajustados para `tol = 0.001` e `sparsity = 0.01`. Os resultados desta análise estão detalhados na Tabela 5.13.

Tabela 5.13: DFs descobertas pelo FDX no *dataset* 'ncvoter' com parâmetros ajustados (`tol=0.001`, `eps=0.05`, `sparsity=0.01`), ranqueadas pela métrica μ^+.

fd	mu_plus	rfi_prime_plus	g3_prime
['first_name']->gender	0.858	0.850	0.949
['city', 'last_name']->street_address	0.315	0.313	0.329
['city']->last_name	0.017	0.047	0.062
['race', 'download_month']->register_date	0.004	0.030	0.154
['name_prefix']->name_suffix	0.0	1.0	1.0
['name_prefix', 'name_suffix']->race	0.0	1.0	1.0
['name_prefix', 'name_suffix', 'race']->ethnic	0.0	1.0	1.0
['name_prefix', 'race']->download_month	0.0	0.981	0.993
['street_address']->middle_name	0.0	0.0279	0.309

Ao alterar alguns parâmetros, vemos mudanças significativas, como o exemplo da DF 'gender' -> 'first_name' da Tabela 5.6, mas que com novos parâmetros, foi corretamente invertida, obtendo, na Tabela 5.13, 'first_name' -> 'gender', que faz mais sentido semântico. Ainda assim, em *datasets* como 'ncvoter', mesmo fazendo diversas alterações nos parâmetros, é difícil obter bons resultados.

6 CONCLUSÃO

Este trabalho instituiu um protocolo para a comparação de abordagens de descoberta e o ranqueamento de DFs e DFAs. Através da aplicação dos algoritmos de descoberta HyFD, Pyro e FDX em múltiplos conjuntos de dados com características distintas, e da avaliação subsequente utilizando as métricas de interesse g'_3, RFI'^+ e μ^+, foi possível conduzir uma análise comparativa e aprofundada, cujas conclusões abordam diretamente às questões de pesquisa que nortearam este estudo.

A investigação sobre a **eficácia das métricas de DFAs quando aplicadas a DFs exatas** demonstrou que, para fins de ranqueamento, tais métricas possuem valor informativo limitado. Em grande parte dos cenários analisados, as métricas tenderam a atribuir o valor máximo para qualquer DF exata, tornando-as ineficazes para a tarefa de distinguir a relevância entre elas. Contudo, esta característica pode ser explorada de maneira secundária, seja para validar a corretude de algoritmos de descoberta de DFs exatas, ou como um ponto de partida para o desenvolvimento de novas métricas especializadas.

Ao analisar **como o tamanho e a unicidade do determinante afetam as métricas**, os resultados revelaram uma sensibilidade particular da métrica μ^+. Demonstrou-se que valores mais altos de μ^+ estão fortemente correlacionados a determinantes com alta unicidade. Essa propriedade torna a métrica μ^+ uma ferramenta valiosa e um forte indicador para a identificação de conjuntos de atributos que são chaves candidatas em um esquema de dados.

A pesquisa também abordou a questão de se **as métricas atribuem pontuações altas a DFs de projeto**, ou seja, aquelas com maior valor semântico. A análise semântica dos ranqueamentos gerados indicou que, embora nenhuma métrica seja infalível, a μ^+ obteve sucesso notável. Em diversos conjuntos de dados, ela posicionou nas primeiras colocações as DFs que, intuitivamente, seriam escolhidas para guiar o processo de normalização. No entanto, o estudo adverte que o ranqueamento não deve ser seguido cegamente, pois dependências de menor relevância semântica ainda podem obter altas pontuações, reforçando a necessidade de conhecimento do domínio. As outras métricas analisadas (g'_3 e RFI'^+), ao darem pontuações muito altas a um maior volume de DFs em comparação a μ^+, acabam poluindo o topo de seus ranqueamentos com DFs sobreajustadas.

Finalmente, para responder **em quais contextos cada abordagem de descoberta é mais eficaz**, a comparação direta dos resultados dos algoritmos para os mesmos conjuntos de dados ofereceu percepções cruciais. A análise evidenciou, por exemplo, um contraste marcante entre o FDX, que se mostrou muito sensível às características intrínsecas dos dados e parâmetros de entrada, e o Pyro, que apresentou maior consistência em seus resultados. Essas observações auxiliam o leitor a selecionar o algoritmo e os parâmetros que melhor se alinham às suas necessidades específicas.

Este trabalho não apenas forneceu uma comparação entre ferramentas, mas também pode orientar analistas de dados em tarefas de normalização e engenharia de dados. As discussões e os resultados apresentados capacitam o leitor a tomar decisões mais informadas sobre a escolha de algoritmos, seus parâmetros e a interpretação de métricas de relevância, culminando em um processo de descoberta de conhecimento mais assertivo e de maior qualidade.

6.1 TRABALHOS FUTUROS

Este trabalho possui algumas limitações que abrem oportunidades para seguir explorando o tema em trabalhos futuros. Algumas delas são:

- **Análise de desempenho computacional:** Realizar um estudo comparativo focado no tempo de execução e no consumo de memória de cada algoritmo. Esta análise de eficiência e escalabilidade forneceria critérios práticos para a seleção da abordagem mais adequada em cenários com restrições de recursos computacionais, complementando as métricas de avaliação de DFs já investigadas.
- **Experimentação com *datasets* de larga escala:** Avaliar a robustez e o comportamento dos algoritmos em datasets com dimensões significativamente maiores, tanto em número de tuplas (linhas) quanto de atributos (colunas). Tal investigação pode ser particularmente interessante para o FDX (abordagem estatística), cujo comportamento e precisão podem variar substancialmente com o volume de dados de entrada.
- **Desenvolvimento de novas métricas de avaliação:** Explorar a criação de novas métricas para avaliação e ranqueamento de dependências funcionais. Esta pesquisa poderia seguir duas vertentes: a concepção de métricas híbridas, que combinem características das métricas existentes para uma avaliação mais holística; e a investigação de novos paradigmas, como a incorporação da relevância semântica das DFs ou o seu impacto em tarefas de aprendizado de máquina subsequentes.

REFERÊNCIAS

[1] Caruccio, L., Deufemia, V. e Polese, G. (2016). Relaxed functional dependencies—a survey of approaches. *IEEE Trans. on Knowl. and Data Eng.*, 28(1):147–165.

[2] Chu, X., Ilyas, I. e Papotti, P. (2013). Discovering denial constraints. *VLDB*.

[3] Davies, S. e Russell, S. J. (1994). Np-completeness of searches for smallest possible feature sets.

[4] Elmasri, R. e Navathe, S. B. (2017). *Fundamentals of database systems*. Pearson, 7th edition.

[5] Fan, W., Geerts, F., Li, J. e Xiong, M. (2011). Discovering conditional functional dependencies. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*.

[6] Friedman, J., Hastie, T. e Tibshirani, R. (2007). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.

[7] Giannella, C. e Robertson, E. (2004). On approximation measures for functional dependencies. *Inf. Syst.*, 29(6):483–507.

[8] Kivinen, J. e Mannila, H. (1995). Approximate inference of functional dependencies from relations. *Theoretical Computer Science*, 149(1):129–149. Fourth International Conference on Database Theory (ICDT '92).

[9] Kruse, S. e Naumann, F. (2018). Efficient discovery of approximate dependencies. *VLDB*.

[10] Liu, J., Li, J., Liu, C. e Chen, Y. (2012). Discover dependencies from data - a review. */L EE TKDE*.

[11] Mandros, P., Boley, M. e Vreeken, J. (2017). Discovering reliable approximate functional dependencies. Em *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, página 355–363, New York, NY, USA. Association for Computing Machinery.

[12] Mandros, P., Boley, M. e Vreeken, J. (2020). Discovering dependencies with reliable mutual information. *Knowl. Inf. Syst.*, 62(11):4223–4253.

[13] Papenbrock, T., Bergmann, T., Finke, M., Zwiener, J. e Naumann, F. (2015a). Data profiling with metanome. *Proceedings of the VLDB Endowment*, 8(12):2150–8097/15/08.

[14] Papenbrock, T., Ehrlich, J., Marten, J., Neubert, T., Rudolph, J.-P., Schonberg, M., Zwiener, J. e Schonberg, M. (2015b). Functional dependency discovery: An experimental evaluation of seven algorithms. *VLDB*.

[15] Papenbrock, T. e Naumann, F. (2016). A hybrid approach to functional dependency discovery. Em *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, página 821–833, New York, NY, USA. Association for Computing Machinery.

[16] Parciak, M., Weytjens, S., Hens, N., Neven, F., Peeters, L. M. e Vansummeren, S. (2024). Measuring Approximate Functional Dependencies: A Comparative Study . Em *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, páginas 3505–3518, Los Alamitos, CA, USA. IEEE Computer Society.

[17] Pena, E. H. M. e De Almeida, E. C. (2021). Discovery and Application of Data Dependencies. Em *Anais do XXXIV Concurso de Teses e Dissertações da SBC (CTD-SBC 2021)*, páginas 1–6, Brasil. Sociedade Brasileira de Computação.

[18] Piatetsky-Shapiro, G. e Matheus, C. (1993). Measuring data dependencies in large databases. Em *Proceedings of the 2nd International Conference on Knowledge Discovery in Databases, pp. 162—173, AAAI Press, 1993*.

[19] Roulston, M. S. (1999). Estimating the errors on measured entropy and mutual information. *Physica D: Nonlinear Phenomena*, 125(3):285–294.

[20] Rovetta, A. (2020). Raiders of the lost correlation: A guide on using pearson and spearman coefficients to detect hidden correlations in medical sciences. *Cureus 12(11): e11794. doi:10.7759/cureus.11794*.

[21] Sánchez, D., Serrano, J. M., Blanco, I., Martín-Bautista, M. J. e Vila, M.-A. (2008). Using association rules to mine for strong approximate dependencies. *Springer Science+Business Media, LLC 2008*.

[22] Zhang, Y., Guo, Z. e Rekatsinas, T. (2020). A statistical perspective on discovering functional dependencies in noisy data. *ACM*.