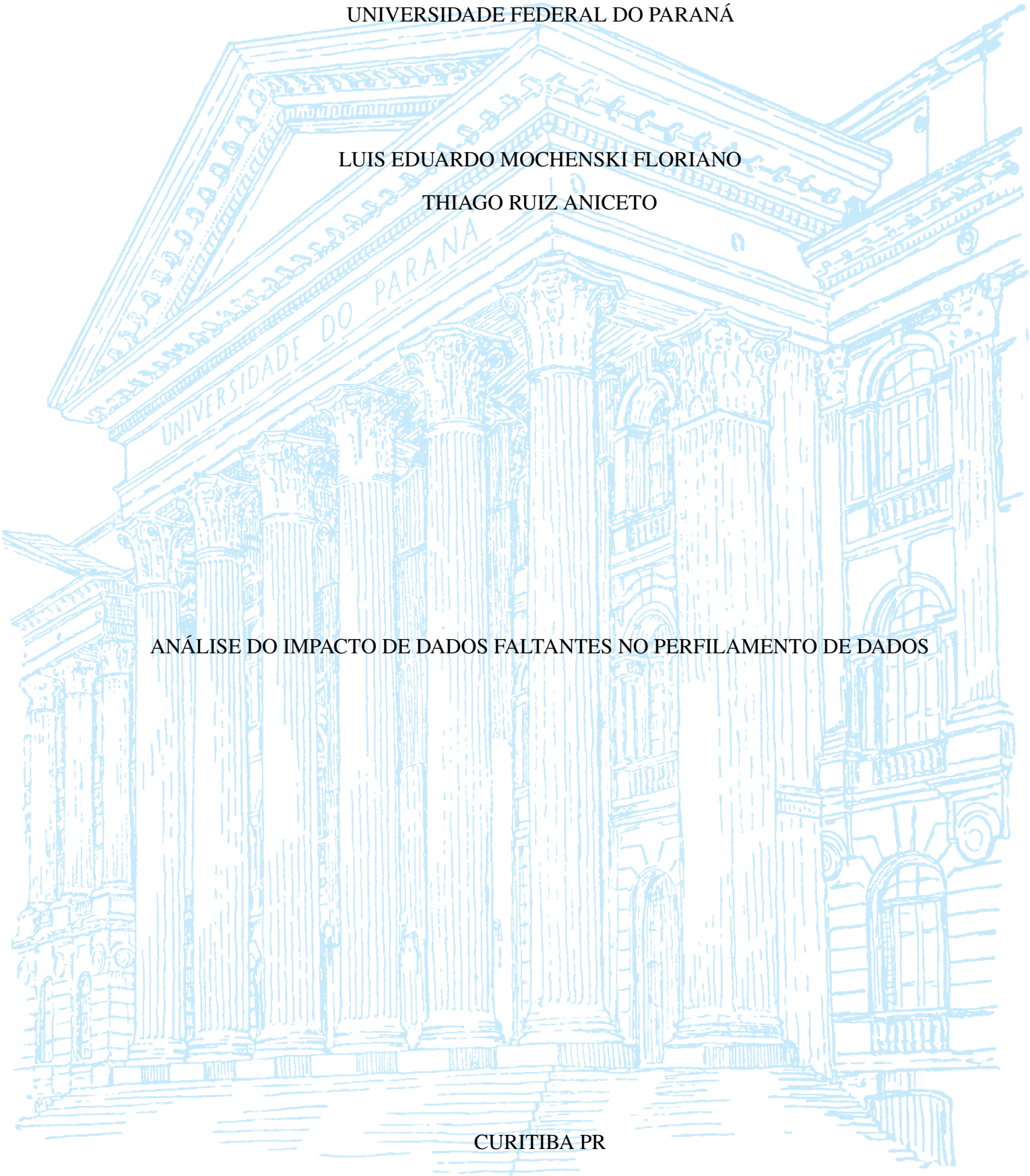
UNIVERSIDADE FEDERAL DO PARANÁ

LUIS EDUARDO MOCHENSKI FLORIANO

THIAGO RUIZ ANICETO




# ANÁLISE DO IMPACTO DE DADOS FALTANTES NO PERFILAMENTO DE DADOS

CURITIBA PR

2023

LUIS EDUARDO MOCHENSKI FLORIANO
THIAGO RUIZ ANICETO

# ANÁLISE DO IMPACTO DE DADOS FALTANTES NO PERFILAMENTO DE DADOS

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: *Ciência da Computação*.

Orientador: Prof. Dr. Marcos Sfair Sunyé e Prof. Dr. Eduardo Cunha de Almeida.

CURITIBA PR

2023

# Ficha catalográfica

Substituir o arquivo `0-iniciais/catalografica.pdf` pela ficha catalográfica fornecida pela Biblioteca da UFPR (PDF em formato A4).

**Instruções para obter a ficha catalográfica e fazer o depósito legal da tese/dissertação (contribuição de André Hochuli, abril 2019. Links atualizados Wellton Costa, Nov 2022):**

1. Estas instruções se aplicam a dissertações de mestrado e teses de doutorado. Trabalhos de conclusão de curso de graduação e textos de qualificação não precisam segui-las.
2. Verificar se está usando a versão mais recente do modelo do PPGInf e atualizar, se for necessário (`https://gitlab.c3sl.ufpr.br/maziero/tese`).
3. conferir o *checklist* de formato do Sistema de Bibliotecas da UFPR, em `https://bibliotecas.ufpr.br/servicos/normalizacao/`
4. Enviar e-mail para "`referencia.bct@ufpr.br`" com o arquivo PDF da dissertação/tese, solicitando a respectiva ficha catalográfica.
5. Ao receber a ficha, inseri-la em seu documento (substituir o arquivo `0-iniciais/catalografica.pdf` do diretório do modelo).
6. Emitir a Certidão Negativa (CND) de débito junto a biblioteca, em `https://bibliotecas.ufpr.br/servicos/certidao-negativa/`
7. Avisar a secretaria do PPGInf que você está pronto para o depósito. Eles irão mudar sua titulação no SIGA, o que irá liberar uma opção no SIGA pra você fazer o depósito legal.
8. Acesse o SIGA (`http://www.prppg.ufpr.br/siga`) e preencha com cuidado os dados solicitados para o depósito da tese.
9. Aguarde a confirmação da Biblioteca.
10. Após a aprovação do pedido, informe a secretaria do PPGInf que a dissertação/tese foi depositada pela biblioteca. Será então liberado no SIGA um link para a confirmação dos dados para a emissão do diploma.

# Ficha de aprovação

Substituir o arquivo 0-iniciais/aprovacao.pdf pela ficha de aprovação fornecida pela secretaria do programa, em formato PDF A4.

*Dedicamos este trabalho a todos os membros do curso de Bacharelado em Ciência da Computação da Universidade Federal do Paraná, a quem temos a honra de fazer parte.*

## AGRADECIMENTOS

Gostaríamos de expressar nossa profunda gratidão ao Prof. Dr. Eduardo H. M. Pena, ao Prof. Dr. Eduardo Cunha de Almeida, ao Prof. Dr. Marcos Sfair Sunyé e a todos os colaboradores e familiares envolvidos neste processo, cujas valiosas contribuições desempenharam um papel fundamental no desenvolvimento e aprimoramento do nosso Trabalho de Conclusão de Curso.

Expressamos nossa sincera gratidão pela valiosa oportunidade de estudar na Universidade Federal do Paraná e desejamos estender nossos mais profundos agradecimentos a todos os professores que dedicaram seu tempo ao ensino. Cada instante vivenciado nesta instituição, sob a orientação desses educadores, contribuiu de maneira significativa para a nossa formação e crescimento acadêmico e profissional.

## RESUMO

A Engenharia de Dados é responsável pelo tratamento e processamento de dados. Isto envolve a descoberta de regras de negócio, que abrange a área de *Data Profiling*. Porém a presença de dados nulos nas bases de dados, pode impactar negativamente a descoberta destas regras. Este impacto atua tanto no tempo de execução, quanto na qualidade e quantidade de resultados. Sendo assim, neste trabalho foram reunidos 5 algoritmos, para representar cada uma das áreas de descoberta de dependências em *Data Profiling* - dependências funcionais (FD), dependências de coluna única (UCC), dependências de negação (DC), dependências de inclusão (IND) e dependências de ordem (OD) - e 3 mecânicas de poluição de dados nulos - *Missing At Random* (MAR), *Missing Not At Random* (MNAR) e *Missing Completely At Random* (MCAR). Dessa forma, cada algoritmo selecionado foi comparado com cada uma das mecânicas de dados nulos, para 8 *datasets* distintos em domínio, quantidade de registros e colunas. Ao final, é discutido o impacto dos dados nulos, frente a cada um dos algoritmos e cada uma das mecânicas.



Palavras-chave: Dados Nulos. Perfilamento de Dados. Engenharia de Dados.

**ABSTRACT**

Data Engineering is responsible for the treatment and processing of data. This involves the discovery of business rules, which encompasses the area of Data Profiling. However, the presence of null data in databases can negatively impact the discovery of these rules. This impact affects both runtime and the quality and quantity of results. Therefore, in this work, 5 algorithms were gathered to represent each of the areas of dependency discovery in Data Profiling - functional dependencies (FD), single-column dependencies (UCC), denial constraint (DC), inclusion dependencies (IND), and order dependencies (OD) - and 3 mechanisms for introducing null data - Missing At Random (MAR), Missing Not At Random (MNAR), and Missing Completely At Random (MCAR). In this way, each selected algorithm was compared with each of the null data mechanisms for 8 different datasets in terms of subject, quantity of records, and columns. In the end, the impact of null data is discussed in relation to each algorithm and each mechanism.

Keywords: Null data. Data Profiling. Data Engineering.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE ACRÔNIMOS

| | |
|---|---|
| DINF | Departamento de Informática |
| PPGINF | Programa de Pós-Graduação em Informática |
| UFPR | Universidade Federal do Paraná |
| FD | Functional Dependency |
| DC | Dependência de Negação |
| UCC | Unique Combination Column |
| OD | Order Dependency |
| IND | Inclusion Dependency |
| PLI | Position List Indexes |
| MAR | Missing at Random |
| MNAR | Missing Not at Random |
| MCAR | Missing Completely at Random |
| HPI | Hasso Plattner Institute |
| PK | Primary Key |
| FK | Foreign Key |
| CSV | Comma-Separated Values |
| CPU | Central Processing Unit |
| GPU | Graphics Processing Unit |
| RAM | Random Access Memory |

**SUMÁRIO**

# 1 INTRODUÇÃO

A engenharia de dados é a área responsável por transformar os dados brutos em dados refinados por meio de um *pipeline* de atividades. Este refinamento depende das regras de negócio de cada base de dados. Para isso a técnica de perfilamento dos dados auxilia a descoberta de regras de negócio. No entanto, a presença de dados nulos possui um impacto direto na performance e qualidade de resultados dos algoritmos usados para este fim. Neste trabalho foram realizados experimentos para evidenciar o comportamento dos algoritmos, do estado da arte de perfilamento de dados, frente aos dados nulos. Para isto, foram estudados os tipos de dependências envolvidas em perfilamento de dados e seus algoritmos, bem como as características de dados nulos em uma base de dados.

Em primeiro lugar, foram estudados os tipos de dependência do estado da arte do perfilamento de dados. São elas: dependências funcionais (FD) (Papenbrock et al., 2015b), dependências de coluna única (UCC) (Papenbrock, 2017), dependências de negação (DC) (Pena et al., 2019), dependências de inclusão (IND) (Papenbrock et al., 2015c) e dependências de ordem (OD) (Langer e Naumann, 2016). Para exemplificar cada uma das dependências, considere a seguinte relação $R_N$ de veículos - Tabela 1.1.

Tabela 1.1: Relação de Veículos

| Tupla | Renavam | Carro | Marca | Ano | Valor |
|---|---|---|---|---|---|
| t1 | 13260962389 | Gol | Volkswagen | 2005 | 15000 |
| t2 | 98765432101 | Gol | Volkswagen | 2007 | 18000 |
| t3 | 27158493820 | Gol | Volkswagen | 2010 | 20000 |
| t4 | 35698712541 | Mobi | Volkswagen | 2012 | 15000 |
| t5 | 44300268437 | Mobi | Fiat | 2015 | 30000 |
| t6 | 51789012345 | Mobi | Fiat | 2030 | 20000 |

Uma FD presente nesta relação seria que o atributo Renavam define funcionalmente os atributos Carro, Marca e Ano - $\{Renavam\} \rightarrow \{Carro, Marca, Ano\}$. Já uma UCC é capaz de definir uma chave primária, por exemplo $\{Renavam, Carro\}$. Uma OD, pode ser verificada ao comparar $Renavam \geq Ano$. Uma IND, representa uma FK *Foreign Key*, para isso considere a relação $R_V$ na Tabela 1.2 - note que a coluna $Renavam$ representa uma FK representada pela IND: $R_N\,[Renavam] \subseteq R_V\,[Renavam]$.

Tabela 1.2: Relação de Valores dos Veículos

| Tupla | Renavam | Valor |
|---|---|---|
| t1 | 13260962389 | 15000 |
| t2 | 98765432101 | 18000 |
| t3 | 27158493820 | 20000 |
| t4 | 35698712541 | 15000 |
| t5 | 44300268437 | 30000 |
| t6 | 51789012345 | 20000 |

A DC é capaz de generalizar as outras dependências por uma desigualdade. A generalização da FD anterior seria - $\varphi : \neg(t.Renavam \neq t'.Renavam \wedge t.Carro = t'.Carro \wedge$

$t.Marca = t'.Marca \wedge t.Ano = t'.Ano)$. Já uma generalização de uma UCC seria: $\varphi : \neg(t.Renavam! = t'.Renavam)$. As outras dependências também possuem suas generalizações.

Para cada uma delas há um algoritmo dominante e que cumpre o objetivo da descoberta da respectiva dependência. Em DF, foi considerado o HyFD (Papenbrock et al., 2015b); Para UCC o HyUCC (Papenbrock, 2017); em DC, foram estudados três algoritmos - Hydra (Bleifuß et al., 2017), DCFinder (Pena et al., 2019) e ECP (Pena et al., 2022), mas apenas este último foi considerado nos testes; para IND, o algoritmo BINDER (Papenbrock et al., 2015c); e por fim, para OD foi utilizado o algoritmo ORDER (Langer e Naumann, 2016).

Após a seleção dos algoritmos, parte-se para a compreensão dos dados nulos. Para isso deve se considerar não apenas a sua semântica (*string* vazia, *null*, *NA*, etc.), mas também a forma com que eles estão dispostos na base de dados. Neste sentido, houve um estudo das boas práticas em *data cleaning* (Osborne, 2013), em especial sobre as diferentes mecânicas de poluição de dados nulos *Missing At Random* (MAR), *Missing Not At Random* (MNAR) e *Missing Completely At Random* (MCAR).

A dependência MCAR, considera a presença completamente aleatória, isso pode representar uma falha humana ou de software. Para simular este cenário, considere a imputação de 3 dados nulos, nas tuplas $t_2$, $t_4$ e $t_5$ no exemplo da Tabela 1.1 normalizado com a Tabela 1.2. As novas tuplas estão na Tabela 1.4.

Tabela 1.3: Relação de Veículos

| Tupla | Renavam | Carro | Marca | Ano | Valor |
|---|---|---|---|---|---|
| t2 | null | Gol | Volkswagen | 2007 | 18000 |
| t4 | 35698712541 | Mobi | null | 2012 | 15000 |
| t5 | 44300268437 | Mobi | Fiat | 2015 | null |

A presença de dados nulos pode ser classificada como MAR, neste caso, ainda são aleatórios, mas podem estar atrelados à um determinado padrão dentro da base. Por exemplo, podemos imaginar que as tuplas $t_2$ e $t_4$ não possuem $Renavam$ definido e por causa disso, não se pode definir o $Carro$, nem a $Marca$ nem o $Ano$ - note que este foi o mesmo exemplo da FD.

Tabela 1.4: Relação de Veículos

| Tupla | Renavam | Carro | Marca | Ano | Valor |
|---|---|---|---|---|---|
| t2 | null | Gol | Volkswagen | 2007 | 18000 |
| t4 | 35698712541 | Mobi | null | 2012 | 15000 |
| t5 | 44300268437 | Mobi | Fiat | 2015 | null |

A mecânica dos nulos também pode ser não aleatória, que é o caso do MNAR - podem estar atrelados a algum acontecimento externo. Por exemplo, observe que $t_6$ possui um carro fabricado no ano de 2030 e nesse caso a fabricante pode optar por deixar o preço sem definição. Logo a tupla $t_6$, poderia ser representada da seguinte forma na Tabela 1.5.

Tabela 1.5: Relação de Veículos

| Tupla | Renavam | Carro | Marca | Ano | Valor |
|---|---|---|---|---|---|
| t6 | 51789012345 | Mobi | Fiat | 2030 | null |

A importância de compreender o dado nulo se dá porque cada um dos algoritmos tem uma forma diferente de considerá-los em seus programas. Alguns ignoram (Papenbrock et al.,

2015c), outros tratam (Papenbrock et al., 2015b; Pena et al., 2022; Papenbrock, 2017) e há aqueles que não são capazes de executar com dados faltantes (Langer e Naumann, 2016).

Dentro da área de perfilamento de dados, há trabalhos que comparam a performance de vários algoritmos de FDs (Papenbrock et al., 2015b) e também consideram métricas de qualidade e comportamento de dados faltantes (Berti-Équille et al., 2018). Já os algoritmos de descobertas de DCs tem duas abordagens - DCs exatas (Hydra) e aproximadas (DcFinder, ECP) - as DCs ex tatas consideram uma base de dados sem dados faltantes, enquanto as DCs aproximadas consideram uma função limitante para tolerar algumas violações, sem desconsiderar a regra descoberta. Outros algoritmos como HyUCC e HyFD consideram que dados nulos são iguais $null = null$, mas nem sempre isso é condizente com a realidade dos dados - por exemplo, se $Renavam$ é uma PK e tiver dois nulos na coluna, isso pode caracterizar uma brecha para a descoberta falhar. Já em IND, o algoritmo BINDER implementa uma estratégia em que o dado nulo é ignorado - neste caso qualquer nulo simplesmente reduz a base de dados e a quantidade de resultados a medida que as faltas aumentam. Há também um estudo que considera 13 algoritmos de IND (Dürsch et al., 2019) - neste estudo há a afirmação de que dados nulos em descoberta de IND ainda é um campo em aberto - mas não há nenhum experimento que mostre o real impacto deles nos algoritmos.

Dado estes cenários, há falta de um estudo que reúna os principais algoritmos de *data profiling* e evidenciem o impacto dos dados faltantes em todas as dependências. Sendo assim, este trabalho contempla uma análise de 5 dependências e seus principais algoritmos - frente aos dados faltantes. Afinal, a existência ou não de dados nulos na base, muda a entrada de cada algoritmo e por consequência a sua saída. Logo, cabe também uma análise comparativa das saídas que proporcionem informações referentes à quantidade e qualidade dos resultados. Adiciona-se também a estas métricas, o tempo de execução, que pode variar em virtude de diferentes entradas e presença de dados nulos.

## 1.1 CONTRIBUIÇÃO

O presente trabalho contribui com a execução destes algoritmos, com 8 bases de dados de tamanhos distintos, variando a disposição dos nulos em quantidade e em mecânica de poluição. Este experimento é completo com uma etapa de análise e discussão sobre os resultados.

## 1.2 DESAFIOS

Um dos desafios enfrentados por este trabalho foi a seleção de bases de dados e a extração de métricas dos resultados.

Encontrar bases de dados de diferentes tamanhos e diferentes áreas, e garantir que elas fossem padronizadas para rodar em todos os algoritmos. As bases utilizadas foram extraídas do portal do instituto *Hasso-Plattner* (HPI).

Adicionalmente, a dificuldade de implantar a coleta de métricas mais elaboradas em todos os algoritmos. Métricas como *succinctness, coverage e interestingness* não foram coletadas.

## 1.3 ORGANIZAÇÃO DO DOCUMENTO

O documento está organizado da seguinte forma: Capítulo 2 - Fundamentação teórica sobre as dependências em perfilamento de dados e a natureza dos dados nulos; Capítulo 3 - Prova de conceito realizada com os algoritmos de descobertas de DCs e a plataforma Metanome, além

de desenvolvimentos e condução dos experimentos; Capítulo 4 - Experimentos, validação dos resultados, análises e discussões; Capítulo 5 - Conclusão e trabalhos futuros.

# 2 FUNDAMENTAÇÃO TEÓRICA

Para compreender o campo de estudo de *Data Profiling*, foram estudadas 5 dependências que dão significância às regras de negócio presentes em uma base de dados.

## 2.1 DEPENDÊNCIAS FUNCIONAIS

A Dependência Funcional (FD) é um conceito adotado na área de banco de dados para definir de maneira formal a relação entre dois atributos, ou conjunto de atributos, em uma tabela. Através delas é possível descrever regras de uma base de dados de forma que um conjunto de valores dos atributos $X$ de uma tabela define unicamente um conjunto de atributos $A$, conforme (Papenbrock et al., 2015b).:

$$\text{FD: } X \rightarrow A$$

As Dependências Funcionais desempenham um importante papel na normalização de bancos de dados, contribuindo para a eliminação de redundâncias e a organização eficiente dos dados. A decomposição de tabelas por meio da identificação de FDs permite projetar esquemas mais eficientes e consistentes.

Um exemplo de uma dependência funcional, considerando as relações da Tabela 1.1, seria Renavam → Carro, Ano. Essa dependência funcional sugere que, dada um determinado Renavam, é possível determinar de forma única o modelo do carro e o ano correspondente.

## 2.2 DEPENDÊNCIAS DE COLUNA ÚNICA

Dependências de coluna única (UCCs) são grupos de atributos em conjuntos de dados relacionais que não contêm nenhuma entrada de valor mais de uma vez. (Papenbrock, 2017) Conhecer essas combinações é importante, pois elas revelam potenciais chaves primárias (PK) e ajudam nas tarefas de gerenciamento de dados, como normalização de esquema, otimização de consultas *(queries)*, modelagem de dados e limpeza de dados. Formalmente, uma Combinação Única, pode ser definido como (Abedjan e Naumann, 2023): dado um esquema de banco de dados relacional S = C1, C2,..., Cm com colunas $C_i$ e uma instância $r \subseteq C1 \times ... \times Cm$, uma combinação de colunas $K \subseteq R$ é única, se para todos os $t1, t2$ pertencentes a $r$:

$$(t1 \neq t2) \Rightarrow (t1[K] \neq t2[K])$$

Para ilustrar com um exemplo prático, considere a Tabela 1.1. Neste exemplo, uma possível UCC seria a combinação {Renavam}, pois cada número de Renavam aparece apenas uma vez na tabela, indicando que é potencialmente uma chave primária única.

## 2.3 DEPENDÊNCIAS DE INCLUSÃO

A Dependência de Inclusão (IND) é definida como duas listas de atributos $X$ e $Y$ sobre uma esquema relacional com relações $R_i$, de forma que: $R_j[X] \subseteq R_k[Y]$ - forma curta: $(X \subseteq Y)$. Em outras palavras, é uma declaração que indica que todos os registros $r$ em $R_j[X]$, também está contido em $R_k[Y]$ (Dürsch et al., 2019). O lado esquerdo $X$ de uma IND são os atributos dependentes, enquanto lado direito $Y$ formam os atributos referenciados.

Tanto os atributos dependentes quanto os referenciados, devem ter o mesmo tamanho, ou seja, $n = |X| = |Y|$. No caso de $n = 1$, uma dependência unária é definida, por outro lado, quando $n > 1$ então é uma dependência n-ária.

Um exemplo de uma dependência unária, considerando as relações das Tabelas 1.1 e 1.2 - $R_N$ e $R_V$, respectivamente - seria $R_N\,[Renavam] \subseteq R_V\,[Renavam]$. Já uma dependência n-ária, considera que há mais de uma coluna que pode ser considerada, portanto considere a Tabela 2.1 como a relação $R_{V2}$.

Tabela 2.1: Relação de Valores 2

| Tupla | Renavam | Carro | Valor |
|---|---|---|---|
| t1 | 13260962389 | Gol | 15000 |
| t2 | 13260962389 | Gol | 18000 |
| t3 | 27158493820 | Gol | 20000 |
| t4 | 35698712541 | Mobi | 15000 |
| t5 | 44300268437 | Mobi | 30000 |
| t6 | 51789012345 | Mobi | 20000 |

Dessa forma, uma IND n-ária em pontencial seria $R_N\,[Renavam, Carro] \subseteq R_{V2}\,[Renavam, Carro]$, considerando $n = 2$.

Esse tipo de dependência é utilizada para expressar relacionamento entre tabelas. Um destes relacionamentos é a chave estrangeria (FK). Além disso ela pode ser utilizada para otimização de consultas *(queries)*, checagem de integridade dos dados e projeto de esquemas relacionais.

## 2.4 DEPENDÊNCIAS DE ORDEM

A Dependência de Ordem desempenha um papel importante na modelagem e análise de dados, permitindo a identificação de padrões de ordenação entre os atributos de um banco de dados. Diferentemente das dependências funcionais, que expressam relações entre valores individuais, as dependências de ordem consideram a ordenação global das tuplas com base em determinados atributos (Szlichta et al., 2018).

Formalmente, uma dependência de ordem $X \rightarrow Y$ em um esquema $S$ é satisfeita para uma instância $r$ de $S$ se, ao ordenar as tuplas de $r$ com base nos atributos de $X$, também obtemos um ordenamento dos atributos de $Y$.

Por exemplo, uma dependência de ordem $(\mathrm{A}, \mathrm{B}, \mathrm{C}) \rightarrow (\mathrm{D})$ expressa que, ao ordenarmos as tuplas de uma tabela com base nos atributos A, B e C, obtemos um ordenamento também pelo atributo D.

Para a Tabela 1.1 fornecida, a dependência de ordem Ano $\rightarrow$ Valor é satisfeita para os carros do tipo *Gol*, uma vez que, ao ordenarmos as tuplas com base no atributo Ano, obtemos um ordenamento também pelo atributo Valor.

## 2.5 DEPENDÊNCIAS DE NEGAÇÃO

A DC consiste em uma generalização de uma FD, sendo capaz de representar as regras de negócio, através de desigualdades entre os atributos de uma base de dados. Em (Pena et al., 2021) uma DC segue a definição:

$$\varphi : \forall t, t' \in r, \neg(p_1 \wedge ... \wedge p_n)$$

De forma que $t$ e $t'$ são tuplas de uma relação $r$ em banco de dados, um predicado $p$ no formato $t.A = t'.B$ onde $A$ é atributo de uma relação e o conjunto de operadores $\{=, \neq, >, <, \geq, \leq\}$.

Sendo assim podemos separar as dependências de negação em diferentes categorias baseado nos operadores, para generalizar UCC, FD e OD.

Para definir cada uma das restrições dos operadores de igualdade, diferença e desigualdade, considere o seguinte exemplo na Tabela 1.1. Nela existem quatro atributos: Renavam que representa a chave primária da tabela, Marca do Carro, Ano de lançamento e Valor atual em R$.

Para este conjunto de dados serão definidas três DCs: A primeira generaliza o conceito de chave primária, e a segunda e terceira definem regras de negócios envolvendo 2 e 3 predicados, respectivamente.

### 2.5.1 Atributo único na relação

Para generalizar a regra de chave primária é possível definir que uma chave $A$ de uma tupla $t$, não pode ser igual a chave $A$ de uma outra tupla $t'$, no mesmo conjunto de dados. Ou seja:

$$\varphi : \neg(t.A = t'.A)$$

Dessa forma a regra "Não podem haver tuplas com Renavam iguais", traduzida em uma DC, resulta em $\varphi_1$.

$$\varphi_1 : \neg(t.Renavam = t'.Renavam)$$

Logo, duas tuplas diferentes cuja expressão $t.Renavam = t'.Renavam$ for verdadeira, tornará $\varphi_1$ falsa e a DC não será satisfeita. Isso indica que há uma inconsistência nos dados da tabela.

Na Tabela 1.1 exitem carros diferentes com o mesmo renavam "13260962389". Observe as tuplas $t1$ e $t2$ que ferem esta regra.

### 2.5.2 Atributos diferentes em uma relação

Regras que envolvem atributos diferentes em uma relação, também podem ser generalizadas. Para isso, basta incluir o operador $\neq$ na expressão. Além disso é possível considerar mais predicados na relação. Ou seja:

$$\varphi : \neg(t.A = t'.A \wedge t.B \neq t'.B)$$

Assim, a regra "Não deve existir marcas diferentes e carros iguais" é traduzida como $\varphi_2$.

$$\varphi_2 : \neg(t.Marca \neq t'.Marca \wedge t.Carro = t'.Carro)$$

Analisando a Tabela 1.1, as tuplas $t_4$ e $t_5$ ferem a regra por considerar o carro "Mobi" da marca "Volkswagen" e "Fiat".

### 2.5.3 Desigualdade entre atributos

A última regra demanda que atributos numéricos sejam analisados. Isso permite que o uso dos operadores de desigualdade seja possível. Ou seja, para atributos numéricos todos os operadores em $O_{numrico}$ podem ser utilizados, ao passo que para atributos, apenas os operadores $O_{texto}$ são válidos.

$$O_{numrico} : \{=, \neq, <, >, \leq, \geq\}$$

$$O_{texto} : \{=, \neq\}$$

Com isso a regra “Para carros iguais, o mais novo deve valer mais” pode ser generalizada. Em $\varphi_3$ será necessário o uso de desigualdade para o atributo $Valor$ e $Ano$:

$$\varphi_3 : \neg(t.Carro = t'.Carro \wedge t.Ano < t'Ano \wedge t.Valor > t'.Valor)$$

Assim, caso haja um carro igual a outro, com anos diferentes, sendo que o mais novo vale menos, então isso fere a regra definida por $\varphi_3$. Observe as tuplas $t5$ e $t6$ que ferem a DC, pois o carro “Mobi” do ano de “2015” tem o valor de “30000” enquanto o “Mobi” mais novo do ano “2020” vale “20000” que é um valor menor. A expressão é avaliada da seguinte forma:

$$\varphi_3 : \neg(t.Carro = t'.Carro \wedge t.Ano < t'Ano \wedge t.Valor > t'.Valor)$$

$$\varphi_3 : \neg(\text{“}Mobi\text{”} = \text{“}Mobi\text{”} \wedge 2015 < 2020 \wedge 30000 > 20000)$$

$$\varphi_3 : \neg(Verdadeiro \wedge Verdadeiro \wedge Verdadeiro)$$

$$\varphi_3 : \neg(Verdadeiro)$$

$$\varphi_3 : Falso$$

Para efeitos de exemplo, não é considerado o estado físico do carro, apenas a sua idade é suficiente para compreensão das diferentes aplicações de DC’s.

### 2.5.4 Dependência de Negação Aproximada

As dependências de negação aproximadas, também conhecidas como DCs aproximadas, são aquelas que permitem uma certa flexibilidade e tolerância para um número limitado de violações. Em configurações onde os dados podem apresentar inconsistências, as DCs aproximadas são projetadas para admitir um grau de inconsistência, mas ainda manter sua validade para a maioria dos dados. Essa abordagem tem por objetivo encontrar um equilíbrio entre a rigidez das restrições exatas e a realidade dos dados imperfeitos, proporcionando uma maior capacidade de lidar com situações reais (Pena et al., 2019).

### 2.5.5 Dependência de Negação Exata

As dependências de negação exatas, também chamadas de DCs exatas, são aquelas que requerem que todas as restrições sejam totalmente satisfeitas, sem exceções. Em configurações ideais, as DCs exatas são aplicadas para garantir a consistência absoluta dos dados. Essas restrições não admitem nenhuma violação e exigem que todas as condições sejam estritamente cumpridas, proporcionando uma abordagem rígida para a validação dos dados (Pena et al., 2019).

## 2.6 APLICAÇÃO DE DEPENDÊNCIAS DE NEGAÇÃO

Existem três etapas essenciais para incorporar a utilização das DCs dentro de um processo de limpeza de dados. O primeiro passo é descobrir quais são as dependências de um determinado conjunto de dados (Pena et al., 2019, 2022; Bleifuß et al., 2017). O segundo é,

a partir das regras descobertas, varrer os dados para encontrar as violações delas (Pena et al., 2020, 2021). Por fim a etapa de correção ou limpeza dos dados que violam as regras (Chu et al., 2013b).

### 2.6.1 Descobertas de dependências de negação

As descobertas destas dependências de negação podem ser feitas de forma manual, assim como no exemplo. Assim, é tomado como ponto de partida uma regra de negócio, que é transformada em uma restrição.

O contrário também é possível, ou seja, partir de uma base de dados, encontrar as DC's e então definir quais destas restrições são de fato regras de negócio. No entanto, é necessário levar em consideração que os dados podem estar sujos e por causa disso o resultado da descoberta pode não ser exato.

Para facilitar esta descoberta, o DC Finder (Pena et al., 2019) foi proposto. Um algoritmo que pode ser utilizado para encontrar DCs de forma eficiente e que leva em consideração a descoberta exata e aproximada. Esse algoritmo é detalhado na seção 2.11.2.

Além deste algoritmo, existe o Hydra (Bleifuß et al., 2017) que incorpora outras otimizações ao construir o espaço de evidências, detalhado na seção 2.11.3.

Ambos estes algoritmos descobrem as DCs de maneira diferentes, no entanto, tanto o DC Finder quanto Hydra aplicam a mesma descoberta de espaço de predicados desenvolvido por (Chu et al., 2013a). Esse processo chave é detalhado na seção 2.11.1.1.

### 2.6.2 Descobertas de violações

A descoberta de violações, por outro lado, não está atrelada a descoberta de restrições, mas sim em encontrar o conjunto de dados que viola elas.

Essa etapa de limpeza de dados é custosa. Um dos aspectos que deixa ela demorada é a quantidade de dados que devem ser encontrados e quantidade de DCs que restringem a base de dados. A solução proposta (Chu et al., 2013b) utiliza um sistema holístico, que faz a busca por completo, e monta um hiper-grafo de violações, Figura 2.1.

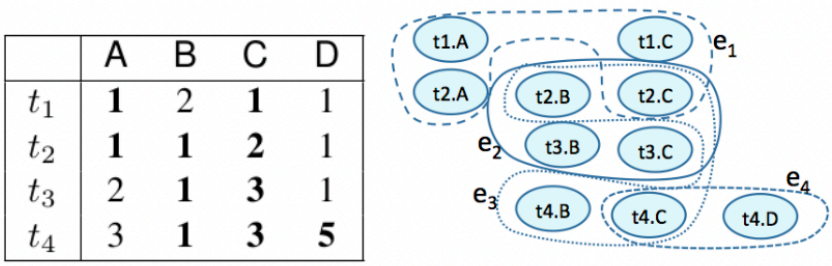



Figura 2.1: Hiper-grafo de violações (Chu et al., 2013b)

Neste grafo os nodos são células que violam a regra e as arestas representam a ligação entre dois nodos que violam a mesma regra.

Exitem outras alternativas que também lidam com a detecção como o *Viofinder* (Pena et al., 2020) e FACET (Pena et al., 2021).

### 2.6.3 Correção de dados

Em (Chu et al., 2013b), também é explicado como que as violações podem ser corrigidas. Para isso é feito um mapeamento das violações no hiper-grafo Figura 2.1 (a) e em seguida é aplicado uma algoritmo para descobrir o conjunto de vértices de cobertura mínima, ou seja, o conjunto mínimo de vértices que contém todas as arestas do grafo. Para o exemplo da Figura 2.1 (b) os vértices $t_2[C]$ e $t_4[C]$.

Por fim é aplicado um um algoritmo de reparação holística. Este algoritmo leva em consideração as DCs atreladas a cada atributo para reparar aquela célula, a este espaço é dado o nome de contexto de reparação. Esse contexto é repetido até que toda a correção de dados seja realizada.

## 2.7 ALGORITMOS DE DESCOBERTA DE DEPENDÊNCIAS FUNCIONAIS

Devido à importância das dependências funcionais, foram propostos diversos algoritmos de descoberta existentes e muitos deles não conseguem lidar eficientemente com conjuntos de dados do mundo real quando o número de registros e de colunas existentes é elevado. A limitação surge devido à otimização desses algoritmos para lidar com muitos registros ou muitos atributos, o que os torna inadequados para conjuntos de dados maiores onde a necessidade de dependências funcionais é mais crítica (Papenbrock et al., 2015b).

Para lidar com esses *datasets* maiores, o algoritmo HyFD se destaca. O funcionamento do algoritmo pode se resumir em algumas etapas:

**Passo 1: Extração e Cálculo Inicial**

O desempenho desta fase é otimizado para eficiência em termos de colunas.

1. Definição de um conjunto de dados de entrada como $D$.
2. Extração de um subconjunto $S$ de registros não aleatórios de $D$.
3. Cálculo das Dependências Funcionais de $S$, denotadas por $FD(S)$.

**Passo 2: Validação e Aprimoramento**

Essa fase é otimizada para eficiência em termos de linhas, utilizando as FDs descobertas anteriormente para podar o espaço de busca.

1. Definição de $FD(D)$ como o conjunto de FDs no conjunto de dados completo $D$.
2. Validação das FDs descobertas em $S$ em relação a $D$, denotadas por $FD(S) \subseteq FD(D)$.
3. Aprimoramento das FDs em $S$ que ainda não são válidas no conjunto de dados completo.

Se a validação no item 2 se tornar ineficiente ou incompleta o algoritmo é capaz de alternar de volta para a primeira fase. Se não, continua a partir dos resultados descobertos até o momento.

O algoritmo HyFD lida com valores nulos de duas maneiras: $null = null$ e $null \neq null$. Essa escolha pode afetar a avaliação de dependências funcionais (FD). Nos experimentos lidos, optam por $null = null$ para manter consistência com abordagens de trabalhos relacionados. O algoritmo suporta ambas as configurações, permitindo a alternância entre elas durante o pré-processamento e amostragem com um parâmetro específico.

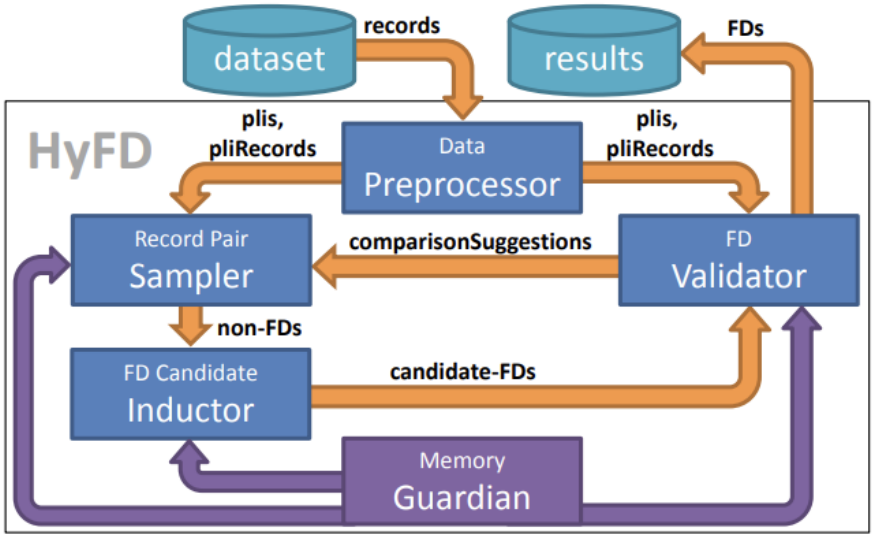



**Figure 2: Overview of HyFD and its components.**

Figura 2.2: Arquitetura do algoritmo HyFD (Papenbrock et al., 2015b)

## 2.8 ALGORITMOS DE DESCOBERTA DE DEPENDÊNCIAS DE COLUNA ÚNICA

A descoberta de UCCs é muito semelhante ao problema de descoberta de FDs mencionado no item anterior, já que os únicos determinam funcionalmente todas as outras colunas individuais dentro de uma tabela.

A descoberta de UCCs é uma tarefa computacional cara. Por esta razão, existe uma limitação dos algoritmos conhecidos, executando apenas para pequenos conjuntos de dados. Um algoritmo de descoberta de UCC chamado HyUCC superou esse problema, sendo agora capaz de processar com eficiência conjuntos de dados que são muito maiores, conforme descrito em (Papenbrock, 2017).

O HyUCC é o algoritmo irmão do HyFD, que foi modificado em certos componentes selecionados para encontrar UCCs em vez de FDs.

O HyUCC difere-se na descoberta de combinações únicas de colunas, em vez de dependências funcionais, em 3 aspectos: utilização de uma estrutura de árvore de prefixos (trie) para armazenar as UCCs, uma validação customizada das UCCs e regras específicas para a eliminação dessas dependências. (Papenbrock, 2017)

A mesma forma de lidar com os dados nulos explicada na seção 2.7 é feita no HyUCC.

## 2.9 ALGORITMOS DE DESCOBERTA DE DEPENDÊNCIA DE ORDEM

A descoberta de dependência de ordem, pode se tornar complexa para grandes *datasets*, devido ao grande espaço de busca e características internas desse tipo de dependência. A partir disso, o algoritmo ORDER é apresentado com uma abordagem eficiente para descobrir todas as dependências lexicográficas de ordem $n$-árias sob o operador '$<$' e todas as dependências de ordem '$\leq$' em um conjunto de dados (Langer e Naumann, 2016).

Seu funcionamento ocorre ao percorrer a treliça - uma estrutura que representa o conjunto de todos os candidatos existentes - de todas as possíveis permutações de atributos de maneira nivelada, da base para o topo. Utilizando a estratégia a priori, como explicado em (Langer e Naumann, 2016), o algoritmo gera candidatos de tamanho $l$ a partir de candidatos

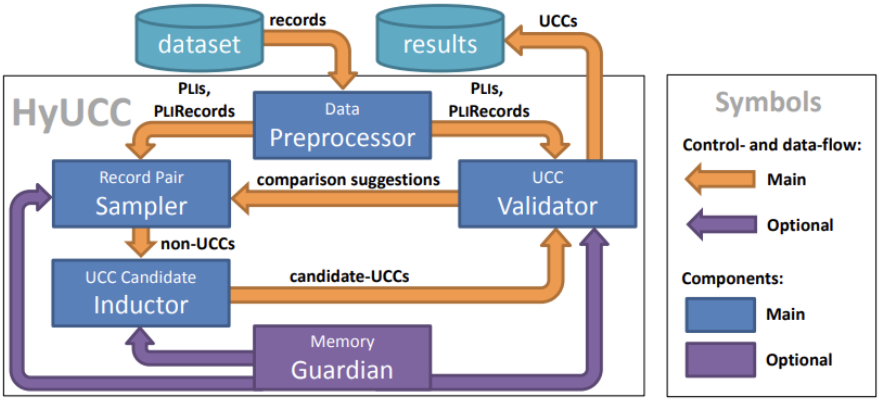



Fig. 2: Overview of HyUCC and its components.

Figura 2.3: Arquitetura do algoritmo HyUCC (Papenbrock, 2017)

de tamanho $l - 1$ e verifica a validade desses candidatos para o nível atual. O conhecimento adquirido é então utilizado para podar o espaço de busca através de regras antes de gerar o próximo nível.

Quando não há mais candidatos restantes, o algoritmo termina, retornando todas as dependências de ordem válidas na tabela.

## 2.10 ALGORITMOS DE DESCOBERTA DE DEPENDÊNCIAS DE INCLUSÃO

A descoberta de IND envolve a avaliação de dependências entre diferentes Tabelas 2.3. Para realizar este trabalho existem algoritmos com diferentes estratégias, performance e limitações (Dürsch et al., 2019). Um algoritmo que se destaca pela sua performance e pelo seu propósito genérico e o BINDER.

O algoritmo BINDER é uma abordagem de divisão e conquista para a descoberta de dependência de inclusão (Papenbrock et al., 2015c). O produto deste algoritmo é uma lista de atributos onde cada item da lista possui um lado esquerdo da relação (atributos dependentes) e lado direito (atributos referenciados). Este algoritmo é capaz de detectar tanto dependências unárias quanto n-árias. Uma dependência unária considera relação de 1 dependente com 1 referenciado, enquanto uma dependência n-ária considera $n$ dependentes com $n$ referenciados.

Uma característica importante é que o próprio algoritmo tem uma estratégia frente aos dados nulos, que é de ignorá-los, uma vez que estes não são relevantes para a descoberta de IND.

## 2.11 ALGORITMOS DE DESCOBERTAS DE DEPENDÊNCIAS DE NEGAÇÃO

Na descoberta de DCs foram estudados 3 algoritmos Hydra (Bleifuß et al., 2017), DcFinder (Pena et al., 2019), e ECP (Pena et al., 2022). O Hydra possui a descoberta de negação exata, enquanto DCFinder incorporta a descoberta aproximada. Além disso, o ECP é um algoritmo focado em melhorar o desempenho na etapa de construção de espaço de evidências.

### 2.11.1 Estrutura geral de um algoritmo de descobertas de DC

Um algoritmo de descoberta de DCs possui 3 (Pena et al., 2022) etapas em sua estrutura. A primeira envolve a construção do espaço de predicados. Esta serve como base para construção do espaço de evidências. E a última é a enumeração de DCs.

#### 2.11.1.1 Construção de espaço de predicados

Uma DC, é composta por um conjunto de predicados. Sendo assim, a construção do espaço de predicados é a primeira etapa para derivar DCs candidatas. Para essa construção ocorrer, são considerados 2 tipos de colunas em uma relação: categóricas e numéricas. As colunas categóricas podem derivar predicados com o conjunto de operadores $\{=, \neq\}$, enquanto as colunas numéricas, considera o conjunto $\{=, \neq >, <, \geq, \leq\}$.

Um exemplo em 2.4 mostra como essa definição ocorre para as colunas *I(string)*, *M(string)* e *S(Double)*.

**EXAMPLE** 7. *Consider the following Employee table with three attributes: Employee ID (*I*), Manager ID (*M*), and Salary(*S*).*

| *TID* | *I(String)* | *M(String)* | *S(Double)* |
|---|---|---|---|
| $t_9$ | *A1* | *A1* | *50* |
| $t_{10}$ | *A2* | *A1* | *40* |
| $t_{11}$ | *A3* | *A1* | *40* |

*We build the following predicate space **P** for it.*

| | | |
|---|---|---|
| $P_1 : t_\alpha.I = t_\beta.I$ | $P_5 : t_\alpha.S = t_\beta.S$ | $P_9 : t_\alpha.S < t_\beta.S$ |
| $P_2 : t_\alpha.I \neq t_\beta.I$ | $P_6 : t_\alpha.S \neq t_\beta.S$ | $P_{10} : t_\alpha.S \geq t_\beta.S$ |
| $P_3 : t_\alpha.M = t_\beta.M$ | $P_7 : t_\alpha.S > t_\beta.S$ | $P_{11} : t_\alpha.I = t_\alpha.M$ |
| $P_4 : t_\alpha.M \neq t_\beta.M$ | $P_8 : t_\alpha.S \leq t_\beta.S$ | $P_{12} : t_\alpha.I \neq t_\alpha.M$ |
| $P_{13} : t_\alpha.I = t_\beta.M$ | $P_{14} : t_\alpha.I \neq t_\beta.M$ | |

Figura 2.4: Espaço de predicados (Chu et al., 2013b)

Aumentar esse espaço de predicados, significa aumentar a quantidade de DCs em potencial, e por consequência o custo computacional para processar a descoberta. Dessa forma, diminuir este espaço pode reduzir o tempo de processamento.

A partir deste espaço de predicados é construído um espaço de evidencias.

#### 2.11.1.2 Construção do conjunto de evidências

Um conjunto de evidências é composto por pedaços de evidências $e$. Um pedaço de evidências é um subconjunto do espaço de predicados $P$ satisfeito por duas tuplas $t$ e $t'$. Sendo assim, o espaço de evidências completo $E_r$ é o conjunto dos pedaços de evidências de todas as tuplas da relação $r$, dado o espaço de predicados $P$.

É neste ponto que métricas são coletadas para a descoberta de DCs aproximadas, uma vez que é necessário tolerar algumas tuplas que não satisfazem a regra e portanto, não fazem parte do conjunto de evidências.

Outro aspecto que pode impactar a construção do conjunto de evidências é a existência de dados faltantes. A presença de dados faltantes impacta diretamente na satisfação de duas tuplas para um conjunto de predicados, ou seja, o pedaço de evidência. Podendo tanto deixar de satisfazer pedaços de evidências, ou então passar a satisfazer novos predicados.

#### 2.11.1.3 Enumeração de DCs

A Enumeração de DCs é a última das 3 etapas. Esta etapa consiste em verificar o espaço de predicados, para verificar se há pedaços de evidências faltantes, em relação à $r$. A forma mais

eficiente de verificar esse espaço ainda é um campo de estudo em aberto, mas há relação com a busca em hiper-grafos (Chu et al., 2013b).

### 2.11.2 Algoritmo DC Finder

O artigo "*Discovery of Approximate (and Exact) Denial Constraints*" (Pena et al., 2019), descreve uma técnica para descoberta de DCs em conjuntos de dados.

A descoberta de DCs é complexa, pois cada DC é expressa como um conjunto de predicados em vez de um conjunto de atributos, ou seja, quanto mais colunas em uma tabela, mais serão os predicados que devem ser considerados. Afinal, um atributo texto pode ter comparações com $\{=, \neq\}$ e atributos numéricos com $\{=, \neq, <, >, \leq, \geq\}$. Isso é perceptível nos exemplos gerados na seção 2.5.

O DcFinder é o algoritmo para a descoberta de DCs exatas e aproximadas. A primeira são descobertas usando bases de dados ideais, ou seja, sem inconsistências - erros ou dados nulos. A aproximada são descobertas considerando que a bases de dados possui inconsistências e, para isso, o algoritmo é relaxado, ou seja, tolera inconsistências nos dados com o objetivo de facilitar a descoberta de DCs.

A arquitetura geral do algoritmo, na Figura 2.5, consiste lista de posições (PLIs), espaço de predicados, conjunto de evidências, busca de DC's, listagem de DC's relevantes.

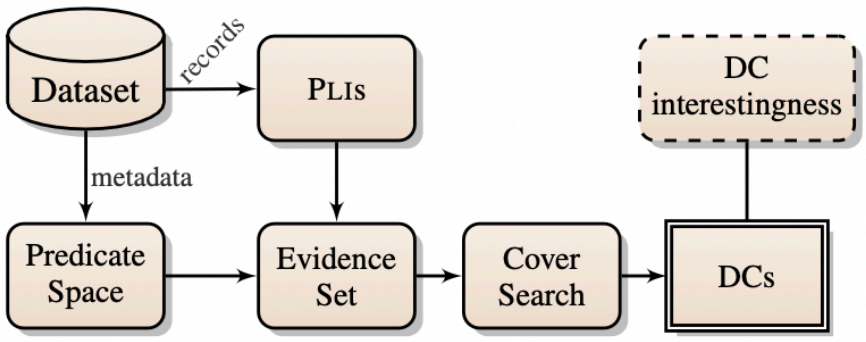



**Figure 1: Building blocks of DCFINDER.**

Figura 2.5: Arquitetura do algoritmo DcFinder (Pena et al., 2019)

A primeira tarefa realizada é a criação da lista de predicados com base nos dados fornecidos. Em seguida, ele constrói estruturas de dados chamadas índices de lista de posições (PLIs) para os registros no conjunto de dados. Os predicados com baixa seletividade, ou seja, aqueles que são satisfeitos por muitos pares de tuplas, são agrupados em um conjunto improvável.

O algoritmo presume que uma evidência satisfaz os predicados menos seletivos e em seguida aloca matrizes de evidências para cada elemento. Estes elementos contém o conjunto de predicados "provavelmente satisfeito". Para isso o DcFinder usa PLIs para encontrar referências a pares de tuplas que não são prováveis de satisfazer os predicados.

Ele executa operações lógicas para cada uma dessas referências para trazer as evidências para um estado consistente. No artigo, essa reconstrução de evidências consiste em inicializar um *array B*, onde cada *index* representa a evidência de um par de tuplas. Depois algoritmos de detecção de inconsistências e refinamentos são aplicados para deixar esse *array* em um estado final.

Por fim os candidatos são encontrados resolvendo o problema de encontrar o conjunto minimal *covers*, no espaço de evidências.

### 2.11.3 Algoritmo Hydra

O algoritmo Hydra é uma solução que tenta melhorar a performance de uma DC eliminando redundâncias nas fases iniciais do processamento. Essa redundâncias ocorre entre par de tuplas e no espaço de predicados (Bleifuß et al., 2017). Hydra otimiza este processo para obter maior performance.

Hydra recebe como entrada uma tabela e um conjunto de predicados. A partir disso, as seguintes etapas no processamento começam, Figura 2.6.

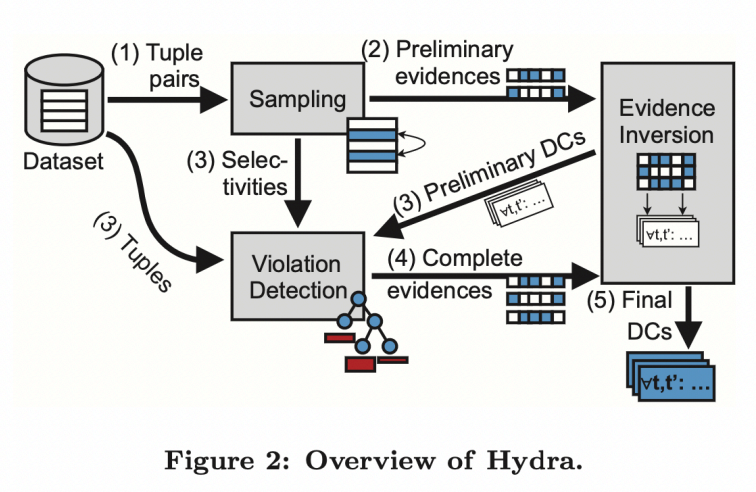



Figura 2.6: Arquitetura do algoritmo Hydra (Bleifuß et al., 2017)

- *Dataset*. A entrada do algoritmo é uma base de dados e um espaço de predicados.
- *Sampling*. Consiste em arranjar uma amostra de dados da tabela, descobrindo um conjunto de evidencias (violações de DCs) preliminares sobre os dados.
- *Evidence Inversion*. Depois da descoberta preliminar, no passo 2, um algoritmo de inversão de evidências é aplicado. Depois esse resultado é enviado novamente para redescobrir as violações.
- *Violation Detection*. Por fim, ao receber o conjunto preliminar de DCs, o algoritmo monta um conjunto completo de evidências levando em consideração as tuplas que não foram consideradas na descoberta preliminar.

Dessa forma o algoritmo Hydra consegue obter uma performance que supera a complexidade quadrática do estado da arte de descoberta de violações. Por outro lado, isso é verdade apenas para dados exatos, ou seja, não é válido para DCs aproximadas como o DC Finder.

### 2.11.4 Algoritmo ECP

O algoritmo ECP *(Evidence Context Pipeline)* está relacionado diretamente com a construção do espaço de evidências. Ao contrário de algoritmos como DcFinder e Hydra, que

possuem um tempo quadrático para o processamento, o ECP pode processar as evidências de forma simultânea, reduzindo a memória e tempo de processamento necessários.

Outro ponto a ser considerado é que a presença do nulo no ECP possui uma tratativa (Pena et al., 2022). No caso de variáveis categóricas, um nulo é tratado como uma *string* vazia, já uma variável numérica nula é tratada como $-\infty$. Essa abordagem pode ter diferentes influências dependendo da natureza do dado imputado.

## 2.12 A PLATAFORMA METANOME

Os algoritmos explicados anteriormente estão disponíveis na plataforma Metanome (Papenbrock et al., 2015a). Este é o software que gerência as bases de dados e executa os algoritmos. Além destes dois descritos na seção 2.11 existem outros algoritmos para outros contextos de perfilamento de dados. Estes estão disponíveis em (Metanome, 2018).

A plataforma Metanome consiste em uma interface web *Frontend* e o servidor de dados *Backend*, Figura 2.7.

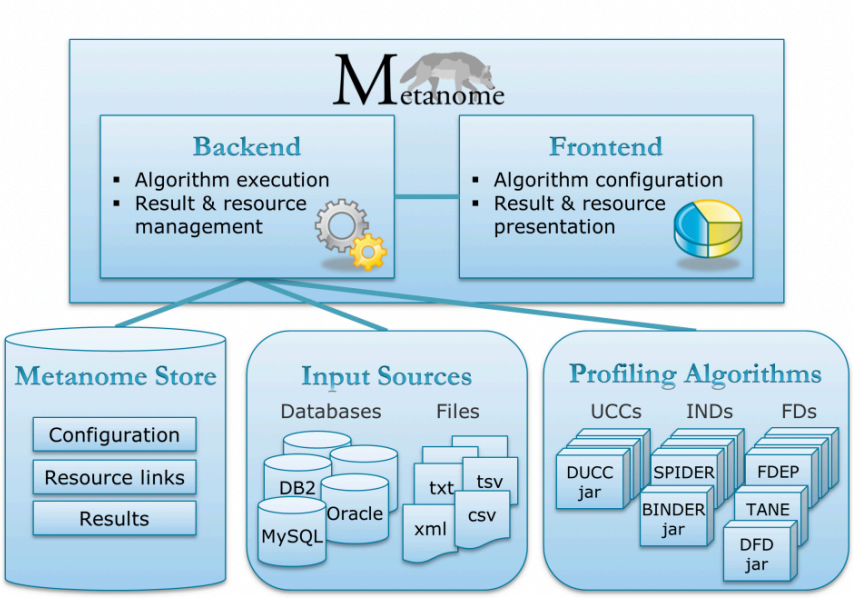



Figure 2: Architecture of the Metanome profiling platform.

Figura 2.7: Diagrama da arquitetura do Metanome (Papenbrock et al., 2015a)

A interface web gerencia a configuração dos algoritmos e disponibiliza uma visualização de dados e listagem de DCs, dado um algoritmo e uma base de dados de entrada. Já o servidor é responsável por armazenar os dados de configuração de entrada do *Frontend*, as bases de dados e os algoritmos.

As tabelas de dados e algoritmos de experimento neste documento foram testados seguindo estes passos.

1. inclusão dos algoritmos ($.jar$) no Metanome.
2. inclusão das bases de dados ($.csv$).
3. aplicação dos algoritmos nas bases de dados.

Os experimentos e seus objetivos foram documentados no capítulo 3 e seus resultados estão no apêndice A.

## 2.13 O METANOME CLI

Além da interface detalhada na seção "A plataforma Metanome", há uma outra maneira de executar os algoritmos. A forma é por meio da ferramenta Metanome CLI, utilizando a linha de comando (CLI, 2017). Essa abordagem oferece uma maior facilidade na automação de tarefas e na criação de experimentos de *data profiling* em um ou mais conjuntos de dados. Ao integrar esse método com outros códigos, torna-se possível realizar uma análise completa de dados de forma eficiente.

## 2.14 A PRESENÇA DE DADOS NULOS EM LIMPEZA DE DADOS

Ao analisar dados, é imprescindível considerar a possibilidade de dados nulos. Inclusive, existe a diferença entre dados nulos e dados faltantes. Em (Osborne, 2013) é explicado sobre MAR, MNAR e MCAR, que são três diferentes formas de considerar os dados nulos.

A presença de dados nulos também afeta a descoberta de dependências funcionais. Em (Berti-Équille et al., 2018) é feita uma análise sobre esse impacto, utilizando diferentes estratégias para tratar os dados nulos antes de detectar as dependências funcionais.

### 2.14.1 Mecanismos de dados nulos

A existência de dados nulos tem diferentes fontes. É possível considerar que foi uma falha humana, por esquecimento. Da mesma forma que é possível considerar uma falha no software, causando a inserção de dados nulos. Ou ainda, pode ser que o dado em questão de fato não exista, por exemplo, a data de casamento de um usuário solteiro, (Osborne, 2013). A questão é que os dados podem ser imputados de forma arbitrária ou proposital.

Em (Osborne, 2013) explica três mecanismos de imputação de dados nulos, são eles: *Missing at Random* (MAR), *Missing not at Random* (MNAR) e *Missing Completely at Random* (MCAR).

#### *2.14.1.1 MCAR*

Esse mecanismo mostra que dados faltantes são aleatórios e que nenhuma outra variável impacta nestes dados nulos.

Um exemplos seria: considere que uma disciplina $X$ aplica três provas no semestre $\{p_1, p_2, p_3\}$. Se o aluno $A$ faltou a primeira prova, mas fez as outras duas, sua grade teria o seguinte resultado $\{null, 10, 7\}$. E no semestre seguinte, vamos supor que perdeu a segunda, $\{6, null, 8\}$. Note que não há relação entre os dados nulos e um não implica no outro.

#### *2.14.1.2 MAR*

Similar ao MCAR, o MAR considera que os dados faltantes são aleatórios e não impactam no resultado das análises. No entanto, considera-se que a ausência de dados se dá por algum motivo anterior. Nesse caso, o dado nulo também não é relevante.

No mesmo exemplo do aluno, podemos considerar que a perda da $p_1$ em $\{p_1, p_2, p_3\}$ implica em não poder fazer o restante do semestre, logo $\{null, null, null\}$. Note que não é completamente aleatório, apenas o primeiro resultado, enquanto os outros são premeditados.

#### *2.14.1.3 MNAR*

Em MNAR, isso é diferente. Neste caso, a falta de um dado é relevante, pois sua perda pode impactar nos resultados finais.

Por exemplo: aplicar um formulário de *feedback* para alunos da matéria X ao final do semestre. Nesse caso, os alunos que tiveram resultados positivos no semestre podem se sentir motivados para responder o *feedback*, ao passo que os que tiveram resultados negativos podem se sentir desmotivados para preencher o formulário. Portanto, a nota do semestre pode influenciar na falta de dados, ou seja, não foram perdidos de maneira arbitrária.

### 2.14.2 Estratégia para análise de dados nulos em dependências funcionais

Em (Berti-Équille et al., 2018) é feita uma análise de impacto de dados nulos na descoberta de dependências funcionais. Para isso são aplicadas 4 estratégias: Ignorar dados nulos, Semântica de dados nulos, FDs aproximadas e Imputação de dados. Ao final do artigo são avaliados quais os impactos de cada estratégia avaliando métrica de genuinidade. Para analisar a qualidade na descobertas de DCs, existem outras métricas similares descritas em 2.15.

#### *2.14.2.1 Ignorar dados nulos*

Este procedimento ignora uma tupla caso algum de seus atributos seja nulo. No entanto há dois pontos negativos: primeiro, a base de dados pode ter muitos dados nulos; segundo, a base de dados sem os nulos, podem induzir a descoberta de FDs não existentes no algoritmo original.

#### *2.14.2.2 Semântica de dados nulos*

Outra alternativa considera a criação de uma semântica para dados nulos. Isto é, é possível considerar os operadores $ON = \{ NULL\text{-}EQ , NULL\text{-}NOT\text{-}EQ \}$ para compara dados nulos. Esse modelo implica em considerar que os dados, apesar de nulos, poderiam ser iguais ou diferentes, caso não fosse nulos.

#### *2.14.2.3 FDs aproximadas*

A estratégia seguinte utiliza FDs aproximadas, ou seja, considera um grau de tolerância para FDs inválidas, ou seja, permite que algumas tuplas violem uma FD, sem que ela seja descartada do conjunto de possíveis FDs válidas.

#### *2.14.2.4 Imputação de dados*

Por fim a imputação de dados considera a correção dos dados utilizando ferramentas probabilísticas e estatísticas, para a criação de dados que substituam os nulos.

#### *2.14.2.5 Biblioteca para inserção de nulos*

Para a inserção de dados nulos, foi utilizado a biblioteca do Python Jenga (Jenga, 2023). Essa biblioteca permite a inclusão de dados nulos, utilizando os diferentes mecanismos (MCAR, MNAR e MAR) para um *dataframe* de entrada. Além disso permite o controle da porcentagem de dados nulos.

## 2.15 MÉTRICAS DE AVALIAÇÃO DE DEPENDÊNCIAS DE NEGAÇÃO

Embora os algoritmos de descoberta de DCs sejam capazes de eliminar DCs triviais, não mínimas e implícitas, ainda podemos nos deparar com um número excessivo de DCs retornadas. Para enfrentar esse problema, é necessário um método de classificação para identificar as DCs mais relevantes e significativas. Neste contexto, foi proposto um esquema de ranking de DCs baseado em três métricas: Succinctness, Coverage e Interestingness (Chu et al., 2013a).

### 2.15.1 Succinctness

A métrica Succinctness mede o quão concisa é uma DC, ou seja, o quão bem ela pode ser descrita com o menor número possível de restrições. Essa métrica é motivada pelo princípio da navalha de Occam, que sugere que, entre várias explicações possíveis, a mais simples é a mais provável de ser correta (Chu et al., 2013a). O objetivo é evitar a complexidade excessiva e o *overfitting* do modelo.

Para calcular a Succinctness de uma DC $\varphi$, é utilizado a seguinte fórmula:

$$\mathrm{Succ}(\varphi) = \frac{\mathrm{Min}(\{\mathrm{Len}(\alpha)|\forall\alpha\})}{\mathrm{Len}(\varphi)}$$

### 2.15.2 Coverage

A Coverage é uma métrica onde é atuante na mineração de dados e as funções de pontuação são projetadas para medir a significância estatística dos dados. Para calcular essa métrica, é necessário determinar quantas vezes a DC é verdadeira ou válida nos dados de entrada. Isso reflete a proporção de instâncias ou transações que são abrangidas pela DC.

Para calcular a Coverage de uma DC $\varphi$, utilizamos a seguinte fórmula:

$$\mathrm{Coverage}(\varphi) = \frac{\sum_{k=0}^{\mathrm{Len}(\varphi)-1} |k\mathrm{E}| \cdot w(k)}{\sum_{k=0}^{\mathrm{Len}(\varphi)-1} |k\mathrm{E}|}$$

onde $|k\mathrm{E}|$ representa o número de evidências de cobertura de $k$ restrições satisfeitas por pares de instâncias e $w(k)$ é um peso que reflete a importância relativa dessas evidências.

### 2.15.3 Interestingness

A métrica Interestingness é uma pontuação que combina as métricas de Succinctness e Coverage para fornecer uma medida geral de interesse ou relevância de uma DC. A ideia é que DCs que sejam concisas e tenham alta cobertura sejam consideradas mais interessantes.

A pontuação de Interestingness, denotada como Inter($\varphi$), é calculada como uma combinação linear ponderada das métricas de Succinctness e Coverage:

$$Inter(\varphi) = a \cdot Coverage(\varphi) + (1 - a) \cdot Succ(\varphi) \tag{2.1}$$

onde $a$ é um parâmetro que controla o peso entre as duas métricas. O parâmetro $a$ controla o peso atribuído a cada métrica na pontuação final de Interestingness. Se $a$ for definido como 0, a métrica Coverage é desconsiderada, atribuindo peso total à métrica de Succinctness. Por outro lado, se $a$ for definido como 1, a métrica Succinctness é desconsiderada, atribuindo peso total à métrica de Coverage. Quando $a$ é definido como 0.5, as duas métricas têm pesos

iguais, refletindo uma consideração equilibrada entre concisão e cobertura na pontuação de Interestingness.

### 2.15.3 Cálculo das Métricas em uma DC

Para ilustrar a aplicação das métricas de Succinctness, Coverage e Interestingness, consideramos a seguinte DC:

$$\varphi : \neg(t.Carro = t'.Carro \wedge t.Ano < t'Ano \wedge t.Valor > t'.Valor)$$

Agora, realizaremos o cálculo das métricas para esta DC.

#### *2.15.3.1 Succinctness*

Para calcular a métrica de Succinctness, a primeira etapa é calcular Len($\varphi$).

Para o comprimento da DC fornecida, vamos considerar o alfabeto formado pelos seguintes elementos:

$$A = [, (, ), =, , <, >, ., t, Carro, t', Ano, Valor]$$

Considerando que:

$$Len(\varphi) = |a|a \in A, a \in \varphi|$$

Temos o seguinte conjunto:

$$|t, Carro, =, t', Ano, <, Valor, > |$$

Portanto,

$$Len(\varphi) = 8 \tag{2.2}$$

Agora, vamos calcular comprimento mínimo das restrições individuais que compõem a DC:

Levando em consideração que temos 3 predicados existentes na DC e que

$$\begin{gathered} Len(t.Carro = t'.Carro) = 4 \\ Len(t.Ano < t'.Ano) = 4 \\ Len(t.Valor > t'.Valor) = 4 \end{gathered} \tag{2.3}$$

Então:

$$\text{Succ}(\varphi) = \frac{4}{8} = 0,5$$

#### *2.15.3.2 Coverage*

A primeira etapa de cálculo da métrica de Coverage, é calcular a cobertura das variáveis individualmente na base de dados alvo. Vamos usar a Tabela 2.1, onde possui 6 tuplas.

- Variável t.Carro:

  A restrição t.Carro = t'.Carro é satisfeita pelos pares t1 e t2 (com Renavam 13260962389 e Carro Gol) e t4 e t5 (com Renavam 35698712541 e Carro Mobi).

  Portanto,

$$|t.CarroE| = 2$$

- Variável t.ano:

  A restrição t.ano < t'.ano é satisfeita pelos pares t1 e t2 (com Ano 2005 e 2007) e t4 e t5 (com Ano 2012 e 2015).

  Portanto,

$$|t.anoE| = 2$$

- Variável t.valor:

  A restrição t.valor > t'.valor é satisfeita pelos pares t1 e t3 (com Valor 15000 e 20000) e t4 e t5 (com Valor 15000 e 30000).

  Portanto,

$$|t.valorE| = 2$$

  Para definirmos o peso (w(k)) para a DC fornecida, levando em conta o conjunto de dados é necessário:

  – Determinar o número de tuplas que satisfazem a DC:

$$t1 : (Gol = Gol \wedge 2005 < 2007 \wedge 15000 > 18000)$$
$$t2 : (Gol = Gol \wedge 2007 < 2010 \wedge 18000 > 20000)$$

    Portanto, temos **2 exemplos** positivos cobertos.

  – Determinar o número total de tuplas no conjunto de dados. Verificando a Tabela 2.1, vemos que ela possui **6 tuplas**.

  – E assim, calcular o peso w(k):

$$w(k) = 2/6 = 0,333$$

    Substituindo na fórmula final:

$$\text{Coverage}(\varphi) = \frac{1 * 0,333 + 1 * 0,333 + 1 * 0,333}{1 + 1 + 1} = 0,333$$

#### *2.15.3.3 Interestingness*

Utilizando os valores obtidos das outras métricas e usando a fórmula de Interestingness definida em (Chu et al., 2013a), obtemos como resultado:

$$Inter(\varphi) = a \cdot Coverage(\varphi) + (1 - a) \cdot Succ(\varphi)$$

$$Inter(\varphi) = a \cdot 0,333 + (1 - a) \cdot 0,5$$

$$Inter(\varphi) = 0,5 \cdot 0,333 + (1 - 0,5) \cdot 0,5$$

$$Inter(\varphi) = 0,4165$$

A métrica de *Interestingness* (Inter) para a descrição $\varphi$ obteve um valor de 0,4165. Essa pontuação é uma medida geral de relevância da dependência de negação $\varphi$ com base nas métricas de *Succinctness* e *Coverage*.

O resultado indica que a dependência $\varphi$ possui um nível moderado de interesse, com um equilíbrio entre concisão e cobertura, sugerindo que a dependência é relevante e pode ser considerada interessante para a análise dos dados.

## 2.16 MÉTRICAS DE AVALIAÇÃO DE DEPENDÊNCIAS FUNCIONAIS

Dependências Funcionais são fundamentais na modelagem e organização de dados em bancos de dados relacionais. Ao realizar a descoberta de FDs, é comum deparar-se com um conjunto de relações, algumas das quais podem ser triviais, não mínimas ou implícitas.

Na busca por FDs relevantes, é necessário avaliar a confiabilidade dessas descobertas. Nesse contexto, destaca-se uma métrica importante: a **genuinidade**. Essa métrica visa classificar as FDs, distinguindo entre aquelas que são verdadeiras e úteis, e aquelas que podem ser consideradas menos confiáveis (Berti-Équille et al., 2018)

Ao aplicar uma métrica de genuinidade para cada FD descoberta, podemos selecionar apenas as FDs consideradas genuínas ou muito prováveis de serem genuínas para aplicação em situações práticas. A utilização dessas métricas de avaliação contribui para uma abordagem mais refinada na escolha de FDs em ambientes de banco de dados.

# 3 DESENVOLVIMENTO

Foi realizado uma prova de conceito com o objetivo de verificar se de fato os dados faltantes tem impacto nos algoritmos de *data profiling*. Foi considerado a utilização de bases sintéticas e dois algoritmos de Dependências de Negação (DC).

Com o objetivo de aprofundar os experimentos sobre o impacto de dados nulos, foi considerado a utilização de 8 bases de dados e de 5 tipos de algoritmos de perfilamento de dados. Além de que, foram combinadas as 3 mecânicas de imputação de dados nulos 2.14.1.

## 3.1 PROVA DE CONCEITO

Ao estudar sobre o algoritmo DcFinder, foi perceptível o impacto da técnica exata e aproximada para detecção de DCs. No entanto, não foram considerados testes com dados nulos. Para compreender este impacto no algoritmo, foram criados três cenários de testes para provar o conceito de que dados nulos tem impacto negativo no algoritmo.

### 3.1.1 Testes de impacto de dados nulos no DcFinder

Realizamos experimentos com DcFinder, utilizando base de dados com erros, sem erros e com dados nulos. O objetivo é mostrar o impacto em dois pontos: Na quantidade de DCs resultantes e na qualidade delas.

No caso da base de dados com dados nulos, adotamos a estratégia de inserção de dados faltantes de forma aleatória usando o mecanismo MCAR. Isso nos permitiu simular uma situação em que os dados estão ausentes de maneira aleatória e independente de outras variáveis.

Além disso, foram considerados dados com erros, ou seja, dados que foram inseridos no *csv* de maneira errada, desrespeitando as regras de negócio da tabela ideal.

#### *3.1.1.1 Testes utilizando tabelas com erros*

Realizamos um experimento com o DcFinder, utilizando como entrada o conjunto de dados descrito na Tabela 1.1. Como resultado, foram geradas 14 restrições de negação, disponíveis no apêndice A.1.1, das quais se destacam:

- $\neg[t0.Ano = t1.Ano]$
- $\neg[t0.Carro \neq t1.Carro \wedge t0.Renavam = t1.Renavam]$

Com estes dois resultados é possível estabelecer uma interpretação sobre os dados. No primeiro item, Ano foi interpretada como sendo uma chave primária, o que faz sentido para os dados apresentados, dado que o ano não se repete na coluna. Note que Renavam não foi encontrado devido a duplicidade nas tuplas $t_1$ e $t_2$.

Por outro lado, no segundo item, é possível interpretar que não podem existir carros diferentes com Renavans iguais. Essa regra de negócio não havia sido mapeada antes, mas ela é importante para garantir que não haja duplicidade na identificação de um carro.

Tabela 3.1: Conjunto de Dados de Veículos sem a presença de valores nulos

| Tupla | Renavam | Carro | Marca | Ano | Valor |
|---|---|---|---|---|---|
| t1 | 1 | Gol | Volkswagen | 2005 | 15000 |
| t2 | 2 | Gol | Volkswagen | 2007 | 18000 |
| t3 | 3 | Gol | Volkswagen | 2010 | 20000 |
| t4 | 4 | Mobi | Fiat | 2012 | 15000 |
| t5 | 5 | Mobi | Fiat | 2015 | 20000 |
| t6 | 6 | Mobi | Fiat | 2020 | 30000 |

#### *3.1.1.2 Testes utilizando tabelas sem erros*

Na Tabela 3.1 estes dados duplicados foram corrigidos e o algoritmo do $DCFinder$ foi reaplicado.

Neste teste, 16 restrições foram encontradas. Filtrando elas por relevância no seu significado, destacam-se:

- $\neg[t0.Renavam = t1.Renavam]$
- $\neg[t0.Carro = t1.Carro \wedge t0.Marca \neq t1.Marca]$

Agora foi considerado Renavam como chave única no primeiro item. E no seguinte, carros com nomes iguais não podem ter de marcas diferentes. Os demais resultados podem ser encontrados no apêndice A.1.2.

### 3.1.2 Testes utilizando tabelas com valores nulos

Por fim, testes foram realizados passando dados nulos. A tabela

Tabela 3.2: Conjunto de Dados de Veículos com a presença de valores nulos

| Tupla | Renavam | Carro | Marca | Ano | Valor |
|---|---|---|---|---|---|
| t1 | 1 | Gol | Volkswagen | 2005 | 15000 |
| t2 | null | Gol | Volkswagen | 2007 | 18000 |
| t3 | 3 | Gol | Volkswagen | 2010 | 20000 |
| t4 | 4 | Mobi | null | 2012 | 15000 |
| t5 | 5 | Mobi | Fiat | 2015 | null |
| t6 | 6 | Mobi | Fiat | 2020 | 30000 |

Os resultados do teste anterior se repetem para os dois casos:

- $\neg[t0.Renavam = t1.Renavam]$
- $\neg[t0.Carro = t1.Carro \wedge t0.Marca \neq t1.Marca]$

No entanto a quantidade de restrições encontradas pelo algoritmo é significativamente menor. Passando de 16 restrições para 6 restrições, menos da metade. Os demais resultados podem ser encontrados no apêndice A.1.3.

### 3.1.3 Testes de impacto de dados nulos no hydra

Também conduzimos experimentos com o Hydra, utilizando os mesmos conjuntos de dados anteriores: um com erros, sem erros e com dados nulos. Nosso objetivo foi analisar o impacto dessas diferentes condições em dois aspectos principais: a quantidade de dependência de negação resultantes e a qualidade das mesmas.

No caso do conjunto de dados com dados nulos, adotamos a estratégia MCAR.

#### *3.1.3.1 Testes utilizando tabelas com erros*

Utilizando como entrada o conjunto de dados descrito na Tabela 1.1, foram geradas 16 dependências de negação, disponíveis no apêndice A.2.1 Entre elas, destacamos as seguintes DCs resultantes:

- $\neg[t0.Carro = t1.Carro \wedge t0.Valor = t1.Valor]$
- $\neg[t0.Carro \neq t1.Carro \wedge t0.Renavam = t1.Renavam]$

A primeira dependência de negação indica que não é aceito ter registros em que o carro da primeira tupla seja igual ao carro da segunda e o valor da primeira tupla seja igual ao valor da segunda tupla. A segunda restrição, assim como a gerada pelo algoritmo DcFinder, segue a mesma interpretação descrita em 3.1.1.1.

#### *3.1.3.2 Testes utilizando tabelas sem erros*

Utilizando como entrada o conjunto de dados descrito na Tabela 3.1, foram geradas 12 dependências de negação, disponíveis no apêndice A.2.2. Entre elas, destacamos as seguintes DCs resultantes:

- $\neg[t0.Renavam = t1.Renavam]$
- $\neg[t0.Carro = t1.Carro \wedge t0.Marca \neq t1.Marca]$

A primeira dependência não aceita a existência de registros com o mesmo número de Renavam, assegurando a unicidade desse identificador para cada veículo.

A segunda dependência impede a ocorrência de registros em que o nome do carro seja igual, mas as marcas sejam diferentes. O que faz sentido nesse cenário.

#### *3.1.3.3 Testes utilizando tabelas com valores nulos*

Utilizando como entrada o conjunto de dados descrito na Tabela 3.2, foram geradas 6 dependências de negação, disponíveis no apêndice A.2.3. Os resultados mais relevantes, é o mesmo que a tabela sem erros:

- $\neg[t0.Renavam = t1.Renavam]$
- $\neg[t0.Carro = t1.Carro \wedge t0.Marca \neq t1.Marca]$

Os resultados indicam que a presença de dados nulos afetam a geração de dependências de negação, reduzindo a quantidade encontrada.

### 3.1.4 Considerações da prova de conceito

Através da prova de conceito envolvendo testes com bases de dados sintéticas, foi possível interpretar que a presença de dados nulos no DcFinder e no Hydra tem um impacto na quantidade e qualidade dos resultados obtidos. Note na Tabela 3.3 que ambos os algoritmos tem resultados inferiores quando há a presença de dados nulos.

Tabela 3.3: Resultado de experimentos com DC Finder e Hydra

| | DC Finder | Hydra | DCs iguais |
|---|---|---|---|
| ERRO | 14 | 16 | 8 |
| IDEAL | 16 | 12 | 7 |
| NULOS | 7 | 6 | 4 |

A observação de que a quantidade de dependências de negação geradas foi menor para a base de dados com dados nulos indica que a presença desses dados influencia diretamente o processo de geração de dependências. A inserção de dados nulos de forma aleatória utilizando o mecanismo MCAR permitiu simular uma situação em que os dados estão ausentes de maneira aleatória. Portanto, é importante considerar a presença de dados nulos e compreender seu impacto durante a execução do DcFinder, Hydra e outros algoritmos de *data profiling*, a fim de garantir resultados mais eficientes e abrangentes.

## 3.2 BASES DE DADOS ESCOLHIDAS

Com o objetivo de explorar características distintas entre bases de dados, incluindo diferentes tamanhos, número de colunas e tipos de dados para refletir casos reais e diversas situações, escolhemos oito conjuntos de dados limpos para conduzir os nossos experimentos com cada algoritmo - com a exceção do BINDER, onde realizamos uma seleção de um dos 8 *datasets* e mais quatro bases de dados de fora da Tabela 3.4. A escolha dos conjuntos de dados foi orientada por alguns critérios. Primeiramente, buscamos garantir a representatividade de casos do mundo real. A diversidade de tamanhos de conjuntos de dados foi considerada para avaliar como os algoritmos se comportam diante de volumes variados de dados em conjunto com a variação no número de colunas. A inclusão de diferentes tipos de dados, como numéricos, categóricos e textuais, também foi um dos critérios. A seleção específica do BINDER a partir de um dos oito datasets foi feita considerando fatores como complexidade e relevância para os objetivos do estudo, já que necessita de mais de uma base. A seguir, listamos todas as bases de dados:

Tabela 3.4: Informações sobre as bases usadas nos algoritmos ECP/INCS, Order, HyUCC e HyFD

| **Dataset** | **Colunas** | **Registros** |
|---|---|---|
| echocardiogram | 13 | 131 |
| iris | 5 | 149 |
| Cathchainlist | 10 | 16,750 |
| adult | 15 | 32,560 |
| airports | 18 | 76,641 |
| Census6 | 6 | 99,760 |
| flights | 20 | 499,999 |
| Tax | 15 | 1,000,000 |

Tabela 3.5: Informações sobre as bases usadas no algoritmo Binder

| **Dataset** | **Colunas** | **Registros** |
|---|---|---|
| Cathnames | 3 | 1,792 |
| Cathchainlist | 10 | 16,750 |
| Cathdomainlist | 10 | 67,053 |
| Cathdomainseqs | 2 | 67,053 |
| Cathtesmaexp | 90 | 299,999 |

O conjunto de dados Iris, uma referência clássica de Fisher em 1936, traz informações de medidas de sépalas e pétalas de três espécies de íris. Já o conjunto *adults*, prevê se a renda ultrapassa 50 mil/ano com dados do censo, incluindo idade e ocupação. No contexto médico, o conjunto *Echocardiogram* foi escolhido e contém informações de classificação de sobrevivência após um ano de um ataque cardíaco, utilizando diversos atributos médicos. Adotamos o conjunto *Airports* que possui informações de análise global de aeroportos, enquanto a base *Flights* proporciona informações abrangentes sobre voos. O conjunto de dados *Tax*, mostra detalhes de impostos individuais, e informações fiscais.

As bases de dados *Cathnames*, *Cathchainlist*, *Cathdomainlist*, *Cathdomainseqs* e Cathtesmaexp são conjuntos do projeto CATH, que utiliza os modelos *AlphaFold* (v2) para prever domínios estruturais em 21 organismos. *Cathnames* oferece identificadores associados a esses domínios, enquanto *Cathchainlist* fornece uma lista organizada das cadeias polipeptídicas correspondentes. *Cathdomainlist* cataloga e organiza os domínios, e *Cathdomainseqs* compila as sequências associadas a esses domínios. Por último, *Cathtesmaexp* enriquece a compreensão funcional, ao oferecer informações detalhadas sobre expressões tesaurísticas ligadas aos domínios estruturais.

## 3.3 ALGORITMOS ESCOLHIDOS

Na Seção 2 foram estudados diferentes tipos de dependências e alguns algoritmos para cada dependência.

Ao avaliar os algoritmos do Metanome (Metanome, 2023), foi compreendido que os melhores para realizar os experimentos são:

- (DC) ECP/INCS - Por ECP possuir uma performance melhor quando comparado com DCFinder e Hydra (Pena et al., 2022). Foi considerado o método de enumeração INCS para os estudos.
- (OD) ORDER - O algoritmo demonstra eficiência e escalabilidade. Sua capacidade de implementar regras de poda resulta em uma redução significativa no espaço de busca e boa performance (Langer e Naumann, 2016). Na plataforma Metanome, possui somente essa opção na descoberta de dependências de ordem.
- (UCC) HyUCC - Optamos pelo uso do algoritmo HyUCC devido à sua eficiência superior em comparação ao DUCC e outros algoritmos, tanto em escalabilidade de linha quanto de coluna. O algoritmo destaca por sua capacidade de ignorar candidatos UCC de nível inferior durante a fase de amostragem, resultando em melhor desempenho. (Abedjan e Naumann, 2023)

- (FD) HyFD - Escolhemos o HyFD devido à sua eficiência e sua capacidade de lidar com conjuntos de dados extensos na descoberta de dependências funcionais. o HyFD apresenta desempenho superior em termos de consumo de memória e tempo de execução do que algoritmos como Tane, DFD e FDs. Consideramos a escolha ideal para atender às demandas dos nossos experimentos (Papenbrock et al., 2015b).
- (IND) BINDER - Pelo seu propósito geral de encontrar tanto dependências unárias quanto n-árias, além de possuir performance superior a outros algoritmos como SPIDER e MIND (Papenbrock et al., 2015c).

É importante mencionar que, com exceção do ECP/INCS, todos os algoritmos estão disponíveis na plataforma do Metanome (Metanome, 2023) e foram executados com a utilização do Metanome CLI (CLI, 2017).

## 3.4 MECÂNICAS DE POLUIÇÃO ESCOLHIDAS

Com o objetivo de verificar o comportamento dos algoritmos frente ao impacto dos nulos, foram consideradas os três mecanismos de imputação de dados nulos: MCAR, MNAR e MAR.

Em cada uma das mecânicas, o percentual de dados nulos foi variado da seguinte forma: 0%(sem nulos), 0.1%, 0.5%, 1%, 2%, 3%, 4%, 5%, 6%, 7%, 8%, 9%, 10% e 20%. Com essa variação foi possível avaliar o impacto quando se varia pouco os dados 0% à 1%, quando há uma variação constante 1% a 10%, e por fim, em 20% de dados nulos.

É importante notar que todas as bases de dados estão em formato (.csv) e as imputações dos dados nulos foram feitas com o auxílio da biblioteca Jenga (Jenga, 2023). Sendo assim, um dado nulo é caracterizado por uma célula vazia deste csv, ou seja, não foram consideradas sintaxes como: “null”, “NA”, “?”, ou qualquer outra sintaxe de nulo.

## 3.5 SCRIPT

Foi criado um `script` para executar cada algoritmo, para cada base, para cada mecênica e para cada percentual de dados nulos.

Para garantir maior consistência nos resultados, cada algoritmo é executado 10 vezes para a mesma configuração. O motivo disso é que a imputação de dados nulos é feita de maneira aleatória. Isso traz um cenário mais próximo da aleatoriedade dos dados no mundo real. Dessa forma, cada rodada pode gerar diferentes resultados, devido a posição distinta do nulo na tabela. Ao final é feito uma média dos resultados para as análises.

### 3.5.1 Script de poluição

O *script* recebe como parâmetro de entrada: o caminho do csv limpo, a mecânica de imputação de dados nulos e o percentual de dados nulos.

Com isso, utilizando o pacote Jenga (Jenga, 2023), é possível gerar a saída de um novo arquivo csv poluído.

### 3.5.2 Armazenamento de resultados

Os resultados dos algoritmos são armazenados em disco. Para organizar os resultados há uma estrutura de pastas a ser respeitada 3.6. Essa estrutura agrupa os resultados por algoritmo, base de dados, mecânica e rodada.

Tabela 3.6: Estrutura de pastas geradas pelo algoritmo

```
/
└── resultados-script
    └── YYYY-MM-dd-hh-mm-ss
        ├── BINDER
        │   └── adults
        │       ├── MCAR
        │       │   ├── 1
        │       │   │   └── resultados
        │       │   ├── ...
        │       │   └── 10
        │       ├── MNAR
        │       └── MAR
        ├── ECP
        ├── HyFD
        ├── HyUCC
        └── ORDER
```

Após todos os algoritmos terminarem a sua execução, os resultados são compilados.

### 3.5.3 Compilação de resultados

Essa compilação consiste em percorrer a estrutura de pastas, coletar as informações de resultados e salvar em um csv único de resultado.

O csv resultante deste *script* contemplará todas as métricas necessárias para as futuras análises e comparações sobre o impacto de dados nulos.

## 3.6 MÉTRICAS COLETADAS

As métricas coletas são: tempo, quantidade de regras e quantidade de regras agrupadas por tamanho.

# 4 EXPERIMENTOS

O experimento realizado foi a execução do `script` em uma Máquina Virtual máquina virtual com CPU de 4 núcleos, 16GB de memória RAM, sem GPU e sistema operacional Ubuntu 22.04 64 bits. O resultado do experimento pode ser influenciado pela virtualização, e algumas variações observadas podem ser atribuídas, em parte, ao ambiente virtualizado e suas configurações como na métrica de tempo de execução. Porém, mantemos o ambiente controlado e separado durante a execução do experimento, garantindo resultados consistentes.

Após a execução de todos os algoritmos, foi observado que nem todas as configurações tiveram resultado. É possível observar nas Tabelas 4.1 e 4.2 que o algoritmo ORDER só teve resultados para *echocardiogram* e *Cathchainlist*. O algoritmo HyUCC não teve resultados com a base *adutls*. Por fim, a base *Census* não teve resultados em nenhum dos algoritmos.

Tabela 4.1: Status dos Experimentos - Parte 1

| **Experimento** | **echocardiogram** | **iris** | **Cathchainlist** | **adult** |
|---|---|---|---|---|
| HyFD | Concluído | Concluído | Concluído | Concluído |
| HyUCC | Concluído | Concluído | Concluído | Sem Resultados |
| ECP/INCS | Concluído | Concluído | Concluído | Concluído |
| ORDER | Concluído | Sem Resultados | Concluído | Sem Resultados |

Tabela 4.2: Status dos Experimentos - Parte 2

| **Experimento** | **airports** | **flights** | **Tax** | **Census6** |
|---|---|---|---|---|
| HyFD | Concluído | Concluído | Concluído | Sem Resultados |
| HyUCC | Concluído | Concluído | Sem Resultados | Sem Resultados |
| ECP/INCS | Concluído | Concluído | Concluído | Sem Resultados |
| ORDER | Sem Resultados | Sem Resultados | Sem Resultados | Sem Resultados |

O algoritmo BINDER, por aceitar múltiplos csv's, rodou apenas para as bases de dados da categoria *Cath*, tanto para dependências unárias quanto n-árias.

Tabela 4.3: Status dos Experimentos - BINDER

| **Experimento** | **Cathnames** | **Cathchainlist** | **Cathdomainlist** | **Cathdomainseqs** | **Cathtesmaexp** |
|---|---|---|---|---|---|
| BINDER | Concluído | Concluído | Concluído | Concluído | Concluído |

Em resumo, foram 5 algoritmos e 8 bases de dados. Note que todos eles foram executados variando as configurações de imputação e quantidade de dados nulos.

## 4.1 ANÁLISES

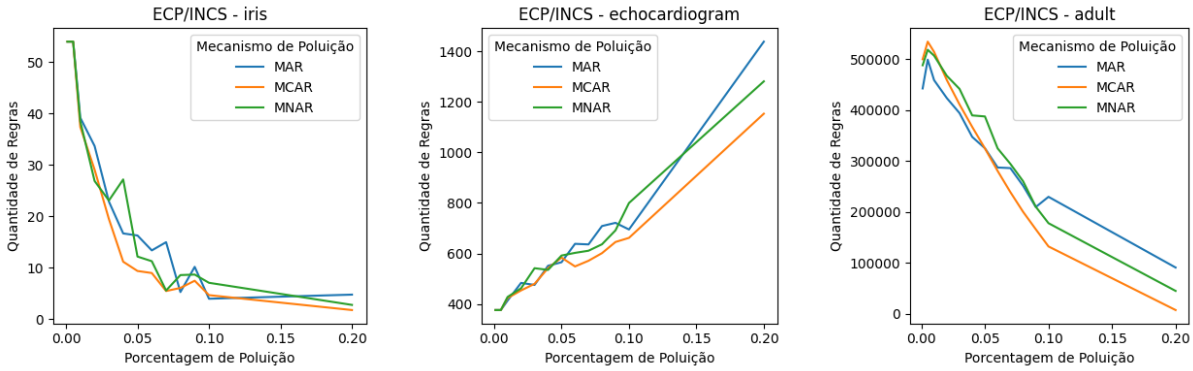
A seguir serão analisados todos os resultados obtidos na fase experimental. Isto é, uma visão do comportamento do algoritmo, performance, quantidade de resultados e regras agrupadas por tamanho. Por fim, algumas considerações e discussões sobre peculiaridades de cada algoritmo.

### 4.1.1 Comportamento ECP/INCS

A etapa de experimentação com o algoritmo ECP combinado com a enumeração INCS teve comportamentos diferentes para cada *dataset*. O comportamento do mecanismo de poluição variou entre *conjunto de dados* pequenos e grandes. Já a variação da porcentagem de nulos, para alguns *datasets* aumentou o tempo de execução, e outros diminuiu - o mesmo ocorre para a quantidade.

#### *4.1.1.1 Quantidade de resultados e tempo em relação a mecânica de nulos*

Os resultados para *echocadiogram*, *iris* e *adults* tiveram valores similares entre as três mecânicas de poluição ao variar a porcentagem de nulos. Observe a Figura 4.1. Note que estas três bases de dados são pequenas no caso de *echocadiogram*, *iris* e de médio porte *adults* em comparação com o conjunto de todas as bases de teste.




(a) Iris (b) Echocardiogram (c) Adults

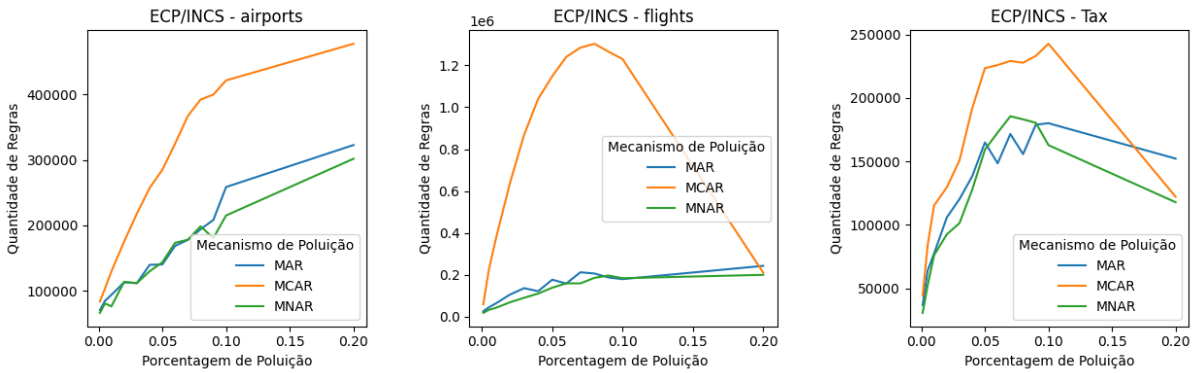
Figura 4.1: Relação entre a Quantidade de Predicados e a Porcentagem de Poluição - ECP/INCS

No entanto, a influencia da mecânica MCAR é mais significativa ao observar os resultados para *datasets* maiores, enquanto MAR e MNAR possuem valroes mais próximos. Note na Figura 4.2. Note que os três *datasets* - *airports*, *flights* e *tax* aumentam o número de regras ao chegar em 20% de dados faltantes em relação à base limpa com 0%. Isso se dá porque todas as mecânicas influenciam na diretamente na satisfação das regras ao criar o conjunto de evidências.

Note que isso pode estar atrelado a abordagem de DCs aproximadas, onde há um limite de tolerância que permite que uma porção das tuplas não satisfaça todos os pedaços de evidência. A inserção de nulos na base faz com que mais regras sejam consideradas ao montar o conjunto de evidências.

Por outro lado, nota-se para as bases *flights* e *tax* uma queda na quantidade de regras próximo dos 10% até os 20%. Compreendeu-se que esta queda se dá pelo limite de tolerância para DCs aproximadas. Ou seja, após uma certa porcentagem de nulos é atingido, o número de tuplas que não satisfazem uma DC em potencial é maior do que este limitante, logo a DC é desconsiderada.

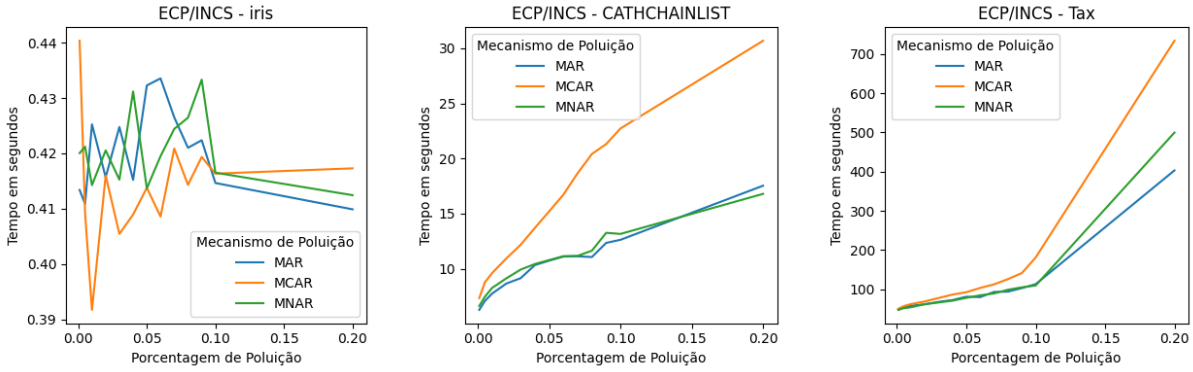
Em relação ao tempo, todos aumentam a medida que os dados nulos crescem. Esse fato se dá pelo aumento de regras em potencial no conjunto de evidências, que pode demandar mais processamento nessa etapa como também na etapa de enumeração de DCs. Novamente, a influência da mecânica MCAR só é clara com *databases* maiores. Na Figura 4.3, note que para csvs pequenos como *Iris*, a oscilação dos valores dificulta a analise, por outro lado, para *datasets*




(a) Airports (b) Flights (c) Tax

Figura 4.2: Relação entre a Quantidade de Predicados e a Porcentagem de Poluição - ECP/INCS Parte 2

maiores como *Flights* e *Tax*, o tempo tem maior previsibilidade, sendo unanime que poluições com MCAR tem um crescimento mais acentuado.




(a) Iris (b) Cathchainlist (c) Tax

Figura 4.3: Relação entre Tempo em segundos e Porcentagem de Poluição - ECP/INCS

### *4.1.1.2 Variação do tamanho das regras em relação a fração de nulos*

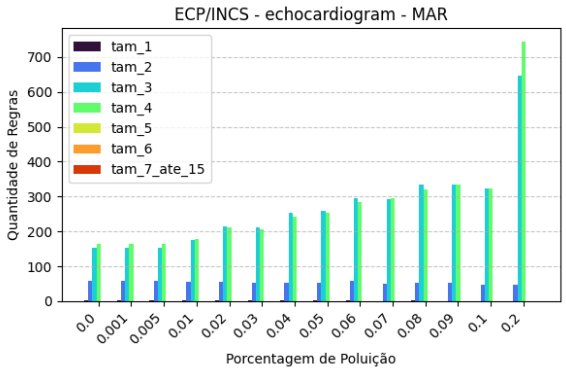
Uma característica comum entre todos os *datasets* e todas as mecânicas, ao variar a quantidade de nulos, é que: A quantidade de resultados muda, mas o tamanho dos regras não. Na Figura 4.4, note que em MAR - *Echocardiogram*, a quantidade de regras varia, mas sempre dentro dos tamanhos 2, 3 e 4 regras. O mesmo ocorre em MCAR - *Cathchainlist* e MNAR - *TAX* com os tamanhos 5, 6 e maiores que 7.

### *4.1.1.3 Considerações*

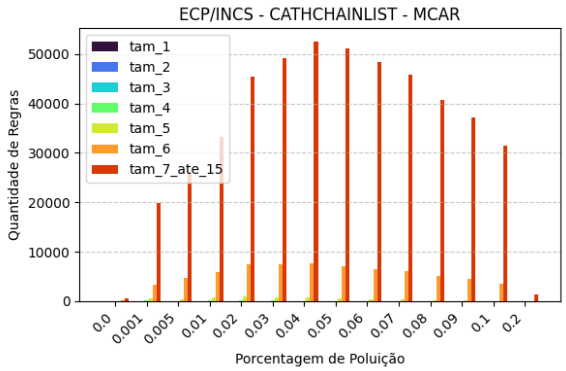
Um ponto comum para os *datasets* maiores, foi a influência mecânica MCAR, que teve maior oscilação, a medida que o percentual de nulos aumenta.

Quanto ao aumento ou diminuição dos resultados, há outras variáveis a serem consideradas como: o tamanho da base, a quantidade de colunas, o tipo de cada coluna e a própria natureza do conteúdo da base. Ou seja, cada base possui sua característica única que pode influenciar na descoberta de DCs.

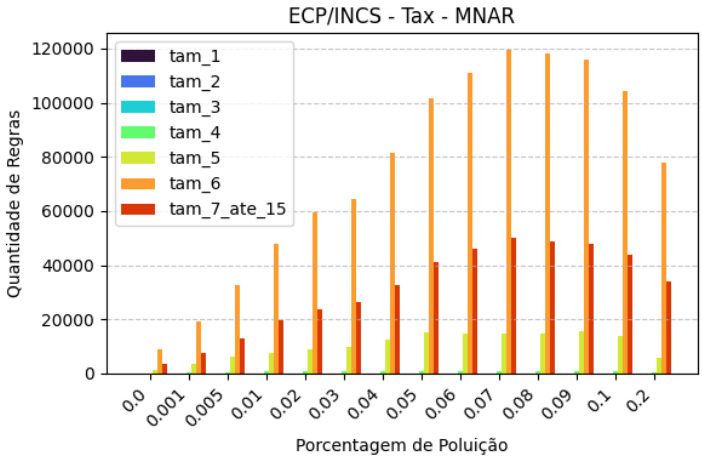
No caso da quantidade de linhas da base, a performance é influenciada quadraticamente e para a quantidade de colunas, exponencialmente (Pena et al., 2022).




(a) MAR - Echocardiogram




(b) MCAR - Cathchainlist




(c) MNAR - Tax

Figura 4.4: Relação entre Tamanho das Regras e Porcentagem de Poluição - ECP/INCS

Ainda nesse aspecto, podem variar o tipo de coluna entre categórica e numérica, que muda a quantidade de operadores analisados para os atributos. Isso significa que, uma base de dados com mais colunas numéricas, dará origem a um espaço de predicados maior e terá mais evidências a serem analisadas. Por outro lado, um *dataset* com mais colunas categóricas, terá menos operadores, menos predicados e menos evidências.

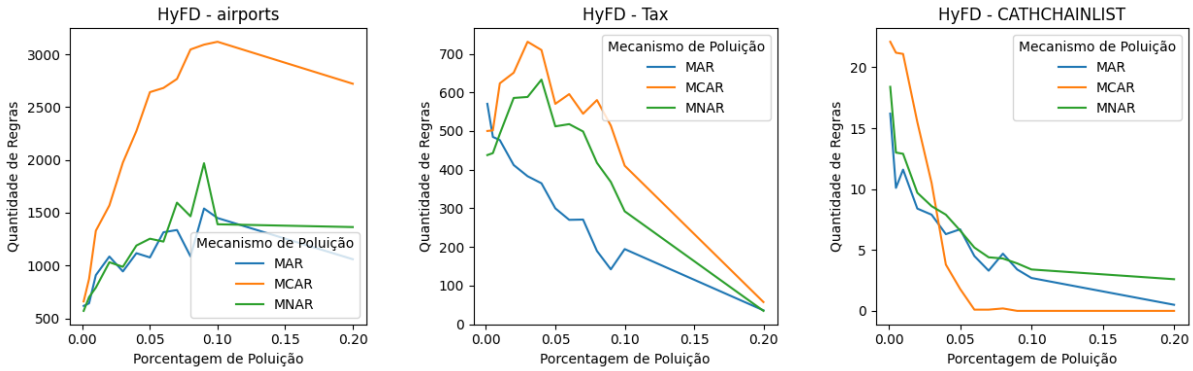
### 4.1.2 Comportamento HyFD

#### *4.1.2.1 Quantidade de resultados e tempo em relação a mecânica de nulos*

A análise dos resultados revela padrões distintos na descoberta de dependências funcionais (FDs) em diferentes conjuntos de dados.

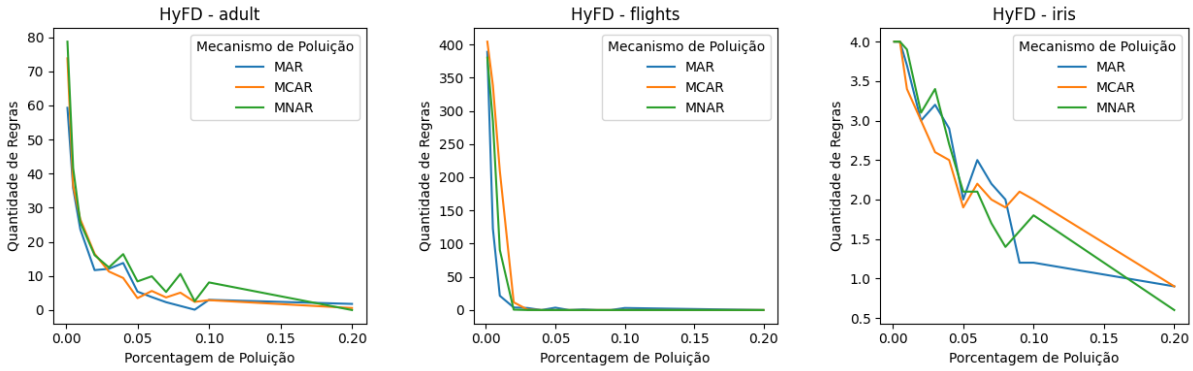
Observando as Figuras 4.5 e 4.6, a tendência geral é uma diminuição na quantidade de FDs a medida que a poluição por dados nulos aumenta, com oscilações de crescimento nas primeiras poluições. Isso mostra uma sensibilidade da descoberta de FDs a presença de valores ausentes, fazendo sentido com a natureza da dependência funcional, que se baseia na relação entre valores em colunas específicas.

O comportamento diferente no conjunto de dados *echocardiogram*, onde a quantidade de FDs aumentou constantemente, pode ser atribuído à natureza dos dados, Figura 4.7. Analisando o *dataset*, observarmos a presença de colunas com valor constante em todas as linhas. De acordo




(a) Airports (b) Tax (c) Cathchainlist

Figura 4.5: Relação entre a Quantidade de Predicados e a Porcentagem de Poluição - HyFD




(a) Adult (b) Flights (c) Iris

Figura 4.6: Relação entre a Quantidade de Predicados e a Porcentagem de Poluição - HyFD Parte 2

com (Papenbrock et al., 2015b) o algoritmo HyFD adota a semântica *null* = *null*. Notamos que essa escolha pode resultar em uma descoberta maior de Dependências Funcionais (FDs). Essa variação é ocasionada pela presença de valores nulos, que introduzem dinâmica para a coluna constante, levando à identificação de novas FDs e, consequentemente, ao aumento do número total de FDs descobertas.

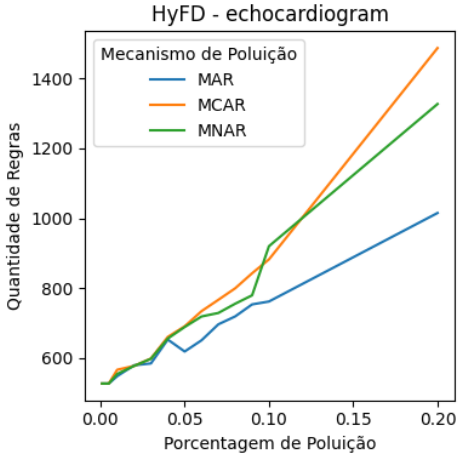
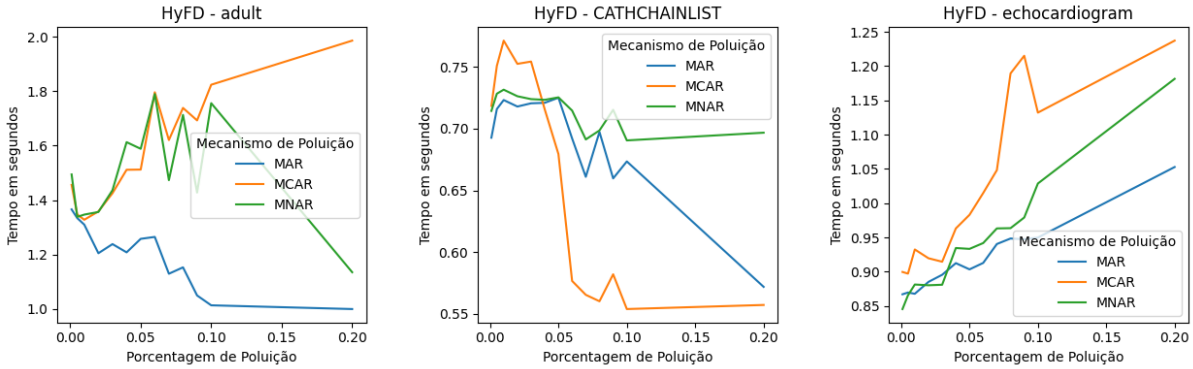



Figura 4.7: Relação entre a Quantidade de Predicados e a Porcentagem de Poluição - HyFD Parte 3

Ao analisar o desempenho de execução do algoritmo HyFD nos conjuntos de dados e sob diferentes mecanismos de poluição (MNAR, MAR, MCAR), observamos padrões distintos - Figura 4.8.

No conjunto *adult*, o tempo de execução inicialmente aumentou, mas entre 5% e 10% de poluição os mecanismos MNAR e MAR diminuíram, enquanto o MCAR continuou a crescer. Para o *dataset flights*, houve uma redução no tempo de execução na primeira poluição, mas a partir de 10%, o MCAR cresceu constantemente. A análise do *cathchainlist* revelou uma queda inicial no tempo de execução, especialmente notável no MCAR, o mecanismo mais eficiente entre os três neste caso. Em *echocardiogram* e *airports*, ao contrário de outros conjuntos, o tempo de execução continuou a crescer com o aumento da poluição, sem mostrar sinais de redução. Nos conjuntos *tax* e *iris*, ocorreram oscilações até 10% de poluição, seguidas de uma diminuição para todos os mecanismos, sendo o MCAR o mais demorado.

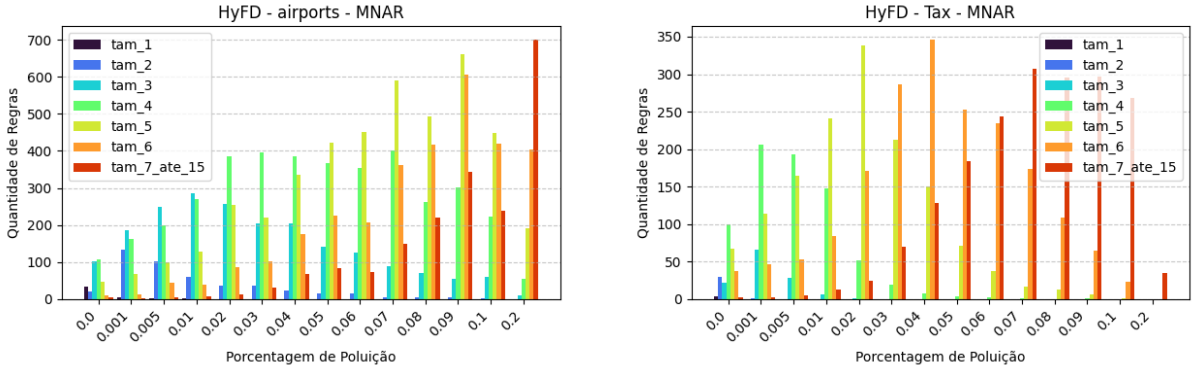



(a) Adult (b) Cathchainlist (c) Echocardiogram

Figura 4.8: Relação entre Tempo em segundos e Porcentagem de Poluição - HyFD

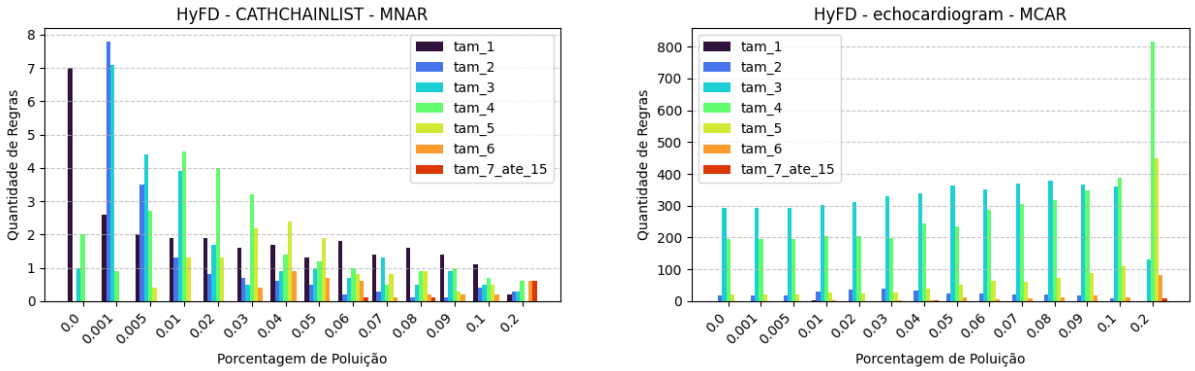
### *4.1.2.2 Variação do tamanho das regras em relação a fração de nulos*

Em todos os conjuntos de dados e cenários de poluição, observamos uma tendência de aumento no tamanho das dependências funcionais na medida que a poluição é aumentada - Figura 4.9 e 4.10. Essa observação pode estar relacionada à maior complexidade introduzida pelos valores nulos. Esse acontecimento resulta na formulação de dependências funcionais mais complexas, caracterizadas por regras mais extensas.




(a) Airports (b) Tax

Figura 4.9: Relação entre Tamanho das Regras e Porcentagem de Poluição - HyFD




(a) Cathchainlist

(b) Echocardiogram

Figura 4.10: Relação entre Tamanho das Regras e Porcentagem de Poluição - HyFD Parte 2

A análise comparativa do tamanho das regras entre diferentes mecanismos de poluição (MNAR, MAR, MCAR) revelou certas oscilações onde aumentava a fração de nulos. Em especial, no caso do conjunto de dados *Airports*, observamos um aumento das regras logo nos primeiros níveis de poluição nos predicados sob o mecanismo MCAR em comparação com os mecanismos MNAR e MAR.

#### *4.1.2.3 Considerações*

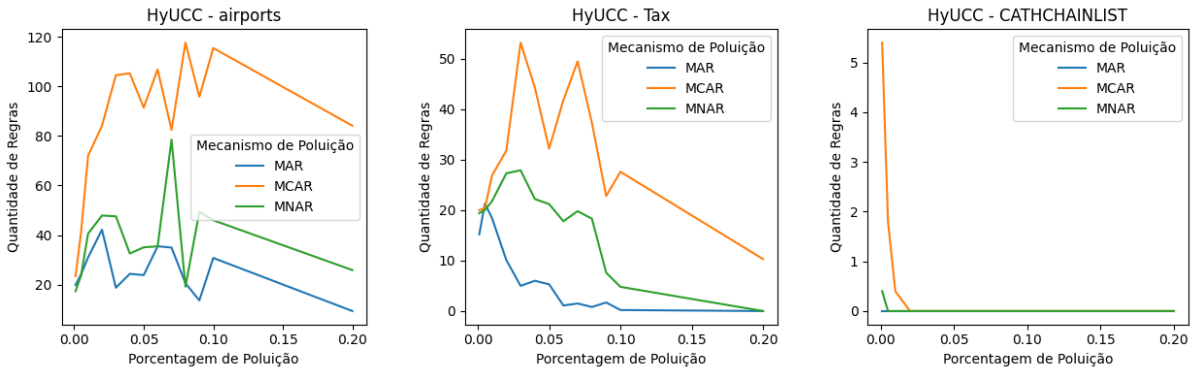
A presença de dados nulos pode dificultar a identificação de relações significativas entre colunas. Na medida que mais valores nulos são introduzidos, as regras para descrever as dependências entre colunas podem se tornar mais longas e difíceis de identificar. Ao adotar a semântica *null* = *null*, a abordagem considera que dois valores nulos são iguais (Papenbrock et al., 2015b). Isso influencia a forma como o algoritmo HyFD descobre e valida as FDs nos dados.

No caso do mecanismo de poluição MCAR, mostrou que ele teve mais variações ao longo da quantidade de poluição. A aleatoriedade na inserção de valores nulos pode levar a resultados mais variados em comparação com outros mecanismos de poluição, na qual a introdução de valores nulos seguem padrões mais específicos.

### 4.1.3 Comportamento HyUCC

#### *4.1.3.1 Quantidade de resultados e tempo em relação a mecânica de nulos*

A introdução de valores nulos em conjuntos de dados maiores inicialmente parece aumentar a quantidade de combinações possíveis. Mas, ao passo que a poluição por nulos continua, vimos um padrão no qual algumas dessas combinações torna redundantes e não contribuem para a identificação de UCC. Essa dinâmica é especialmente evidente em *datasets* maiores, nos quais a complexidade combinatória é mais acentuada, resultando em uma ampla variedade de combinações que demandam mais tempo para serem identificadas e reduzidas, como é o caso do *airports* e *tax* - Figura 4.11.




(a) Airports (b) Tax (c) Cathchainlist

Figura 4.11: Relação entre a Quantidade de Predicados e a Porcentagem de Poluição - HyUCC

A natureza e característica dos dados desempenham um papel importante. Em casos como o conjunto de dados *Echocardiogram*, a presença naturalmente elevada de UCC e no número de colunas pode resultar em uma resistência inicial a diminuição da quantidade de regras, observando a Figura 4.12. Adicionado ao fato de existir uma coluna constante, explicada anteriormente. Nesses casos, é possível que seja necessário um aumento significativo na poluição por nulos para que a redução na quantidade de UCC se torne evidente.

Analisando a métrica de tempo de execução do algoritmo HyUCC, Figura 4.13, o resultado mostra alguns padrões diferentes para conjuntos de dados e mecanismos de poluição.

No caso do conjunto de dados *airports*, vimos um aumento consistente no tempo de execução à medida que a quantidade de poluição aumenta, destacando o mecanismo MCAR, que

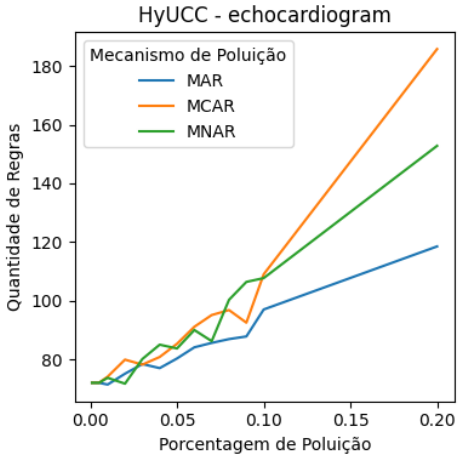
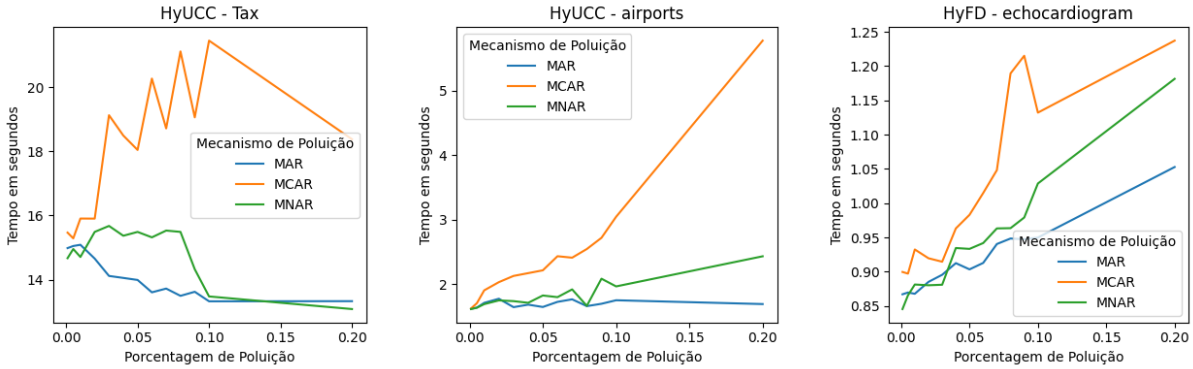



Figura 4.12: Relação entre a Quantidade de Predicados e a Porcentagem de Poluição - HyUCC Parte 2

apresentou um aumento maior em relação aos demais. No *Catchcainlist*, ocorreu uma diminuição no tempo de execução, acompanhada por oscilações entre os mecanismos de poluição.

Na base *echocardiogram*, o tempo de execução aumentou conforme a poluição se intensificou. O *dataset iris* exibiu grandes oscilações no tempo de execução, com um aumento significativo no mecanismo MCAR a partir de 5% de poluição. Para o conjunto *tax*, houve um aumento notável no tempo de execução para o mecanismo MCAR, seguido de uma queda a partir dos 10% de poluição, enquanto os mecanismos MAR e MNAR diminuíram conforme a porcentagem de poluição aumentava.

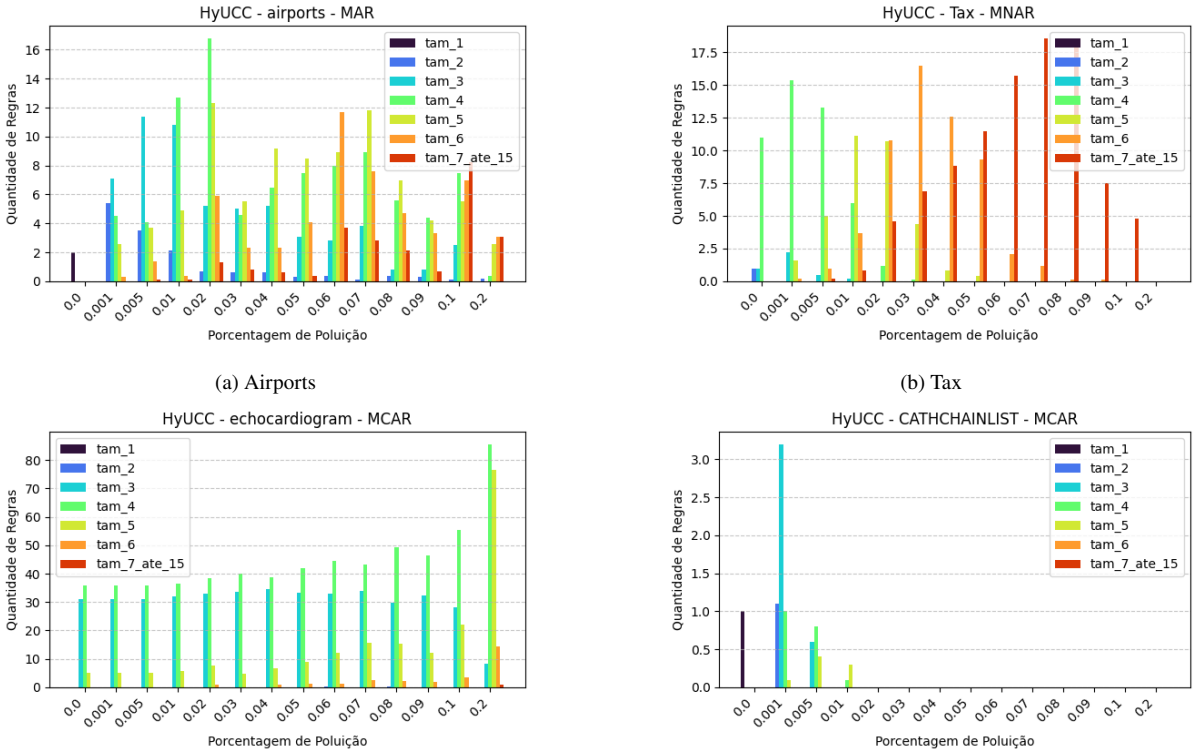



(a) Tax (b) Airports (c) Echocardiogram

Figura 4.13: Relação entre Tempo em segundos e Porcentagem de Poluição - HyUCC

### *4.1.3.2 Variação do tamanho das regras em relação a fração de nulos*

Em todos os conjuntos de dados e mecanismos de poluição, existe um padrão geral de aumento nos tamanhos das regras conforme a poluição aumenta. Isso, na nossa análise, está relacionado com a maior complexidade das regras que o nulo adiciona, que introduz mais combinações possíveis de valores em diferentes colunas, levando a predicados mais complexos (regras maiores) para suportar os erros.




(a) Airports

(b) Tax

(c) Echocardiogram

(d) Cathchainlist

Figura 4.14: Relação entre Tamanho das Regras e Porcentagem de Poluição - HyUCC

#### *4.1.3.3 Considerações*

Um fato em comum das análises foi a a predominância do mecanismo MCAR em gerar mais resultados na identificação de (UCC). Isso pode ser atribuída, a menor dependência de padrões existentes e a menores restrições na formação de UCC na poluição dos dados.

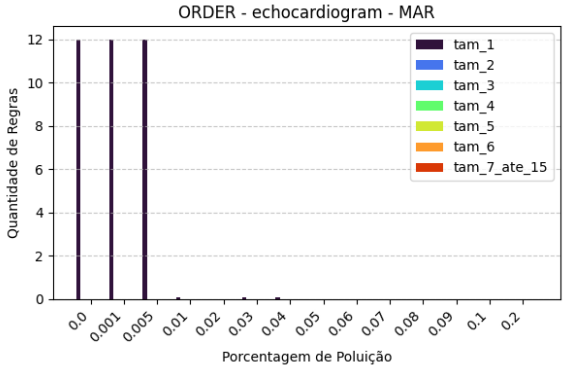
A aleatoriedade completa do MCAR permite uma formação mais aberta de combinações, enquanto a independência de padrões existentes amplia a variedade de resultados.

### 4.1.4 Comportamento ORDER

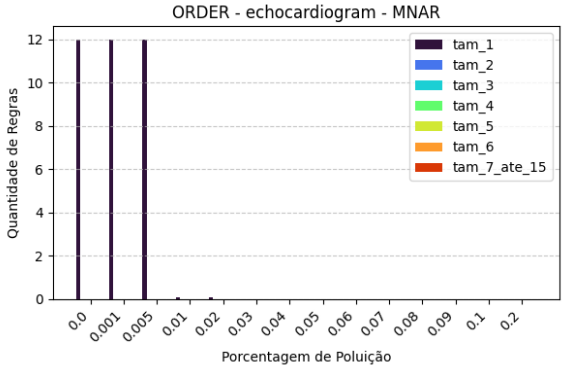
#### *4.1.4.1 Quantidade de resultados e tempo em relação a mecânica de nulos*

Analisamos os resultados referentes às dependências de ordem nos conjuntos de dados citados na Tabela 3.4.

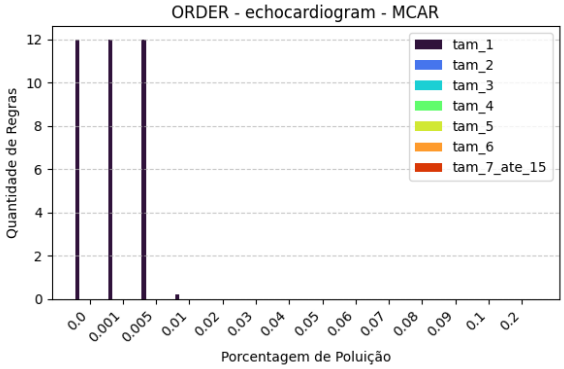
Somente *echocardiogram* e *catchainlist* tiveram dependências de ordem encontradas, considerando três mecanismos de poluição (MNAR, MAR, MCAR). Os demais *datasets* utilizados para experimentação, não obtiveram resultados.




(a) Echocardiogram MAR



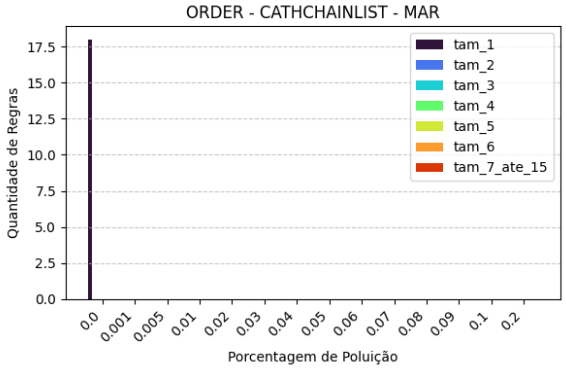

(b) Echocardiogram MNAR



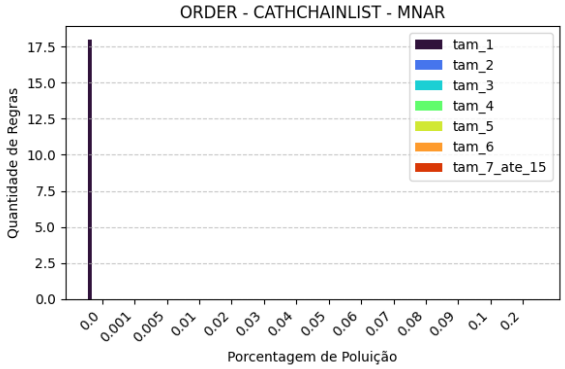

(c) Echocardiogram MCAR

Figura 4.15: Relação entre Tamanho das Regras e Porcentagem de Poluição - ORDER

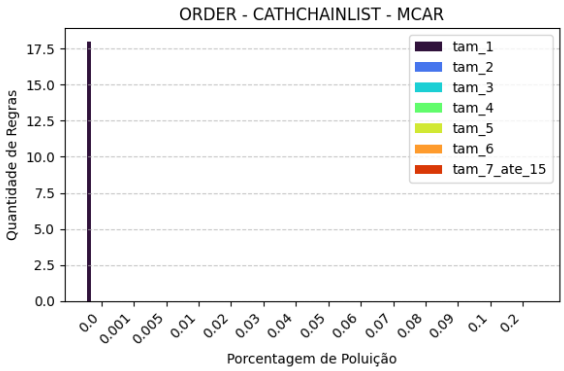
No caso do *Echocardiogram*, observamos na Figura 4.15 que, independentemente do mecanismo de poluição, foram identificados poucas regras (12 no total), todos de tamanho 1 na faixa de 0.1% a 2%.




(a) Cathchainlist MAR



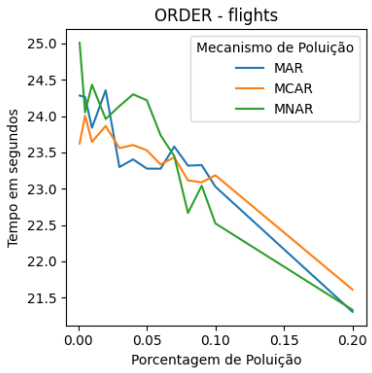

(b) Cathchainlist MNAR



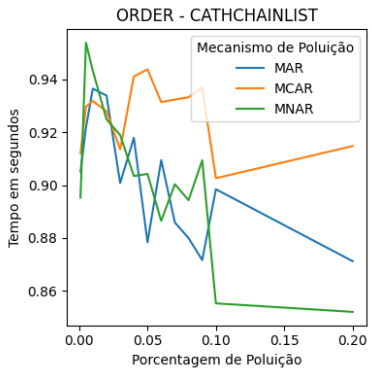

(c) Cathchainlist MCAR

Figura 4.16: Relação entre Tamanho das Regras e Porcentagem de Poluição - ORDER Parte 2

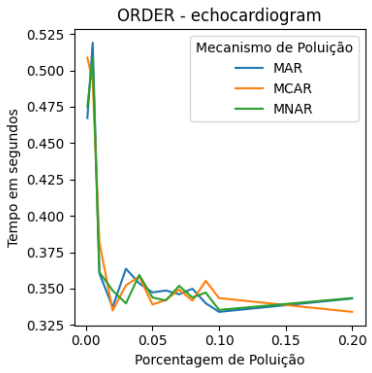
No conjunto de dados *Cathchainlist*, foram identificadas 18 regras de tamanho 1 para os três mecanismos de poluição. O gráfico ilustrado na Figura 4.16 mostrou que ao poluir minimamente a base, não tivemos mais resultados quantitativos.




(a) Flights




(b) Cathchainlist




(c) Echocardiogram

Figura 4.17: Relação entre Tempo em segundos e Porcentagem de Poluição - ORDER

Analisando o tempo em cada *dataset* (Figura 4.17), na proporção que aumentávamos a porcentagem de poluição, observamos um comportamento de diminuição no tempo de compilação, levando em conta oscilações no percurso. Esse comportamento foi observado na maior parte dos *datasets*, indicando uma eficiência crescente do algoritmo ao passo que a quantidade de poluição aumentava.

#### *4.1.4.2 Considerações*

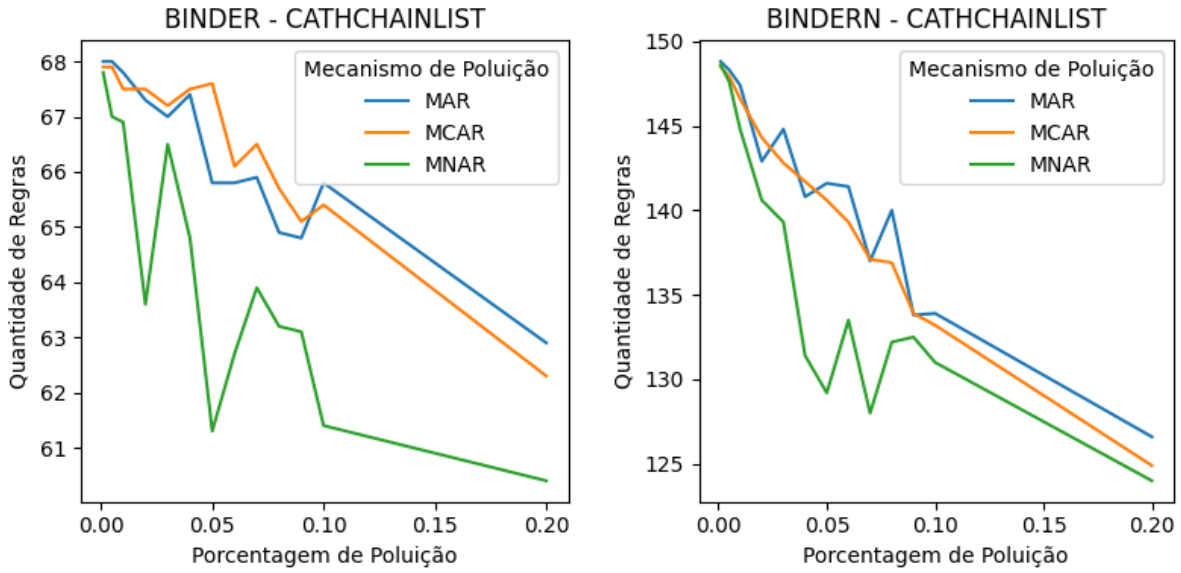
A abordagem adotada em (Langer e Naumann, 2016) e (Szlichta et al., 2018), que consiste na exclusão das colunas contendo um ou mais valores nulos nos conjuntos de dados, sugere uma estratégia focada na análise de dependências de ordem envolvendo apenas valores não nulos. No entanto, é relevante considerar que essa abordagem pode ser aprimorada, uma vez que a presença de valores nulos é uma realidade comum em conjuntos de dados do mundo real. A implementação de uma semântica ou método específico para lidar com dados nulos poderia enriquecer a análise de dependências de ordem.

### 4.1.5 Comportamento BINDER

Os experimentos com o algoritmo BINDER mostrou comportamentos diferentes quanto a quantidade em relação às mecânicas de nulos e seus percentuais. Por outro lado ele mostrou uniformidade entre os resultados da descoberta de predicados unários e n-ários.

#### *4.1.5.1 Quantidade de resultados e tempo em relação a mecânica de nulos*

Em relação a quantidade de resultados, há uma queda em todas as mecânicas de dados nulos, sendo MNAR a mais oscilante quando comparado com MCAR E MAR, Figura 4.18. Esse cenário é similar tanto para dependências unárias quanto n-árias.



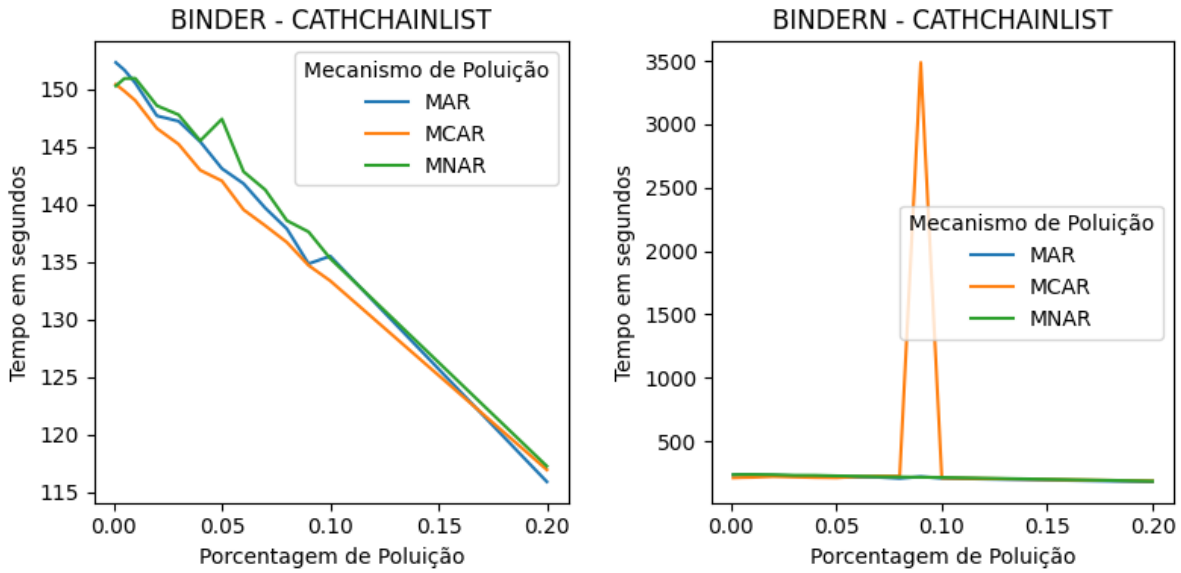

(a) Dependência unária

(b) Dependência n-ária

Figura 4.18: Relação entre a Quantidade de Predicados e a Porcentagem de Poluição - BINDER

Por outro lado, essa discrepância entre os tipos de mecânicas não ocorre ao analisar o tempo com a porcentagem de poluição, Figura 4.19.

Ao verificar a Figura 4.19 para a Dependência n-ária, observamos um grande aumento no tempo de execução próximo aos 10% de poluição. Esse acréscimo foi identificado durante a execução do *script* na máquina virtual que utilizamos para realizar os experimentos. Esse aumento não esperado no tempo de execução mostra a ocorrência de um comportamento excepcional na máquina virtual utilizada.



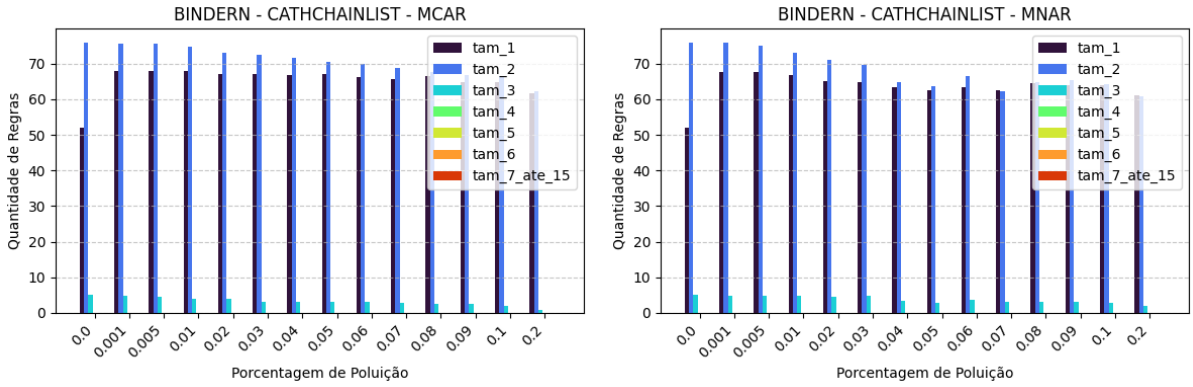

(a) Dependência unária

(b) Dependência n-ária

Figura 4.19: Relação entre Tempo em segundos e Porcentagem de Poluição - BINDER

### *4.1.5.2 Variação do tamanho das regras em relação a fração de nulos*

Ao observar os resultados contemplados na Figura 4.18, há uma oscilação quando os dados nulos são imputados com MNAR. Uma outra visão pertinente, é a variação entre as dependências unárias. Na Figura 4.20 é possível ver a distribuição dos resultados por quantidade de regras.




(a) Dependência n-ária

(b) Dependência n-ária

Figura 4.20: Relação entre Tamanho das Regras e Porcentagem de Poluição - BINDER (N-ária)

Nota-se que para MCAR o resultado decresce de maneira uniforme independente da quantidade de regras no resultado, ao passo que para MNAR a oscilação não é uniforme. Em MAR, o resultado é similar a MCAR.

### *4.1.5.3 Considerações*

O algoritmo BINDER é pouco afetado pela variação de dados nulos, quando se trata de quantidade de resultados. Isso se dá pela estratégia adotada pelo próprio algoritmo de ignorar os dados nulos, uma vez que estes não são relevantes para a descoberta de IND (Papenbrock et al.,

2015c). A redução do tempo é compatível com o fato de que: ao ignorar dados nulos, há menos dados para processar e portanto menos ramificações na estratégia de divisão e conquista. Por fim, ainda que de forma sutil, há variação maior na quantidade de resultados, quando os dados são influenciados pela mecânica MNAR do que pelas outras.

Sobre a base de dados, o BINDER foi testado unicamente com a base de dados de estrutura de proteína "CATHCHAIN". Logo, como trabalhos futuros, a análise com mais bases de dados com tamanhos variados pode trazer resultados distintos.

### 4.1.6 Considerações gerais

Após analisar os algoritmos diante a dinâmica dos nulos, é notável que cada um deles terá comportamentos distintos. Alguns destes comportamentos serão previsíveis, outros não.

Na descoberta de dependências de negação com ECP/INCS, cada mecânica de dados nulos mostrou comportamentos diferentes. Para *datasets* pequenos isso é pouco claro, mas para bases grandes a mecânica MCAR possui maior oscilação. Já o tempo, é crescente em todos os casos a medida que os dados ausentes aumentam. Por fim, a variação de predicados muda em quantidade, mas os tamanhos permanecem os mesmos.

A quantidade de resultados do algoritmo HyFD, tende a diminuir a medida que os dados nulos aumentam e a intensidade dessa diminuição varia de acordo com a mecânica de poluição - sendo MCAR, novamente a que teve mais variações. Porém, a natureza dos dados pode influenciar, como é o caso da base *echocardiogram*, com colunas de valores constante, como explicado anteriormente. Já em relação ao tempo, não há uma previsibilidade, cada mecânica de poluição tem um comportamento diferente, algumas aumentam e outras diminuem em relação a cada *dataset*. Finalmente, a variação de predicados no HyFD diminui em quantidade e, ao contrário do ECP, os tamanhos das regras aumentam.

O algoritmo HyUCC, seguiu as mesmas características do HyFD quanto a relevância da mecânica MCAR e ao tempo dos algoritmos. Da mesma forma, as regras, que aumentam de tamanho a medida que o a poluição aumenta.

Para o algoritmo ORDER, apenas duas bases tiveram resultados e mesmo assim, a quantidade de resultados zerou, após incluir dados nulos. Por fim, o tempo, que na maioria das bases diminuiu a medida que os nulos aumentaram.

Em relação ao BINDER, que foi executado apenas para uma base de dados, notou-se uma diminuição da quantidade e do tempo de execução ao longo da porcentagem de poluição. Sendo que a mecânica de destaque foi MNAR, que teve mais variações. Já em relação aos tamanhos das regras, não houve grande variação, apenas na quantidade de resultados.

A variação observada nos resultados de todos os algoritmos refletem a complexidade pertencente dos dados e também suas estratégias de lidar com valores nulos em cada mecanismo de poluição e seus algoritmos. Um resumo sucinto da análise é a Tabela 4.4.

Tabela 4.4: Tabela do resultado dos algoritmos em relação às métricas

| **Algoritmo** | **Tempo** | **Quantidade** | **Tamanho** | **Mecânica de destaque** |
|---|---|---|---|---|
| ECP/INCS | Aumenta | Aumenta | Aumenta | MCAR |
| HyFD | Diminui | Diminui | Aumenta | MCAR |
| HyUCC | Oscila | Diminui | Aumenta | MCAR |
| ORDER | Diminui | Diminui | Constante | MCAR |
| BINDER | Diminui | Diminui | Diminui | MNAR |

Por fim, vale reforçar que a natureza dos próprios dados tem influência nos algoritmos. Mas, ao combinar isso com, mecânica de nulos e porcentagem de nulos, o comportamento pode variar bastante. Então, é importante que essas informações sejam consideradas ao avaliar o impacto de nulos em *Data Profiling*. Os gráficos restantes estão no Apêndice A.

### 4.1.7 Comportamento das dependências e sua relação com os tipos de nulos

Cada algoritmo mostrou um comportamento diferente frente aos tipos de nulos, isso reflete a forma com que as próprias dependências se relacionam com as mecânicas MCAR, MAR e MNAR.

Esse comportamento de cada dependência frente as mecânicas podem ser comparados quando se mostra o impacto na complexidade dos resultados e de performance na etapa de descoberta. Sendo assim, o aumento na quantidade e tamanho das regras são indicativos do aumento na complexidade dos resultados e também na perda de significância das regras, da mesma forma que o tempo de processamento indica o impacto na performance.

No caso da descoberta de DCs, ficou evidenciado que todas as métricas (tempo, quantidade e tamanho de regras) aumentam. Logo é possível afirmar que: a medida que os dados nulos aumentam na base de dados, aumenta também a complexidade das regras, visto que elas aumentam em tamanho; a significância diminui, visto que elas aumentam em quantidade, o que pode indicar uma falta de precisão para definir regras de negócios que de fato sejam relevante para o conjunto de dados; e há uma piora direta na performance dos algoritmos, que indica que a presença de dados faltantes pode tornar lento o processo de perfilamento de dados, por meio de descoberta de DCs. Além disso, houve a métrica MCAR que teve mais influência nos resultados, ou seja, dados nulos com a disposição completamente aleatória pioram ainda mais a performance e resultados do que dados que possuem certa previsibilidade como o MAR e MNAR. É importante notar que o MCAR pode representar falhas de *software* e *hardware*, que indicam dois fatores que claramente pioram a descoberta de DCs.

Embora as DCs sejam capazes de generalizar os outros tipos de dependências, elas também são as mais impactadas pelos nulos. Cabe destacar que uma análise sobre o próprio *dataset* é útil para compreender se os dados faltantes são oriundos de um fator completamente aleatório, ou não. Dessa forma, é possível ponderar, com mais clareza, se a descoberta de DC é de fato a melhor dependência para o perfilamento de dados naquela base.

A descoberta de FDs e UCCs também mostram que o fator de aleatoriedade é o que mais impacta na descoberta. No entanto, os nossos experimentos mostraram um impacto diferente na complexidade dos resultados. Nota-se que em ambos os algoritmos, há uma queda na quantidade de resultados e diminuição do tempo de processamento do algoritmo, o que por um lado pode facilitar as análises. Isso pode indicar que se o problema de perfilamento em questão exige um tempo de processamento mais rápido, FDs e UCCs podem desempenhar melhor frente aos dados nulos, quando comparado com DCs. Por outro lado, o tamanho das regras também aumenta, que pode deixar as regras com menos significância no processo de perfilamento de dados.

As ODs tiveram poucos resultados em nossos experimentos, logo não há como afirmar que as métricas coletadas de fato refletem o comportamento do tipo de dados nulos em dependências de ordem. Cabe aqui uma experimentação mais aprofundada sobre diferentes abordagens para o tratamento de dados nulos no algoritmo. Talvez tratar dado nulo com um valor mais comum da coluna, ou então a média. Abordagens como a sintaxe de $null = null$ ou $null \neq null$ também podem ser exploradas. Outras, como a substituição realizada em DCs para variáveis categóricas e numéricas, também pode ter um impacto diferente.

A dependência que não teve MCAR como mecânica de destaque, foi a IND. A detecção de IND é pouco afetada pelos dados nulos. Uma vez que eles não são relevantes para a descoberta

(Papenbrock et al., 2015c). De fato os experimentos refletiram este cenário. No entanto é importante citar que o é substancial em uma IND é que o conteúdo de um conjunto de colunas esteja contido no conjunto de outras colunas, para que seja considerada uma relação entre os atributos. Note que neste caso, MNAR não traz o fator de aleatoriedade que MAR e MCAR trazem. Ou seja, enquanto MAR e MCAR são bem distribuídos no *dataset*, o MNAR pode estar concentrado em algum padrão presente na característica dos dados. Essa concentração pode reduzir algum valor importante em uma das colunas comparadas, a ponto de não ser possível declarar que uma coluna está contida na outra. Isso pode caracterizar um descarte de uma possível IND.

Por fim, cabe destacar que a aleatoriedade nos dados nulos é o que mais impacta na descoberta das dependências. Essa aleatoriedade é descrita pela mecânica MCAR e representa falhas de *software* e *hardware*. Por outro lado, as mecânicas MAR e MNAR, por estarem associadas a própria característica dos dados, possuem menos aleatoriedade do que MCAR, e isso pode facilitar no processo de descoberta.

# 5 CONCLUSÃO

Este trabalho explorou o impacto da presença de dados nulos na descoberta de dependências em *Data Profiling*, focando em cinco algoritmos representativos: HyFD, HyUCC, ECP, ORDER e BINDER, e três mecânicas de poluição de dados nulos: MAR, MNAR e MCAR. Os resultados evidenciaram que cada algoritmo reage de maneira distinta a dinâmica dos dados nulos, apresentando comportamentos variados em termos de quantidade e qualidade de resultados, além do tempo de execução.

A mecânica MCAR teve mais influência sobre os algoritmos HyFD, HyUCC, ECP, ORDER. Enquanto o BINDER teve um comportamento mais impactado com a métrica MNAR. Sendo assim, é clara a indicação de que diferentes mecânicas de dados nulos têm influência variada sobre a quantidade de resultados e tempo, dependendo da dependência analisada.

Em relação a variação de tamanho de predicados o ECP, BINDER e ORDER tiveram pouca variação. Por outro lado HyUCC e HyFD tiveram a complexidade de seus resultados aumentada a medida que a quantidade de nulos aumentava.

É importante ressaltar que cada *dataset* possui características distintas, e isso influencia diretamente na performance e resultados. Mesmo assim, foi possível identificar padrões nos resultados - dentro do mesmo algoritmo e entre os algoritmos.

Assim, os resultados refletem não apenas a complexidade pertencente nos *datasets*, mas também a influência das mecânica de valores nulos nos algoritmos de *Data Profiling*. Destaca-se a necessidade de considerar estratégias específicas para lidar com nulos em cada tipo de dependência identificada, com o objetivo de aprimorar a precisão e refinamento na análise de dados com valores ausentes.

Portanto, ao avaliar o impacto de valores nulos em *Data Profiling*, é importante reconhecer a influência de cada mecanismo em cada tipo de dependência. Pois cada dependência vai reagir de uma forma específica. Essa compreensão pode levar a diferentes estratégias para tratamento de dados nulos em uma etapa de pre-processamento dos dados. Ou então, fornecer insumos para o refinamento do algoritmo ao considerar a mecânica de dados nulos que mais impacta seu desempenho.

## 5.1 TRABALHOS FUTUROS

Algumas sugestões de trabalhos futuros são:

1. **Aprofundar nas Estratégias de Tratamento de Nulos:**
   - Explorar estratégias específicas e práticas para lidar com valores nulos em cada tipo de dependência identificada.
2. **Estudo de Métricas Adicionais:**
   - Explorar métricas mais elaboradas, como *succinctness, coverage e interestingness*, para uma análise mais abrangente do desempenho dos algoritmos de *Data Profiling* na presença de dados nulos.
   - Explorar métricas de desempenho, como uso de memória, uso de CPU e uso de disco.
3. **Análise de Outros Algoritmos:**

- Incluir outros algoritmos de *Data Profiling* do Metanome e comparar seu desempenho em relação à presença de dados nulos.

4. **Análise de Outras Bases de Dados:**

   - Estudar o comportamento de mais bases de dados, de diferentes características.
   - Estudar o comportamento de uma base em relação aos anos.

# REFERÊNCIAS


Abedjan, Z. e Naumann, F. (2023). Advancing the discovery of unique column combinations. Relatório técnico, Hasso Plattner Institute, Potsdam, Germany.

Berti-Équille, L., Harmouch, H., Naumann, F., Novelli, N. e Thirumuruganathan, S. (2018). Discovery of genuine functional dependencies from relational data with missing values. *Proceedings of the VLDB Endowment*, 11(8):880–892.

Bleifuß, T., Kruse, S. e Naumann, F. (2017). Efficient denial constraint discovery with hydra. *Proceedings of the VLDB Endowment*, 11(3):311–323.

Chu, X., Ilyas, I. F. e Papotti, P. (2013a). Discovering denial constraints. *Proceedings of the VLDB Endowment*, 6(13):1498–1509.

Chu, X., Ilyas, I. F. e Papotti, P. (2013b). Holistic data cleaning: Putting violations into context. Em *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, páginas 458–469. IEEE.

CLI, M. (2017). Repositório do metanome cli. `https://github.com/HPI-Information-Systems/Metanome/tree/metanome_cli/metanome-cli`. Acessado em 30/05/2023.

Dürsch, F., Stebner, A., Windheuser, F., Fischer, M., Friedrich, T., Strelow, N., Bleifuß, T., Harmouch, H., Jiang, L., Papenbrock, T. et al. (2019). Inclusion dependency discovery: An experimental evaluation of thirteen algorithms. Em *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, páginas 219–228.

Jenga (2023). Repositório do pacote jenga. `https://github.com/schelterlabs/jenga`. Acessado em 15/11/2023.

Langer, P. e Naumann, F. (2016). Efficient order dependency discovery. *VLDB Journal*, 25(2):223–241.

Metanome (2018). Repositório de algoritmos metanome. `https://github.com/HPI-Information-Systems/metanome-algorithms/tree/hydra`. Acessado em 30/05/2023.

Metanome (2023). Data profiling metanome. `https://hpi.de/naumann/projects/data-profiling-and-analytics/metanome-data-profiling.html`. Acessado em 18/11/2023.

Osborne, J. W. (2013). *Best practices in data cleaning: A complete guide to everything you need to do before and after collecting your data*. Sage.

Papenbrock, Thorsten, N. F. (2017). A hybrid approach for efficient unique column combination discovery.

Papenbrock, T., Bergmann, T., Finke, M., Zwiener, J. e Naumann, F. (2015a). Data profiling with metanome. *Proceedings of the VLDB Endowment*, 8(12):1860–1863.

Papenbrock, T., Ehrlich, J., Marten, J., Neubert, T., Rudolph, J.-P., Schönberg, M., Zwiener, J. e Naumann, F. (2015b). Functional dependency discovery: An experimental evaluation of seven algorithms. *Proceedings of the VLDB Endowment*, 8(10):1082–1093.

Papenbrock, T., Kruse, S., Quiané-Ruiz, J.-A. e Naumann, F. (2015c). Divide & conquer-based inclusion dependency discovery. *Proceedings of the VLDB Endowment*, 8(7):774–785.

Pena, E. H., de Almeida, E. C. e Naumann, F. (2019). Discovery of approximate (and exact) denial constraints. *Proceedings of the VLDB Endowment*, 13(3):266–278.

Pena, E. H., de Almeida, E. C. e Naumann, F. (2021). Fast detection of denial constraint violations. *Proceedings of the VLDB Endowment*, 15(4):859–871.

Pena, E. H., Lucas Filho, E. R., de Almeida, E. C. e Naumann, F. (2020). Efficient detection of data dependency violations. Em *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, páginas 1235–1244.

Pena, E. H., Porto, F. e Naumann, F. (2022). Fast algorithms for denial constraint discovery. *Proceedings of the VLDB Endowment*, 16(4):684–696.

Szlichta, J., Godfrey, P., Golab, L., Kargar, M. e Srivastava, D. (2018). Effective and complete discovery of bidirectional order dependencies via set-based axioms. *The VLDB Journal*, 27(4):573–591.

# APÊNDICE A – RESULTADOS DE ALGORITMOS

## A.1 RESULTADOS DE DESCOBERTA DE DC'S COM METANOME E DC FINDER

### A.1.1 Resultado obtido utilizando tabela com erros

- $\neg[t0.Ano = t1.Ano]$
- $\neg[t0.Ano \leq t1.Ano \wedge t0.Renavam \geq t1.Renavam]$
- $\neg[t0.Carro \neq t1.Carro \wedge t0.Renavam = t1.Renavam]$
- $\neg[t0.Carro \neq t1.Carro \wedge t0.Valor < t1.Valor \wedge t0.Ano \leq t1.Ano]$
- $\neg[t0.Renavam = t1.Renavam \wedge t0.Valor \leq t1.Valor]$
- $\neg[t0.Renavam \leq t1.Renavam \wedge t0.Carro \neq t1.Carro \wedge t0.Valor < t1.Valor]$
- $\neg[t0.Marca \neq t1.Marca \wedge t0.Carro \neq t1.Carro \wedge t0.Ano \leq t1.Ano]$
- $\neg[t0.Marca \neq t1.Marca \wedge t0.Carro \neq t1.Carro \wedge t0.Valor < t1.Valor]$
- $\neg[t0.Marca \neq t1.Marca \wedge t0.Renavam = t1.Renavam]$
- $\neg[t0.Marca \neq t1.Marca \wedge t0.Renavam \leq t1.Renavam \wedge t0.Carro \neq t1.Carro]$
- $\neg[t0.Tupla = t1.Tupla]$
- $\neg[t0.Valor = t1.Valor \wedge t0.Carro = t1.Carro]$
- $\neg[t0.Valor \geq t1.Valor \wedge t0.Marca \neq t1.Marca \wedge t0.Ano \leq t1.Ano]$
- $\neg[t0.Valor \geq t1.Valor \wedge t0.Marca \neq t1.Marca \wedge t0.Renavam \leq t1.Renavam]$

### A.1.2 Resultado obtido utilizando tabela sem erros

- $\neg[t0.Renavam = t1.Renavam]$
- $\neg[t0.Valor < t1.Valor \wedge t0.Marca \neq t1.Marca \wedge t0.Renavam \leq t1.Renavam]$
- $\neg[t0.Ano \leq t1.Ano \wedge t0.Valor < t1.Valor \wedge t0.Carro \neq t1.Carro]$
- $\neg[t0.Renavam \leq t1.Renavam \wedge t0.Valor \geq t1.Valor \wedge t0.Marca = t1.Marca]$
- $\neg[t0.Tupla = t1.Tupla]$
- $\neg[t0.Ano = t1.Ano]$
- $\neg[t0.Carro = t1.Carro \wedge t0.Renavam \leq t1.Renavam \wedge t0.Valor \geq t1.Valor]$
- $\neg[t0.Carro = t1.Carro \wedge t0.Valor = t1.Valor]$
- $\neg[t0.Renavam \leq t1.Renavam \wedge t0.Ano \geq t1.Ano]$

- $\neg[t0.Valor < t1.Valor \wedge t0.Renavam \leq t1.Renavam \wedge t0.Carro \neq t1.Carro]$
- $\neg[t0.Valor = t1.Valor \wedge t0.Marca = t1.Marca]$
- $\neg[t0.Ano \leq t1.Ano \wedge t0.Valor \geq t1.Valor \wedge t0.Marca = t1.Marca]$
- $\neg[t0.Ano \leq t1.Ano \wedge t0.Valor < t1.Valor \wedge t0.Marca \neq t1.Marca]$
- $\neg[t0.Carro = t1.Carro \wedge t0.Ano \leq t1.Ano \wedge t0.Valor \geq t1.Valor]$
- $\neg[t0.Carro = t1.Carro \wedge t0.Marca \neq t1.Marca]$
- $\neg[t0.Carro \neq t1.Carro \wedge t0.Marca = t1.Marca]$

### A.1.3 Resultado obtido utilizando tabela com dados nulos

- $\neg[t0.Marca = t1.Marca \wedge t0.Carro \neq t1.Carro]$
- $\neg[t0.Ano = t1.Ano]$
- $\neg[t0.Carro = t1.Carro \wedge t0.Valor = t1.Valor]$
- $\neg[t0.Tupla = t1.Tupla]$
- $\neg[t0.Marca = t1.Marca \wedge t0.Valor = t1.Valor]$
- $\neg[t0.Ano \leq t1.Ano \wedge t0.Carro \neq t1.Carro]$
- $\neg[t0.Renavam = t1.Renavam]$

## A.2 RESULTADOS DE DESCOBERTA DE DC'S COM METANOME E HYDRA

### A.2.1 Resultado obtido utilizando tabela com erros

- $\neg[t0.Id = t1.Id \wedge t0.Valor = t1.Valor]$
- $\neg[t0.Carro \neq t1.Carro \wedge t0.Marca = t1.Marca \wedge t0.Ano \leq t1.Ano \wedge t0.Valor < t1.Valor]$
- $\neg[t0.Ano \leq t1.Ano \wedge t0.Marca \neq t1.Marca \wedge t0.Valor > t1.Valor]$
- $\neg[t0.Carro \neq t1.Carro \wedge t0.Marca = t1.Marca \wedge t0.Valor < t1.Valor \wedge t0.Renavam \leq t1.Renavam]$
- $\neg[t0.Valor \leq t1.Valor \wedge t0.Carro = t1.Carro \wedge t0.Marca \neq t1.Marca \wedge t0.Ano \geq t1.Ano]$
- $\neg[t0.Carro \neq t1.Carro \wedge t0.Renavam = t1.Renavam]$
- $\neg[t0.Carro = t1.Carro \wedge t0.Valor = t1.Valor]$
- $\neg[t0.Renavam \leq t1.Renavam \wedge t0.Marca \neq t1.Marca \wedge t0.Ano \geq t1.Ano]$
- $\neg[t0.Ano = t1.Ano]$
- $\neg[t0.Renavam \leq t1.Renavam \wedge t0.Marca \neq t1.Marca \wedge t0.Valor > t1.Valor]$

- $\neg[t0.Valor \leq t1.Valor \wedge t0.Carro = t1.Carro \wedge t0.Renavam \geq t1.Renavam \wedge t0.Marca \neq t1.Marca]$
- $\neg[t0.Valor \leq t1.Valor \wedge t0.Renavam \leq t1.Renavam \wedge t0.Ano \geq t1.Ano]$
- $\neg[t0.Renavam > t1.Renavam \wedge t0.Ano \leq t1.Ano]$
- $\neg[t0.Renavam = t1.Renavam \wedge t0.Marca \neq t1.Marca]$
- $\neg[t0.Carro \neq t1.Carro \wedge t0.Renavam \leq t1.Renavam \wedge t0.Ano \geq t1.Ano]$
- $\neg[t0.Tupla = t1.Tupla]$

### A.2.2 Resultado obtido utilizando tabela sem erros

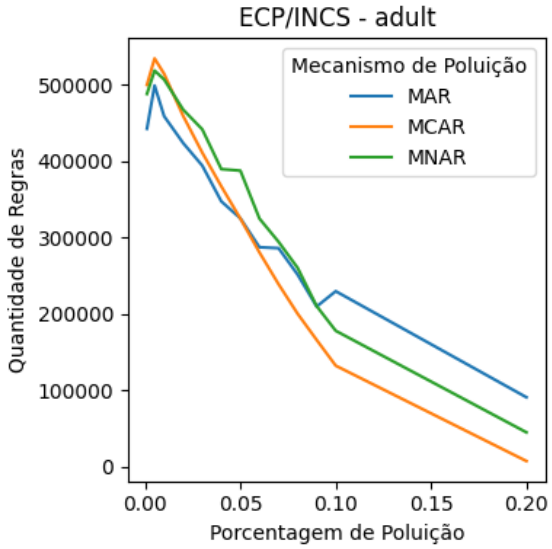
- $\neg[t0.Ano \leq t1.Ano \wedge t0.Renavam \geq t1.Renavam]$
- $\neg[t0.Tupla = t1.Tupla]$
- $\neg[t0.Carro = t1.Carro \wedge t0.Valor = t1.Valor]$
- $\neg[t0.Valor = t1.Valor \wedge t0.Marca = t1.Marca]$
- $\neg[t0.Renavam = t1.Renavam]$
- $\neg[t0.Ano = t1.Ano]$
- $\neg[t0.Carro = t1.Carro \wedge t0.Marca \neq t1.Marca]$
- $\neg[t0.Carro \neq t1.Carro \wedge t0.Marca = t1.Marca]$
- $\neg[t0.Valor \leq t1.Valor \wedge t0.Marca = t1.Marca \wedge t0.Renavam \geq t1.Renavam]$
- $\neg[t0.Ano \geq t1.Ano \wedge t0.Valor \leq t1.Valor \wedge t0.Marca = t1.Marca]$
- $\neg[t0.Carro = t1.Carro \wedge t0.Valor \leq t1.Valor \wedge t0.Renavam \geq t1.Renavam]$
- $\neg[t0.Carro = t1.Carro \wedge t0.Ano \geq t1.Ano \wedge t0.Valor \leq t1.Valor]$

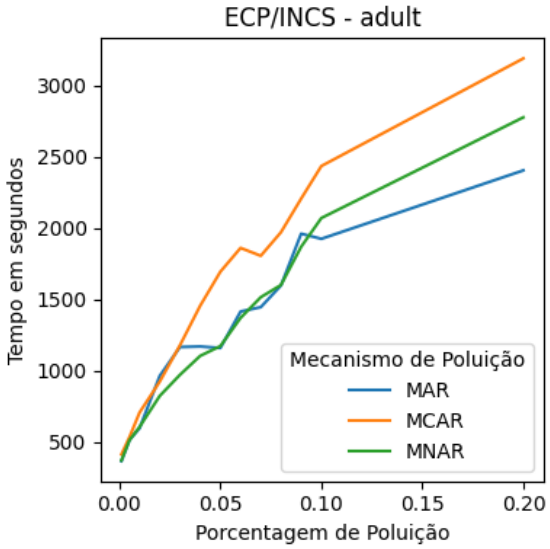
### A.2.3 Resultado obtido utilizando tabela com dados nulos

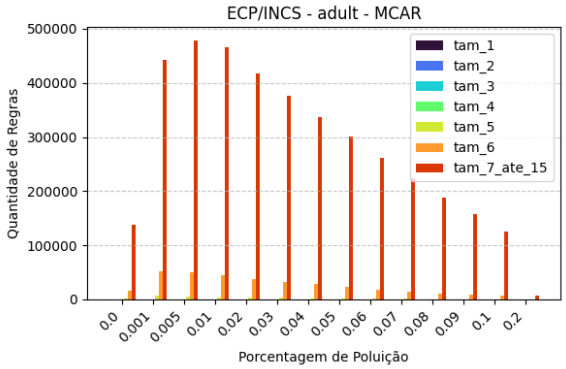
- $\neg[t0.Renavam = t1.Renavam]$
- $\neg[t0.Ano = t1.Ano]$
- $\neg[t0.Carro \neq t1.Carro \wedge t0.Marca = t1.Marca]$
- $\neg[t0.Tupla = t1.Tupla]$
- $\neg[t0.Valor = t1.Valor \wedge t0.Marca = t1.Marca]$
- $\neg[t0.Carro = t1.Carro \wedge t0.Valor = t1.Valor]$

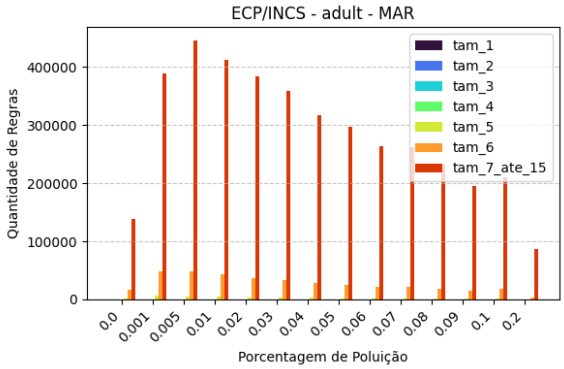
## A.3 RESULTADOS ECP/INCS

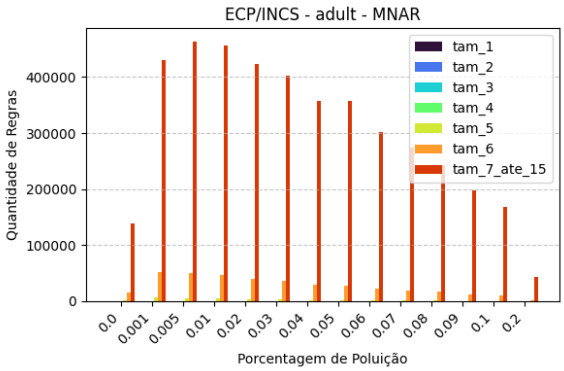
### A.3.1 Resultados ECP/INCS - Adults



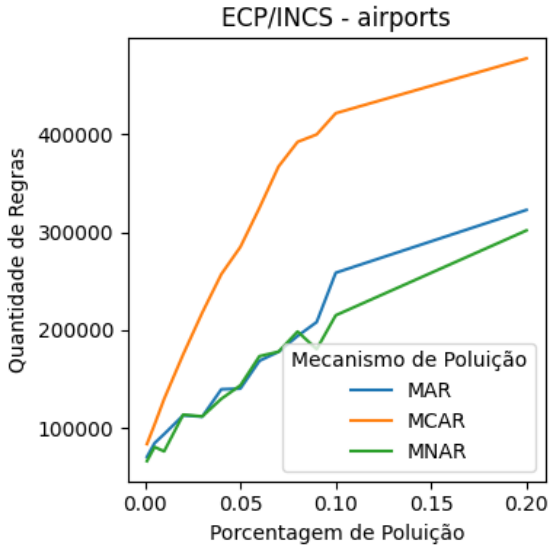



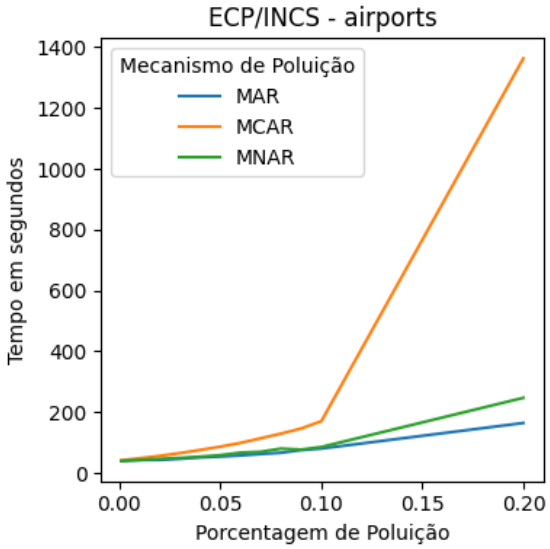



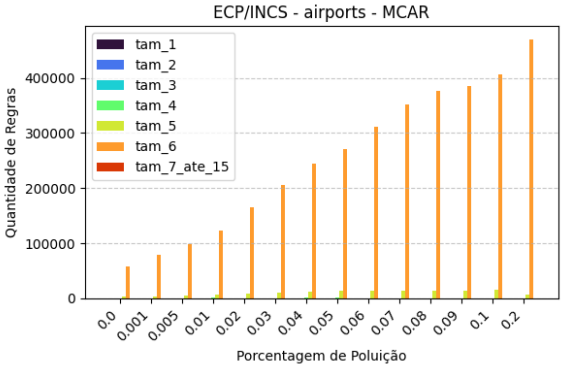



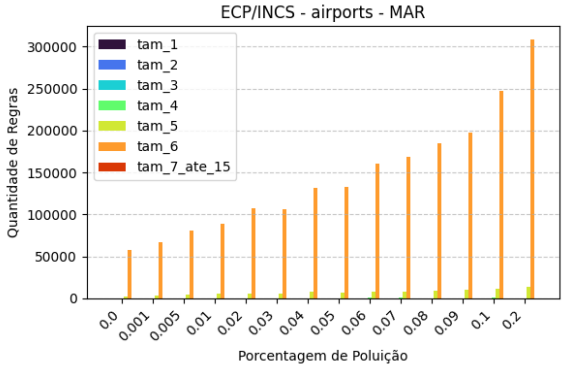



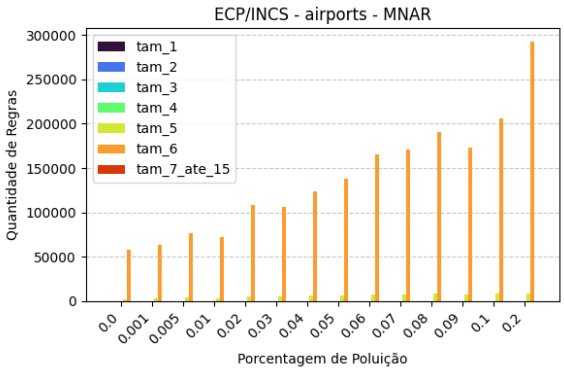
### A.3.2 Resultados ECP/INCS - Airports



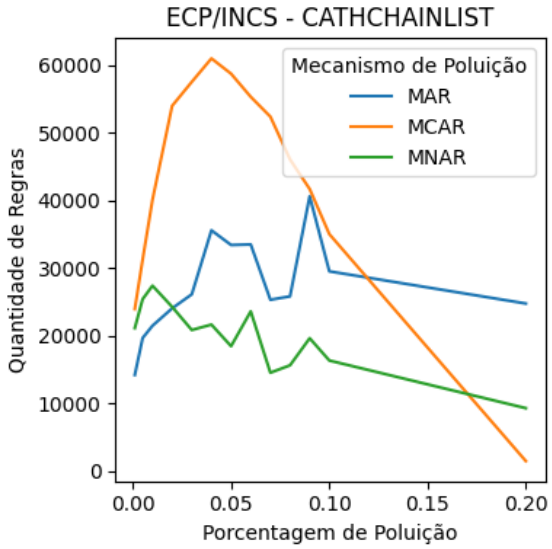



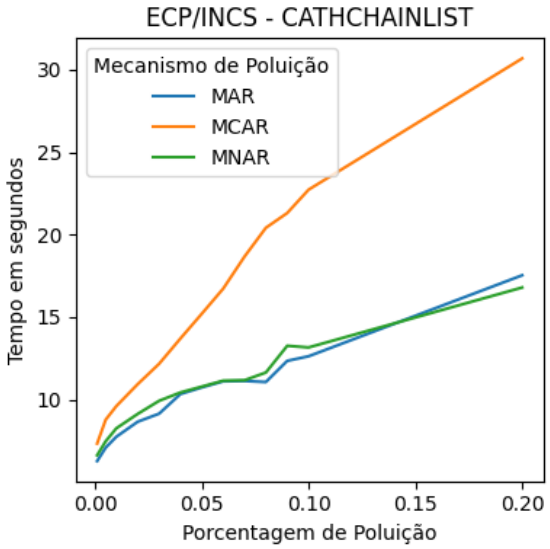



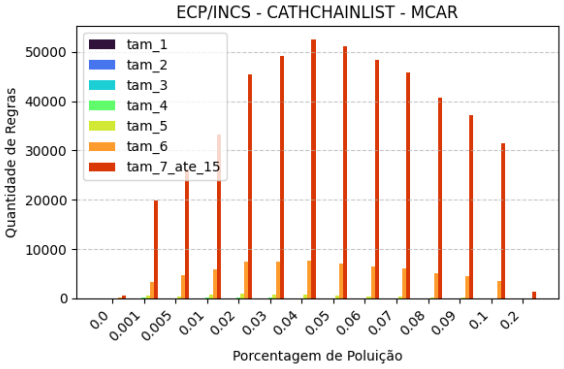



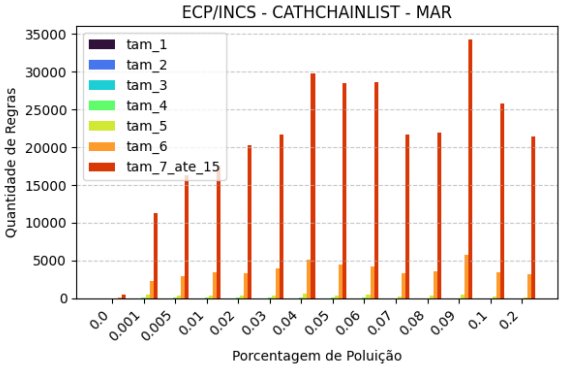



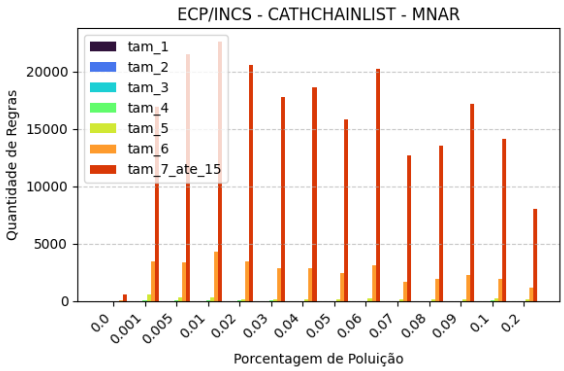
### A.3.3 Resultados ECP/INCS - Cathchainlist



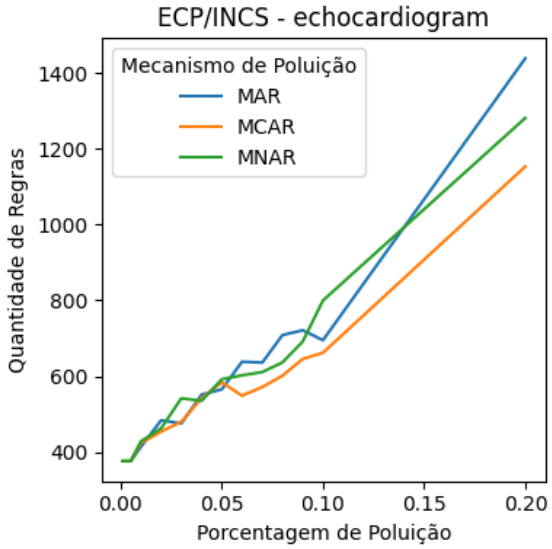



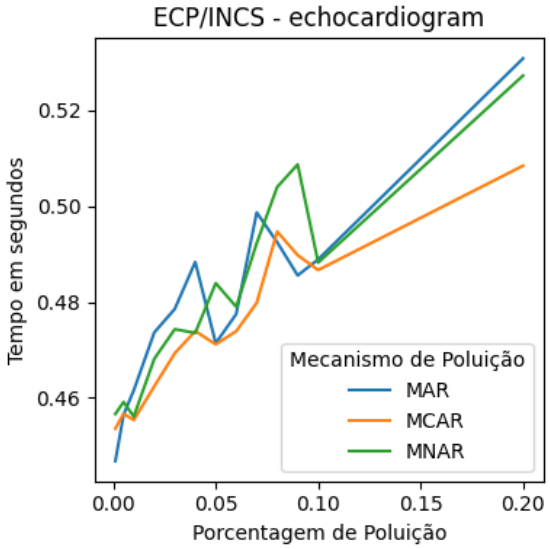



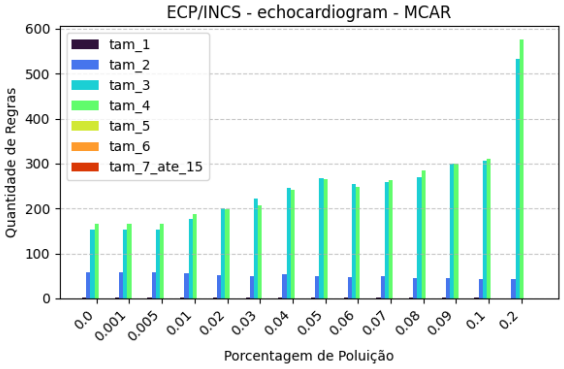



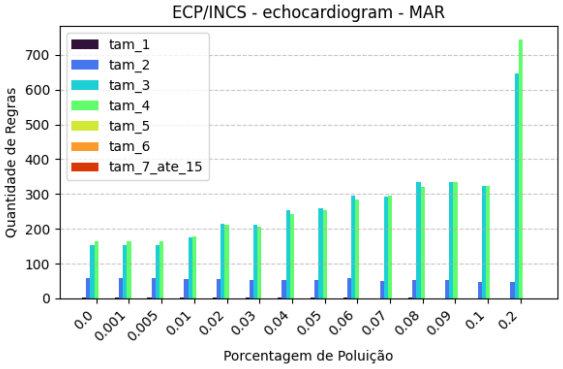



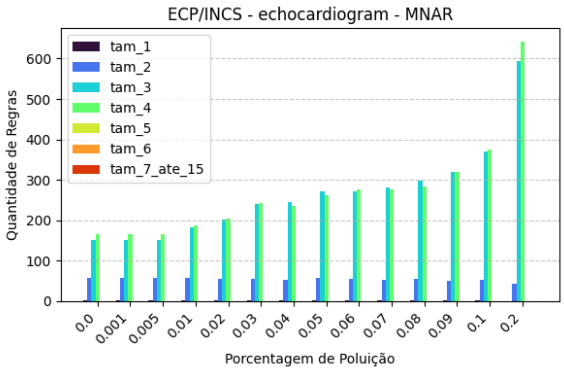
### A.3.4 Resultados ECP/INCS - Echocardiogram



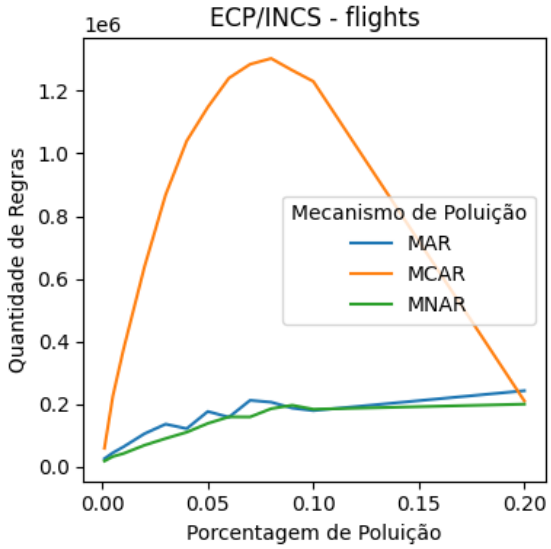



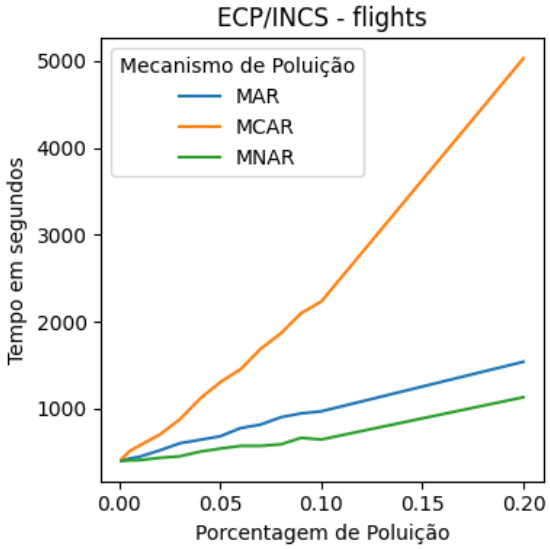



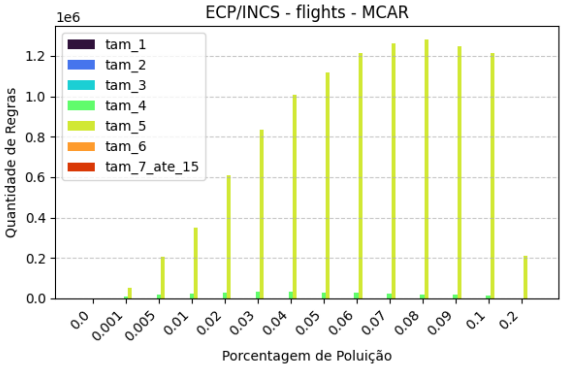



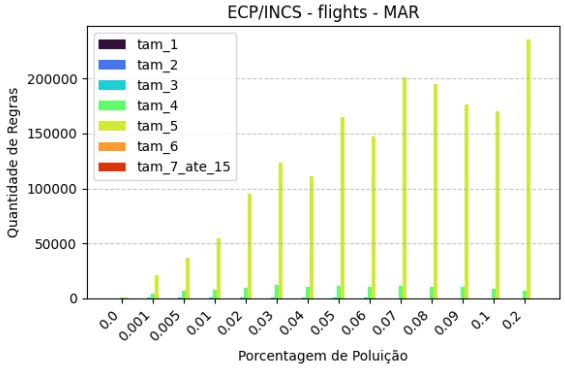



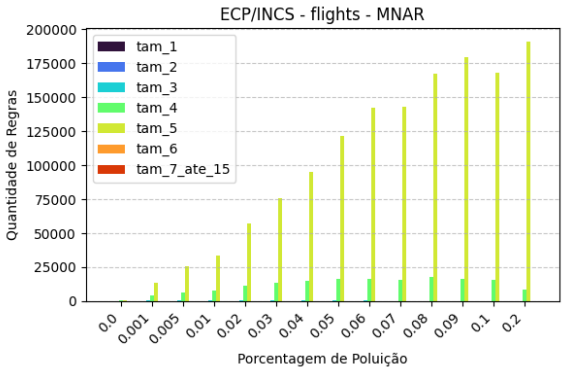
### A.3.5 Resultados ECP/INCS - Flights



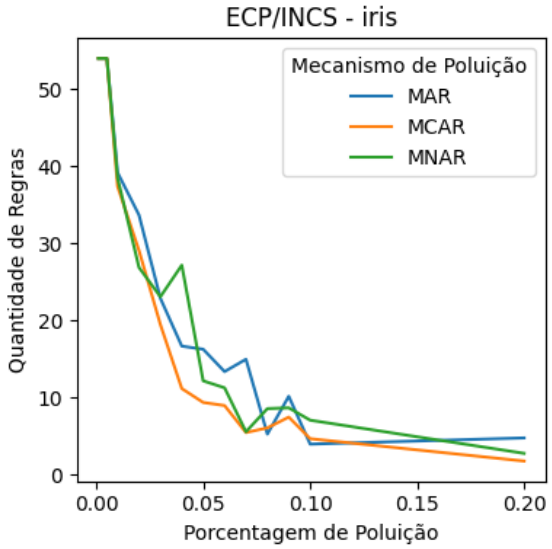



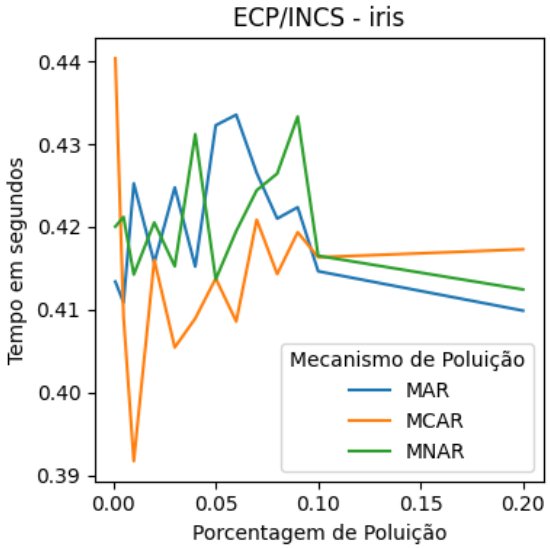



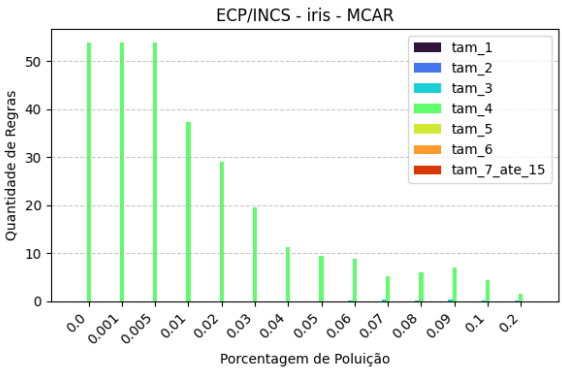



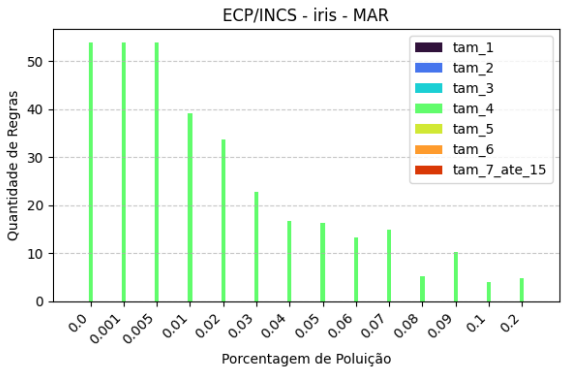



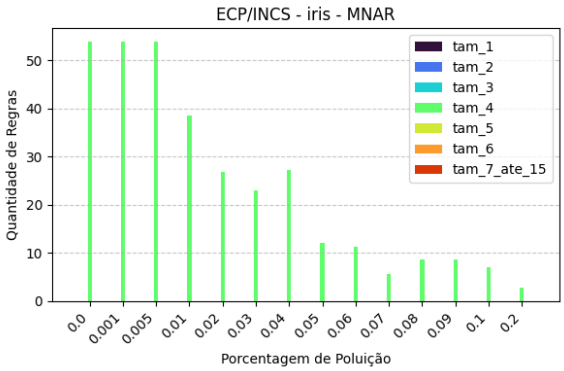
### A.3.6 Resultados ECP/INCS - Iris



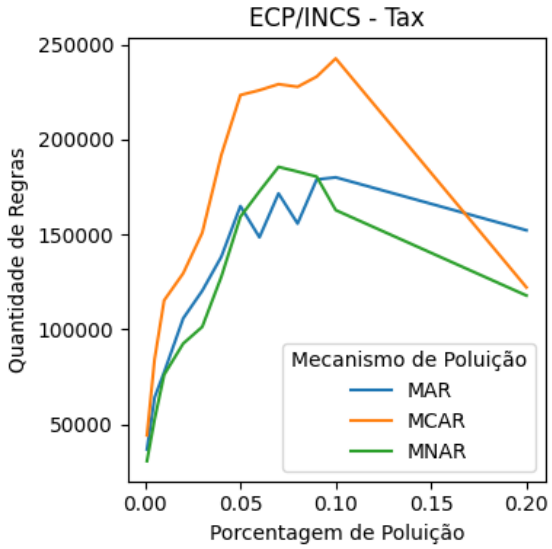



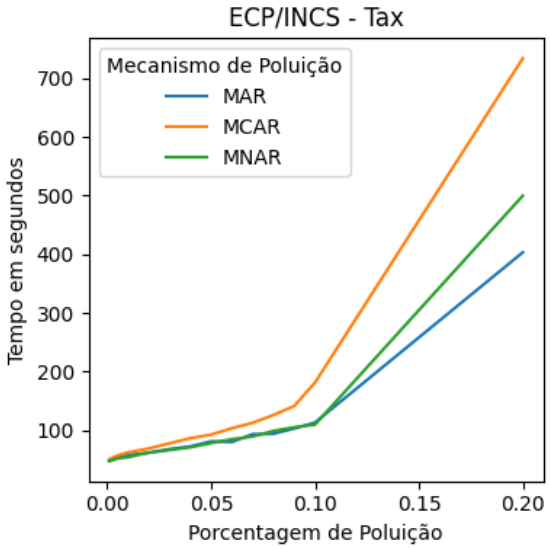



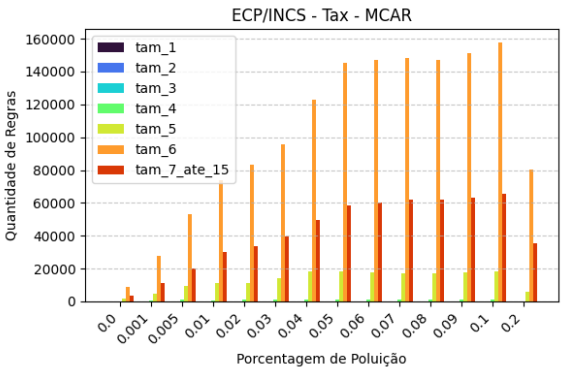



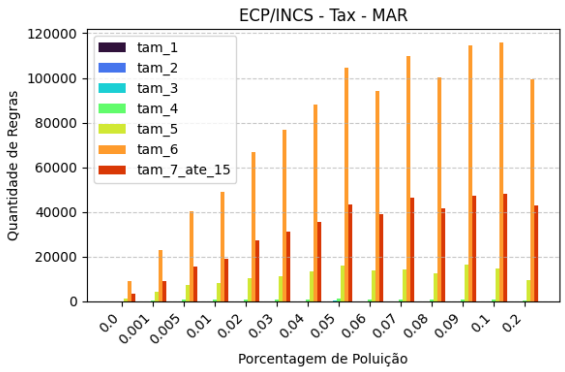



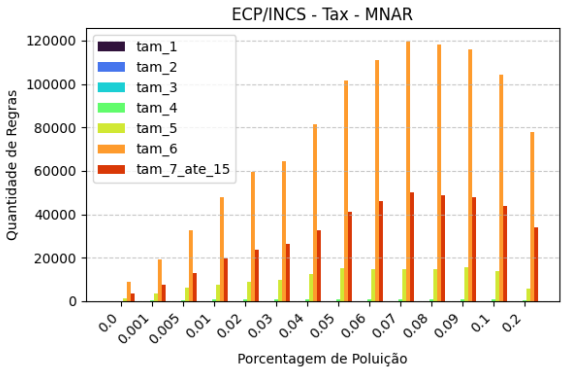
### A.3.7 Resultados ECP/INCS - Tax



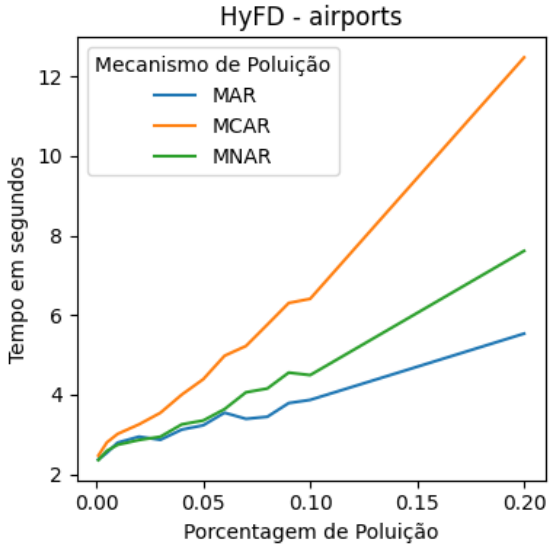



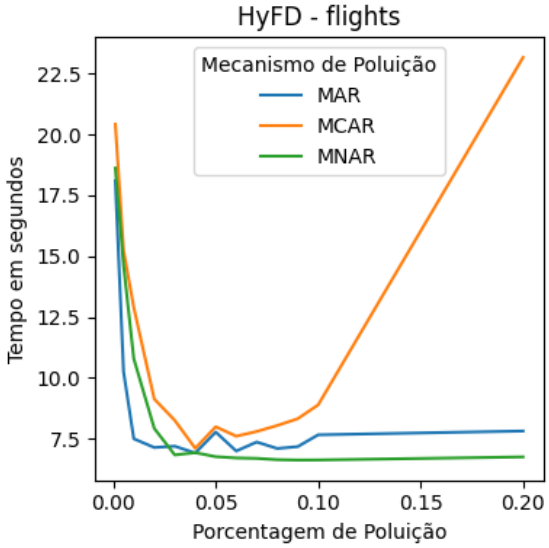



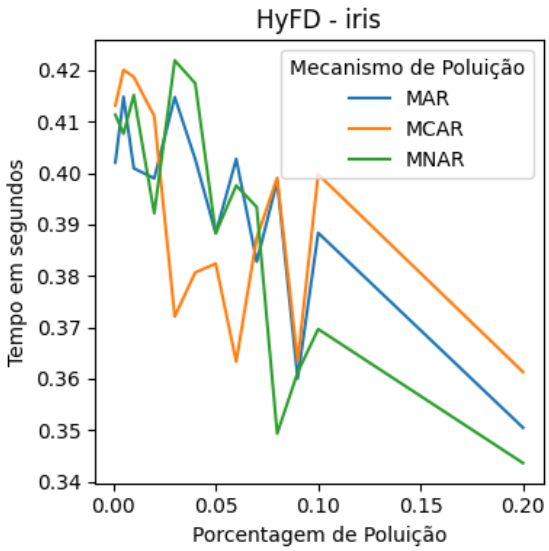



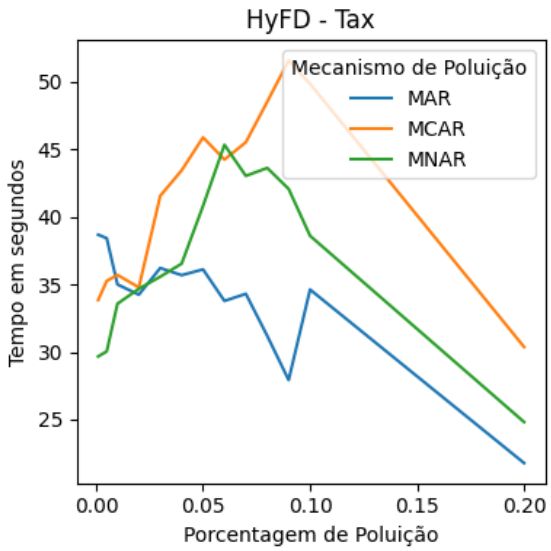



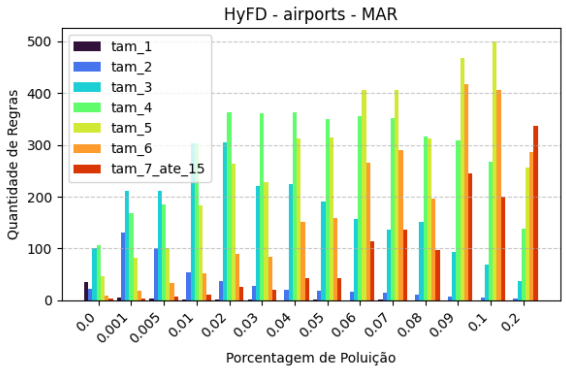
## A.4 RESULTADOS HYFD

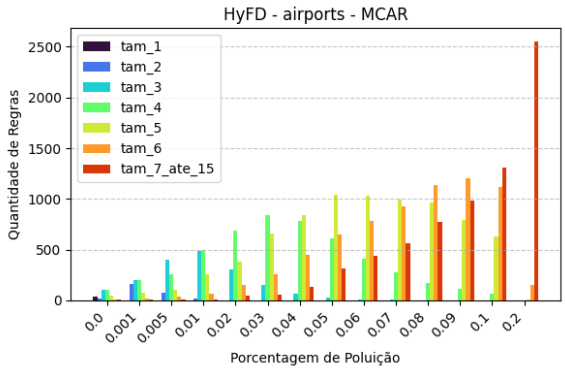


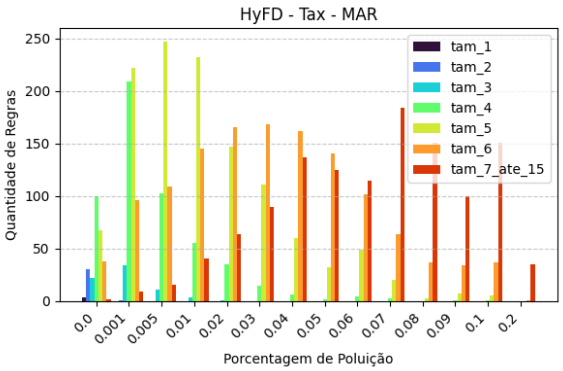



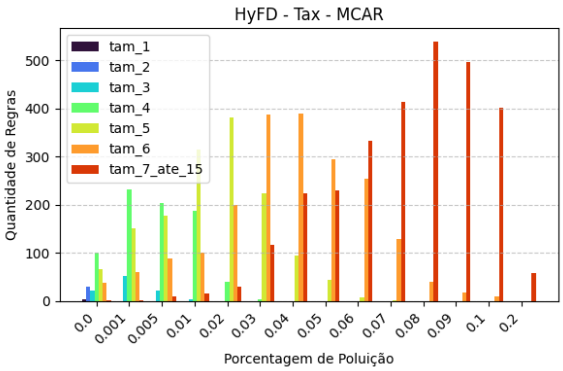



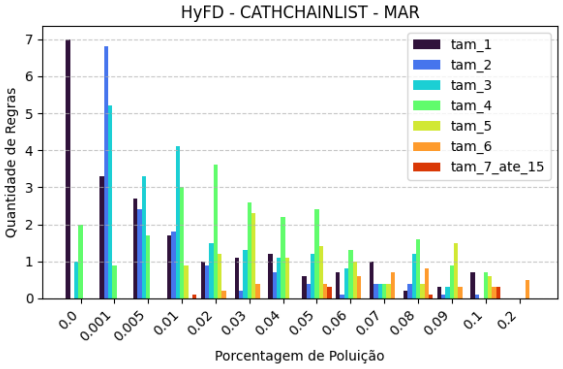



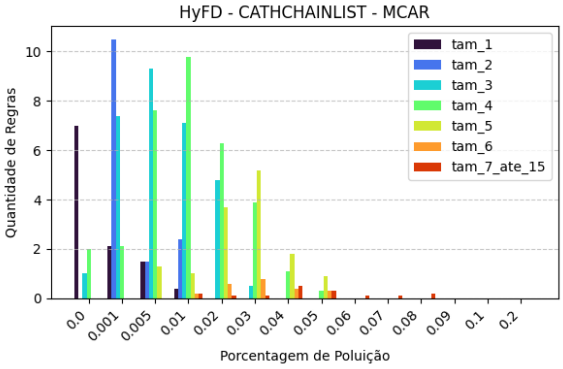



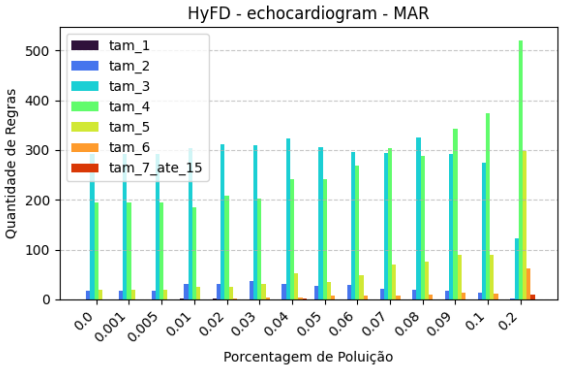



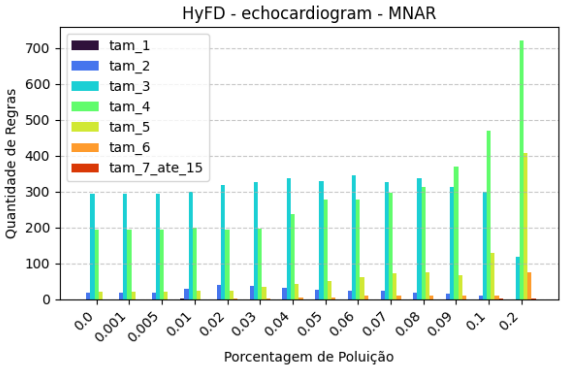

HyFD - Tax - MAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15

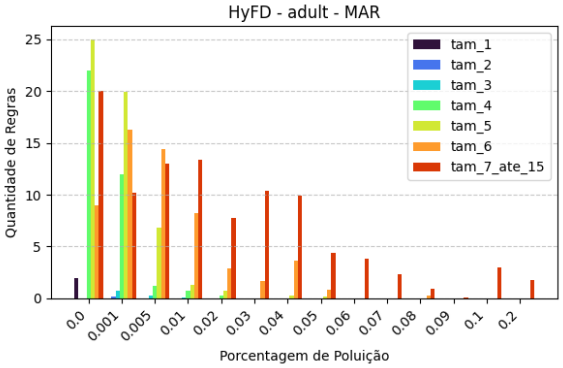

HyFD - Tax - MCAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15

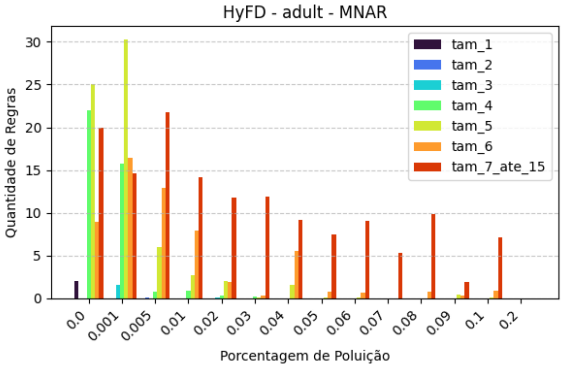

HyFD - CATHCHAINLIST - MAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15

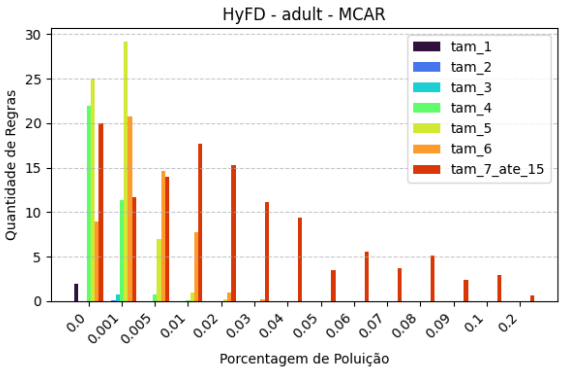

HyFD - CATHCHAINLIST - MCAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15

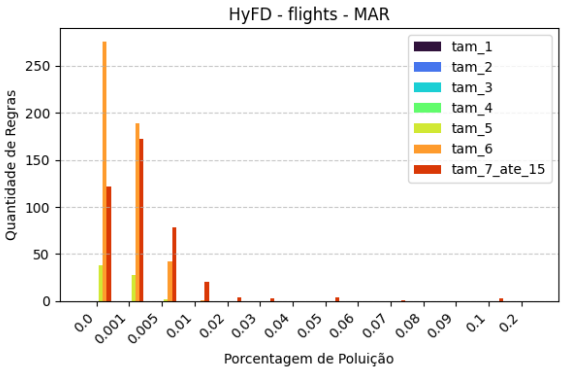

HyFD - echocardiogram - MAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15

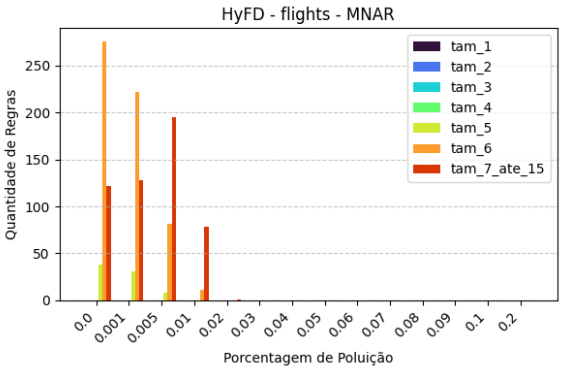

HyFD - echocardiogram - MNAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15

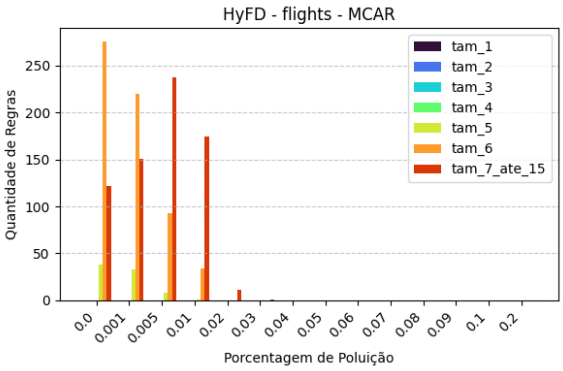

HyFD - adult - MAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15

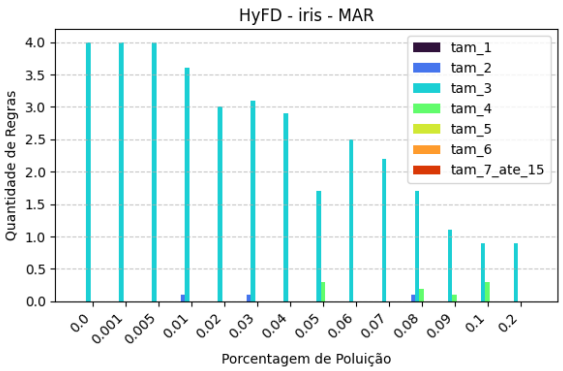

HyFD - adult - MNAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15

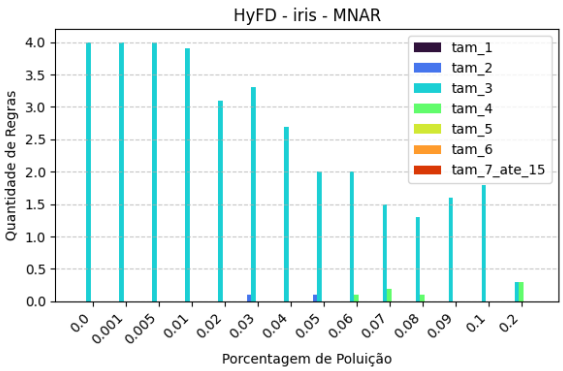

HyFD - adult - MCAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15

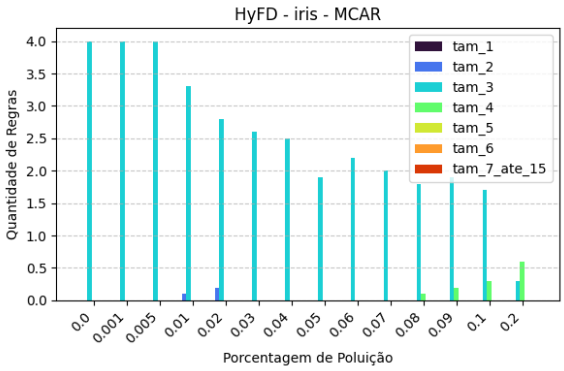

HyFD - flights - MAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15


HyFD - flights - MNAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15

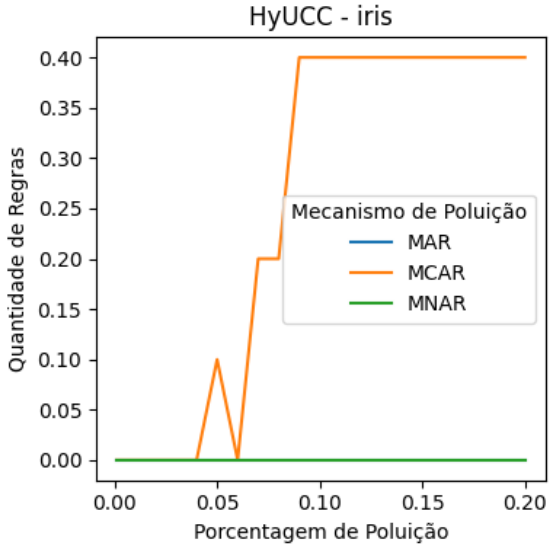

HyFD - flights - MCAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15

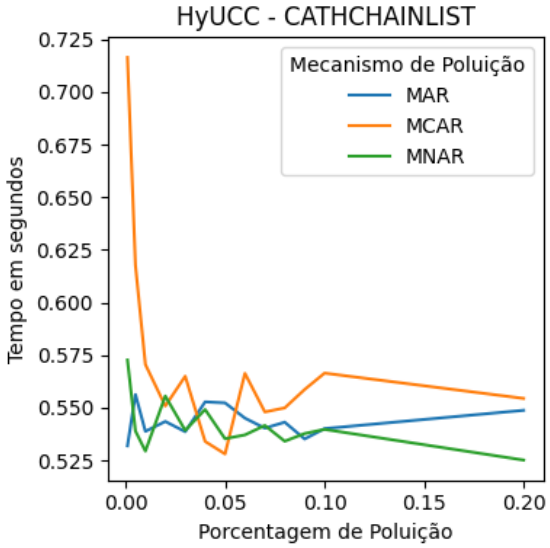

HyFD - iris - MAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15

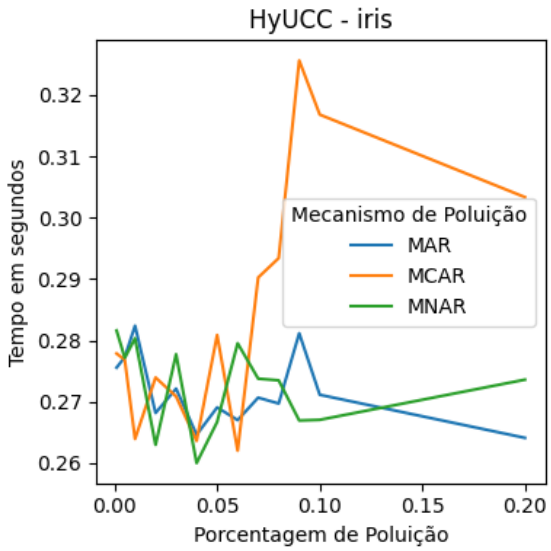

HyFD - iris - MNAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15

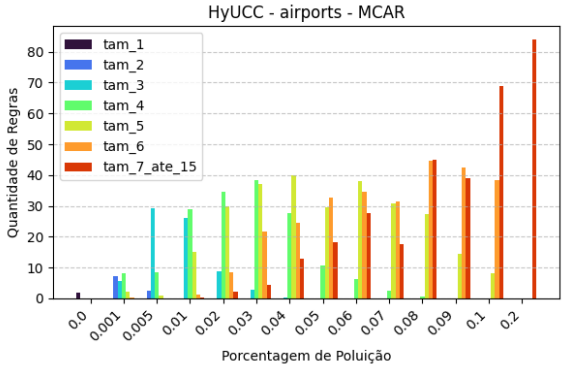

HyFD - iris - MCAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15

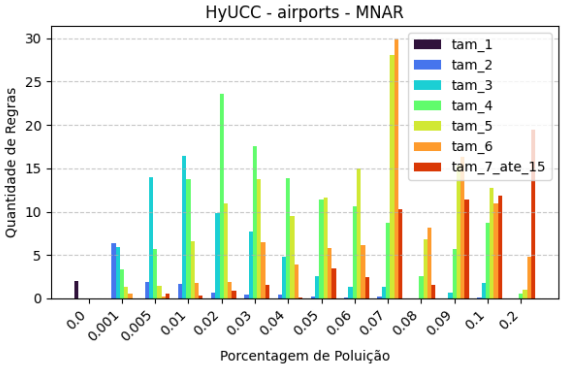
## A.5 RESULTADOS HYUCC

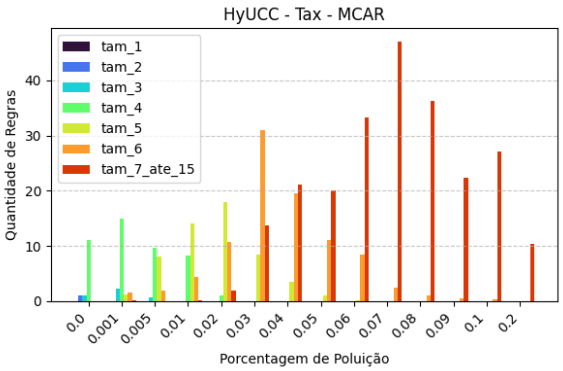


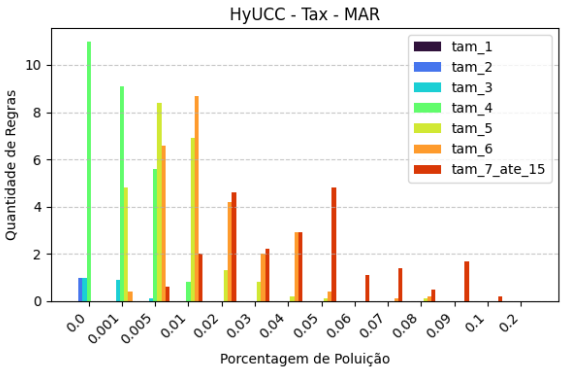



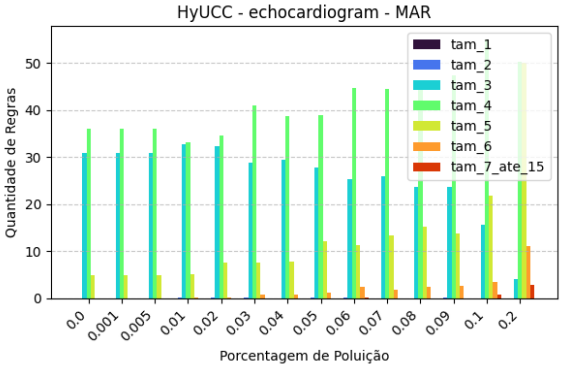



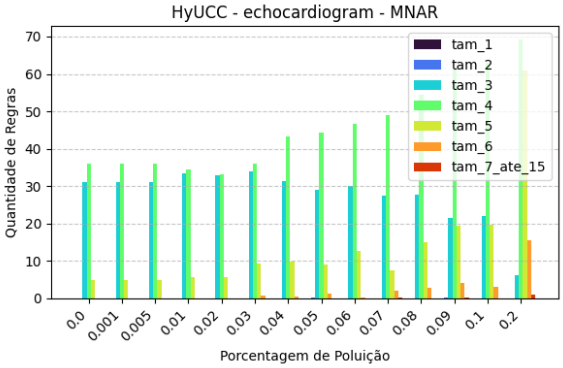



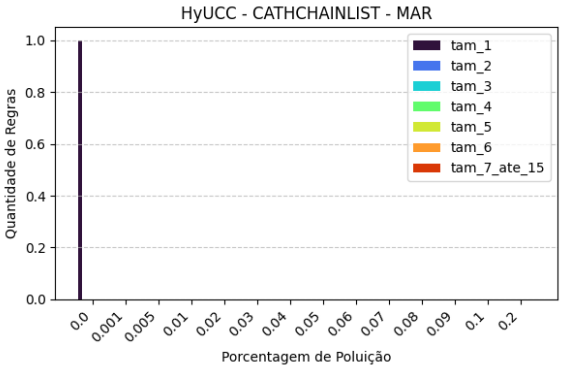



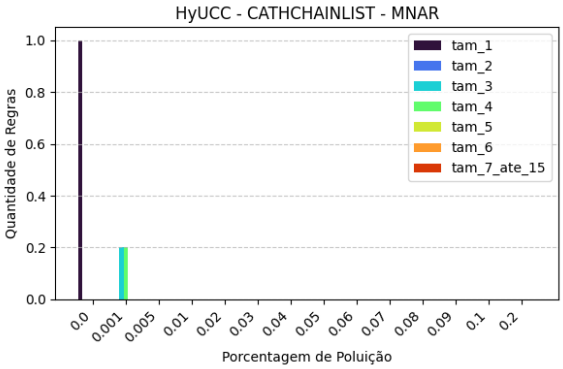

HyUCC - Tax - MCAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15

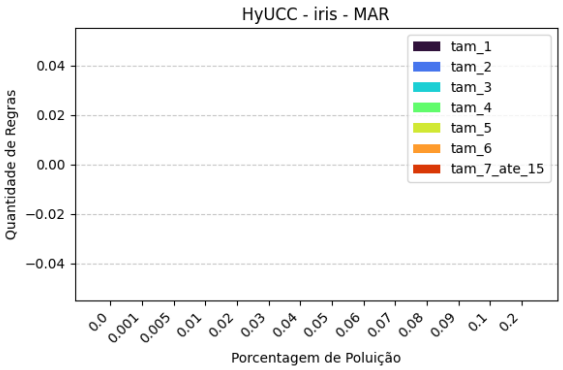

HyUCC - Tax - MAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15

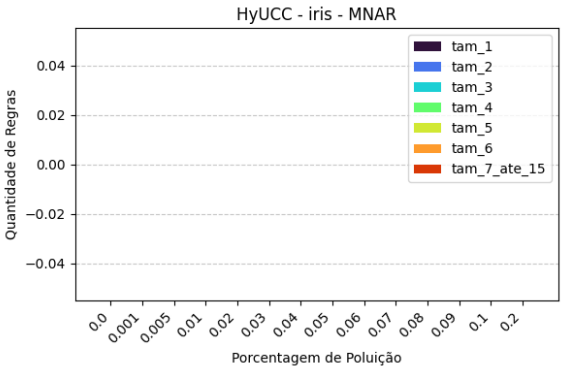

HyUCC - echocardiogram - MAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15

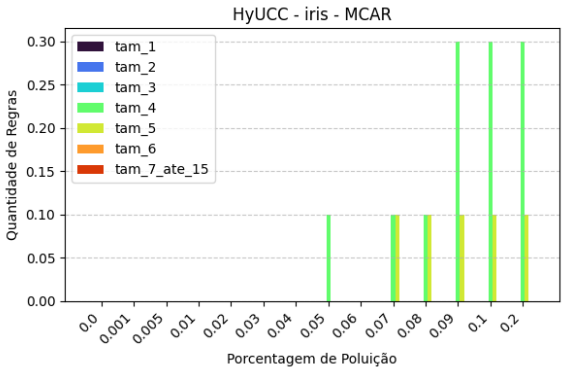

HyUCC - echocardiogram - MNAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15


HyUCC - CATHCHAINLIST - MAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15

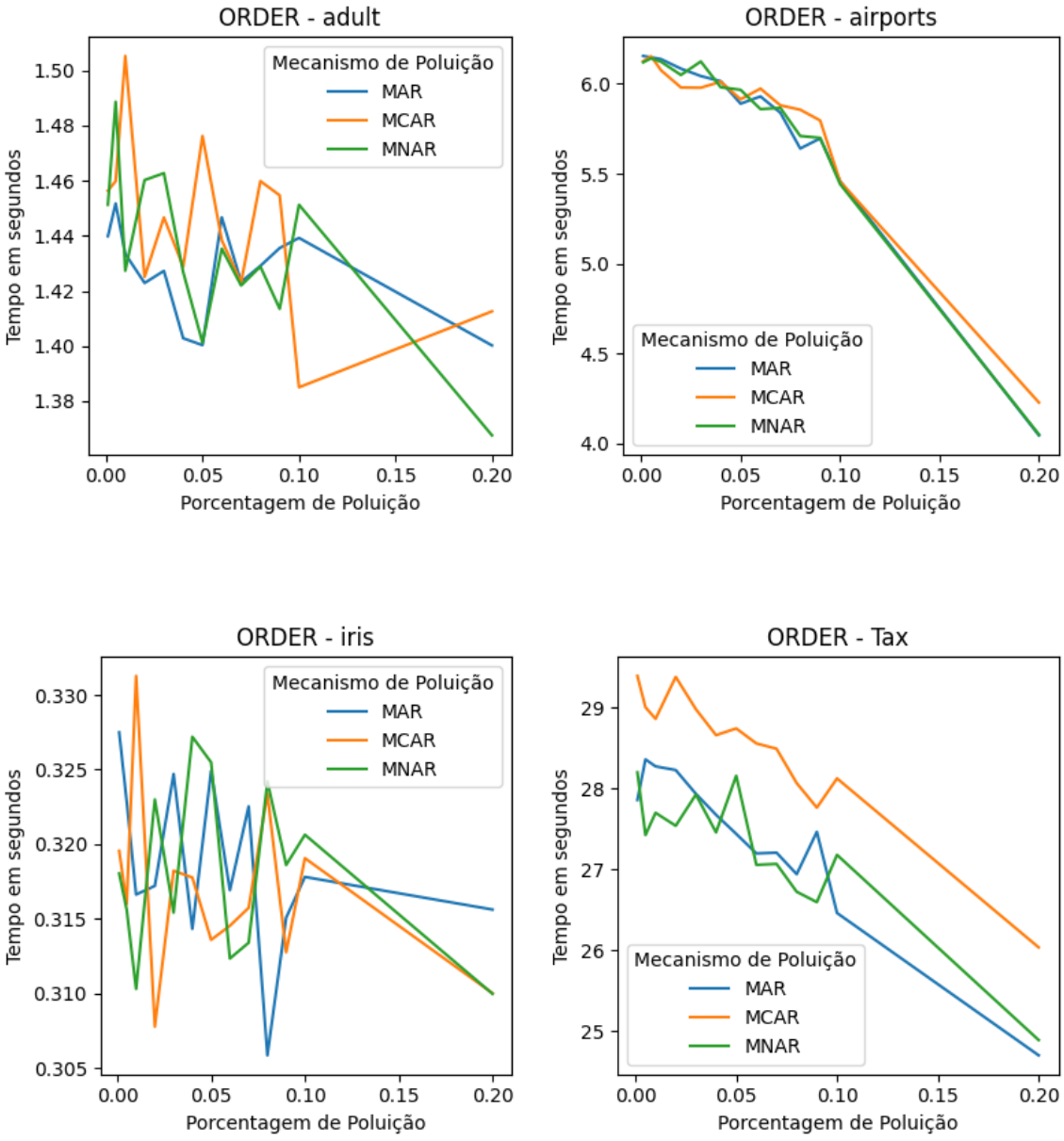

HyUCC - CATHCHAINLIST - MNAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15


HyUCC - iris - MAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15


HyUCC - iris - MNAR
Quantidade de Regras
Porcentagem de Poluição
tam_1
tam_2
tam_3
tam_4
tam_5
tam_6
tam_7_ate_15

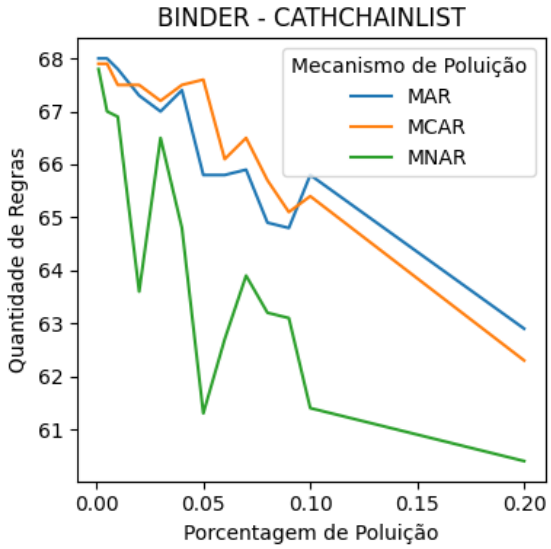


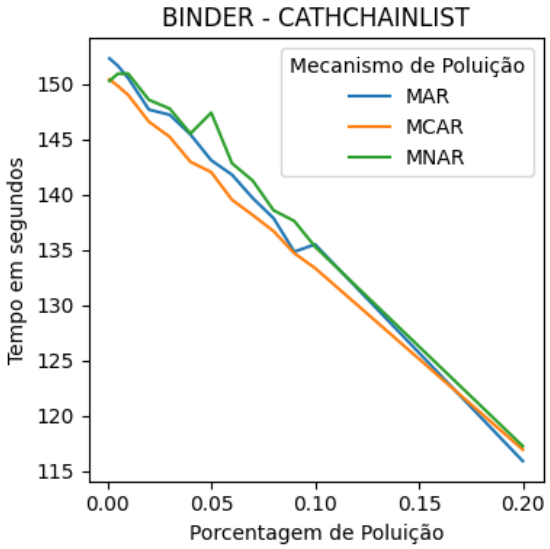

## A.6 RESULTADOS ORDER

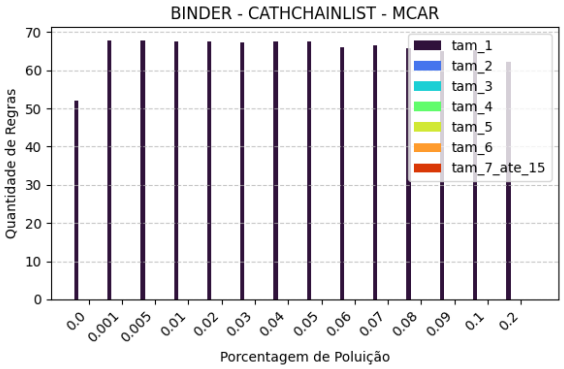


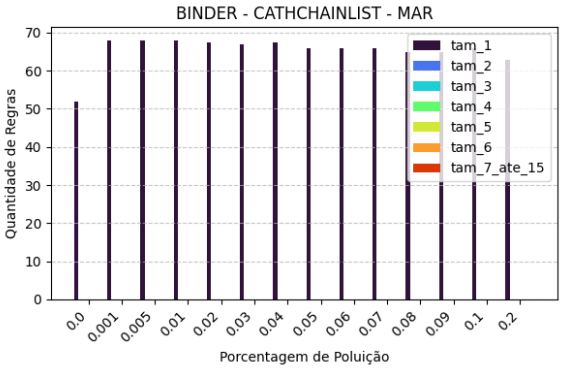
## A.7 RESULTADOS BINDER

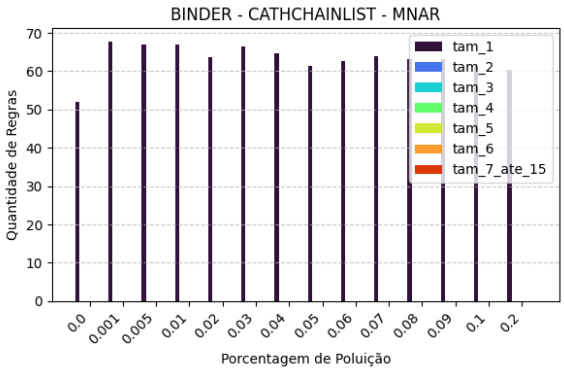
### A.7.1 Resultados BINDER - Unário - Cathchainlist



### A.7.2 Resultados BINDER - N-ário - Cathchainlist