Domain Knowledge and Functions in Data Science

Application to Hydroelectricity Production

Pierre Faure--Giovagnoli[1,2]

[1]Univ Lyon, INSA Lyon, CNRS, UCBL, LIRIS UMR 5205, Villeurbanne, France
[2]Compagnie Nationale du Rhône, Lyon, France

Thesis defense, November 2023

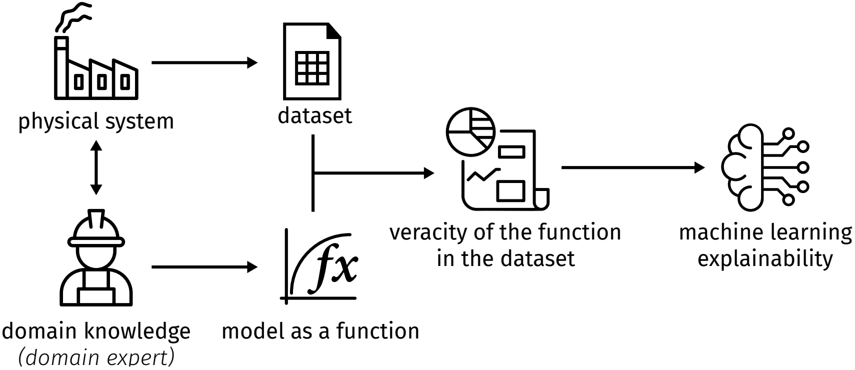
Sihem	AMER-YAHIA	Reviewer
Themis	PALPANAS	Reviewer
Frédérique	LAFOREST	Examiner
Pierre	SENELLART	Examiner
Marius	BOZGA	Examiner
Jean-Marc	PETIT	Advisor
Vasile-Marian	SCUTURICI	Advisor
Pierre	ROUMIEU	Guest

Funded by the **CNR**

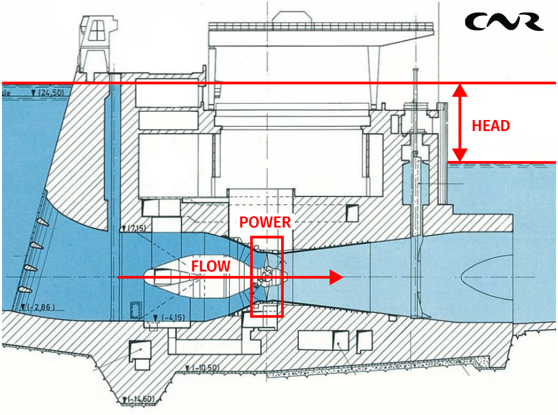
▹ Contents

1. Context
2. Framework presentation: *from functions to the relaxed g_3 indicator*
3. Contributions
   - Complexity analysis using the *properties of equality*
   - Algorithmics and the FASTG$_3$ python library
   - Application to supervised learning, the ADESIT web application
4. Conclusion and perspectives

▶ Context


physical system
dataset
veracity of the function
in the dataset
machine learning
explainability
domain knowledge
(domain expert)
model as a function

- 3 variables:
  - power (Megawatts)
  - flow ($m^3 \cdot s^{-1}$)
  - head (m)
- 2 constants:
  - water density ρ ($kg \cdot m^{-3}$)
  - turbine efficiency η (no unit)



Running example ▸ Domain knowledge [Cengel et al., 2010]

$$\text{power} = f_{\eta,\rho}(\text{flow}, \text{head}) = \eta \cdot \rho \cdot \text{flow} \cdot \text{head}$$

Context ▹ What about the recorded data?

FUNCTION

$$\text{power} = f_{\eta,\rho}(\text{flow}, \text{head})$$

?

DATASET

r	flow	head	power
t_1	2.5	10.1	22.9
t_2	2.7	10.4	23.2
t_3	2.6	10.3	23.0
t_4	2.5	10.2	23.3
t_5	2.6	10.1	23.1
t_6	2.6	10.3	22.9

- **How to evaluate the veracity of f in r?**
  Our study is three-fold:
  1. What is the complexity of this problem?
  2. How to solve it efficiently?
  3. How does that satisfaction relates to supervised learning?

- From functions to the relaxed g_3 indicator

- We focus on the deterministic nature of functions:

Property ▸ Function unicity

A function in the form $C = f(X)$ assigns to each element of X **exactly one element** of C.

- Thus, we measure the existence of *any function* with given inputs and outputs.

Running example ▸ Inputs and outputs

We do not consider the formula itself but only the inputs and outputs:

$$\boxed{\text{power} = f_{\eta,\rho}(\text{flow}, \text{elevation})} = \eta \cdot \rho \cdot \text{flow} \cdot \text{elevation}$$

- For a function $C = f(X)$, a functional dependency (FD) $X \rightarrow C$ expresses the same unicity constraint:

Definition ▸ Satisfaction of crisp FDs [Armstrong, 1974]

$X \rightarrow C$ is satisfied in a relation r (noted $r \models X \rightarrow C$) if:

$$\forall t_1, t_2 \in r, t_1[X] = t_2[X] \Rightarrow t_1[C] = t_2[C]$$

- We use FDs to study the existence of functions in data.

Running example ▸ From function to crisp FD

Thus, we can convert the function to a crisp FD:

$$\text{power} = f_{\eta,\rho}(\text{flow}, \text{head}) \xRightarrow{\textit{becomes}} \text{flow}, \text{head} \rightarrow \text{power}$$

- A counterexample violates the FD and **its associated function**!

Definition ▶ Counterexample

A counterexample of a FD in the form $X \rightarrow C$ is a pair of tuples which have similar values on X and dissimilar values on C.

Running example ▶ Our first counterexample

	X		C
r	flow	head	power
t_1	2.5	10.1	22.9
t_2	2.7	10.4	23.2
t_3	2.6	10.3	23.0
t_4	2.5	10.2	23.3
t_5	2.6	10.1	23.1
t_6	2.6	10.3	22.9

$\{(t_3, t_6)\} \not\models \text{flow}, \text{head} \rightarrow \text{power}$

- Real-life problems
  - 👎 may not hold on the *whole dataset*
  - 👎 equality is *too restrictive*
- Solutions
  - 👍 use a *coverage indicator* to measure the *partial* validity
  - 👍 use *predicates* instead of equality

- A coverage indicator measures the veracity of a FD in a relation.
  - This provides a greater nuance over the classical binary FD satisfaction.

- Most common: *the g_3 indicator* [Kivinen and al., 1995]:

  The g_3 indicator is the minimum proportion of tuples to remove from a relation such that no counterexample remains.

- More formally:

Definition ▸ g_3 indicator

For a relation r and a FD in the form $X \rightarrow C$:

$$g_3(X \rightarrow C, r) = 1 - \frac{\max(|\{s \mid s \subseteq r, s \models X \rightarrow C\}|)}{|r|}$$

Running example ▶ Computing g_3 with crisp FDs

id	flow	head	power
t_1	2.5	10.1	22.9
t_2	2.7	10.4	23.2
~~t_3~~	~~2.6~~	~~10.3~~	~~23.0~~
t_4	2.5	10.2	23.3
t_5	2.6	10.1	23.1
t_6	2.6	10.3	22.9

φ : flow, head $\rightarrow$ power

$\{(t_3, t_6)\} \not\models \varphi$

$g_3(\varphi, r) = \frac{1}{6}$

<u>Reminder</u>

The g_3 indicator is the <u>minimum</u> proportion of tuples to remove from a relation such that no counterexample remains.

- *Crisp equality* not sufficient in real life $\Rightarrow$ replace equality by *predicates*.
- Each attribute A is equipped with a *binary predicate* comparing every two values in the *domain* (dom) of A: $\phi_A: \mathtt{dom}(A) \times \mathtt{dom}(A) \longrightarrow \{\mathtt{true}, \mathtt{false}\}$
- Similar to [Caruccio and al., 2015], the satisfaction can be redefined:

Definition ▸ Satisfaction of non-crisp FDs

The satisfaction of a FD $X \longrightarrow C$ in a relation r in regard to a set of predicates Φ (noted $r \models_\Phi X \longrightarrow C$) is defined as:

$$\forall t_1, t_2 \in r, \bigwedge_{A_i \in X} \phi_i(t_1[A_i], t_2[A_i]) \Rightarrow \phi_C(t_1[C], t_2[C])$$

- Covers many FD relaxations from literature [Caruccio and al., 2015, Song et al., 2020].

Running example ▶ Defining predicates

To take sensor uncertainties into account, we can associate an absolute distance to each attribute:

$$\phi_{\text{flow}}(x,y) = \phi_{\text{head}}(x,y) = \phi_{\text{power}}(x,y) = \begin{cases} \texttt{true} & \text{if } |x - y| \leq 0.1 \\ \texttt{false} & \text{otherwise.} \end{cases}$$

- We can adapt the definition of g_3 to FDs with predicates:

Definition ▶ g_3 indicator with predicates

For a relation r, a FD in the form $X \rightarrow C$ and a set of predicates Φ:

$$g_3{}^{\Phi}(X \rightarrow C, r) = 1 - \frac{\max(|\{s \mid s \subseteq r, s \models_{\Phi} X \rightarrow C\}|)}{|r|}$$

Running example ▸ Computing g_3 with non-crisp FDs

$$\varphi : \text{flow}, \text{head} \rightarrow \text{power}$$

$$\phi_{\text{flow}}(x,y) = \phi_{\text{head}}(x,y) = \phi_{\text{power}}(x,y) = \begin{cases} \texttt{true} & \text{if } |x-y| \leq 0.1 \\ \texttt{false} & \text{otherwise.} \end{cases}$$

r	flow	head	power
t_1	2.5	10.1	22.9
~~t_2~~	~~2.7~~	~~10.4~~	~~23.2~~
t_3	2.6	10.3	23.0
~~t_4~~	~~2.5~~	~~10.2~~	~~23.3~~
~~t_5~~	~~2.6~~	~~10.1~~	~~23.1~~
t_6	2.6	10.3	22.9

$$\{(t_1,t_5),(t_1,t_4),(t_4,t_5),(t_4,t_3),(t_4,t_6),(t_5,t_6),(t_3,t_2),(t_2,t_6)\} \not\models_{\Phi} \varphi$$

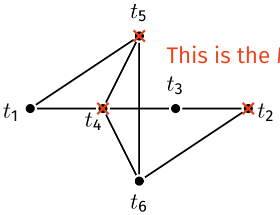
$$g_3^{\Phi}(\varphi, r) = \frac{3}{6} = 0.5$$

Running example ▶ Switching to conflict graph

$$\varphi : \text{flow}, \text{head} \rightarrow \text{power}$$

$$\phi_{\text{flow}}(x, y) = \phi_{\text{head}}(x, y) = \phi_{\text{power}}(x, y) = \begin{cases} \texttt{true} & \text{if } |x - y| \leq 0.1 \\ \texttt{false} & \text{otherwise.} \end{cases}$$

r	flow	head	power
t_1	2.5	10.1	22.9
~~t_2~~	~~2.7~~	~~10.4~~	~~23.2~~
t_3	2.6	10.3	23.0
~~t_4~~	~~2.5~~	~~10.2~~	~~23.3~~
~~t_5~~	~~2.6~~	~~10.1~~	~~23.1~~
t_6	2.6	10.3	22.9



This is the MVC!

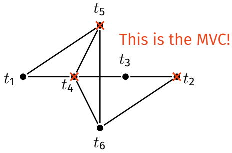
$$g_3^{\Phi}(r, \varphi) = |\text{MVC}(\text{CG}_{\Phi}(r, \varphi))| / |r|$$

- This is called the conflict graph (CG) [Bertossi, 2011].

$$\varphi : \text{flow}, \text{head} \rightarrow \text{power}$$

$$\phi_{\text{flow}}(x,y) = \phi_{\text{head}}(x,y) = \phi_{\text{power}}(x,y) = \begin{cases} \texttt{true} & \text{if } |x-y| \leq 0.1 \\ \texttt{false} & \text{otherwise.} \end{cases}$$

r	flow	head	power
t_1	2.5	10.1	22.9
~~t_2~~	~~2.7~~	~~10.4~~	~~23.2~~
t_3	2.6	10.3	23.0
~~t_4~~	~~2.5~~	~~10.2~~	~~23.3~~
~~t_5~~	~~2.6~~	~~10.1~~	~~23.1~~
t_6	2.6	10.3	22.9



$g_3^{\Phi}(r,\varphi) = |\text{MVC}(\text{CG}_{\Phi}(r,\varphi))|/|r|$

- g_3 corresponds to the size of a minimum vertex cover (MVC) in CG [Song, 2010].

- Hardness of computing g_3:
  - 👍 Crisp FDs: Polynomial (e.g. [Huhtala et al., 1999]).
  - 👎 Non-crisp FDs: NP-Hard (reduction derived from [Song, 2010]).

$C = f(X) \longrightarrow X \rightarrow C$

crisp satisfaction

$$\forall t_1, t_2 \in r, t_1[X] = t_2[X] \Rightarrow t_1[C] = t_2[C] \xrightarrow{\text{poly}} g_3$$

non-crisp satisfaction

$$\forall t_1, t_2 \in r, \bigwedge_{A_i \in X} \phi_i(t_1[A_i], t_2[A_i]) \Rightarrow \phi_c(t_1[C], t_2[C]) \xrightarrow[\text{NP-hard}]{} g_3^{\Phi}$$

Let's examine this in more detail!

- Complexity analysis

- For studying the hardness of computing g_3, with use the decision version:

Problem ▶ Error Validation Problem with Predicates (EVPP)

In: a relation scheme with predicates (R, Φ), a relation r and a FD $X \rightarrow A$ over R, $k \in \mathbb{R}$.

Out: `YES` if $g_3^\Phi(X \rightarrow A, r) \leq k$, `NO` otherwise.

- The results naturally extends to the optimization problem.

- about the complexity of EVPP:
  - polynomial for usual FDs with equality [Huhtala et al., 1999].
  - NP-complete for non-crisp FDs [Faure--Giovagnoli et al., 2022].
- *what makes the problem tractable (or not)?*
  - *idea:* study the impact of (common) *predicates properties* on EVPP:

    (ref): $\phi_A(x,x) = \texttt{true}$

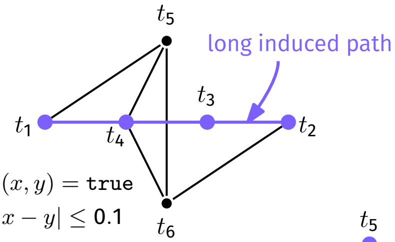
    (sym): $\phi_A(x,y) = \texttt{true}$ implies $\phi_A(y,x) = \texttt{true}$

    (tra): $\phi_A(x,y) = \phi_A(y,z) = \texttt{true}$ implies $\phi_A(x,z) = \texttt{true}$

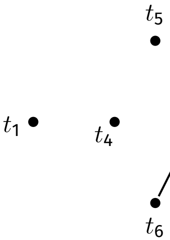
    (asym): $\phi_A(x,y) = \phi_A(y,x) = \texttt{true}$ implies $x = y$
  - *goal:* a quick-reference map of EVPP complexity

- The *properties* of the predicates bound the *structure* of the conflict-graph!

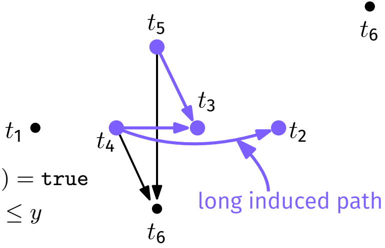
$\mathrm{CG}_\Phi(r, \text{flow}, \text{head} \rightarrow \text{power})$ with $\phi_{power} = \phi_{flow} = \phi_{head}$



$\phi_{power}(x, y) = \texttt{true} \iff |x - y| \leq 0.1$

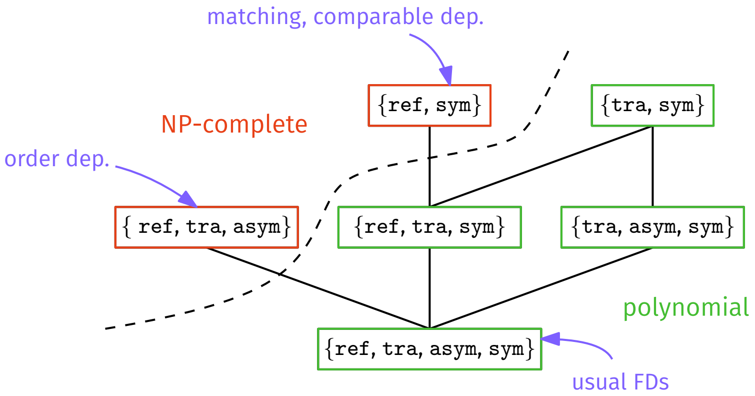


$\phi_{power}(x, y) = \texttt{true} \iff x = y$

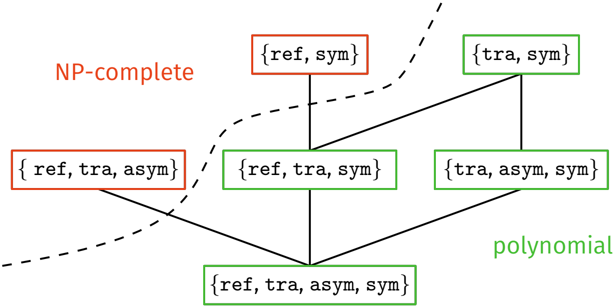


$\phi_{power}(x, y) = \texttt{true} \iff x \leq y$

- The *properties* of the predicates bound the *structure* of the conflict-graph! [Faure--Giovagnoli et al., 2023]



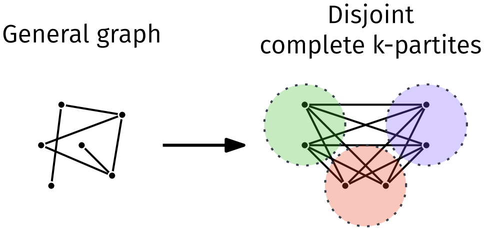
- Algorithmics



- Two cases:
  1. Polynomial algorithms for tra. and sym. predicates.
  2. The general case, a NP-hard problem.

Algorithmics ▶ Tra. et sym. predicates (polynomial)

The graph is now constrained:



- Very efficient polynomial exact and approx. algorithms can be developed!

$g_3(A \rightarrow C, r)$ can be computed in polynomial time [Kivinen and al., 1995]:

r	A	C
t_0	0	0
t_1	0	1
t_2	0	1
t_3	1	1
t_4	1	0

1. Group by antecedents
2. Find the most frequent element in each group
3. Count the tuples in minority
   - Those are the tuples to suppress to remove all counterexamples
4. Normalize by the size of the relation: $g_3(A \rightarrow C, r) = \frac{|\{t_0, t_3\}|}{|r|} = \frac{2}{5}$

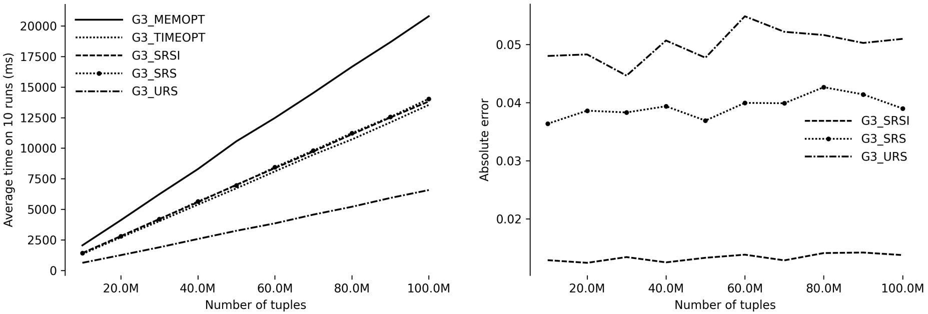
Two alternatives for the *Group By*:

- Hashing
  - Keep all groups in memory while tracking the most frequent element in each group
  - Linear complexity in $|r|$
  - High memory usage
- Sorting
  - Sort the dataset and then iterate through the tuples in one pass
  - Log-linear complexity in $|r|$
  - Can be low in memory usage via external sorting

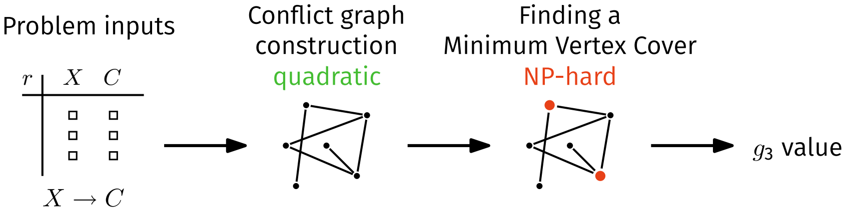
In large datasets, sampling procedures:

- Uniform Random Sampling
  - Exact algorithm with a random subset of the full relation
- Stratified Random Sampling (adapted from [Cormode and al., 2009])
  1. First pass: estimate the size of each group on random subset of the full relation
  2. Second pass: reservoir sample fixed number of tuples in each group to find most frequent elements
  3. Compute g_3 with weighted average
- Improved Stratified Random Sampling
  - Same process as before but sample a variable number of tuples in second pass:
    - The number is proportional to the estimated size of the group (step 1)
    - Based on Serfling's inequality [Serfling, 1974] - *Hoeffding's with finite population correction*

Exact and approximate algorithms for computing g_3 with tra. and sym. predicates:

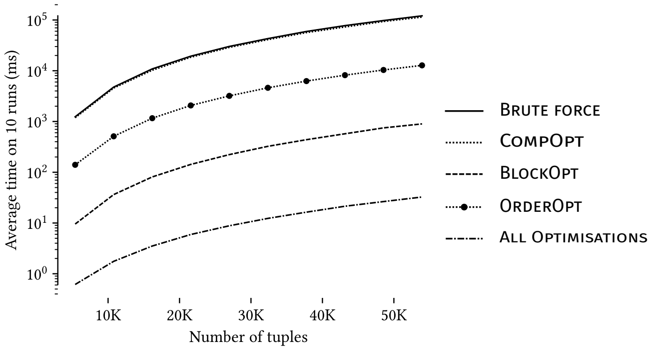


Algorithmics ▸ General case (NP-hard)

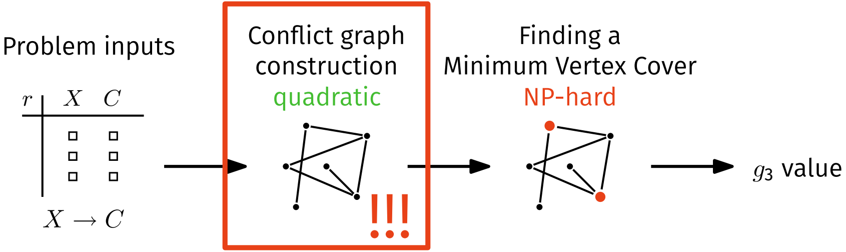


- Two steps:
  1. Constructing the conflict graph.
     - Nodes are the tuples.
     - Edges are constructed via *counterexample enumeration*.
       Costly quadratic process in $|r|$
       Potential optimizations drawn from record linkage and similarity joins
  2. Evaluating a *Minimum Vertex Cover*.
     - Exact solvers - exponential in the number of edges (e.g. [Hespe et al., 2020])
     - Solvers with heuristics - no guarantees (e.g. [Cai et al., 2013])
     - Approximation algorithms - Edge Deletion, Greedy Independent Cover...

Comparison of various optimizations for constructing the conflict graph:



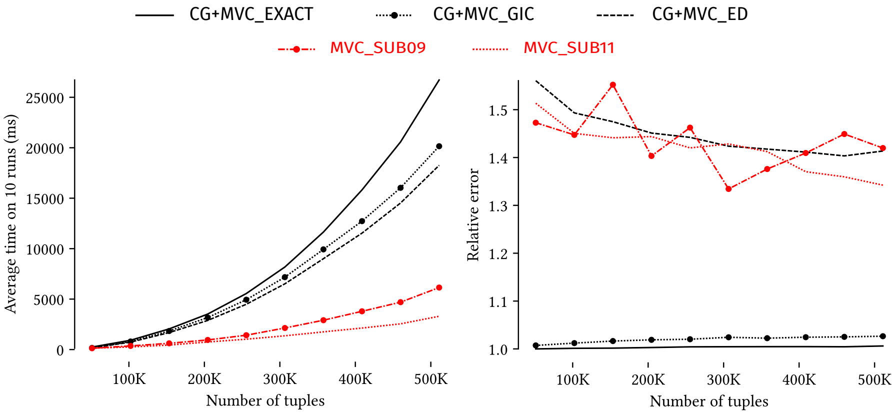
Problem: the conflict graph construction is the bottleneck!



Solution: sublinear algorithms.

- They **do not** construct the whole graph.
- On-the-fly counterexample enumeration.
- Algorithms adapted from [Yoshida et al., 2009] and [Onak et al., 2012].
  - Good time performance
  - Average accuracy

Exact, approximate and sublinear algorithms for computing g_3 in the general case:



Algorithmics ▶ FASTG3

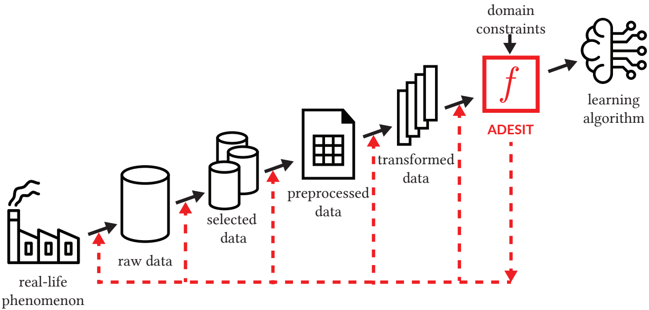
- **Python library** for computing the relaxed g_3 indicator.
- **Open-source** available on GitHub: github.com/datavalor/fastg3
- Implements all the algorithms mentioned previously.
- **Implemented in C++** with intuitive Python interface.

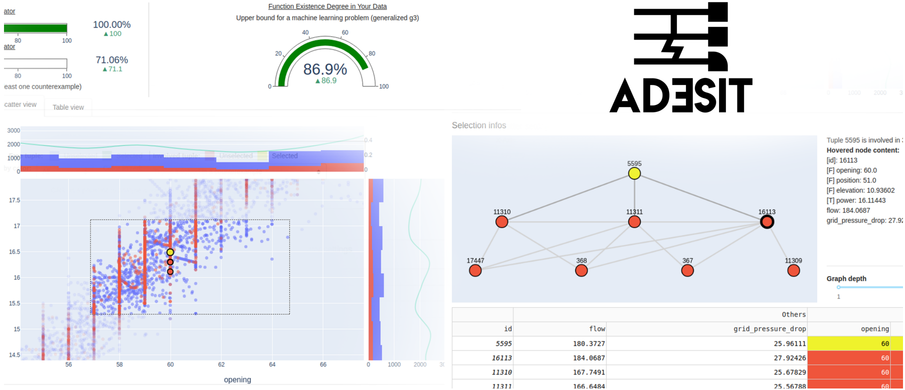
- Counterexample analysis for supervised learning

- In supervised learning, we *learn* a function. Does it really exist?

- Consider a supervised learning problem we want to learn C from features X from relation r (i.i.d.).
  - [Le Guilly et al., 2020] shows that $g_3(r, X \rightarrow C)$ bounds the accuracy of any model.
  - When $|r|$ tends to infinity, it corresponds the Bayes error rate for this process!

- **Our proposition: ADESIT.** A tool for interactive counterexample analysis.





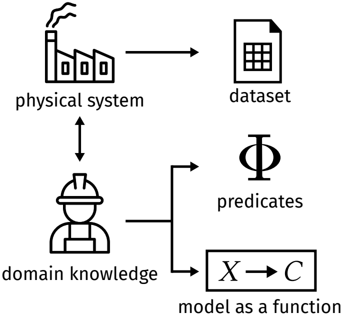
- Web application for **counterexample analysis**.
- Demonstration available at: adesit.liris.cnrs.fr
- **Open-source** available on GitHub: github.com/datavalor/adesit
- Based on FASTG3.

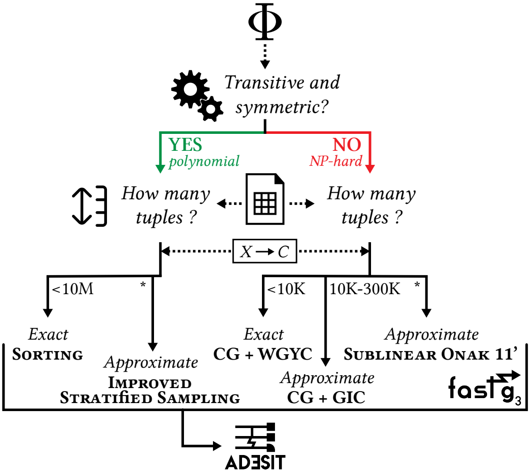
▶ Conclusion and perspectives

- Framework for measuring the existence of a function in a dataset.
  - *Functions existence* can be modeled by *functional dependencies*.
  - *Equality* can be replaced by *predicates*.
  - The g_3*-error* measures the veracity of a *FD*/function in a dataset.

- Contributions
  - Complexity dichotomy based on properties of equality [Faure--Giovagnoli et al., 2023].
    - Polynomial when predicates at least tra. and sym.
  - Algorithmic solutions for computing the g_3 indicator [Faure--Giovagnoli et al., 2022].
    - The polynomial case: scalable, good sampling approaches.
    - The NP-hard case: less scalable due to CG, sublinear faster but less accurate.
    - The FASTG3 python library.
  - Application to supervised learning [Faure--Giovagnoli et al., 2021].
    - The ADESIT web application.
    - Link to accuracy and Bayes error.

Inputs

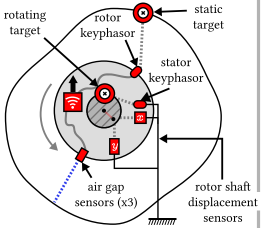

physical system
dataset
domain knowledge
predicates
X → C
model as a function

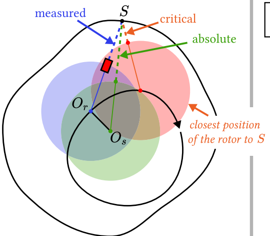

Φ
Transitive and symmetric?
YES
polynomial
NO
NP-hard
How many tuples ?
How many tuples ?
X → C
<10M
*
<10K
10K-300K
*
Exact
Sorting
Approximate
Improved Stratified Sampling
Exact
CG + WGYC
Approximate
CG + GIC
Approximate
Sublinear Onak 11'
fastg3
AD3SIT

- Link between the Bayes error and the relaxed g_3 indicator
  - What happens when you relax equality?

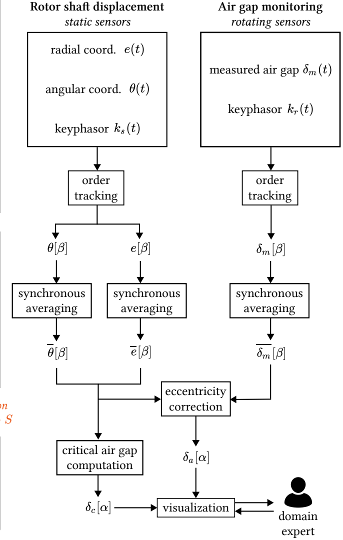
- Designing a new sub-linear algorithm with better approximation in practice...
  - What makes an algorithm possible to adapt into sublinear?
  - Replacing edge deletion with Sorted List Right [Laforest et al., 2008].

Problem

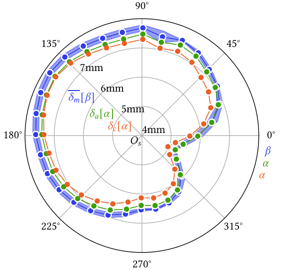


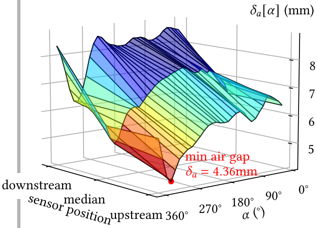


Solution



Results





Thank you for listening!

- Armstrong, William Ward
  Dependency Structures of Data Base Relationships
  IFIP congress, 1974.
- Serfling, Robert J
  Probability inequalities for the sum in sampling without replacement
  The Annals of Statistics, 1974.
- Kivinen, Jyrki and Mannila, Heikki
  Approximate inference of functional dependencies from relations
  Theoretical Computer Science, 1995.
- Papadimitriou, Christos H., and Kenneth Steiglitz
  Combinatorial optimization: algorithms and complexity
  Courier Corporation, 1998.
- Y. Huhtala, J. Kärkkäinen, P. Porkka, H. Toivonen
  TANE: An efficient algorithm for discovering functional and approximate dependencies.
  The computer journal, vol. 42, p. 100–111, 1999.
- Bassée, Renaud and Wijsen, Jef
  Neighborhood dependencies for prediction
  Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2001.
- Parnas, Michal, and Dana Ron
  Approximating the minimum vertex cover in sublinear time and a onnection to distributed algorithms.
  Theoretical Computer Science, 2007.

- Nguyen, Huy N., and Krzysztof Onak.
  Constant-time approximation algorithms via local improvements.
  49th Annual IEEE Symposium on Foundations of Computer Science, 2008.
- Delbot, François and Laforest, Christian
  A better list heuristic for vertex cover
  Information Processing Letters, 2008.
- Cormode, Graham and Golab, Lukasz and Flip, Korn and McGregor, Andrew and Srivastava, Divesh and Zhang, Xi
  Estimating the Confidence of Conditional Functional Dependencies
  SIGMOD International Conference on Management of Data, 2009.
- Yoshida, Yuichi and Yamamoto, Masaki and Ito, Hiro
  An improved constant-time approximation algorithm for maximum˜ matchings
  ACM symposium on Theory of computing, 2009.
- Cengel, Yunus A
  Fluid mechanics
  Tata McGraw-Hill Education, 2010.
- Song, Shaoxu
  Data dependencies in the presence of difference
  Hong Kong University of Science and Technology, 2010.
- L. Bertossi
  Database repairing and consistent query answering.
  Synthesis Lectures on Data Management, vol. 3, p. 1–121, 2011.

- Song, Shaoxu and Chen, Lei
  Differential dependencies: Reasoning and discovery
  ACM Transactions on Database Systems, 2011.
- Levene, Mark and Loizou, George
  A guided tour of relational databases and beyond
  Springer Science & Business Media, 2012.
- Onak, Krzysztof and Ron, Dana and Rosen, Michal and Rubinfeld, Ronitt
  A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size
  ACM-SIAM symposium on Discrete Algorithms, 2012.
- Baixeries, Jaume and Kaytoue, Mehdi and Napoli, Amedeo
  Computing similarity dependencies with pattern structures
  Conference on Concept Lattices and their Applications-CLA, 2013.
- Cai, Shaowei and Su, Kaile and Luo, Chuan and Sattar, Abdul
  NuMVC: An efficient local search algorithm for minimum vertex cover
  Journal of Artificial Intelligence Research, 2013.
- Caruccio, Loredana and Deufemia, Vincenzo and Polese, Giuseppe
  Relaxed functional dependencies—a survey of approaches
  IEEE Transactions on Knowledge and Data Engineering, 2015.
- S. Song, F. Gao, R. Huang, and C. Wang
  Data Dependencies over Big Data: A Family Tree.
  IEEE Transactions on Knowledge and Data Engineering, 2020.

- Hespe, Demian and Lamm, Sebastian and Schulz, Christian and Strash, Darren
  WeGotYouCovered: The Winning Solver from the PACE 2019 Challenge
  SIAM Workshop on Combinatorial Scientific Computing, 2020.
- Marie Le Guilly, Jean-Marc Petit and Vasile-Marian Scuturici
  Evaluating Classification Feasibility Using Functional Dependencies
  Trans. Large Scale Data Knowl. Centered Syst., 2020.
- Faure--Giovagnoli, Pierre and Petit, Jean-Marc and Scuturici, Vasile-Marian and Le Guilly, Marie
  ADESIT: Visualize the Limits of your Data in a Machine Learning Process
  International Conference on Very Large Data Bases, 2021.
- Faure--Giovagnoli, Pierre and Petit, Jean-Marc and Scuturici, Vasile-Marian
  Assessing the Existence of a Function in your Dataset with the g3 Indicator
  38th IEEE International Conference on Data Engineering, 2022.
- S. Vilmin, P. Faure--Giovagnoli, J-M. Petit, V-M. Scuturici
  Functional dependencies with predicates: what makes the g3-error easy to compute?
  ICCS 2023