Numéro d'ordre : 2023ISAL0093

Institut National des Sciences Appliquées de Lyon

Ecole Doctorale N°512 Infomaths, Lyon

Thèse présentée par

Pierre FAURE--GIOVAGNOLI

Pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

Spécialité INFORMATIQUE

Domain Knowledge and Functions in Data Science

Application to Hydroelectricity Production

Soutenue publiquement le 24 novembre 2023 devant le jury constitué de :

Sihem AMER-YAHIA	*Directrice de recherche*	*Université Grenoble Alpes*	RAPPORTEUSE
Themis PALPANAS	*Professeur*	*Université Paris Cité*	RAPPORTEUR
Frédérique LAFOREST	*Professeure*	*INSA Lyon*	EXAMINATRICE
Pierre SENELLART	*Professeur*	*École Normale Supérieure*	EXAMINATEUR
Marius BOZGA	*Ingénieur de recherche*	*Université Grenoble Alpes*	EXAMINATEUR
Jean-Marc PETIT	*Professeur*	*INSA Lyon*	DIRECTEUR
Vasile-Marian SCUTURICI	*Professeur*	*INSA Lyon*	DIRECTEUR

Étant donné un mur, que se passe-t-il derrière ?
Jean Tardieu

Remerciements

Cette thèse a été une aventure exaltante qui m'a considérablement appris, tant sur le plan scientifique que personnel. Néanmoins, en ces froides journées de novembre, mon voyage doit s'achever. Un voyage qui n'aurait pas pu se faire sans de nombreuses personnes que je tente de remercier, du mieux que je peux, dans les quelques lignes qui suivent.

Mes premiers remerciements vont naturellement à mes deux directeurs de thèse. Vasile-Marian Scuturici et Jean-Marc Petit, merci de m'avoir confié cette thèse, merci d'avoir cru en moi, et merci pour votre accompagnement sur ces trois années qui ont compté parmi les plus belles de ma vie. Vous m'avez beaucoup appris et votre soutien a été essentiel pour mener cette aventure à son terme.

Je tiens aussi à remercier l'ensemble du jury. Merci tout d'abord aux deux rapporteurs, Sihem Amer-Yahia et Themis Palpanas, d'avoir d'accepté de relire mon travail. C'est un honneur d'avoir pu partager mon travail avec vous et vos retours ont été précieux. Merci aussi à Marius Bozga, Frédérique Laforest et Pierre Senellart d'avoir assisté à ma soutenance. Vos remarques et questions alimenteront sans aucun doute mes futures recherches.

Mes remerciements vont également à l'équipe de Datavalor. Benjamin, Vincent et Pierre-Nicolas, votre soutien moral et vos conseils techniques ont été très appréciés. Plus généralement, je remercie l'équipe d'Insavalor qui a toujours été rapide et précise dans ses réponses à mes nombreuses questions.

Merci à toute l'équipe du laboratoire LIRIS pour son aide et son support. Vous avez formé un environnement particulièrement riche et stimulant qui ont alimenté ma passion tout au long de cette thèse. Malgré le COVID qui nous a séparé quelques temps, merci à tous les autres doctorants pour votre présence rassurante au sein du grand bateau de la recherche.

Je remercie également mes amis et ma famille pour leur soutien indéfectible tout au long de cette thèse. Merci tout d'abord à Simon, nos innombrables séances de travail et de philosophie de comptoir dans les cafés vont me manquer. Je n'oublie pas non plus ton aide, tant sur le plan scientifique que de la communication (Beamer et IPE seront à tous jamais mes fidèles compagnons). Jennie, ton humour et ta bonne humeur ont aussi illuminé mes journées. L'un comme l'autre, vous êtes passés de collègues à ami.e.s. Nous avons

partagé repas, cafés, films et jeux au laboratoire comme ailleurs ; merci de tout cœur et à très vite. Merci également à mes ami.e.s lyonnais de longue date qui ont été présents quand j'en avais besoin : Arthur, Adrien, Camille, Charlotte, Mathilde, Rémi, William... Pour les non-Lyonnais et ceux que j'ai oublié, merci également. Merci également de tout cœur à ma famille pour leur soutien de chaque instant. Chers parents, nos nombreux appels ont égayé mes semaines. Merci finalement à Anne-Laure qui m'aura accompagné pendant un peu plus de mes deux premières années. Une pensée pour Plume.

Je ne peux clore ces remerciements sans parler de l'organisme qui finance cette thèse : la Compagnie Nationale du Rhône. Les rencontres effectuées en son sein ont été profondément enrichissantes. Merci à Pierre Roumieu qui m'a accompagné sur tant sur le plan scientifique qu'administratif tout en me faisant découvrir ses superbes modèles réduits de centrales. Merci à Christophe Turbidi pour sa présence et ses projets qui ont su alimenté ma plumme. Merci enfin à Eric Divet d'avoir cru en cette recherche et de l'avoir soutenue tout du long. Merci enfin de m'avoir permi de visiter vos impressionnantes centrales hydro-électrique le long du Rhône et d'avoir donné un aspect plus pratique à ma thèse quand j'en ai ressenti le besoin.

Résumé

Dans cette thèse, nous étudions le lien entre la connaissance métier sous forme d'une fonction et la science des données. Considérons le scénario suivant. Soit $D(y, z_1, \ldots, z_n)$ un ensemble de données, Alice une experte en science des données, Bob un expert du domaine et $y = f(z_1, \ldots, z_n)$ une fonction connue de Bob grâce à ses connaissances métier. Dans cette thèse, nous nous intéressons aux questions suivantes, simples mais cruciales pour Alice. *Comment définir la satisfaction de f dans D ? Comment mesurer efficacement cette satisfaction ? Comment cette satisfaction est-elle liée à la tâche d'apprentissage supervisé consistant à apprendre f à partir de D ?* Il s'avère que ces problèmes sont liés à l'étude des contre-exemples par l'utilisation des dépendances fonctionnelles (DF) et, en particulier, des mesures permettant de quantifier la satisfaction des DFs dans un ensemble de données telles que l'indicateur g_3. Plus précisément, nous considérons le cas où l'égalité est remplacée par des prédicats plus flexibles, une relaxation maintenant courante dans la littérature.

Premièrement, nous examinons la complexité du calcul du g_3. Il est connu que g_3 peut être calculé en temps polynomial lorsqu'on utilise l'égalité, alors qu'il devient NP-difficile lorsqu'on utilise des prédicats généraux. Nous proposons d'affiner cette dichotomie en étudiant l'impact des propriétés communes suivantes : réflexivité, transitivité, symétrie et antisymétrie. Nous montrons que la symétrie et la transitivité sont suffisantes pour garantir que l'erreur g_3 puisse être calculée en temps polynomial. Cependant, la suppression de l'une d'entre elles rend le problème difficile. *Deuxièmement*, nous étudions le calcul de g_3 dans les cas polynomial et NP-difficile identifiés dans la première partie. Nous proposons différentes solutions exactes et approximées pour le calcul de g_3 dans les deux cas. Nous comparons ces solutions dans une étude expérimentale détaillée des performances temporelles et d'approximation. Tous les algorithmes sont également disponibles via FASTG3, une librairie Python open-source implémentée en C++. *Troisièmement*, nous connectons l'étude des contre-exemples et l'indicateur g_3 à l'apprentissage supervisé à l'aide d'une application web appelée ADESIT. ADESIT est destinée à faire partie d'un processus itératif de raffinement des données juste après la sélection des données et juste avant le processus d'apprentissage lui-même. Elle permet d'évaluer la capacité d'un ensemble de données à donner de bons résultats pour un problème d'apprentissage supervisé par le biais de statistiques et d'une exploration visuelle. *Enfin*, nous validons notre approche par une application au problème industriel de la surveillance de l'entrefer dans les générateurs hydrauliques compacts et développons une solution pour le traitement automatique des données enregistrées.

Abstract

In this dissertation, we investigate the link between domain knowledge in the form of functions and data science. Consider the following scenario. Let $D(y, z_1, \ldots, z_n)$ be a dataset, Alice a data scientist, Bob a domain expert and $y = f(z_1, \ldots, z_n)$ a function known to Bob from his background knowledge. In this dissertation, we are interested in the following simple yet crucial questions for Alice. *How to define the satisfaction of f in D? How to measure that satisfaction efficiently? How does this satisfaction relate to the supervised learning task of learning f from D?* It turns out that these problems are related to the study of counterexamples through the use of functional dependencies (FDs) and, in particular, FD measures used to quantify their satisfaction in a dataset such as the g_3 indicator. More specifically, we consider the case where the equality is replaced by more flexible predicates, a relaxation that is now common in the literature.

First, we examine the complexity of computing g_3. It is known that g_3 can be computed in polynomial time when using equality, while it becomes NP-hard when using general predicates. Our goal is to refine this dichotomy by studying the impact of the following common properties: reflexivity, transitivity, symmetry, and antisymmetry. We show that symmetry and transitivity together are sufficient to guarantee that the g_3 can be computed in polynomial time. However, removing one of them makes the problem NP-hard. *Second*, we study the computation of g_3 in the polynomial and NP-hard cases identified previously. We propose different exact and approximate solutions for the computation of g_3 in both cases. We compare these solutions in a detailed experimental study of time performance and approximation accuracy. All the algorithms are also made available via FASTG3, a fast open-source Python library with an underlying C++ implementation. *Third*, we link counterexamples and g_3 to supervised learning with a web application called ADESIT. ADESIT is intended to be part of an iterative data refinement process right after data selection and just before the machine learning process itself. It provides a way to evaluate the ability of a dataset to perform well for a given supervised learning problem through statistical and visual exploration. *Finally*, we validate our approach by applying it to the industrial problem of air gap monitoring in compact hydro-generators, and develop a solution for automatically processing the recorded data.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1. Context

Today's society is a data society. Vast amounts of information are accessed and stored every second, ready to be dissected and analyzed [SS13]. It is no wonder that data science is a fast-growing field of research, even going so far as to call data scientist "*the sexiest job of the* 21st *century*" [DP12]. Nonetheless, it is widely acknowledged that data science techniques alone are not sufficient and that domain knowledge is an essential part of successful data projects [Kit14]. Thus, for the time being, the use of domain knowledge in addition to data science techniques is a critical area of research for extracting more advanced insights from data. However, as noted in [Via13], *data scientists are not domain experts*. Therefore, for data science projects to be successful, data scientists need to be part of cross-functional teams and leverage the knowledge of others. The importance of such collaboration has been stressed many times and several solutions have been proposed to integrate this knowledge [ASM22, YHPM99, DCH97, BSEK06].

This thesis deals with the problem of domain knowledge integration in the context of hydroelectricity production. In particular, it is the result of a partnership between the LIRIS (Laboratoire d'InfoRmatique en Image et Systèmes d'information) computer science research laboratory and the CNR (Compagnie Nationale du Rhône), an important French producer of renewable energy. This partnership takes place within the CNR chair at the "INSA Lyon Fondation". This type of collaboration is an interesting playground for confronting the current state of research with domain experts and industrial problems, in this case in the field of data science. In particular, our various interactions with the CNR experts led us to the following question: how can their theoretical models, in the form of functions, be better verified on the basis of data collected in the field?

In the following, we introduce the CNR and give some elements of hydroelectric production that will be used in this thesis. We then present the subject of this thesis in more detail and list our main contributions. Finally, we present the organization of the manuscript.

1.2. The “Compagnie Nationale du Rhône” (CNR)

Logo of the CNR.

The CNR is one of France’s leading renewable energy producers. It uses solar and wind power, but its main production is based on hydroelectricity. The CNR operates 20 hydroelectric power plants along the French part of the Rhône river, producing up to 3100 MWh [dR22]. For several decades, the CNR has used a wide range of sensors on its machines for direct control of the plants by humans or programmable controllers. For more than 10 years, much of this data has also been stored for later use, but remains largely under-exploited. For this reason, the CNR has invested in several data-centric projects in recent years, with the aim of extracting new knowledge and developing new services from its data. These projects are particularly important to better understand and optimize production, as well as to keep the machines running. The study presented in this thesis is part of this approach.

1.3. Run-of-the-river hydroelectricity, a quick overview

In this dissertation, we exploit elements of domain knowledge from hydroelectricity production. Hence, we now present an overview of the functioning of a run-of-the-river hydroelectric power plant to facilitate the understanding of the following chapters.

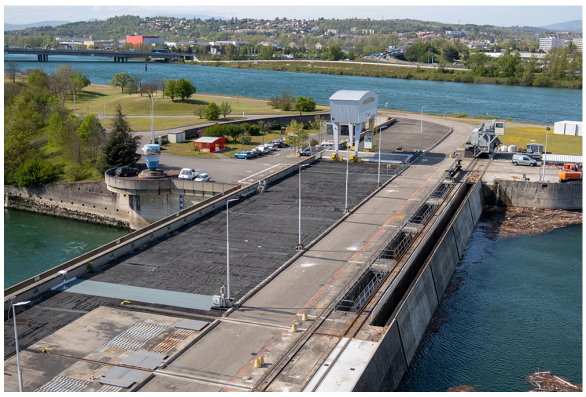


Figure 1.1 – *Example of a CNR run-of-the-river hydroelectric power plant (location: Pierre-Bénite, France).*

Run-of-the-river hydroelectric power plants are optimized to operate on rivers with little or no water storage capacity, such as the Rhône. A photograph of such plant is shown in Figure 1.1. Their design is radically different from classical hydroelectric projects and in particular from hydropower plants with pondage [WMCS84]. In particular, we take the example of a run-of-the-river hydroelectric power plant with bulb turbines which is the most common type used by the CNR. The scheme of such a plant is shown in Figure 1.2.

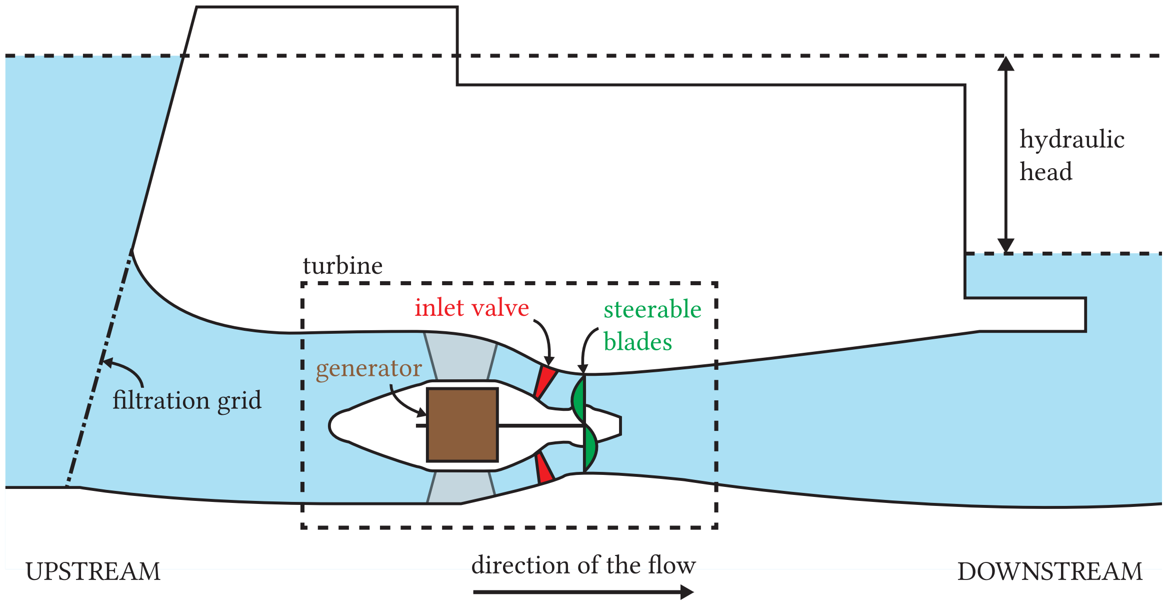


Figure 1.2 – *Schematic of a run-of-the-river hydroelectric power plant with a bulb turbine. This diagram shows one turbine but there are usually several installed in parallel. This is the most common installation at the CNR.*

We now describe the general functioning of such unit. Upstream of the turbine, a filtration grid is used to prevent potentially hazardous objects from entering the water inlet and damaging the turbine. The flow ($m^3 \cdot s^{-1}$) is controlled by a valve installed just before the blades. The hydraulic head (m) is the difference in water level between downstream and upstream of the unit. Together with the flow, the hydraulic head is the most important variable determining the potential power generated by a turbine. In particular, a classic function from hydraulics can be expressed as follows [CC13]:

$$\text{power} = f_{\eta,\rho}(\text{flow}, \text{head}) = \eta \cdot \rho \cdot \text{flow} \cdot \text{head} \tag{1.1}$$

where ρ ($kg \cdot m^{-3}$) is the density of the water and η (no unit) is the turbine's performance which depends on fixed parameters such as the structure of the water inlets or the design of the blades. From domain knowledge, we also know that the power produced depends strongly on the orientation of the blades and the head loss at the filtration grids. In this thesis, we will study the main following problem:

Given a function describing a physical system and data recorded from that system, to what extent is the function "verified" in the data?

Hydraulic energy is used to rotate the head of the turbine to produce electricity through the generator. The generator has two main components: the rotor and the stator. The rotor is driven by the rotating turbine head, creating a rotating electromagnetic field. The stator uses this electromagnetic field to produce an alternating current synchronized to the European grid frequency: 50 Hz. The air gap is the space between the rotor and the stator. In run-of-the-river installations, *compact* generators are generally required in order to disturb the flow of the river as little as possible and to preserve the ecosystem. This compactness is usually achieved at the cost of a small air gap, which sometimes leads to rotor-stator collisions due to progressive *stator deformation*, as happened at the CNR. In this case, the air gap distance is only a few millimeters for turbines of several meters in diameter. In addition, due to real constraints, the rotor is not perfectly aligned with its axis of rotation and its position varies over time. This phenomenon is called *dynamic rotor eccentricity*. In this thesis, the following problem will serve as an example of industrial application:

In the case of stator deformation with dynamic rotor eccentricity, how to process air gap monitoring data so to obtain reliable and operable insights?

1.4. Assessing the existence of a function in a dataset

The CACOH (Centre d'Analyse Comportementale des Ouvrages Hydrauliques) laboratory at the CNR is responsible for analyzing the behaviour of hydraulic structures. During one of our meetings with their engineers, the question arose of confronting the theoretical function presented in function 1.1 with some data recorded on site. This question made sense in the context of this thesis, as it is a ubiquitous means of confronting domain knowledge with data. It led our research to the problem we describe in the following.

Leveraging domain knowledge is the key to extracting insightful information from data about real-world problems. Therefore, when collaborating with domain experts to confront their knowledge with real-world data, data scientists need an effective way to express and measure their theoretical model against a dataset. We focus on one of the most ubiquitous forms of models: *functions*.

Figure 1.3 highlights the pipeline we intend to follow in this dissertation. Our goal is to enable the comparison of data extracted from a physical system with a function given by a domain expert. As described later in Chapter 6, this process also provides insight into the potential success of a machine learning project where the goal becomes *learning a function,* i.e. *a model*. To illustrate our motivation, we use the following running example drawn from the CNR.

Example 1. *We consider the toy dataset r_{toy} in Table 1.1 presenting three attributes of a water turbine: the incoming flow, the hydraulic head and the power produced by the turbine. These attributes have been presented in the previous section. Moreover, we also know thanks to function 1.1 that the power is dependent on the flow and the head. To assess the proper functioning of their turbines, and gain insights into potential technical optimizations, it is crucial to evaluate the veracity of function 1.1 against the data given in r_{toy}.*

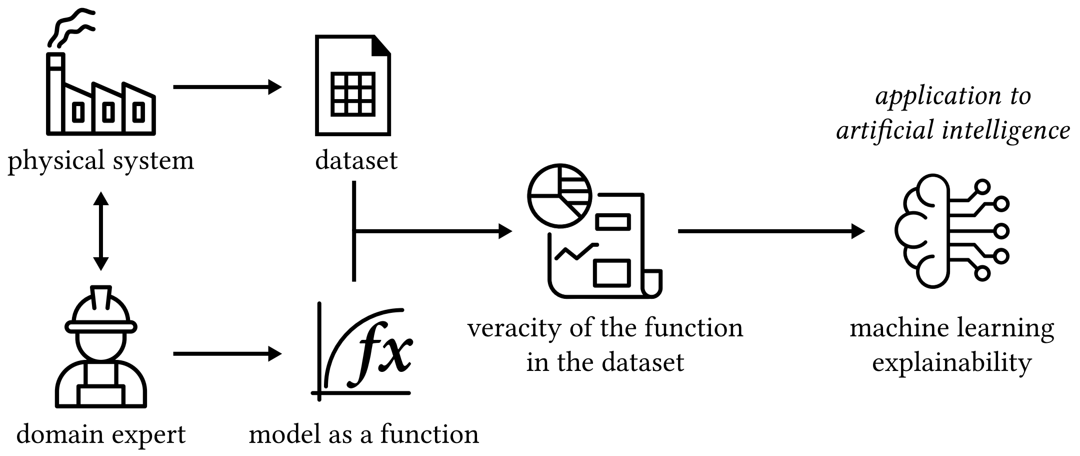

physical system
dataset
application to artificial intelligence
domain expert
model as a function
veracity of the function in the dataset
machine learning explainability

Figure 1.3 – *General process motivating the thesis. The data extracted from a physical system is compared to the function given by a domain expert. Those results can be used to support the machine learning process.*

Table 1.1 – RELATION r_{toy}.

r_{toy}	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

Functions are deterministic by nature: *an input has an unique output*. Functional dependencies (FDs) precisely capture this intuition and offer a comprehensive framework to express constraints between sets of attributes. Introduced in [Arm74], FDs have been extensively used to guarantee integrity in databases. However, their role has gradually extended to many tasks such as data cleaning [BFG+07], data mining [NC01], and query optimization [NK04], to mention but a few.

In its original definition, a FD φ is an expression of the form $X \rightarrow A$ where X is a set of attributes and A a single attribute. In φ, X is often called the *antecedent* (a.k.a. features) and A the *consequent* (a.k.a. target). Such a FD is generally called a *classical*, *exact* or *crisp* FD. We say that a relation r *satisfies* φ when, for each two tuples of r, their equality on the antecedent of φ entails their equality on the consequent. A pair of tuples which does not satisfy the FD is called a *counterexample* of φ in r. In other words, φ models the fact that, in r, the antecedent must completely determine the consequent, just as the inputs of a function must completely define the outputs.

Example 2 (Continued)**.** *Function 1.1 translates as the following functional dependency:*

$$\varphi_f : \mathit{flow}, \mathit{head} \rightarrow \mathit{power}$$

Unfortunately, crisp FDs are often too restrictive to portray accurately many real-life scenarios. As a consequence, extending the satisfaction of FDs has been at the core of many works during the last decades, as witnessed by recent surveys on the topic [CDP15, SGHW20]. Those approaches can be classified according to two principles:

(i) measuring how much a FD deviates from the (deterministic) definition of a function

(ii) redefining the comparabilities between tuples

The first principle (i) aims to quantify the partial validity of a FD in data, rather than assessing its perfect satisfaction. This allows to take into account data quality issues such as outliers, mismeasurements or mistakes, which should not impact the relevance of a FD in the data. To this end, a measure of satisfaction (a.k.a. coverage) needs to be chosen for φ in r. The most common measure is the g_3 indicator, introduced by Kivinen and Mannila [KM95]. More precisely, $g_3(\varphi, r)$ corresponds to the smallest proportion of tuples to remove from r in order to satisfy φ. In the literature, the g_3 indicator is often called the g_3-error [CDP15] or just the error [HKPT99] while its opposite $1 - g_3$ is often referred to as *confidence* [CGF+09, GKK+08, SGHW20]. Furthermore, g_3 is at the core of many FD mining algorithms [KM95, WSC+17, HKPT99, CDP16] and links to supervised learning have been established in literature [LPS20, FGPSLG21].

Example 3 (Continued)**.** *There may be various errors in sensor recordings, or temporary perturbations in the turbine such as branches often succeeding to passing upstream filters. Hence, it is very unlikely that a FD such as φ_f will be satisfied in r_{toy} despite being* almost *valid. In r_{toy}, the pair (t_3, t_6) is the only counterexample to φ_f. Thus, removing 1 tuple (t_3 or t_6) over 6 is sufficient to satisfy φ_f, which gives a g_3 value of $\frac{1}{6}$. It is now up to the data scientist to discuss with domain experts the significance of such g_3 value.*

The second principle (ii) introduces relaxations on the comparison of attribute values. Loosening the strict equality of crisp FDs allows to take into account imprecisions and uncertainties that are inherent to every observation. A classic approach is to replace the equality by predicates as proposed in [CDP15, CCPP17, SGHW20]. By doing so, we obtain a more general type of FD, generally called *relaxed* or *non-crisp* FD. Such a powerful relaxation captures a finer notion of proximity directly related to the problem at hand and the available domain knowledge. Many well-known examples of non-crisp FDs such as Similarity Dependencies [BKN13], Differential Dependencies [SC11] or Neighborhood Dependencies [BW01] can be found in the literature and generally use an association of metric predicates (metric associate with a threshold) and equality predicates, see also [CCPP17]. The substitution of equality by predicates considerably enlarges the spectrum of expression from string and numerical metrics to ordering constraints and probabilistic proximity. We illustrate this extension in the following example.

Example 4 (Continued)**.** *In r_{toy}, several FDs are trivially satisfied since all measures are continuous, and hence, are likely to be unique. In this case, strict equality should be relaxed by incorporating uncertainties to better capture the subtlety of the data. Thus, we can define a predicate which returns* TRUE *if two recorded values are considered similar under the sensor uncertainty and* FALSE *otherwise. In this is example, we keep it simple and apply an absolute uncertainty of* 0.1 *to each attribute such that:*

$$\phi_{power}(x,y) = \phi_{head}(x,y) = \phi_{flow}(x,y) = \begin{cases} \text{TRUE} & \text{if } |x-y| \leq 0.1 \\ \text{FALSE} & \text{otherwise.} \end{cases} \tag{1.2}$$

*First, we observe that the previous pair (t_3, t_6) is no longer a counterexample. Indeed, their values on the power attribute are not distant enough to be considered different when incorporating the uncertainties. However, several new counterexamples arise: (t_1, t_5), (t_1, t_4), (t_4, t_5), (t_4, t_3)... Thus, integrating these uncertainties showcases the possible difficulties of real-life data in matching the target function. To validate φ_f, it is sufficient to remove 3 tuples (*e.g. *t_2, t_4 and t_5) over 6 total which gives a g_3 of $\frac{3}{6} = 0.5$. Such a high error requires discussions with domain experts: what are the possible causes of such deviations from the model?*

Nonetheless, if predicates extend FDs in a meaningful way with respect to real world applications, they also make the computation harder. In fact, contrary to strict equality, computing the g_3-error with binary predicates becomes NP-complete [FGPS22, SCP13]. In particular, this is the case for metric [KSSV09], matching [Fan08], comparable and differential dependencies [SCP13]. Still, there is no detailed analysis of what makes the g_3-error hard to compute when dropping equality for more meaningful predicates. As a consequence, domain experts are left without any insights on which predicates they can use to be able to estimate the validity of their background knowledge in their data. Moreover, no detailed analysis for the computation of g_3 can be found in literature, especially for its scalability with large datasets. Finally, while [LPS20] sketches a link between supervised learning and the g_3-error, the use of predicates as well as the exploitation of FD counterexamples is not found in the literature. To mitigate these absences in the literature, we present the following contributions:

- **Complexity analysis.** First, we investigate the following question: *which properties of predicates make the g_3-error easy to compute?* Indeed, predicates offer a convenient framework to study the impact of common properties such as reflexivity, transitivity, symmetry (the properties of equality), and antisymmetry on the hardness of computing the g_3-error. In this setting, we make the following contributions. First, we show that dropping reflexivity and antisymmetry does not make the g_3-error hard to compute. When removing transitivity, the problem becomes **NP**-complete. This result is intuitive as transitivity plays a crucial role in the computation of the g_3-error for dependencies based on similarity/distance relations [CDP15, SGHW20]. Second, we focus on symmetry. Symmetry has attracted less attention, despite its importance in partial orders and order FDs [DH82, GH83, Ng01]. Even though symmetry seems to have less impact than transitivity in the computation of the g_3-error, we show that when it is removed the problem also becomes **NP**-complete. This result holds in particular for ordered dependencies.

– **Algorithms for computing** g_3. Second, we examine the computation of g_3 in the **NP**-hard and polynomial cases identified above. For both cases, we propose different exact and approximate solutions for the computation of g_3. First, for the **NP**-hard case, we present a detailed computation pipeline with various computation optimizations, including approximation algorithms and adaptations of recent developments in sublinear algorithms for **NP**-hard problems [ORRR12, YYI09]. Second, for the polynomial case where predicates are constrained to be at least transitive and symmetric, we propose exact algorithms and approximate ones based on uniform and stratified random sampling. We also propose an in-depth experimental study of the algorithms presented in terms of time performance and approximation accuracy. All the algorithms are also made available through FASTG3, an open-source Python library designed to be intuitive and efficient thanks to an underlying C++ implementation.

– **Interactive counterexample exploration.** Third, we present how our solution can be used to support the supervised learning process. Notably, we show how to use counterexamples and the g_3 indicator to assist the upstream process before training a learning algorithm. Our main contribution is the development of a web application to interact with a dataset under the scope of counterexamples: ADESIT. ADESIT is based on the FASTG3 library and proposes interactive visualizations and indicators to help understand why the learning might fail and in which situations.

We also dedicate a part of this thesis to validate our approach on a real-world problem from the CNR. More precisely, we show how counterexamples and the g_3 indicator can be used as a go/no-go step for a data science project. To do this, we take advantage of an auxiliary project carried out with the CNR's Industry 4.0 structure, which is working on the use of data for predictive maintenance. Following a meeting with engineers from this structure, we decided to study the problem of air gap monitoring in compact hydro-generators such as those made by CNR. The aim is to propose a faster and more reliable solution with complementary information on stator deformations and critical air gap distances to support maintenance. Before developing a data science solution for this problem, we explain how we used ADESIT to verify some primary hypotheses and assumptions. We then explain our solution in more detail and apply it to some data from the CNR.

1.5. Manuscript organization.

The organization of the manuscript reads as follows. In Chapter 2, we give some preliminaries in the fields of graph theory and database theory. Chapter 3 introduces the notions of counterexample and conflict graph as well as the problem of computing the g_3 indicator with predicates and its associated semantics. Chapter 4 is devoted to an in-depth analysis of the complexity of the problem of computing g_3 with respect to a set of properties. In the light of these results, Chapter 5 offers algorithmic solutions for its exact and approximate computation: first in in the general case but also when the predicates are restricted to be at least transitive and symmetric. We also perform experiments on several datasets to analyze the time performance and the accuracy of the algorithms. In Chapter 6, we present ADESIT, a web application for analyzing a dataset in view of a function and we illustrate its use with a running example from the CNR. In Chapter 7, we validate our approach on the problem of air gap monitoring. We use counterexample analysis as a preliminary go/no-go step for a CNR project on air gap monitoring in compact hydro-generators, and present a data science solution for automatically processing such data. Finally, we present some related work, conclude this thesis and suggest some avenues for future study.

Chapter 2

Preliminaries

In this chapter, we introduce the main notations used throughout this dissertation. In particular, we introduce notions of graph and database theory. All the objects we consider are finite.

Graph. We begin with some definitions on graphs [Ber73] and ordered sets [DP02]. A *graph* G is a pair (V,E) where V is a set of *vertices* and E is a collection of pairs of vertices called *edges*. An edge of the form (u,u) is called a *loop*. The graph G is *directed* if edges are ordered pairs of elements. Unless otherwise stated, we consider loopless undirected graphs. Let $G=(V,E)$ be a graph, and let $V' \subseteq V$. The graph $G[V'] = (V',E')$ with $E' = \{(u,v) \in E \mid \{u,v\} \subseteq V'\}$ is the graph *induced* by V' with respect to G. A *path* in G is a sequence $u_1,\dots,u_n$ of vertices such that $(u_i,u_{i+1}) \in E$ for each $1 \leq i < n$. The *length* of a path is its number of edges. An *independent set* of G is a subset I of V such that no two vertices in I are connected by an edge of G. Dually, a *clique* of G is a subset K of V such that every pair of distinct vertices in K are connected by an edge of G. An independent set is *maximal* if it is inclusion-wise maximal among all independent sets. It is *maximum* if it is an independent set of maximal cardinality. The size of a maximum independent set in G is the *independence number* of G, written $\alpha(G)$. The problem of computing $\alpha(G)$ is a well-known **NP**-complete problem defined as follows:

Maximum Independent Set (MIS)

Input:	A graph $G=(V,E)$, $k \in \mathbb{N}$.
Output:	`yes` if $\alpha(G) \geq k$, `no` otherwise.

The set complement of an independent set is a *vertex cover* of G. The *covering number* $\beta(G)$ of G is the minimum cardinality of a vertex cover of G. The problem of computing $\beta(G)$ is the dual of MIS:

Minimum Vertex Cover (MVC)

Input:	A graph $G = (V, E)$, $k \in \mathbb{N}$.
Output:	`yes` if $\beta(G) \leq k$, `no` otherwise.

A graph G is a *co-graph* if it has no induced subgraph equal to a path of length 3 (called P4). A 2-subdivision of a graph results from inserting 2 new vertices in every edge, that is from replacing each edge (u, v) with a P4 u, x, y, v. A graph is a 2-subdivision if it is the 2-subdivision of some graph. A *partially ordered set* or *poset* is a pair $P = (V, \leq)$ where V is a set and $\leq$ a reflexive, transitive, and antisymmetric binary relation. The relation $\leq$ is called a *partial order*. If for every $x, y \in V$, $x \leq y$ or $y \leq x$ holds, $\leq$ is a *total order*. A poset P is associated to a directed graph $G(P) = (V, E)$ where $(u_i, u_j) \in E$ exactly when $u_i \neq u_j$ and $u_i \leq u_j$. An undirected graph $G = (V, E)$ is a *comparability graph* if there exists a way to direct the edges of G so that the resulting directed graph corresponds to a poset.

Database. We move to terminology from database theory [LL12]. We use capital first letters of the alphabet (A, B, C, ...) to denote single attributes and capital last letters (..., X, Y, Z) for attribute sets. Let U be a universe of attributes with each attribute $A \in U$ taking value in a domain denoted by $\mathsf{dom}(A)$. Let $R \subseteq U$ be relation schema. The domain of R is $\mathsf{dom}(R) = \bigcup_{A \in R} \mathsf{dom}(A)$. The active domain of a relation r over R is the set of constant values that appear in the tuples of r. Sometimes, especially in examples, we write a set as a concatenation of its elements (*e.g.* AB corresponds to $\{A, B\}$).
A *tuple* t over R is a mapping $t \colon R \to \mathsf{dom}(R)$ such that $t(A) \in \mathsf{dom}(A)$ for every $A \in R$. The *projection* of a tuple t on a subset X of R is the restriction of t to X, written $t[X]$. A *relation* r over R is a finite set of tuples over R. We write $t[A]$ as a shortcut for $t[\{A\}]$. The size $|r|$ of r is its number of tuples, the cardinality of the set. For convenience, we will often put $|r| = n$.
A *functional dependency* over R is an expression $X \to A$ where $X \cup \{A\} \subseteq R$. Given a relation r over R, we say that r *satisfies* $X \to A$, denoted by $r \models X \to A$, if for every pair of tuples (t_1, t_2) of r, $t_1[X] = t_2[X]$ implies $t_1[A] = t_2[A]$. In case r does not satisfy $X \to A$, we write $r \not\models X \to A$.
Similar to [HKPT99], we denote the *equivalence class* of a tuple $t \in r$ with respect to a given set $X \subseteq R$ by $[t]_X$ such that $[t]_X = \{u \in r \mid t[A] = u[A] \text{ for all } A \in X\}$. The set $\Omega_X(r) = \{[t]_X \mid t \in r\}$ of equivalence classes is a *partition* of r under X.

Chapter 3

The g_3-error with predicates

This chapter introduces the g_3 indicator as well as the notions of counterexamples and conflict-graph. First, we describe its classical version as introduced in [KM95]. Second, we present our framework for extending the classical equality and show how it applies to g_3. We also present the current knowledge about the hardness of this problem.

3.1. The g_3-error

In this section, we introduce the classical g_3-error, along with its connection with vertex covers in graphs through counterexamples and conflict-graphs [Ber11].

Let r be a relation over R and $X \rightarrow A$ a FD. The g_3-error, or simply g_3, quantifies the degree to which $X \rightarrow A$ holds in r. We write it $g_3(X \rightarrow A, r)$. It has been introduced by Kivinen and Mannila [KM95], and it is frequently used to estimate the partial validity of a FD in a dataset [CDP15, CGF+09, FGPS22, HKPT99]. It is the minimum proportion of tuples to remove from r to satisfy $X \rightarrow A$. Formally:

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

In particular, if $r \models X \rightarrow A$, we have $g_3(X \rightarrow A, r) = 0$. For the most part, we will consider the optimization problem of computing the g_3-error where the goal is to compute its exact value. In some situations and especially hardness analysis, we will rather examine its decision version. We refer to the decision problem of computing $g_3(X \rightarrow A, r)$ as the *error validation problem* [CDP15, SCP13]. It reads as follows:

Error Validation Problem (EVP)

Input:	A FD $X \rightarrow A$ over R, a relation r over R, $k \in \mathbb{R}$.
Output:	`yes` if $g_3(X \rightarrow A, r) \leq k$, `no` otherwise.

It is known that there is a strong relationship between this problem and the task of computing the size of an MVC in a graph [CDP15, FGPS22]. To appreciate the relationship between the EVP and the MVC, we need to introduce the notions of *counterexample* and *conflict-graph* [Ber11, FGPS22]. A *counterexample* to $X \to A$ in r is a pair of tuples (t_1, t_2) such that for all $B \in X$, we have $t_1[B] = t_2[B]$ and $t_1[A] \neq t_2[A]$. Such pair of tuples is sometimes called a violating pair [KM95]. In other words, the pair (t_1, t_2) is a counterexample if $(t_1, t_2) \not\models X \to A$. The *conflict-graph* of $X \to A$ with respect to r is the graph $\mathsf{CG}(X \to A, r) = (r, E)$ where a pair of tuples (t_1, t_2) in r belongs to E when it is a counterexample to $X \to A$ in r. A vertex cover of $\mathsf{CG}(X \to A, r)$ is precisely the subrelation to remove from r to satisfy $X \to A$. Therefore, computing $g_3(X \to A, r)$ reduces to finding the size of a minimum vertex cover in $\mathsf{CG}(X \to A, r)$. More formally, we have the following proposition:

PROPOSITION 1. *Let R be a relation schema and φ a FD. We have:*

$$g_3(\varphi, r) = \frac{\beta(\mathsf{CG}(\varphi, r))}{|r|}$$

Proof. Let $\mathsf{CG}(\varphi, r) = (V, E)$. Let C be an MVC in (V, E). By definition, there exists no edge in E such that both endpoints are in $V \backslash C$ and therefore no counterexample in the corresponding set of tuples. Moreover, C is also the minimum set of vertices/tuples which can be removed to get rid of all counterexamples. Observing that $|V| = |r|$, the result follows:

$$g_3(\varphi, r) = \frac{|C|}{|V|} = \frac{\beta(\mathsf{CG}(\varphi, r))}{|r|}$$

■

However, MVC is an NP-complete problem [GJ79] but computing $g_3(X \to A, r)$ takes polynomial time in the size of r and $X \to A$ [HKPT99]. This is due to the properties of equality, namely reflexivity, transitivity and symmetry. $\mathsf{CG}(X \to A, r)$ is a disjoint union of complete k-partite graphs, and hence a co-graph [FGPS22]. In this class of graphs, solving MIS is polynomial [GRT97]. Such graph structures are illustrated in the following two examples.

Example 5. *In the example presented in Figure 3.1, we can observe on the right the conflict graph corresponding to the table on the left and the FD $A \to B$. Two distinct complete k-partite graphs are visible.*

Example 6 (Continued from Example 4)**.** *As in Example 3, we use the relation r_{toy} presented in Table 1.1 and the FD φ_f : flow, head $\to$ power. In Figure 3.2 is presented the corresponding conflict graph $\mathsf{CG}(\varphi_f, r_{toy})$. The only counterexample to φ_f in r_{toy} is (t_3, t_6) and corresponds to the only edge of the graph. One possible MVC of size* 1 *is encircled which corresponds to the only tuple to remove to get rid of all counterexamples.*

r	A	B	C
t_1	1	1	1
t_2	1	1	2
t_3	1	2	3
t_4	2	3	4
t_5	2	4	5
t_6	2	3	6
t_7	2	5	7
t_8	2	5	8

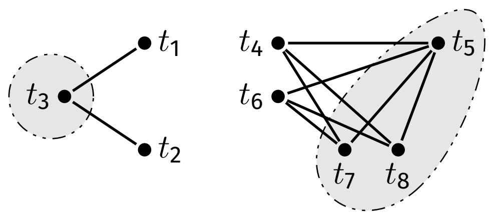


Figure 3.1 – *A relation r and the conflict graph* $\mathsf{CG}(A \to B, r)$. *An MVC is encircled.*

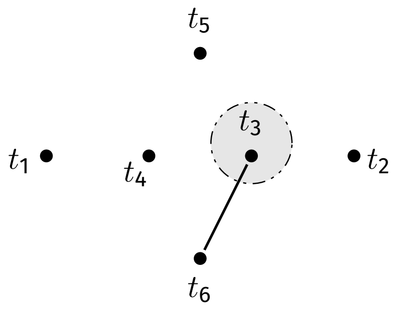


Figure 3.2 – *Conflict-graph* $\mathsf{CG}(\varphi_f, r_{toy})$. *An MVC is encircled.*

The observations made in this section led us to look more closely at the impact of common value comparison properties on the structure of conflict graphs and the resulting complexity of computing g_3. First, we need to introduce predicates to relax equality, and to define a more general version of the error validation problem accordingly.

3.2. Predicates to relax equality

In this section, we equip each attribute of a relation schema with a binary predicate. We define the new g_3-error and the corresponding error validation problem.

Let R be a relation schema. For each $A \in R$, let $\phi_A \colon \mathsf{dom}(A) \times \mathsf{dom}(A) \to \{\text{TRUE}, \text{FALSE}\}$ be a predicate. For instance, the predicate ϕ_A can be equality, a distance, or a similarity relation. We assume that predicates are black-box oracles that can be computed in polynomial time in the size of their input. Let Φ be a set of predicates, one for each attribute in R. The pair (R, Φ) is a *relation schema with predicates*. In a relation schema with predicates, relations and FDs are unchanged. However, the way a relation satisfies (or not) a FD can easily be adapted to Φ.

DEFINITION 1 (satisfaction with predicates). *Let* (R, Φ) *be a relation schema with predicates, r a relation over R and* $X \to A$ *a FD over R. The relation r* satisfies $X \to A$ *with respect to* Φ, *denoted by* $r \models_\Phi X \to A$, *if for every pair of tuples* (t_1, t_2) *of r, the following* *formula holds:*

$$\bigwedge_{B\in X} \phi_B(t_1[B], t_2[B]) \implies \phi_A(t_1[A], t_2[A])$$

The g_3-error also has to adapt to Φ. The *g_3-error with predicates* of $X \to A$ with respect to r, denoted by $g_3^\Phi(X \to A, r)$ is defined by:

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

From the definition of $g_3^\Phi(X \to A, r)$, we derive the extension of the error validation problem from equality to predicates:

Error Validation Problem with Predicates (EVPP)

Input:	A relation schema with predicates (R, Φ), a FD $X \to A$ over (R, Φ), a relation r over (R, Φ), $k \in \mathbb{R}$.
Output:	`yes` if $g_3^\Phi(X \to A, r) \leq k$, `no` otherwise.

Observe that according to the definition of satisfaction with predicates (Definition 1), counterexamples and conflict-graphs remain well-defined. However, for a given predicate ϕ_A, $\phi_A(x, y) = \phi_A(y, x)$ is not required to be true, meaning that we have to consider ordered pairs of tuples. That is, an ordered pair of tuples (t_1, t_2) in r is a counterexample to $X \to A$ if $\bigwedge_{B\in X} \phi_B(t_1[B], t_2[B]) = \text{TRUE}$ but $\phi_A(t_1[A], t_2[A]) \neq \text{TRUE}$, denoted $(t_1, t_2) \not\models_\Phi X \to A$.

We call $\mathsf{CG}_\Phi(X \to A, r)$ the conflict-graph of $X \to A$ in r. In general, $\mathsf{CG}_\Phi(X \to A, r)$ is directed. It is undirected if the predicates of Φ are symmetric. In particular, computing $g_3^\Phi(X \to A, r)$ still amounts to finding the size of a minimum vertex cover in $\mathsf{CG}_\Phi(X \to A, r)$. The next proposition formally states the correspondence between the g_3-error with predicates and the size of an MVC in the associated conflict-graph. The proof is similar to that of Proposition 1 and is omitted.

Proposition 2. *Let (R, Φ) be a relation schema with predicates and φ a FD. We have:*

$$g_3^\Phi(\varphi, r) = \frac{\beta(\mathsf{CG}_\Phi(\varphi, r))}{|r|}$$

Example 7 (Continued)**.** *We now add to the previous example the same predicates as in Example 4 (absolute uncertainties on each attribute). The new conflict graph $\mathsf{CG}_\Phi(\varphi_f, r_{toy})$ is presented in Figure 3.3. Remark that the graph is no longer a co-graph (*e.g. *t_1, t_4, t_3, t_2). This is due to the new predicates which are different from the equality. An MVC of size* 3 *is encircled for a g_3 of $\frac{1}{2}$ as in Example 4.*

Thus, there is also a strong relationship between the EVPP and the MVC, similar to the one between the EVP and the MVC. Nonetheless, unlike the EVP, the problem EVPP is NP-complete (corollary from [SCP13]).

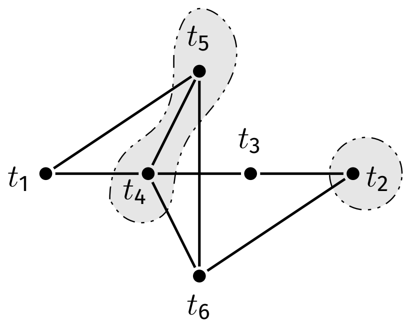


Figure 3.3 – *Conflict-graph* $\mathsf{CG}_\Phi(\varphi_f, r_{toy})$*. An MVC is encircled.*

For complexity and algorithmic purposes, we will sometimes restrict the expressiveness our model by imposing some properties to the predicates. We now describe the main properties of interest for this thesis. Let (R,Φ) be a relation schema with predicates. Let $A \in R$ and ϕ_A be the corresponding predicate. For all $x,y,z \in \mathsf{dom}(A)$, we consider the following properties:

(`ref`) $\phi_A(x,x) = \text{TRUE}$ (reflexivity)

(`tra`) $\phi_A(x,y) = \phi_A(y,z) = \text{TRUE}$ implies $\phi_A(x,z) = \text{TRUE}$ (transitivity)

(`sym`) $\phi_A(x,y) = \phi_A(y,x)$ (symmetry)

(`asym`) $\phi_A(x,y) = \phi_A(y,x) = \text{TRUE}$ implies $x = y$ (antisymmetry)

Note that symmetry and antisymmetry together imply transitivity, since in this case, $\phi_A(x,y) = \text{TRUE}$ entails $x = y$. In the next chapter, we study this gap of difficulty between the EVP and the EVPP in regard to those properties of predicates.

Chapter 4

Complexity of computing g_3

4.1. Introduction

As mentioned before, although predicates extend FDs in powerful and meaningful ways with respect to real-world applications, they also make computation more difficult. In fact, contrary to strict equality, computing the g_3-error with general binary predicates becomes NP-complete [FGPS22, SCP13]. In particular, it has been proven for metric [KSSV09], matching [Fan08], comparable and differential dependencies [SCP13]. Still, there is no detailed analysis of what makes the g_3-error hard to compute when dropping equality for more flexible predicates. As a result, domain experts are left without any insights on which predicates they can use to be able to estimate efficiently the validity of their background knowledge in their data.

This last problem constitutes the motivation for our contribution. In this chapter, we investigate the following question: *which properties of predicates make the g_3-error easy to compute?* Predicates provide a convenient framework for studying the impact of common properties such as reflexivity, transitivity, symmetry (the properties of equality), and antisymmetry on the hardness of computing the g_3-error. In this setting, we make the following contributions:

– First, we show that dropping reflexivity and antisymmetry does not make the g_3-error hard to compute. When removing transitivity, the problem becomes NP-complete. This is intuitive as transitivity plays a crucial role in the computation of the g_3-error for dependencies based on similarity/distance relations [CDP15, SGHW20], or equivalently on the underlying structure of the conflict graph.

– Second, we focus on symmetry. Symmetry has attracted less attention, despite its importance in partial orders and order FDs [DH82, GH83, Ng01]. Surprisingly, we show that when symmetry property is not satisfied the problem also becomes NP-complete, in particular for ordered dependencies.

The main results of this chapter have been published in [VFGPS23].

Chapter organization. First, we analyze the impact of dropping some properties of equality on the hardness of g_3. Second, we relate our results with existing extensions of FDs. Finally, we conclude with some remarks and open questions for further research.

4.2. Complexity analysis

In this section, we study properties of binary predicates that are commonly used to replace equality. Those properties –namely reflexivity (`ref`), transitivity (`tra`), symmetry (`sym`) and antisymmetry (`asym`)– have been described in Section 3. We show how each of them affects the error validation problem.

4.2.1. Dropping reflexivity and antisymmetry

We first show that symmetry and transitivity are sufficient to make the EVPP solvable in polynomial time. In fact, we prove that the resulting conflict-graph is a co-graph, as with equality.

THEOREM 1. *The problem* EVPP *can be solved in polynomial time if the predicates are transitive (* `tra`*) and symmetric (* `sym`*).*

Proof. Let (R,Φ) be a relation schema with predicates. Let r be relation over R and $X \rightarrow A$ a functional dependency. We assume that each predicate in Φ is transitive and symmetric. We show how to compute the size of a maximum independent set of $\mathsf{CG}_\Phi(r, X \rightarrow A)$ in polynomial time.

As ϕ_A is not necessarily reflexive, a tuple t in r can produce a counterexample (t,t) to $X \rightarrow A$. Indeed, it may happen that $\phi_B(t[B],t[B]) = \text{TRUE}$ for each $B \in X$, but $\phi_A(t[A],t[A]) = \text{FALSE}$. However, it follows that t never belongs to a subrelation s of r satisfying $s \models_\Phi X \rightarrow A$. Thus, let $r' = r \setminus \{t \in r \mid \{t\} \not\models_\Phi X \rightarrow A\}$. Then, a subrelation of r satisfies $X \rightarrow A$ if and only if it is an independent set of $\mathsf{CG}_\Phi(X \rightarrow A, r)$ if and only if it is an independent set of $\mathsf{CG}_\Phi(X \rightarrow A, r')$. Consequently, computing $g_3^\Phi(r, X \rightarrow A)$ reduces to solve MIS in $\mathsf{CG}_\Phi(X \rightarrow A, r')$.

We prove now that $\mathsf{CG}_\Phi(X \rightarrow A, r')$ is a co-graph. Assume for contradiction that $\mathsf{CG}_\Phi(X \rightarrow A, r')$ has an induced path P with 4 elements, say t_1,t_2,t_3,t_4 with edges (t_1,t_2), (t_2,t_3) and (t_3,t_4). Remind that edges of $\mathsf{CG}_\Phi(X \rightarrow A, r')$ are counterexamples to $X \rightarrow A$ in r'. Hence, by symmetry and transitivity of the predicates of Φ, we deduce that for each pair (i,j) in $\{1,2,3,4\}$, $\bigwedge_{B\in X} \phi_B(t_i[B],t_j[B]) = \text{TRUE}$. Let us focus on $\{t_1,t_3,t_4\}$. Thus, we have $\bigwedge_{B\in X} \phi_B(t_3[B],t_1[B]) = \bigwedge_{B\in X} \phi_B(t_1[B],t_4[B]) = \text{TRUE}$. However, neither (t_1,t_3)

nor (t_1, t_4) belong to $\mathsf{CG}_\Phi(r', X \to A)$ since P is an induced path by assumption. We deduce that $\phi_A(t_3[A], t_1[A]) = \phi_A(t_1[A], t_4[A]) = \text{TRUE}$ must hold. However, the transitivity of ϕ_A implies $\phi_A(t_3[A], t_4[A]) = \text{TRUE}$, a contradiction with (t_3, t_4) being an edge of $\mathsf{CG}_\Phi(X \to A, r')$. We deduce that $\mathsf{CG}_\Phi(X \to A, r')$ cannot contain an induced P_4, and that it is indeed a co-graph. As MIS can be solved in polynomial time for co-graphs [GRT97], the theorem follows. ■

It is possible to encounter non-reflexive predicates when dealing with strict orders or with binary predicates derived from `SQL` equality. In the 3-valued logic of `SQL`, comparing the `null` value with itself evaluates to FALSE rather than TRUE. With this regard, it could be natural for domain experts to use a predicate which is transitive, symmetric and non-reflexive, or more exactly reflexive almost everywhere but on the `null` value. This would allow to take into account missing information without altering the data.

4.2.2. Dropping transitivity

The previous proof heavily makes use of transitivity, which has a strong impact on the edges belonging to the conflict-graph. Intuitively, conflict-graphs can become much more complex when transitivity is dropped. Indeed, we prove an intuitive case: when predicates are not required to be transitive, the EVPP becomes intractable.

THEOREM 2. *The problem* EVPP *is* **NP***-complete even when the predicates are symmetric (*`sym`*) and reflexive (*`ref`*).*

The proof is a reduction from the problem of finding the size of a maximum clique in general graphs (dual to MIS). It uses arguments similar to the proof of Song et al. [SCP13] showing the **NP**-completeness of the EVPP for comparable dependencies. For completeness, it is given in Appendix A, page xxi.

4.2.3. Dropping symmetry

We turn our attention to the case where symmetry is dropped from the predicates. In this context, conflict-graphs are directed. Indeed, an ordered pair of tuples (t_1, t_2) may be a counterexample to a FD, but not (t_2, t_1). Yet, transitivity still contributes to constraining the structure of conflict-graphs, as suggested by the following example.

Example 8. *We consider the relation on the left of Figure 4.1. We equip A, B, C, D with the following predicates:*

- $\phi_C(x, y) = \text{TRUE}$ *if and only if* $x \leq y$.

– *$\phi_A(x,y)$ is defined by*

$$\phi_A(x,y) = \begin{cases} \text{TRUE} & \textit{if } x = y \\ \text{TRUE} & \textit{if } x = 1 \textit{ and } y \in \{2,4\} \\ \text{TRUE} & \textit{if } x = 3 \textit{ and } y = 4 \\ \text{FALSE} & \textit{otherwise.} \end{cases}$$

– *ϕ_B and ϕ_D are the equality.*

Let $\Phi = \{\phi_A, \phi_B, \phi_C, \phi_D\}$. The conflict-graph $\mathsf{CG}_\Phi(C \to A)$ is represented on the right of Figure 4.1. Since ϕ_C is transitive, we have $\phi_C(t_3[C], t_j[C]) = \text{TRUE}$ for each tuple t_j of r. Moreover, $\phi_A(t_3[A], t_6[A]) = \text{FALSE}$ since (t_3, t_6) is a counterexample to $C \to A$. Therefore, the transitivity of ϕ_A implies either $\phi_A(t_3[A], t_4[A]) = \text{FALSE}$ or $\phi_A(t_4[A], t_6[A]) = \text{FALSE}$. Hence, at least one of (t_3, t_4) and (t_4, t_6) must be a counterexample to $C \to A$ too. In the example, this is (t_3, t_4).

r	A	B	C	D
t_1	1	2	1	5
t_2	1	1	2	5
t_3	2	1	1	5
t_4	3	2	3	5
t_5	2	3	4	5
t_6	4	4	5	6

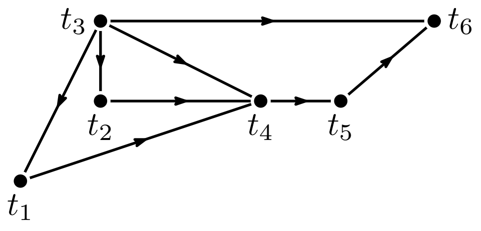


Figure 4.1 – *A relation r and the conflict graph $\mathsf{CG}_\Phi(AB \to C, r)$.*

Nevertheless, if transitivity constrains the complexity of the graph, dropping symmetry still allows new kinds of graph structures. Indeed, in the presence of symmetry, a conflict-graph cannot contain induced paths with more than 3 elements because of transitivity. However, such paths may exist when symmetry is removed.

Example 9. *In the previous example, the tuples t_2, t_4, t_5, t_6 form an induced P_4 of the underlying undirected graph of $\mathsf{CG}_\Phi(r, C \to A)$, even though ϕ_A and ϕ_C enjoy transitivity.*

Therefore, we are left with the following intriguing question: can the loss of symmetry be used to break transitivity, and offer conflict-graphs a structure sufficiently complex to make the EVPP intractable? The following theorem answers this question affirmatively. The proof is provided in Appendix B, page xxiv.

THEOREM 3. *The problem* EVPP *is* ***NP***-*complete even when the predicates are transitive (`tra`), reflexive (`ref`), and antisymmetric (`asym`).*

Theorem 1, Theorem 2 and Theorem 3 characterize the complexity of the EVPP for each combination of predicates properties. In the next section, we discuss the granularity of these, and we use them as a framework to compare the complexity of the EVPP for some known extensions of FDs.

4.3. Discussions

Replacing equality with various predicates to extend the semantic of classical FDs is frequent [CDP15, SGHW20]. Our approach offers to compare these extensions on the EVPP within a unifying framework based on the properties of the predicates they use. We can summarize our results with the hierarchy of classes of predicates given in Figure 4.2.

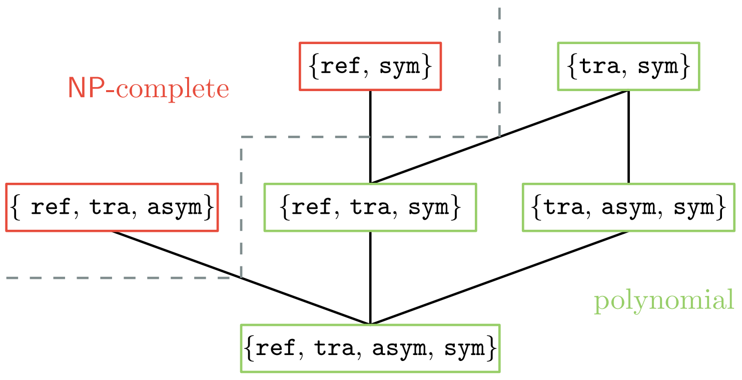


Figure 4.2 – *Complexity of the* EVPP *with respect to the properties of predicates.*

Regarding the computation of the g_3-error, most existing work has focused on similarity/distance predicates, but some extensions of FDs use partial orders as predicates. This is the case for ordered dependencies [DH82, GH83], ordered FDs [Ng01], and also of some sequential dependencies [GKK+09] and denial constraints [BBFL05] for example. To our knowledge, the role of symmetry in the EVPP has received little attention as for sequential dependencies [GKK+09] where a measure other than the g_3-error has been used.

In this context, our work can be used to obtain almost direct results on some existing FDs: we now study ordered dependencies as an example. The predicates of Theorem 3 are reflexive, transitive and antisymmetric. They are therefore partial orders. Consequently, the FD $X \to A$ is an *ordered FD* as defined by Ng in [Ng01]. We get the following corollary:

THEOREM 4. *The problem* EVPP *is* **NP***-complete for ordered FDs.*

Ordered functional dependencies are a restricted case of ordered dependencies [GH83], sequential dependencies [GKK+09], and denial constraints [BBFL05] (see [SGHW20]). The hardness of computing the g_3-error for these dependencies follows from Corollary 4.

The hierarchy is a fairly accurate representation of current knowledge about the EVPP and the distinction between tractable and intractable cases. However, this analysis may need further refinement. Indeed, there may be certain types of FDs with predicates for which the EVPP is tractable in polynomial time, even though their predicates belong to a class for which the problem is **NP**-complete. For examples, suppose that each attribute A in R is equipped with a *total* order ϕ_A. We show in Proposition 3 and Corollary 1 that in this case, the EVPP can be solved in polynomial time, even though the predicates are reflexive, transitive and antisymmetric.

PROPOSITION 3. *Let* (R,Φ) *be a relation schema with predicates. Then, the* EVPP *can be solved in polynomial time for a given FD* $X \to A$ *if* ϕ_B *is transitive for each* $B \in X$ *and* ϕ_A *is a total order.*

Proof. Let (R,Φ) be a relation scheme with predicates and $X \to A$ a functional dependency. Assume that ϕ_B is transitive for each $B \in X$ and that ϕ_A is a total order. Let r be a relation over R. Let $G = (r,E)$ be the undirected graph underlying $\mathsf{CG}_\Phi(X \to A, r)$, that is, $(t_i,t_j) \in E$ if and only if (t_i,t_j) or (t_j,t_i) is an edge of $\mathsf{CG}_\Phi(X \to A, r)$.

We show that G is a comparability graph. To do so, we associate the following predicate $\leq$ to $\mathsf{CG}_\Phi(X \to A, r)$: for each pair t_i,t_j of tuples of r, $t_i \leq t_i$ and $t_i \leq t_j$ if (t_i,t_j) is a counterexample to $X \to A$. We show that $\leq$ is a partial order:

- *reflexivity*. It follows by definition.
- *antisymmetry*. We use contrapositive. Let t_i,t_j be two distinct tuples of r and assume that (t_i,t_j) belongs to $\mathsf{CG}_\Phi(X \to A, r)$. We need to prove that (t_j,t_i) does not belong to $\mathsf{CG}_\Phi(X \to A, r)$, *i.e.* it is not a counterexample to $X \to A$. First, $(t_i,t_j) \in \mathsf{CG}_\Phi(X \to A, r)$ implies that $\phi_A(t_i[A],t_j[A]) = \text{FALSE}$. Then, since ϕ_A is a total ordering, $\phi_A(t_j[A],t_i[A]) = \text{TRUE}$. Consequently, (t_j,t_i) cannot belong to $\mathsf{CG}_\Phi(X \to A, r)$ and $\leq$ is antisymmetric.
- *transitivity*. Let t_i,t_j,t_k be tuples of r such that (t_i,t_j) and (t_j,t_k) are in $\mathsf{CG}_\Phi(X \to A, r)$. The predicates of X being transitive, we have that $\bigwedge_{B\in X} \phi_B(t_i[B],t_k[B]) = \text{TRUE}$. We show that $\phi_A(t_i[A],t_k[A]) = \text{FALSE}$. Since (t_i,t_j) is a counterexample to $X \to A$, we have $\phi_A(t_i[A],t_j[A]) = \text{FALSE}$. As ϕ_A is a total order, we deduce that $\phi_A(t_j[A],t_i[A]) = \text{TRUE}$. Similarly, we obtain $\phi_A(t_k[A],t_j[A]) = \text{TRUE}$. As ϕ_A is transitive, we derive $\phi_A(t_k[A],t_i[A]) = \text{TRUE}$. Now assume for contradiction that $\phi_A(t_i[A],t_k[A]) = \text{TRUE}$. Since, $\phi_A(t_k[A],t_j[A]) = \text{TRUE}$, we derive $\phi_A(t_i[A],t_j[A]) = \text{TRUE}$ by transitivity of ϕ_A, a contradiction. Therefore, $\phi_A(t_i[A],t_k[A]) = \text{FALSE}$. Using the fact that $\bigwedge_{B\in X} \phi_B(t_i[B],t_k[B]) = \text{TRUE}$, we conclude that (t_i,t_k) is also a counterexample to $X \to A$. The transitivity of $\leq$ follows.

Consequently, $\leq$ is a partial order and G is indeed a comparability graph. Since MIS can be solved in polynomial time for comparability graphs [Gol04], the result follows. ■

We can derive the following corollary for total orders, which can be used for ordered dependencies.

COROLLARY 1. *Let (R,Φ) be a relation schema with predicates. The problem* EVPP *can be solved in polynomial time if each predicate in Φ is a total order.*

In particular, this result applies to Golab et al. [GKK+09] where a polynomial-time algorithm is proposed for a variant of g_3 applied to a restricted type of sequential dependencies using total orders on each attribute.

4.4. Extension to the optimization problem

So far, we have focused solely on the EVPP, the decision version of computing g_3. In the following chapter, we study algorithms for computing the exact value of g_3 which corresponds to the optimization problem. For the hardness results on the EVPP to hold for its optimization counterpart, we need to show a polynomial-time self-reduction [Gol10].

Let (R,Φ) be a relation schema with predicates. Consider a relation r over (R,Φ), a FD φ over (R,Φ) and a variable k. The goal is to compute $g_3^{\Phi}(\varphi,r)$ by calling EVPP$((R,\Phi),\varphi,r,k)$ a polynomial number of times in the size of the input. The number of tuples we can remove to satisfy the FD is finite. Thus, the number of values of k is also finite and corresponds to $\frac{m}{|r|}$ with $0 \leq m \leq |r|$. Thus, we need to find the first value of k such that the EVPP becomes FALSE. This can be done in $\log(|r|)$ time by using dichotomy.

Therefore, the hardness results summarized in Figure 4.2 are directly transferable to the optimization problem of computing g_3. The only difference is that the problem is said to be NP-hard instead of NP-complete.

4.5. Conclusion

In this chapter, we have studied the complexity of computing the g_3-error when equality is replaced by more general predicates. We studied four common properties of binary predicates: reflexivity, symmetry, transitivity, and antisymmetry. We have shown that when symmetry and transitivity are taken together, the g_3-error can be computed in polynomial time. Transitivity strongly affects the structure of the conflict-graph and it is not surprising that removing it makes the g_3-error hard to compute. More surprising is that removing symmetry instead of transitivity leads to the same conclusion. This is because removing symmetry makes the conflict-graph oriented. In this case, the orientation of the edges weakens the effect of transitivity, allowing the conflict-graph to be complex enough to make the g_3-error computation problem intractable.

We believe that our approach sheds new light on the g_3-error computation problem, and that it is suitable for estimating the complexity of this problem when defining new types of FDs, by looking at the properties of the predicates used to compare values.

Chapter 5

Computing g_3 with predicates: *exact and approximate algorithms*

5.1. Introduction

In this chapter, we extend the decision problem EVPP to the optimization problem of computing the g_3-error. In this case, the objective becomes to *find* or *estimate* the value of g_3. However, as shown in Section 4.4: if the predicates are not at least transitive and symmetric, computing g_3 is NP-hard. Thus, our goal is to provide efficient algorithms for computing g_3 exactly and approximately in both the NP-hard and polynomial cases. In summary, we present the following contributions:

- **Computing g_3 with predicates: the general case.** To the best of our knowledge, we present the first complete pipeline for computing the g_3 indicator with predicates, a case where the problem is NP-hard. We examine its computational scalability by presenting connections to similar problems in the literature, especially for the conflict-graph construction process which is the bottleneck of the operation in practice. We also propose solutions for its exact computation and, for very large datasets, we adapt approximation algorithms and recent developments in sublinear algorithms for NP-hard problems [YYI09, ORRR12].
- **Computing g_3 in polynomial-time with transitive and symmetric predicates.** Computing g_3 with classic FDs (equality predicate) is a well-known polynomial problem [KM95, HKPT99]. We introduce a pre-processing algorithm that allows the use of known algorithms from the literature and propose practical solutions for its computational scalability, a topic that has received only little attention in previous research. We compare the scalability of two exact algorithms that optimize memory and time complexity respectively. For very large datasets where exact algorithms reach their limit, we analyze the theoretical guarantees of uniform random sampling and study advanced sampling schemes to enable computational scalability. In particular, we

improve the stratified sampling algorithm of Cormode et al. [CGF+09]. This new algorithm shows excellent results in practice.

– **Experiments and open-source library.** We conduct extensive experiments through a study of time performance and approximation accuracy. In particular, we show that our algorithms and optimizations can be applied to very large datasets with reasonable computation time, while maintaining a good level of accuracy for the expected results. We also detail which dataset specificities and problem parameters affect the computation time. All the algorithms are available through FASTG3, an open-source Python library for computing g_3 that provides efficient and scalable C++ implementations [BBC+11] with an intuitive API.

The main results of this chapter have been published in [FGPS22].

Chapter organization. First, we present solutions for computing g_3 in the general NP-hard case, both exactly and approximately for large datasets. Second, we consider the case where the predicates are at least transitive and symmetric, a case where efficient polynomial algorithms can be developed. Finally, we experiment with several datasets to analyze the time performances and the accuracy of the approximation algorithms. Note that the EVPP will also be discussed as some algorithmic optimizations can be applied in some cases. Underlined labels spread throughout the document (*e.g.* `CEE_BruteForce`) are be used in the experiments section (Section 5.4) to refer to specific algorithms.

5.2. The general case

In this section, we propose algorithms for the computation of g_3 in the most general case which is NP-hard. We give exact algorithms and approximate solutions for large datasets. As far as exact computation is concerned, the g_3 and the confidence are strictly equivalent. Henceforth, we will only discuss the former. However, the two problems diverge in terms of approximation and will be treated separately.

5.2.1. Computation overview

Consider a relation schema with predicates (R, Φ) and an FD φ. As shown in Proposition 2, we have:

$$g_3^{\Phi}(r, \varphi) = \frac{\beta(\mathsf{CG}_{\Phi}(r, \varphi))}{|r|}$$

Hence, the g_3-error can be obtained from the composition of two successive operations:

(i) **Counterexample enumeration (CEE).** First, we need to construct the conflict graph $\mathsf{CG}_{\Phi}(r, \varphi)$. If the vertices are simply the tuples, we need to enumerate all counterexamples in r to create the edges. This latter operation is costly as it is quadratic in the number of tuples and optimizations will be examined. In the following, we present algorithms for the CEE with various levels of optimizations and constraints. They are summarized in Table 5.1 and will be detailed in Section 5.2.2.

(ii) **Computing the covering number.** Second, we have to compute/estimate the covering number $\beta(\mathsf{CG}_{\Phi}(r,\varphi))$. The covering number (size of an MVC) can be computed exactly in a reasonable amount of time for a *small* number of edges (*i.e.* a *small* number of counterexamples). However, its exponential complexity quickly becomes overwhelming and approximations must be examined. Two major types of approximations will be considered: *approximation algorithms* and *sublinear algorithms*. The proposed algorithms are summarized in Table 5.2 and will be detailed in Section 5.2.3.

Table 5.1 – SUMMARY OF THE CEE ALGORITHMS.

Name	Attribute space	Predicate type
`CEE_BruteForce`	Any	Any
`CEE_BlockOpt`	Any	Equality
`CEE_CompOpt`	Any	Any
`CEE_OrderOpt`	Totally ordered	Monotonic

Table 5.2 – SUMMARY OF g_3 ALGORITHMS IN THE NP-HARD CASE.

Name[b]	Approx?	Complexity[abcd]
`CEE+EXA_WGYC` [HLSS20]	·	exponential
`CEE+HEUR_NuMVC(t)` [CSLS13]	Yes	$\mathcal{O}(n^2)$+timeout t
`CEE+APPROX_GIC` [HR97]	Yes	$\mathcal{O}(n^2)+\mathcal{O}(n \cdot \log(n)+\lvert E\rvert)$
`CEE+APPROX_2Approx` [PS98]	Yes	$\mathcal{O}(n^2)+\mathcal{O}(\lvert E\rvert \cdot \log(n))$
`APPROX_Sub09`[e] [YYI09]	Yes	$\mathcal{O}(d^4/\epsilon^2)\cdot\mathcal{O}(n)$
`APPROX_Sub11`[e] [ORRR12]	Yes	$\mathcal{O}(\bar{d}^2 \cdot \text{poly}(1/\epsilon))\cdot\mathcal{O}(n)$

[a] n corresponds to the size of the relation $|r|$.
[b] CEE+ prefixes algorithm which require to construct the conflict graph $\mathsf{CG} = (r,E)$ in $\mathcal{O}(n^2)$.
[c] d an $\bar{d}$ denote for the maximum and average degree of the conflict graph.
[d] ϵ denotes for the statistical error parameter of the algorithms.
[e] `APPROX_Sub09` and `APPROX_Sub11` use the on-the-fly CEE which has a complexity of $\mathcal{O}(n)$ for each query.

5.2.2. Counterexample enumeration (CEE)

To construct the conflict graph, we need to enumerate all the counterexamples in the relation, *i.e.* all pairs of tuples with similar antecedents and dissimilar consequents. We will use the fact that for φ in the form $X \longrightarrow A$, $(t_1,t_2) \not\models_\Phi \varphi$ corresponds to the following:

$$\bigwedge_{B\in X} \phi_B(t_1[B],t_2[B]) \wedge \neg\phi_A(t_1[A],t_2[A])$$

Brute force (`CEE_BruteForce`). In the most general scenario, each tuple in r must be compared to all other tuples in a nested loop. This scheme requires n^2 *comparisons*. A *comparison* is the successive pairwise comparison of each attribute of two tuples. This operation requires computing at most $|X|+1$ predicates for each pair of tuples, *i.e.* one predicate for all the antecedents and one for the consequent.

Fortunately, the problem of comparing pairs of records in a dataset has been studied extensively in several areas, ranging from record linkage to similarity joins [HS03,Chr11]. In particular, some propositions found in the literature that allow for significant time gains can be applied and are described below. These potential optimizations depend on the definition of the predicate and attribute spaces. However, even if all these methods can significantly speed up the enumeration process, they do not change the theoretical complexity, which is still bounded by n^2: indeed, it still takes n^2 comparisons if the dataset contains n^2 counterexamples.

Remark 1. Note that storing all resulting counterexamples can also be memory-intensive. For a dataset of 200,000 tuples, if all of them are in violation with all others and stored as *unsigned ints* pairs, approximately 150 gigabytes are required to store them all.

Attributes' comparison order (`CEE_CompOpt`). This first optimization requires no assumption about the predicates or the domain structure of the attributes. In this case, our goal is optimize the *order* of the attributes *inside tuple-to-tuple comparisons*. Indeed, the order in which the attributes are successively checked plays an important role in the complexity of the CEE: *we want the attributes to be sorted according to their "selectivity",* i.e. *in an order that produces the fewest false positives*. Indeed, if the comparison on the first attribute is positive, then the second one is checked and so on until it is either identified as a counterexample or rejected. However, if a pair of tuples is rejected by an attribute, all the predicates previously computed will have been a waste of resources by generating a false positive temporary counterexample. This is particulary important for expensive predicates such as the Levenshtein distance. We should therefore sort the attributes for comparison from those that generate the fewest counterexamples to those that generate the most. To efficiently evaluate the number of potential counterexamples of an attribute in X, we propose to perform the CEE for each attribute on a sample of the datasets of predefined size. More precisely, we isolate each antecedent (*e.g.* $B_0, B_1 \longrightarrow A$ becomes $B_0 \longrightarrow A$ to evaluate B_0) and then perform the CEE on this modified FD. Finally, we sort the attributes in ascending order regarding their computed number of counterexamples and perform the CEE on the full dataset.

Blocking (`CEE_BlockOpt`). The *blocking* indexing method is widely used in the field of record linkage as it allows for massive gains in time complexity (see [SVSF14, PSTP20] for surveys). It consists in grouping together similar values in attributes and performing a nested loop in each resulting block separately. Processing smaller blocks reduces the overall quadratic complexity and can easily be combined with parallelization. In our case, it requires that at least one attribute of X has a predicate corresponding to an *equivalence relation* ({`ref`,`tra`,`sym`}). In fact, an equivalence relation is necessary to group values into blocks (corresponding to equivalence classes). In general, most categorical attributes such as *zip codes* or *phone numbers* are usually compared using equality, which is the most common equivalence relation. Note that the gains of this method depend solely on the content of the attributes used for grouping as more distinct values lead to smaller blocks and therefore less processing time. Let $X_{eq} \subseteq X$ be the subset of *antecedents* with predicates corresponding to *equivalence relations*. After blocking into $B = |r[X_{eq}]|$ blocks, the number of comparisons is bounded by $\mathcal{O}(B \cdot n_{max}^2)$, where n_{max} is the size of the largest block.

Candidate pairs in totally ordered space with monotonic predicate (`CEE_OrderOpt`). To further optimize comparisons within a block, we can restrict the expressiveness of a predicate in X. The aim is to output a set of candidate pairs by first joining on an attribute for which the complexity can be optimized. The full pairwise attribute comparison can then be performed on this restricted set of candidates. In particular, we consider the case where the domain of an antecedent $B_i^o \in X$ is equipped of a *total order* $(\mathsf{dom}(B_i^o), \leq)$. In addition, we consider monotonic symmetric predicates w.r.t $(\mathsf{dom}(B_i^o), \leq)$ such that for elements $x, y, z \in$ $dom(B_i^o)$, we have:

$$x \leq y \leq z \text{ and } \phi_i(x,z) \implies \phi_i(x,y)$$

This property is well-suited for predicates based on a monotone metric and a threshold, which is common in the literature [SC11, KSSV09]. However, it also applies to our hydropower running example with the predicate presented in Formula 5.2, page 42. Moreover, many attributes such as *ages*, *incomes* and *sizes* are, most of the time, ordered numerical attributes.

In this case, it is possible to exploit this property to quickly generate a set of candidate pairs. To do so, we propose an algorithm inspired by [DGZ03], which runs in log-linear time in the size of the relation. This algorithm first sorts the data and, for each tuple t, successively keeps in memory the most distant tuple similar to t on the attribute B_i^o under consideration. Such process follows a *sliding window* strategy where all combinations of tuples within the window are potential candidate counterexamples. Its pseudo-code is proposed in Algorithm 1.

Remark 2. Note that there may be other optimizations specific to some attributes and predicates for a faster CEE. The optimizations proposed in this section are meant to be generic and to provide leads for adapting the process to the specific cases one might encounter.

Algorithm 1 Window sliding algorithm for generating a set of candidate pairs for monotonic predicates in totally ordered space (`CEE_CompOpt`)

```
Require:
    a relation $r$
    $B_i^o$ an attribute defined on a total order $(\mathrm{dom}(A_i), \leq)$
    $\phi_i$ a monotonic predicate associated to $A_i$

1: Let $x_1, x_2, ..., x_n$ be the elements of $r^{B_i^o}$ sorted in increasing order with their indexes $id_1, id_2, .., id_n$
2: $ub \Leftarrow 0$
3: $C \Leftarrow \{\}$
4: for $i = 1, 2, ..., n$ do
5:     $ub \Leftarrow$ by dichotomy, the highest index $k$ $(ub \leq k \leq n)$ such that $\phi_i(x_i, x_k)$ is TRUE
6:     for $j = i, ..., ub$ do
7:         $C = C \cup \{(id_i, id_j)\}$
8:     end for
9: end for
10: return $C$
```

5.2.3. Computing the covering number

Once the conflict graph $\mathsf{CG}_\Phi(r, \varphi) = (r, E)$ resulting from the CEE is constructed, the g_3-error can be evaluated by computing the covering number $\beta(\mathsf{CG}_\Phi(r, \varphi))$. There are two main types of algorithms for solving the MVC [CSLS13]: *exact* algorithms and *heuristic* algorithms. Exact algorithms guarantee the optimality of their solution, but may take an unreasonable amount of time for large graphs. In contrast, heuristic algorithms, such as local-search algorithms, do not provide a guarantee, but are known in practice to propose near-optimal solutions within a reasonable time (usually set by the user) without being constrained by the size of the graph. We use the exact solver `WeGotYouCovered` [HLSS20] (`EXA_WGYC`), winner of PACE 2019, and the local search algorithm `NuMVC` [CSLS13] (`HEUR_NuMVC`(t) where t is the running time), which runs for a given time specified by the user and returns the best solution found.

What about the decision problem? For the decision problem (EVPP), it is possible to optimize the search space by using a fixed-parameter tractable (FPT) algorithm for the MVC. The problem is therefore not to find an MVC but to answer the question: *is there an MVC of size smaller than k?* For example, [CKX06] makes it possible to solve it in $\mathcal{O}(k \cdot n + 1.2738^k)$ with $k = n \cdot \eta_e$. However, this algorithm has a Klam value [DF12] of 190 (maximum value of k for which the algorithm is expected to be practical) which makes it suitable only for small thresholds η_e. To the best of our knowledge, the MIS does not have any FPT algorithm and the confidence validation problem is still intractable after this relaxation [DF12].

5.2.4. Approximation algorithms for the covering number

When datasets become very large, exact algorithms become unusable (see experiments in Section 5.4). Thus, we can use approximation algorithms to obtain *estimates* of the target value, along with some *theoretical guarantees*. In this case, a clear distinction must be made between the g_3-*error* and the *confidence*, since their associated graph problems (the MVC and MIS respectively) *do not share the same approximation guarantees*. If the MVC cannot be approximated with a better factor than 1.3606 [DS05], its dual the MIS is even harder and cannot be approximated within a constant factor [GJ79]. Nevertheless, the many studies on hard problems have proposed solutions which are explored in this section.

Approximation algorithms generally express their guarantees in terms of a ratio, which can be constant or dependent on the problem parameters. This ratio is expressed differently if we consider a minimization problem (*e.g.* MVC) or a maximization problem (*e.g.* MIS). Consider β and α, the *covering number* and the *independence number* of $\mathsf{CG}_\Phi(r,\varphi)$ respectively. Let $\hat{\beta}$ and $\hat{\alpha}$ be their approximation obtained with an approximation algorithm. Given a k-approximation algorithm for these problems, where k is the approximation ratio, we get:

$$\beta \le \hat{\beta} \le k \cdot \beta,$$
$$k^{-1} \cdot \alpha \le \hat{\alpha} \le \alpha,$$

or equivalently

$$g_3 \le \hat{g}_3 \le k \cdot g_3,$$
$$k^{-1} \cdot \mathsf{conf} \le \hat{\mathsf{conf}} \le \mathsf{conf}.$$

The best constant-factor approximation algorithm for the MVC is that of the algorithm of Gavril and Yannakakis. Their procedure (described in [PS98]) greedily computes the size of a maximal matching in a graph. This 2-approximation algorithm has been implemented as `APPROX_2Approx` in this thesis. A more complex algorithm [Kar09] achieves a better factor of $2 - \Theta\left(\frac{1}{\sqrt{\log(|V|)}}\right)$ but depends on the problem parameters. However, an algorithm with a lower approximation factor is not guaranteed to provide the best results in practice. This is why Delbot and Laforest propose an experimental comparison of 6 approximation algorithms which confirms this gap between theory and practice [DL10]. In particular, the Greedy Independent Cover (GIC) algorithm [HR97] emerges as the clear winner of this benchmark and is implemented as `APPROX_GIC`. While it only provides an approximation ratio of at least $\frac{\sqrt{d}}{2}$ (where d is the maximum degree of the graph), it often provides near-perfect approximations on a wide variety of graphs in log-linear time. In this case, these algorithms are very fast to run and the CEE becomes the bottleneck of the operation. Therefore, computing multiple approximations and taking the smallest is a viable option as it only requires the CEE to be performed once. For example, `APPROX_2Approx` can still be useful in cases where `APPROX_GIC` might not perform well.

For the MIS, there is currently no known constant-factor approximation algorithm. However, a well-known algorithm with parameterized approximation factor for this problem is the minimum greedy algorithm. It is presented in [HR97] and serves as an inspiration for the GIC algorithm presented above. Its guarantee is an approximation factor of $\frac{d+2}{3}$. More generally, any upper bound on the MVC acts as a lower bound on the MIS, but without the same approximation guarantees.

It is also often useful to express lower bounds and upper bounds for the MVC and the MIS respectively. First, a *maximal* matching M can be found in linear time [PS98]. In this case, $|M|$ serves as a lower-bound for β and $1-|M|$ is an upper-bound for α. A *maximum* matching can even be found in $\mathcal{O}(\sqrt{n} \cdot |E|)$ with the MV Matching algorithm [MV80], tightening the previous result. Several upper and lower bounds are examined in [Wil11]. We do not implement or experiment with lower bounds in the experiments in Section 5.4.

5.2.5. Mitigating counterexample enumeration thanks to sublinear algorithms

As mentioned earlier, there are very efficient approximation algorithms in the literature for estimating an MVC in linear or log-linear time while the CEE is inevitably quadratic. This is why in practice, the bottleneck in the computation of g_3 comes from the CEE rather than the computation of the MVC itself. This will be confirmed later in our experiments. In this case, sublinear algorithms provide an efficient alternative: *they are able to estimate the size of the MVC without looking at the whole graph.* They only need to construct the part of the graph that the algorithm explores, thus reducing the complexity of the CEE. In this section, we propose an *on-the-fly* adaptation of the CEE to benefit from sublinear algorithms. Then, we present two state-of-the-art sublinear algorithms and briefly describe their implementation in this thesis.

The on-the-fly CEE. As explained in [PR07], common sublinear algorithms for the MVC have only two ways of exploring a graph, they can (1) ask for the degree of a given vertex and (2) ask for the i^{th} neighbor of a vertex. To take advantage of this exploration strategy, the CEE must be performed on-the-fly by fetching all tuples that form a counterexample with a given one *only* on demand. More formally, the tuples that form a counterexample with a tuple $t \in r$ correspond to the neighbors of t in $\mathsf{CG}_{\Phi}(r, \varphi)$. The previous approach to finding these neighbors was to compute the whole conflict-graph using the CEE. However, we have shown that this is particularly expensive, and now explain a way to avoid this operation by computing the neighborhoods of the required tuples one at a time. For a given tuple t, consider the list of tuples that form a counterexample with t. The options (1) and (2) correspond respectively to the *length* and the i^{th} element of this *list*. From an algorithmic point of view, asking for the degree is equivalent to listing all counterexamples and thus listing the complete neighborhood of t. Once this is done, asking for the neighbor of i^{th} is completely free, since it has already been found. In the case where the algorithm asks for the i^{th} neighbor of t before asking for its degree, the operation can be optimized by stopping the enumeration as soon as the i^{th} neighbor is found.

Furthermore, the enumeration process can benefit from all the optimizations proposed in Section 5.2.2 by keeping all relevant information in memory (blocking, ordered attributes, type of join, etc.). The complexity of the *on-the-fly CEE* to retrie all the neighbors of a tuple $t \in r$ ranges from the size of the neighborhood to the total number of tuples in r depending on the optimizations available. It is therefore possible to propose a graph proxy hiding an on-the-fly CEE procedure to any sublinear graph algorithm, saving time and memory in comparison to complete the CEE.

Process overview and implementation. Let $\mathcal{A}(G)$ be an algorithm for computing an approximate MVC of graph G. As initially proposed in [PR07], most sublinear algorithms for the MVC work as follows:

(i) Sample a set of vertices from the graph.

(ii) Decide for each vertex if it belongs to $\mathcal{A}(G)$.

(iii) Generalize the result to estimate the size of the MVC on the full graph.

We understand from this process that the quality of the result depends on the approximation performances of $\mathcal{A}$, knowing that not all algorithms can be made sublinear. Currently, most of the literature focused on adapting the 2-approximation algorithm `APPROX_2Approx` presented in the previous section as it is the best known constant-factor approximation. Thus, *these sublinear algorithms propose an estimate of the result of a 2-approximation algorithm*. Notably, they should provide results equal to or worse than `APPROX_2Approx`. In particular, we implement [YYI09] (`APPROX_Sub09`) and [ORRR12] (`APPROX_Sub11`) with respective query complexities of $\mathcal{O}(\frac{d^4}{\epsilon^2})$ and $\mathcal{O}(\overline{d}^2 \cdot \text{poly}(\frac{1}{\epsilon}))$[1] (where d and $\overline{d}$ are the maximum and average degrees of the graph). Following Hoeffding's inequality, by sampling $m = \min(n, \left\lceil \frac{1}{2\epsilon^2} \ln(\frac{2}{1-\delta}) \right\rceil)$ nodes, those algorithms provide an approximation $\hat{\beta}$ of β such that:

$$p(\beta - n \cdot \epsilon \leq \hat{\beta} \leq 2 \cdot \beta + n \cdot \epsilon) \geq \delta,$$

or

$$p(g_3 - \epsilon \leq \hat{g}_3 \leq 2 \cdot g_3 + \epsilon) \geq \delta.$$

These sublinear algorithms are implemented for testing in the experiments of Section 5.4.4. To the best of our knowledge, this is the first implementation of those algorithms.

[1]Corrected from original paper following discussion with the authors.

5.3. Polynomial algorithms for transitive and symmetric predicates

In this section, we restrict the predicates to be *at least transitive and symmetric*, a case where the g_3-error can be computed in polynomial time as shown in Chapter 4. We explain how to benefit from previous literature and give efficient exact and approximation algorithms to take advantage of these new constraints.

An important difference with Section 5.2 is that *we can process the relation directly without creating the conflict graph.* This can be seen as an optimization as we could still use the previous method of creating the conflict graph and then solving the MVC. In fact, as explained in Section 4.2, the use of transitive and symmetric predicates *constrains* the conflict graph to a certain class in which an MVC can be found in polynomial time. Hence, both this polynomial algorithm for the MVC and the polynomial algorithm for solving g_3 directly are actually closely related. Naturally, due to the ubiquity of classical equality, processing the relation directly is also the first proposition in the literature as it is the easiest way to compute g_3 in this case.

First, we propose a pre-processing step to convert all polynomial cases to a single case for which solutions to compute the g_3 error are known from the literature. Then, we analyse two exact algorithms which optimize time and memory respectively. Finally, we study approximation algorithms for very large datasets. In particular, after analyzing the guarantees of uniform sampling, we propose an improvement of an existing stratified sampling approach [CGF+09]. We also mention the analysis proposed by [KM95] for the EVPP, the corresponding decision problem. A summary of all proposed algorithms is presented in Table 5.3 and will be detailed in the following sections.

Table 5.3 – SUMMARY OF g_3 ALGORITHMS IN THE POLYNOMIAL CASE.

Name	Approx?	Complexity[a]
`EXA_MemOpt`	·	$\mathcal{O}(n \cdot \log(n))$
`EXA_TimeOpt`	·	$\mathcal{O}(n)$
`APPROX_URS(A)`	Yes	$T_{\mathtt{A}}(m)$[b]
`APPROX_SRS`[CGF+09]	Yes	$\mathcal{O}(n)$
`APPROX_SRSI`	Yes	$\mathcal{O}(n)$

[a] n corresponds to the size of the relation $|r|$.

[b] $T_{\mathtt{A}}(m)$ corresponds to the complexity of algorithm $\mathcal{A}$ for a random sample of r of size m.

5.3.1. Computation overview

Let (R, Φ) be a relation schema with predicates *at least transitive* (`tra`) *and symmetric* (`sym`), r a relation over R and $\varphi : X \to A$ a FD. As shown in Section 4.2, in this case the computation of $g_3^{\Phi}(\varphi, r)$ can be done in polynomial time. To solve all polynomial cases shown in

Figure 4.2, we propose to pre-process the relation to make the predicates artificially reflexive (`ref`) by restricting the active domain. In this case, predicates have all the properties of an equivalence relation (*e.g.* classical equality) which allows to obtain the *equivalence classes* described in the preliminaries (Chapter 2). In this case, *the strategy for solving g_3 in polynomial time is known from literature [HKPT99, CGF+09, LPS20].* Indeed, previous works have studied the equality, *i.e.* a case where predicates enjoy all the properties. However, anti-symmetry (`asym`) is not exploited in the algorithms and only the equivalence relation conferred by transitivity, symmetry and reflexivity is used.

We now describe the process for artificially adding the reflexive property to the attributes' predicates. To do this, we simply need to remove all tuples which do not respect reflexivity. Those are the tuples which are not similar to themselves on at least one attribute of φ w.r.t. the predicates. Thus, predicates become artificially reflexive *on the remaining set of tuples*. In other words, by restricting the active domain of the relation, we constrain the graph to respect the exact same structures as if the predicates were indeed transitive, symmetric *and reflexive*. This allows us to apply the same polynomial algorithms which can be found in the literature. Nonetheless, those deleted tuples must be accounted for in the final computation of the g_3-error. Interestingly, as stated in Proposition 4, the role of those tuples differs if their lack of reflexivity emerges from the antecedents or the consequents.

PROPOSITION 4. *Let (R, Φ) be a relation schema with predicates where each predicate in Φ is at least symmetric and transitive. Let r be a relation over R, and $X \to A$ a functional dependency over R. The following properties hold for every tuple t in r:*

(i) $\bigwedge\limits_{B \in X} \phi_B(t[B], t[B]) = \text{FALSE}$ *implies* $(t, t') \models X \to A$ *for every other t' in r*

(ii) $(t, t) \not\models X \to A$ *implies* $(t, t') \not\models X \to A$ *for every other t' in r such that* $\bigwedge\limits_{B \in X} \phi_B(t[B], t'[B]) = \text{TRUE}$

Proof. We prove the properties in order :

(i) Assume for contradiction that there exists t' in r such that $\bigwedge\limits_{B \in X} \phi_B(t'[B], t[B]) = \text{TRUE}$. As $\bigwedge\limits_{B \in X} \phi_B(t[B], t[B]) = \text{FALSE}$, we also have $\bigwedge\limits_{B \in X} \phi_B(t'[B], t[B]) = \text{FALSE}$ due to the transitivity of the predicates. A contradiction. Hence, $(t, t') \models X \to A$ must hold.

(ii) Assume for contradiction that there exists t' in r such that $\bigwedge\limits_{B \in X} \phi_B(t[B], t'[B]) = \text{TRUE}$ and $\phi_A(t'[A], t[A]) = \text{TRUE}$. As $(t, t) \not\models X \to A$, we must have $\phi_A(t[A], t[A]) = \text{FALSE}$. Due to the transitivity of the predicates, we should also have $\phi_A(t'[A], t[A]) = \text{FALSE}$. A contradiction. Hence, $(t, t') \not\models X \to A$ must hold for every $t' \in r$ such that $\bigwedge\limits_{B \in X} \phi_B(t[B], t'[B]) = \text{TRUE}$.

■

From Proposition 4, we can design the pre-processing procedure presented in Algorithm 2. This procedure takes each tuple one by one: if a value of the tuple on one attribute of the FD is not similar to itself, the tuple is *non-reflexive* and is removed from the relation. More precisely, if one attribute in the antecedents is non-reflexive, we remove the tuple and increment a counter (case (i)). If all the attributes in the antecedents are reflexive but the consequent A is non-reflexive, we just remove the tuple (case (ii)). Otherwise, we do nothing. This algorithm returns the new relation and the counter. This pre-processing step is quite light and only requires a linear pass over the data.

Algorithm 2 Relation pre-processing for artificially adding reflexivity to predicates (RefPreproc)

Require:
a relation schema with predicates satisfying tra and sym (R, Φ)
a relation r over (R, Φ)
a functional dependency $X \to A$ over (R, Φ)

```
1:  $C \Leftarrow 0$
2:  for tuple $t \in r$ do
3:      if $\bigwedge_{B \in X} \phi_B(t[B], t[B]) = \text{FALSE}$ then
4:          $C{+}{+}$
5:          $r \Leftarrow r \setminus \{t\}$
6:      else if $\phi_A(t'[A], t[A]) = \text{FALSE}$ then
7:          $r \Leftarrow r \setminus \{t\}$
8:      end if
9:  end for
10: return $C$, $r$
```

We now show how to use this pre-processing step by introducing Algorithm 3. This algorithm can compute g_3 with predicates satisfying transitivity and symmetry by using an algorithm for predicates satisfying reflexivity, transitivity and symmetry, a well-studied case in the literature. A formal description of this algorithm is proposed in Proposition 5.

PROPOSITION 5. *Let (R, Φ) be a relation schema with predicates satisfying tra and sym. Let r be a relation and φ a functional dependency over (R, Φ). Let A be an algorithm for computing g_3 with predicates satisfying ref, tra and sym. We have TraSym$((R, \Phi), \varphi, r, A) = g_3^{\Phi}(\varphi, r)$.*

Proof. Let (R, Φ) be a relation schema with predicates satisfying transitivity and symmetry. Let r be a relation and φ a functional dependency over (R, Φ). Let A be an algorithm for computing g_3 with predicates satisfying reflexivity, transitivity, and symmetry. Let C^{tmp} and r' be the outputs of RefPreproc$((R, \Phi), r, \varphi)$. From Proposition 4, we know that r' does not contain any "non-reflexive" tuple, *i.e.* tuples which are not similar to themselves on at least one attribute w.r.t. Φ. Thus, the predicates from Φ also respect reflexivity on the active domain of r' and we have $\mathtt{A}((R,\Phi),\varphi,r') = g_3^{\Phi}(\varphi,r')$. Adapting this result to the size of r w.r.t. C^{tmp}, we obtain $g_3^{\Phi}(\varphi,r)$. ■

Algorithm 3 g_3 computation with predicates at least transitive and symmetric (`TraSym`)

Require:

a relation schema with predicates satisfying `tra` and `sym` (R,Φ)
a functional dependency φ over (R,Φ)
a relation r over (R,Φ)
an algorithm `A` for computing g_3 with predicates satisfying `ref`, `tra` and `sym`

```
1: $C^{tmp}, r' \Leftarrow \mathtt{RefPreproc}((R,\Phi),r,\varphi)$
2: $C \Leftarrow C^{tmp} + |r'| \cdot (1 - \mathtt{A}((R,\Phi),\varphi,r'))$    ▷ multiplying by $|r'|$ to denormalize $g_3$
3: return $1 - \frac{C}{|r|}$
```

Once the problem is restricted to the classical g_3 with predicates satisfying reflexivity, transitivity and symmetry, *i.e.* corresponding to equivalence relations, g_3 can be computed according to the following steps:

(i) find the equivalence classes of $\Omega_X(r)$,

(ii) for each equivalence class, find the most frequent A value and count its number of occurrences.

Indeed, finding the most frequent A value in a class allows to discard a minimum number of tuples in the corresponding class.

5.3.2. Exact computation

To find the equivalence classes, it is necessary to perform a GROUP-BY operation on X. GROUP-BY operations are generally sort- or hash-based with sorting optimizing memory and hashing time complexity [MSL+15]. We study both strategies. The approach for finding the most frequent element in each equivalence class also depends on the GROUP-BY strategy.

Sorting (`EXA_MemOpt`). After sorting the data, we can compute g_3 in one pass over the data. If the data is also sorted externally and then read in a streaming fashion to find each equivalence class, this method allows for low memory consumption. In particular, the memory required can be adjusted depending on the chunks used for external sorting, with memory consumption ranging from $\mathcal{O}(1)$ to $\mathcal{O}(n)$. The total time complexity is $\mathcal{O}(n \cdot \log(n))$ and corresponds to the sorting operation. The pseudo-code is presented in Algorithm 4.

Algorithm 4 Memory optimized g_3 (EXA_MemOpt)

Require:
a relation schema with predicates satisfying ref, tra and sym (R, Φ)
a functional dependency $X \rightarrow A$ over (R, Φ)
a relation r over (R, Φ)

```
1: Sort $r$ on $X$ then $A$
2: $X_{current}, A_{current} \Leftarrow \emptyset, \emptyset$
3: $C \Leftarrow 0$
4: $c_{max}, c_{current} \Leftarrow 0, 0$
5: **for all** tuple $t \in r$ **do**
6:     **if** $t[X] \neq X_{current}$ **then**
7:         $C \Leftarrow C + c_{max}$
8:         $c_{max}, c_{current} \Leftarrow 0, 0$
9:         $X_{current} \Leftarrow t[X]$
10:        $A_{current} \Leftarrow t[A]$
11:    **end if**
12:    **if** $t[A] = A_{current}$ **then**
13:        $c_{current} \Leftarrow c_{current} + 1$
14:    **else**
15:        $c_{current} \Leftarrow 1$
16:        $Y_{current} \Leftarrow t[A]$
17:    **end if**
18:    **if** $c_{max} < c_{current}$ **then**
19:        $c_{max} = c_{current}$
20:    **end if**
21: **end for**
22: $C \Leftarrow C + c_{max}$
23: **return** $1 - \frac{C}{|r|}$
```

Hashing (EXA_TimeOpt). In this case, we can compute g_3 in one pass over the data by storing each equivalence class in a hash table. Hashing is less memory efficient, but allows a lower theoretical time complexity of $\mathcal{O}(n)$. The theoretical memory requirement is $|\Omega_X(r)| + \sum_{r_i \in \Omega_X(r)} |r_i[A]| = \mathcal{O}(n)$. However, depending on the hashing algorithm, significant memory and time overheads may be required. The pseudo-code is presented in Algorithm 5.

Algorithm 5 Time optimized g_3 (EXA_TimeOpt)

```
Require:
    a relation schema with predicates satisfying ref, tra and sym $(R, \Phi)$
    a functional dependency $X \to A$ over $(R, \Phi)$
    a relation r over $(R, \Phi)$

$C \Leftarrow 0$
$map \Leftarrow \{\}$
for all tuples $t \in r$ do
    if $t[X] \notin map$ then
        $map[t[X]] \Leftarrow \{\}$
    end if
    if $t[A] \notin map[t[X]]$ then
        $map[t[X]][t[A]] \Leftarrow 0$
    end if
    $map[t[X]][t[A]]{+}{+}$
end for
for all key $k_X \in$ map do
    $C \Leftarrow C + \max_{k_A \in map[k_X]}(map[k_X][k_A])$
end for
return $1 - \frac{C}{|r|}$
```

5.3.3. Random sampling

For very large datasets, the computation of g_3 can become overwhelming in terms of time complexity and memory management. We present techniques for its scalability based on random sampling.

Uniform random sampling (APPROX_URS)

We first consider the attractive approach of URS proposed in Algorithm 6. This simple approach is easy to implement, allows for massive gains in computation time and proposes good theoretical guarantees presented in Theorem 5.

Algorithm 6 Uniform sampling g_3 (APPROX_URS)

```
Require:
    a relation schema with predicates satisfying ref, tra and sym $(R, \Phi)$
    a functional dependency $\varphi$ over $(R, \Phi)$
    a relation $r$ over $(R, \Phi)$
    an algorithm A for computing $g_3$ with predicates satisfying ref, tra and sym
1: $m \Leftarrow \min\left(|r|, \left\lceil \frac{1}{2\epsilon^2} \ln(\frac{2}{1-\delta}) \right\rceil\right)$
2: $s \Leftarrow$ uniform random sample of size $m$ drawn from $r$
3: return $\mathtt{A}((R, \Phi), \varphi, s)$
```

THEOREM 5. *Let (R, Φ) be a relation schema with predicates satisfying* ref, tra, sym *and* asym. *Let r be a relation and φ a functional dependency over (R, Φ). Let $\mathtt{A}((R, \Phi), \varphi, r)$ be an algorithm for computing $g_3^\Phi(\varphi, r)$ in $T_{\mathtt{A}}(|r|)$ time. For an error ϵ and a confidence δ,* APPROX_URS$((R, \Phi), \varphi, r, \mathtt{A}, \epsilon, \delta)$ *computes an estimate $\hat{g}_3^\Phi$ of $g_3^\Phi(\varphi, r)$ such that $p(|\hat{g}_3^\Phi - g_3^\Phi| \leq \epsilon) \geq \delta$. Its time complexity is $\mathcal{O}(T_{\mathcal{S}}(m, |r|) + T_{\mathtt{A}}(m))$ with $m = \min\left(|r|, \left\lceil \frac{1}{2 \cdot \epsilon^2} \cdot \ln(\frac{2}{1-\delta}) \right\rceil\right)$ and $T_{\mathcal{S}}(m, |r|)$ is the complexity of sampling m tuples in r.*

The proof is presented in Appendix C. Despite its simplicity, this approach often performs badly owing to the many specificities of real datasets such very small equivalence classes, too many different consequents, etc. (see Section 5.4 for experiments).

Advanced sampling schemes

To obtain a more efficient algorithm in practice, we also examine the advanced sampling schemes presented by Cormode et al. to compute the confidence of the CFDs [CGF+09]. These strategies can be applied almost directly to our use case, since the CFDs only require a filtering phase before processing [CGF+09]. They propose a variety of solutions, including a 2-pass stratified random sampling, which gives the best results in their experiments. Consequently, we implement this algorithm (APPROX_SRS) in this thesis. In particular, we use [Li94] instead of the classical reservoir sampling approach [Vit85]. This strategy speeds up the first pass considerably if the size of the dataset is known. We now describe this algorithm and propose a way to improve it.

For a sample s of r, we obtain two partitions $\Omega_X(s) = \{s_1, s_2, ..., s_\ell\}$ and $\Omega_X(r) = \{r_1, r_2, ..., r_m\}$ with $\ell \leq m$. Observe that, since $s \subseteq r$, each s_i is contained in a unique r_j. For clarity, we assume $s_i \subseteq r_i$. The APPROX_SRS algorithm proposes an estimate of the g_3-error in two passes over r:

(i) In a *first pass*, it builds a sample s by reservoir sampling $p_1 = \frac{2}{\epsilon_1^2} \cdot log(\frac{2}{\delta_1})$ tuples from r. Thanks to the size of each equivalence class s_i in $\Omega_X(s) = \{s_1, s_2, ..., s_\ell\}$, it provides an estimate $\widehat{|r_i|} = \frac{|s_i|}{|s|} \cdot |r|$ of the size of each equivalence class r_i in $\Omega_X(r) = \{r_1, r_2, ..., r_m\}$ with error ϵ_1 and confidence δ_1.

(ii) In a second pass, for each $s_i \in \Omega_X(s)$, it builds a sample u_i of p_2 tuples chosen in r_i with reservoir sampling. Then, the sample u_i is used to compute an estimate $\widehat{g_3^\Phi}(\varphi, r_i) = g_3^\Phi(\varphi, s_i)$ of $g_3^\Phi(\varphi, r_i)$.

Finally, based on s, it computes a weighted average to provide an estimate of g_3 for the full relation such that:

$$\widehat{g_3^\Phi}(X \rightarrow A, r) = 1 - \sum_{s_i \in \Omega_X(s)} \frac{|s_i|}{|s|} \cdot g_3^\Phi(\varphi, s_i).$$

Note that we have not specified the size of the reservoir p_2 in the second pass. In fact, two methods are proposed by Cormode et al. for sampling in the second pass [CGF+09]: reservoir sampling and the space-saving algorithm [MAEA05]. In both cases, the value chosen for p_2 does not take into account the size of the individual equivalence classes, which can lead to extremely over/underestimated sample sizes. In their experiments, the authors use a constant value of $p_2 = 20$. However, while 20 can give a good estimate for small equivalence class sizes, it is not able to correctly estimate g_3 for very large ones (*e.g.* 50,000 tuples). On the contrary, choosing a large value such as 5000 performs better for large equivalence classes, but also samples every element of smaller ones, leading to a decrease in performance. Therefore, we observe that it is not possible to estimate a good constant value for p_2, since small and large groups can coexist in the same dataset. Therefore, we propose to use a variable reservoir size p_i for each equivalence class $r_i \in \Omega_X(r)$, depending on its size $|r_i|$ (improvement implemented in APPROX_SRSI). In particular, we use the estimate $\widehat{|r_i|}$ made in the first pass for the size of each equivalence class r_i and then use Hoeffding's inequality with finite population correction [Ser74] to provide an adaptive sample size p_i such that:

$$p_i = \left\lceil \left(\frac{2 \cdot \epsilon_2^2}{\ln\left(\frac{2}{1-\delta_2}\right)} + \frac{|s|}{|s_i| \cdot |r|} \right)^{-1} \right\rceil \tag{5.1}$$

where ϵ_2 and δ_2 are the error and confidence.

This solution offers two major improvements: there is no need to assume anything about the data beforehand to manually choose p_2 and each reservoir size p_i is chosen to give good guarantees on the approximation while sampling *just enough* tuples. To illustrate this improvement, consider three equivalence classes of sizes 2, 300 and 5000. For $\delta_2 = 0.95$ and $\epsilon_2 = 0.05$, the original method would have required a constant number of samples, such as $p_2 = 20$. With our proposal, and taking into account accurate estimates of the true sizes in the first pass, these groups would require 2, 214 and 643 respectively. We will see that this approach (APPROX_SRSI) works very well in practice and always outperforms APPROX_SRS. The pseudo-code for APPROX_SRSI is given in algorithm 7.

Algorithm 7 Improved stratified random sampling g_3 (APPROX_SRSI) - derived from [CGF+09]

```
Require:
    a relation schema with predicates satisfying ref, tra, sym and asym $(R, \Phi)$
    a relation r over $(R, \Phi)$
    a functional dependency $\varphi : X \rightarrow A$ over $(R, \Phi)$
    error $\epsilon_1$ and confidence $\delta_1$ to use for the first pass
    error $\epsilon_2$ and confidence $\delta_2$ to use for the second pass

1: $C \Leftarrow 0$
2: $s \Leftarrow \emptyset$
3: $p_1 = \frac{2}{\epsilon_1^2} \cdot log(\frac{2}{\delta_1})$
4: for all tuple $t \in r$ do                                   ▷ First pass
5:     reservoir sample $t$ in $s$ of size $p_1$
6: end for
7: for all $s_i \in \Omega_X(s)$ do
8:     $p_i \Leftarrow \left\lceil \left( \frac{2 \cdot \epsilon_2^2}{\ln\left(\frac{2}{1-\delta_2}\right)} + \frac{|s|}{|s_i| \cdot |r|} \right)^{-1} \right\rceil$
9:     $s_i \Leftarrow \emptyset$
10: end for
11: for all tuple $t \in r$ do                                  ▷ Second pass
12:     reservoir sample $t$ in $s_i$ of size $p_i$
13: end for
14: for all stored $s_i$ do
15:     $C \Leftarrow C + \frac{|s_i|}{|s|} \cdot |r| \cdot \max_{x \in s_i[A]} |\{t \mid t[A] = x, t \in s_i\}|$
16: end for
17: return $1 - \frac{C}{|r|}$
```

What about the decision problem? The decision problem (EVPP) with random sampling is treated in the original paper introducing g_3 [KM95]. Notably, it is proven that a uniform sample of size at least $\mathcal{O}(\frac{\sqrt{n}}{k} \cdot \log \frac{1}{\delta})$ is required to solve it with probability δ with k the EVPP threshold parameter. This result also holds for the confidence validation problem. Note that this number is generally pretty high in regard to the size of r.

5.4. Experiments

In this section, we present extensive experiments with the algorithms proposed in this chapter. We study their time performance as well as their approximation accuracy on real and synthetic datasets. First, we introduce FASTG3, the Python library we developed to perform the experiments. Second, we present our datasets and the context of the experiments. Finally, we present and discuss our results.

5.4.1. The FASTG3 Python library

Logo of FASTG3.

FASTG3 is an open-source library for computing the g_3 indicator. This library is proposed as a Python module and achieves very good performance thanks to an underlying C++ implementation based on Cython. In its current version, FASTG3 supports predicates in the form of relative and absolute uncertainties. Such predicates are especially useful when comparing measured values such as sensors values, common in industrial instrumentations and especially at the CNR. Let τ_{abs} and τ_{rel} be respectively the absolute and relative uncertainties of some attribute, such predicate can be defined as follows:

$$\phi^u(x, y; \tau_{abs}, \tau_{rel}) = \begin{cases} \text{TRUE} & \text{if } |x - y| \leq \tau_{abs} + \tau_{rel} \cdot \max(|x|, |y|) \\ \text{FALSE} & \text{otherwise.} \end{cases} \tag{5.2}$$

Those predicates correspond to the NP-hard case of computing g_3 and support all optimizations mentioned in Section 5.2.2 for CEE. FASTG3 also implements the classical equality of crisp FDs.

All algorithms implemented and tested are summarized in Tables 5.2 and 5.3. Multiple optimizations are also proposed and are summarized in Table 5.1. The source code of FASTG3 is available publicly on GitHub (github.com/datavalor/fastg3) and it is composed of about 4500 lines of code. All benchmarks (also available on the repository) are run on the following configuration: *Ubuntu 22.04, Python 3.10, i7-7700k, 32GB of memory.* All given running times are based on an average of ten runs.

5.4.2. Datasets

Two real-life datasets and a synthetic one are used for the experiments:

– `Diamonds`. This public dataset is composed of 53,940 tuples with 9 categorical and numerical attributes describing various properties of a set of diamonds (see Table 5.4). The price of the diamonds is the target value which depends on the other attributes.

– `Hydroturbine`. Composed of 511,017 tuples, this dataset is similar to our running example with 6 numerical attributes describing various properties of a water turbine (see Table 5.5). The power produced by the turbine is the target value which depends on the other attributes. This dataset is given by the CNR, the funder of this thesis, and cannot be made public.

Table 5.4 – ATTRIBUTES OF THE `Diamonds` DATASET.

Name	Type	Description
price	numerical	price in US dollars
carat		weight
x		length in *mm*
y		width in *mm*
z		depth in *mm*
depth		total depth percentage ($\frac{2 \cdot z}{x+y}$)
table		width of top of diamond relative to its widest point
cur	categorical	quality of the cut
color		diamond color
clarity		a measurement of how clear the diamond is

Table 5.5 – ATTRIBUTES OF THE `Hydroturbine` DATASET. SEE SECTION 1.3 FOR A DETAILED DESCRIPTION OF THE ATTRIBUTES.

Name	Type	Description
power	numerical	power produced by the turbine (Megawatts)
flow		flow inside the turbine in $m^3 \cdot s^{-1}$
head		water height difference between upstream and downstream in m
opening		percentage of upstream valve opening

– `Syn`**.** We also use a generator of synthetic datasets making it possible to choose multiple parameters related to the computation of g_3 with equality predicates (crisp FDs). It is defined as $\texttt{Syn}(g = 0.5, n = 1M, e = 300, a = 2, c = 1, u = 0)$ with known g_3 value (g) as well as variable numbers of tuples (n), equivalence classes (e), antecedents (a), and consequents (c) and percentage of unique consequents in each equivalence class while keeping the target g_3 achievable (u). We use the default values in the following experiments when parameters are not defined. x means that the parameter is currently being tested (*e.g.* $\texttt{Syn}(g = x, n = 100)$).

Remark 3. As long as the number of antecedents or consequents is not changed in the experiments, the default FDs presented in the following sections are used. The predicates used in Section 5.4.4 correspond to uncertainties devised with domain experts for `Hydroturbine` and are estimated for `Diamonds` as it is a public dataset. The FDs themselves correspond to potential functions one may encounter, for example, in a prediction problem.

5.4.3. Classical equality (polynomial case)

In this first series of experiments, we explore the case where equality is used, *i.e.* crisp FDs. This is the simplest and most classical solution for comparing values. More precisely, it falls in the polynomial case of our complexity analysis (see Figure 4.2) and in particular in the {`ref`, `tra`, `asym`, `sym`} group. Hence, the preprocessing phase presented in Section 5.3.1 is unnecessary and thus not studied. Either way, this preprocessing algorithm is linear and its examination would not be particularly interesting in our case. This case corresponds to the computation of g_3 with crisp FDs.

Settings

The FDs used in this section for `Diamonds` and `Hydroturbine` are respectively:

- carat, cut, color, clarity, depth $\rightarrow$ price
  - 41,350 equivalence classes, $g_3 = 0.20$
- flow, opening, position $\rightarrow$ power
  - 354,867 equivalence classes, $g_3 = 0.13$

Unless otherwise indicated, sampling algorithms use confidence $\delta = 0.95$ and error $\epsilon = 0.01$. This corresponds to $\mathrm{min}(18445, n)$ tuples sampled in the first pass of `APPROX_SRS` and `APPROX_SRSI` as well as the sample size used by `APPROX_URS`. `APPROX_URS` is used in conjunction with `EXA_TimeOpt`. If `APPROX_SRSI` chooses the reservoir size in the second pass using formula 5.1 (with $\delta = 0.95$ and $\epsilon = 0.05$ in our experiments), a value needs to be chosen for `APPROX_SRS`. We use $p_2 = 100$ which is greater than the original value of 20 used in [CGF+09] but it seemed fairer regarding the variety of our datasets.

Results

Figures 5.1 and 5.2 present the effect of the number of tuples on the time and approximation performances. We observe that for small datasets such as `Diamonds`, the sorting operation is not an issue and that `EXA_TimeOpt` is not better than `EXA_MemOpt`. Interestingly, they achieve the same performance on the larger dataset `Hydroturbine` and we see that `EXA_TimeOpt` largely beats `EXA_MemOpt` on `Syn`$(n = 100M)$. The reasons for this is the presence of large equivalence classes and the small number of unique consequents of `Syn` which allow the hashing algorithm to reallocate memory less often. The results for the `Syn` dataset show the linear scalability of our solutions for very large datasets (tests up until $n = 100M$). As shown in Figure 5.3, the number of antecedents has an important effect on the running time. Indeed, the tuple-to-tuple equality check used in every algorithm (hashing, sorting, etc.) becomes longer as the number of attributes' values to be compared grows ($\approx$ linear effect).

For random sampling, we observe that `APPROX_SRSI` and `APPROX_SRS` have approximately the same running times. They are generally not faster than exact algorithms for small datasets such as `Diamonds` owing to their computation overheads (notably reservoir sampling) but become efficient for large ones. We see that `APPROX_SRSI` is always at least

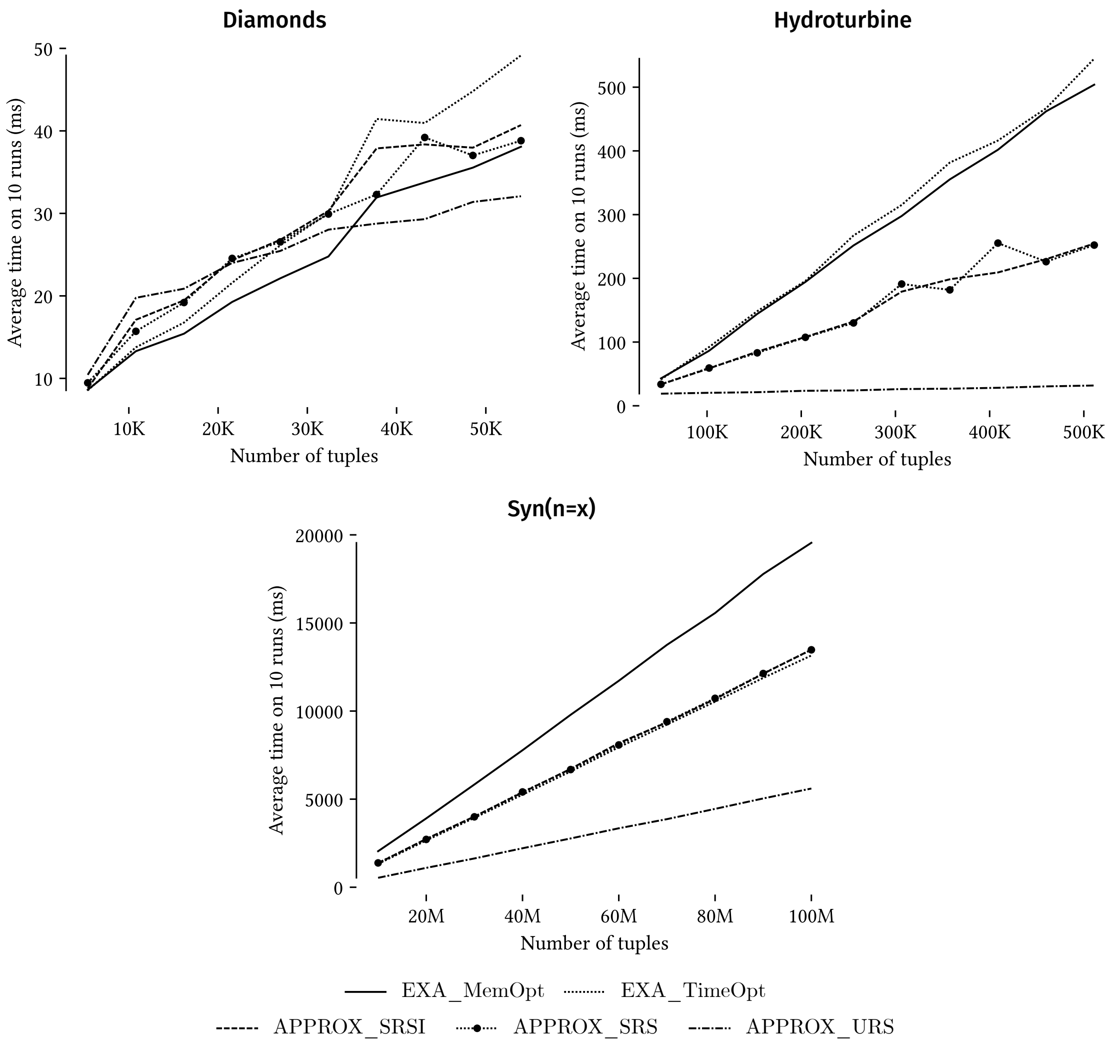


Figure 5.1 – *Influence of the number of tuples on the time performance with equality predicates.*

as accurate as APPROX_SRS and often proposes near-optimal approximations. In the results of the Syn dataset, we notably observe one of the drawbacks of the original APPROX_SRS: with only 300 equivalence classes, the reservoir size of 100 becomes insufficient which considerably deteriorates the approximation quality. In general, APPROX_URS is really fast but is only usable with large equivalence classes. Indeed, when the size of the equivalence classes grows (*i.e.* their number decreases), the proportion of tuples sampled in each one increases, leading to greater accuracy.

Finally, Figure 5.5 presents the effect of the various parameters of the Syn dataset on the approximation accuracy of random sampling algorithms. It confirms that APPROX_URS provides poor approximations for datasets with small equivalence classes and therefore requires a large number of samples. However, APPROX_SRS and APPROX_SRSI work really

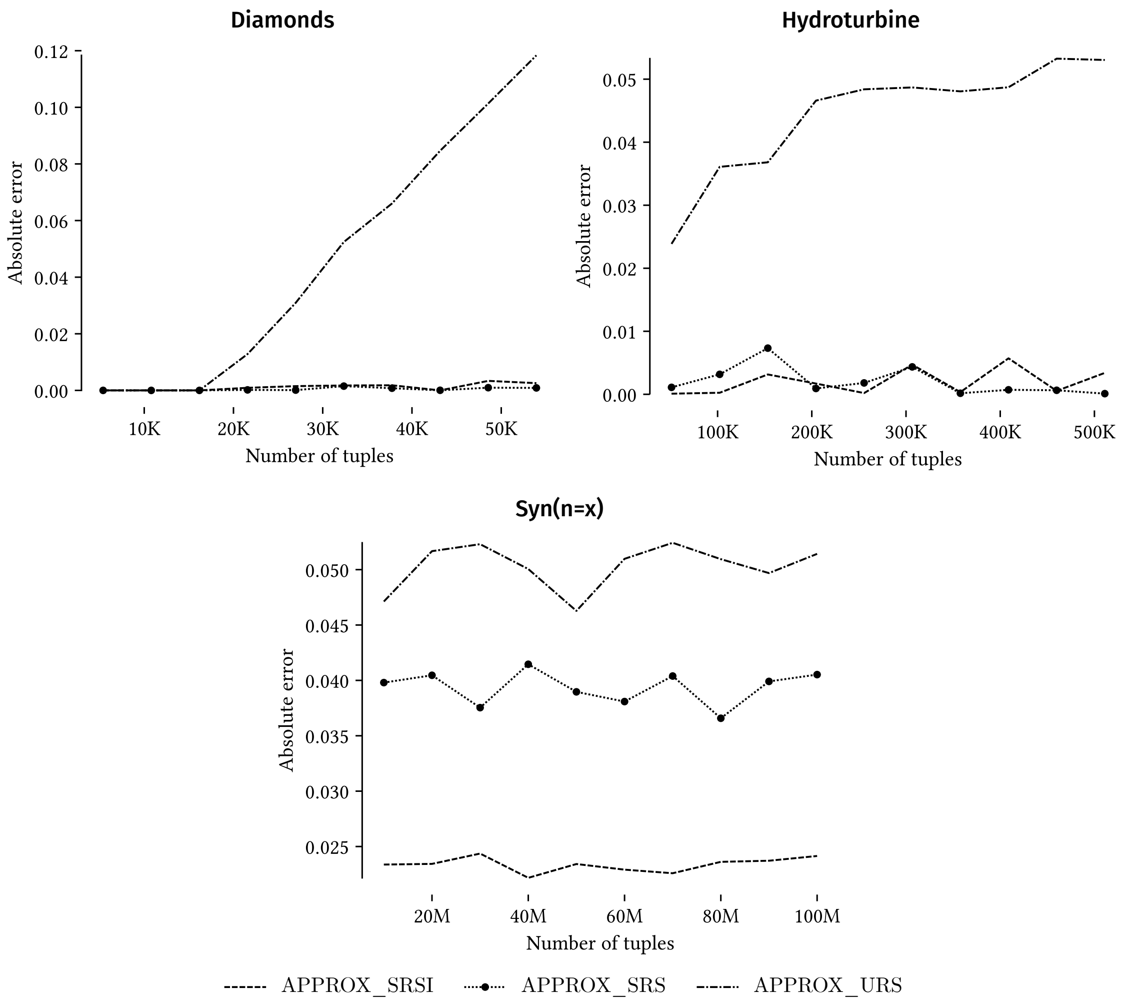


Figure 5.2 – *Influence of the number of tuples on the approximation performance with equality predicates.*

well with a small number of sampled tuples. When varying the parameter g_3, we also observe that algorithms tend to give a worse approximation for g_3 above 0.5. Indeed, at this point, the most frequent element in each equivalence class is not in *strict majority* which is known to be a hard case in *frequent elements problems* (for example, this is the limit of exact *heavy hitters* approaches). Finally, we can see that having very few distinct elements in each equivalence class also degrades the approximation as the most frequent element becomes harder to distinguish from the others.

To summarize, `EXA_MemOpt` and `EXA_TimeOpt` boh perform well with a slight advantage for `EXA_MemOpt` when there are few equivalence classes or unique consequents. For random sampling, `APPROX_SRSI` is preferable in most cases and its confidence and error in both passes could be further reduced to improve its execution time. Finally, g_3 values under 0.5 and with more unique consequents are better approximated.

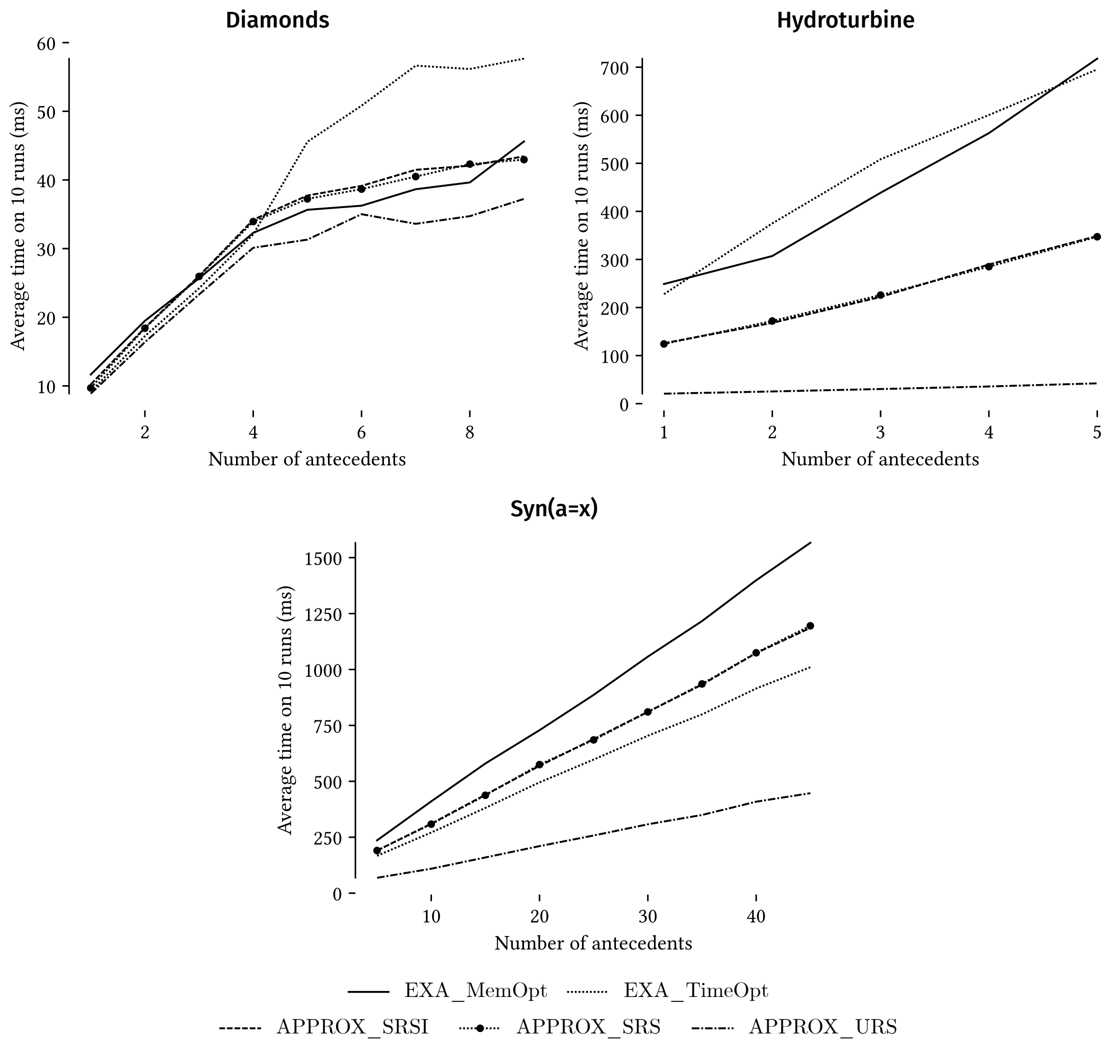


Figure 5.3 – *Influence of the number of antecedents on the time performance with equality predicates.*

5.4.4. Extension to uncertainties (NP-hard case)

As mentioned earlier, strict equality is often too restrictive to be meaningful for real-world comparisons. This is particularly true for continuous attributes such as the sensor ones found in the Diamonds and Hydroturbine datasets. In this case, it is preferable to incorporate the notion of uncertainty using predicates such as those presented in the formula 5.2. This is exactly what we propose in this section. We extend the equivalence to uncertainties that fall into the NP-hard case of our dichotomy.

Settings

The FDs used in this section for Diamonds and Hydroturbine are respectively:

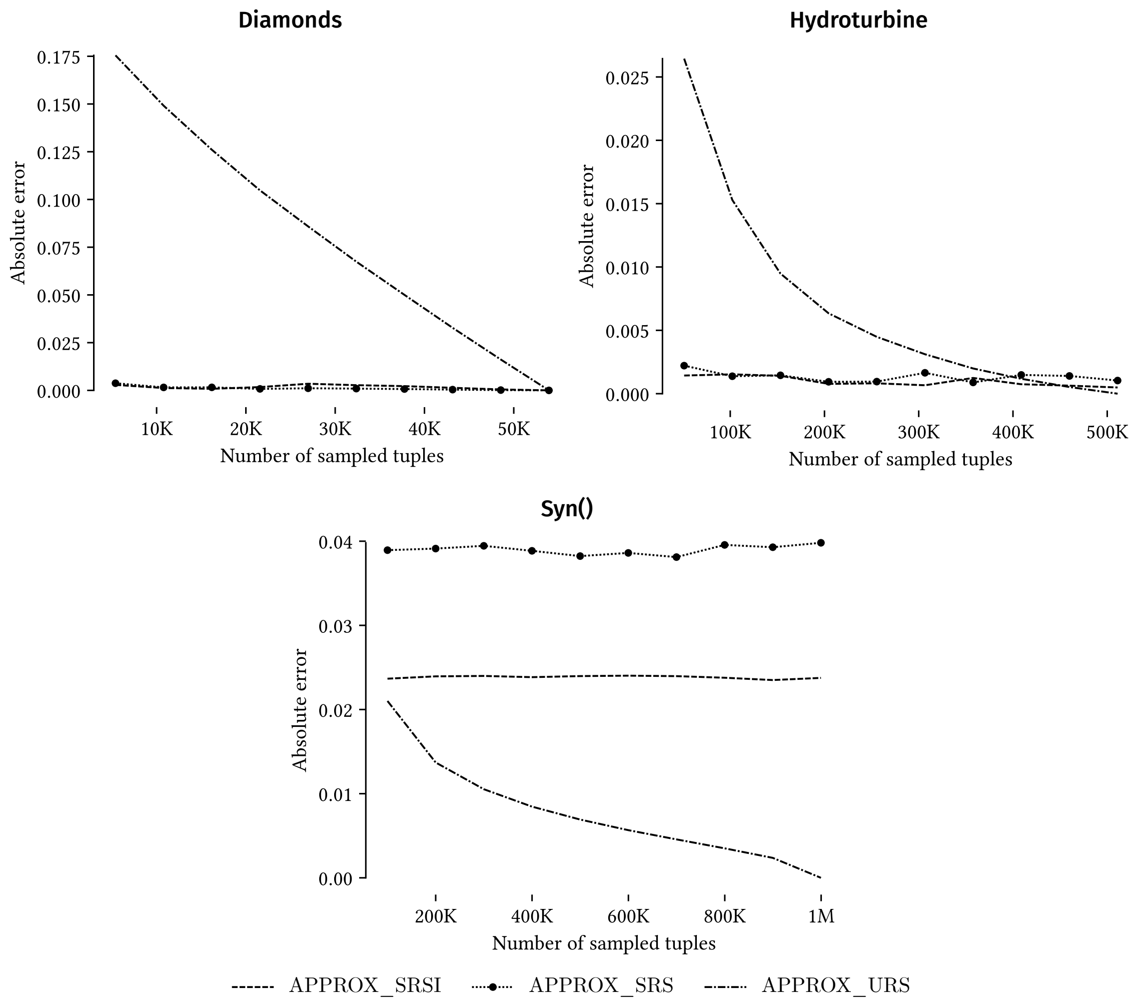


Figure 5.4 – *Influence of the number of samples on the approximation performance with equality predicates.*

- carat, x, y, z, depth, cut, color, clarity → price
  - Predicates:
    - $\phi_{carat}(x,y) = \phi_x(x,y) = \phi_y(x,y) = \phi_z(x,y) = \phi_{depth}(x,y) = \phi^u(x,y;0.05,0)$
    - $\phi_{price}(x,y) = \phi^u(x,y;0.1,0)$
    - Attributes cut, color and clarity use regular equality.
  - 21,182 counterexamples, $g_3 = 0.22$
- flow, opening, elevation → power
  - Predicates:
    - $\phi_{flow}(x,y) = \phi_{power}(x,y) = \phi^u(x,y;0.05,0)$
    - $\phi_{opening}(x,y) = \phi_{elevation}(x,y) = \phi^u(x,y;0.03,0)$

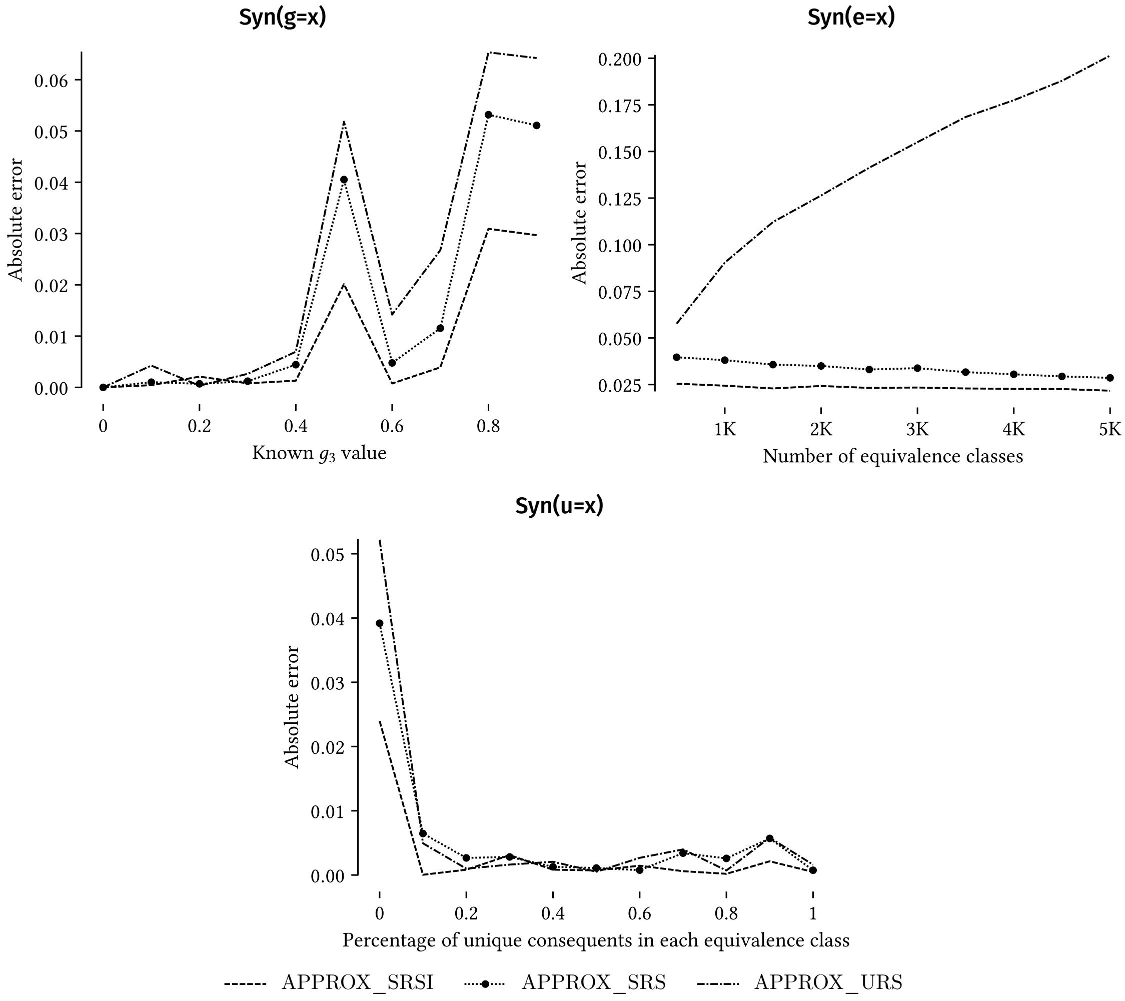


Figure 5.5 – *Influence of Syn parameters on the approximation performance with equality predicates.*

- ○ 2,972,255 counterexamples, $g_3 = 0.31$

NuMVC is used with a constant running time of 1 second (HEUR_NuMVC(1s)) and is therefore not shown on the time performance graphs. Unless otherwise indicated, 2000 tuples/nodes are sampled for the two sublinear algorithms APPROX_Sub09 and APPROX_Sub11.

Results

Counterexample enumeration. Figure 5.6 presents the execution time of the CEE with different levels of optimization for the Diamonds dataset. All these levels are achievable because the FD considered contains categorical antecedents with the equality predicate (cut, color and clarity) as well as at least one totally ordered antecedent with monotonic predicate (one of carat, x, y, z, depth or depth). If blocking is so effective, it is because the

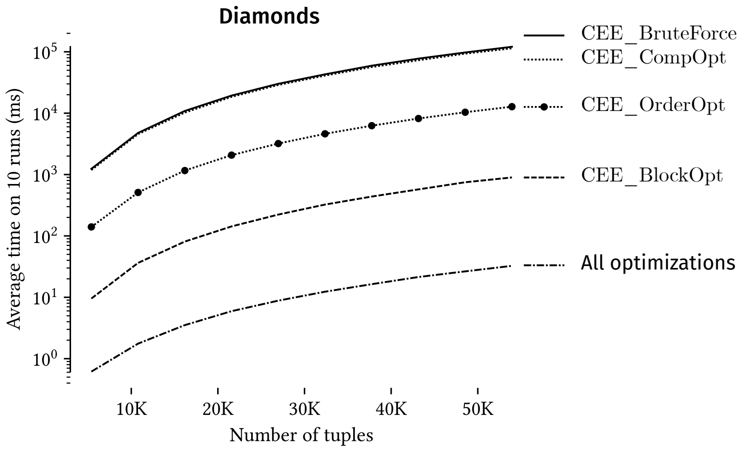


Figure 5.6 – *Counterexample enumeration on the `Diamonds` dataset with various levels of optimizations.*

projection of `Diamonds` on {cut, color, clarity} generates very small blocks which can be processed efficiently. With all optimizations combined, very reasonable computation times are achieved. On the other hand, only `CEE_CompOpt` and `CEE_OrderOpt` can be used for `Hydroturbine` and the high number of rows and especially the high number of counterexamples make the process still more time consuming with $\approx 30s$ total for processing the 200k rows (graph not shown here).

To summarize, the number of counterexamples is by far the most limiting factor. In addition to more counterexamples meaning more *required* comparisons, this is also likely to generate more potential *false positives* (pairs of tuples which are compared to finally conclude that they are not a counterexample) and therefore predicates computed in vain. Therefore, it is possible to perform an optimized CEE on very large datasets with very few counterexamples but it may also be very long with medium datasets which contain a lot of them.

Computing the g_3-error. Figures 5.7 and 5.8 present the time and approximation performances of the computation of the g_3-error. We observe that `APPROX_GIC` offers excellent approximation accuracy. `HEUR_NuMVC(1s)` provides perfect results in constant 1s time which is especially useful for very large graphs with many edges where `EXA_WGYC` takes too much time. In all our tests, there is no case where `APPROX_2Approx` is preferred and it is, in general, closer to its 2-approximation ratio guarantee than the exact value.

We can also observe that sublinear algorithms offer significant time performance benefits by replacing the full CEE by an on-the-fly CEE. We can see that both are almost equivalent with `Diamonds` when `APPROX_Sub11` performs better than `APPROX_Sub09`. In fact, the cubic guarantees of `APPROX_Sub11` are likely to perform better that the quadratic guarantees of `APPROX_Sub09`. It is also reassuring to see that their approximation is always very close to APPROX_2Approx of which they initially propose an estimate. In Figure 5.9, we can observe the approximation accuracy of the two sublinear algorithms dependending on the sample size. We can see that they do not need a large sample size to work well.

To summarize, EXA_WGYC performs well but becomes limited when the dataset contains numerous counterexamples. Nonetheless, APPROX_GIC proposes a fast and accurate estimate of the error almost as competitive as HEUR_NuMVC. If the CEE becomes too long, sublinear algorithms propose a good estimate of APPROX_2Approx in reasonable time, even with a small sample size. In general, APPROX_Sub11 is preferred over APPROX_Sub09.

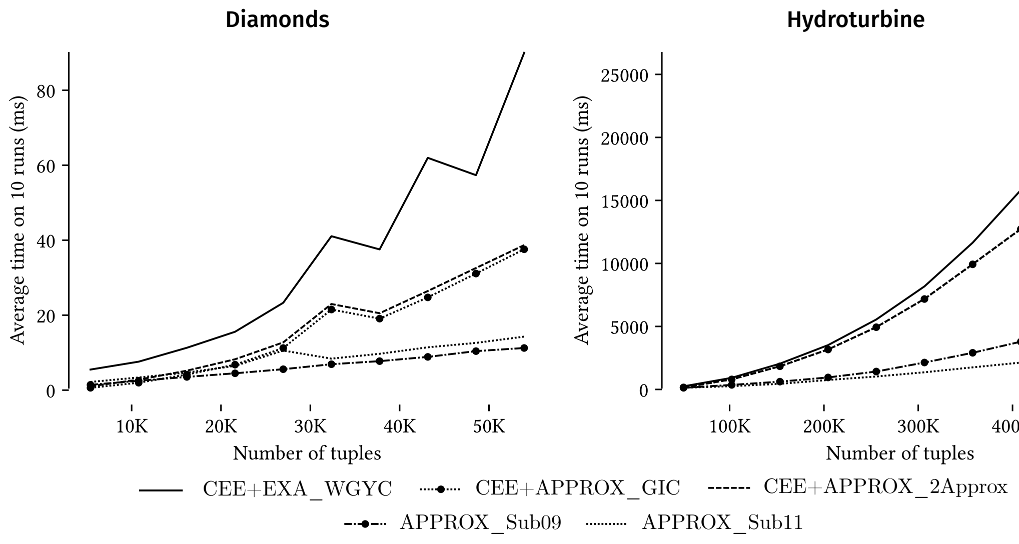


Figure 5.7 – *Influence of the number of tuples on the time performance.*

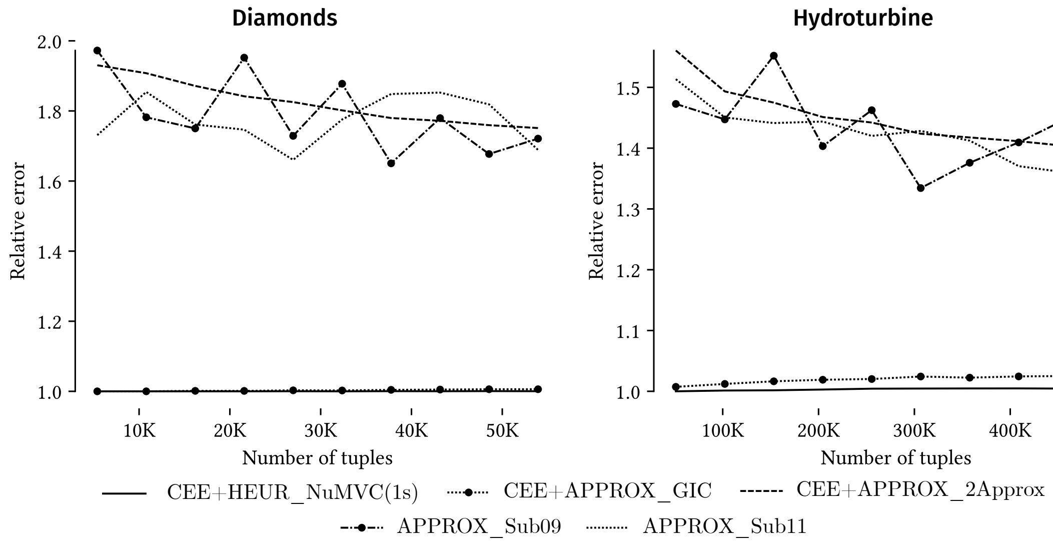


Figure 5.8 – *Influence of the number of tuples on the approximation accuracy.*

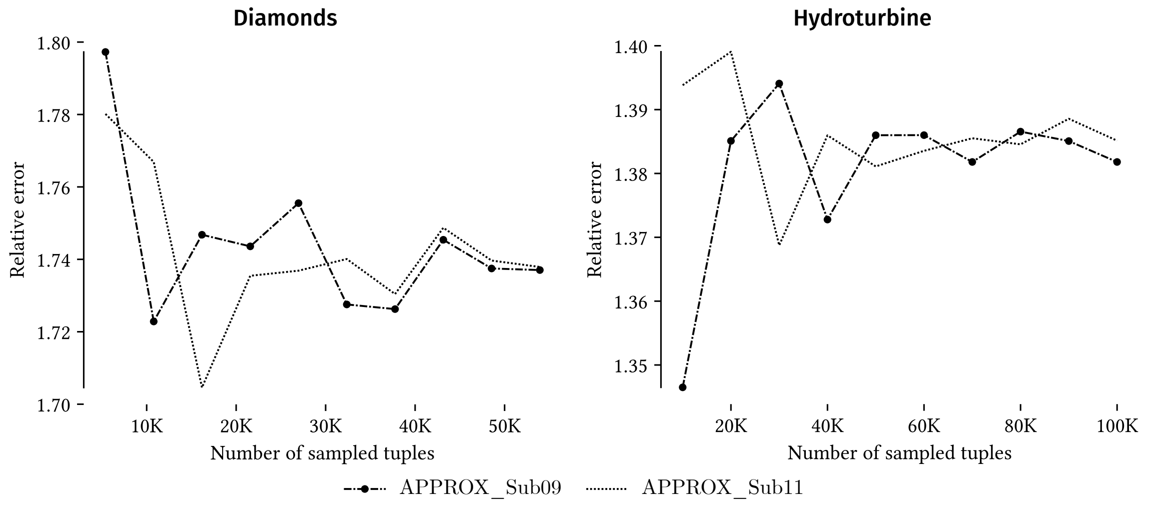


Figure 5.9 – *Influence of the sample size on the approximation accuracy of sublinear algorithms.*

5.4.5. Summary of experiments

Polynomial case. When predicates are constrained to equality in the case of crisp FDs, the computation of g_3 depends on the number of tuples and is highly scalable up to millions of tuples. However, approximation algorithms can be used to avoid iterating over each tuple and thus speed up the computation. `APPROX_URS` appeared to be inadequate but stratified approaches provide an efficient alternative (`APPROX_SRS`), especially when used with a dynamic reservoir size in the second size (`APPROX_SRSI`) to account for different equivalence class sizes. These approximation algorithms are particularly effective for smaller g_3 values.

NP-hard case. In the more general case such as uncertainty predicates, building the conflict graph via the CEE depends on the number of tuples while computing the g_3-error by finding an MVC rather depends on the number of counterexamples. If the CEE can be achieved in reasonable time (*e.g.* many optimizations can be applied), `EXA_WGYC` can be used to compute the error exactly for a small number of counterexamples and `APPROX_GIC` or `HEUR_NuMVC(t)` propose efficient approximation alternatives. When the CEE becomes too long, sublinear algorithms (especially `APPROX_Sub11`) offer a faster alternative at the expense approximation quality.

5.5. Conclusion

In this chapter, we studied scalable techniques to compute the g_3 indicator with (i) general predicates (known to be NP-hard) and (ii) with transitive and symmetric predicates (proven to be polynomial). For the first case (i), two subproblems were identified. First, for the enumeration of the counterexamples which is quadratic in the size of the dataset, blocking techniques, attribute ordering and ordered spaces were studied. Second, an analysis of available approximation algorithms and recent developments in sublinear algorithms were examined for solving the MVC. For the second case (ii), uniform and stratified sampling schemes were proposed and their theoretical guarantees analyzed.

We implemented all the algorithms presented in Section 5. We tested them through extensive experiments. For crisp FDs with classical equality, the propositions were shown to be fairly scalable while keeping a good approximation accuracy for sampling approaches. When incorporating uncertainties with non-crisp FDs, we observed that the bottleneck of the computation lies in the counterexample enumeration process. Sublinear algorithms offer important time savings but they all adapt an 2-approximation algorithm which is known to provide average approximations in practice.

Chapter 6

ADESIT: A web application for interactive counterexample analysis

6.1. Introduction

The previous chapters where devoted to the study of the computation of the g_3-error. We now explore further how such process can be used in practice, mainly to better understand the interplay between a dataset and a function (possibly a machine learning model) represented by a FD. Indeed, if the g_3-error itself provides a degree of satisfaction of a function in a dataset, it is not always sufficient to understand *why* and *where* it does or does not behave well. We now present our proposition for using counterexamples in conjunction with indicators in the frame of supervised learning, where the goal is to *learn a function*.

Consider a supervised learning (SL) problem where the task is to predict a continuous or categorical target A from a set of features X using a set of examples. The goal is to find a function f (a.k.a. model) such that $f(X) \simeq A$. This function must balance between error minimization on the examples and its ability to generalize well on unseen instances to avoid overfitting. To do so, a set of training examples is sent to a learning algorithm to infer such a function which can be evaluated afterwards against a testing dataset. Suppose the resulting accuracy is below expectations: what should be questioned? One might be tempted to try different training parameters or even change the learning algorithm. If this approach makes sense in some cases, challenging the very existence of f should also be one of the primary concerns to prevent an unsuccessful SL process or, conversely, to help the practitioners trust the computed model. In practice, the quality and completeness of the learning examples is well-known to be one of the primary factors of success in SL [JPN+20]. Noisy, inaccurate, or incomplete data can lead to poor models, and it is sometimes even difficult to understand that the learning dataset itself is to blame. If the examples given for training contradict each other or do not contain enough information, even the most advanced algorithm will fail to understand the functioning of the phenomenon it is trying to predict.

In response to these concerns, multiple methods and metrics have been developed in the past allowing to evaluate, improve, and better understand the data and its predictive power. In particular, we focus on the recent studies proposing the use of functional dependencies (FDs) and specifically the analysis of counterexamples to find contradictions in the dataset [CIP13, LPS20]. For the specific case of SL, we understand intuitively that learning examples with equal causes (X) and different outcomes (A) are likely to cause problems during the learning process. As a function, f needs to give a unique answer for a given input. This type of contradiction can be due to noise in the data of A (sensor precision, input error...) but it might also reveal missing or redundant features of X or just the unpredictable nature of the phenomenon. Therefore, finding “regions” with high densities of counterexamples is of great interest for data preprocessing, to gain insight into the intrinsic limitations of the raw dataset and the problem statement itself. Our aim is to guide interactions with domain experts so to build with them the “intimate conviction" that a function exists in the data and that, if applicable, an ML model can be built. If the limits given by counterexample analysis are not suitable for the problem at hand, the process must be refined by working on data acquisition or processing. Thus, these questions are intimately linked to data and feature selection and it highlights the interweaving of these concepts between the database and ML communities.

Logo of ADESIT.

As our main proposition, we present ADESIT (Advanced Data Exploration and Selection Interactive Tool), a web-based intuitive graphical user interface to evaluate the limits of a dataset for a given SL problem. This evaluation is made through statistical measures based on counterexamples and an interactive visual exploration permits the user to see the variations of this measure on different regions of the data: given a dataset, a set of features X and a prediction target A, ADESIT helps the user to understand the predictive power of the data but also potential refinements and improvements in her features or data selection. We propose various statistics on the dataset up to tuple granularity along with an interactive representation of the data and found counterexamples. As illustrated in Figure 6.1, we thought ADESIT as being part of an iterative refinement process. Given a new SL problem, we want the user to quickly be able to evaluate what performances cannot be exceeded and what steps should be taken before approaching the learning process itself.

Due to a technical gap between data scientists and experts, domain knowledge is often underused in the conception of prediction models. However, it is often the domain experts themselves who know the intrinsic theoretical parameters of the prediction problem but want a more practical model closer to reality. To do so, we propose to use counterexamples as a “means of mediation” between data experts and domain experts. As a consequence,

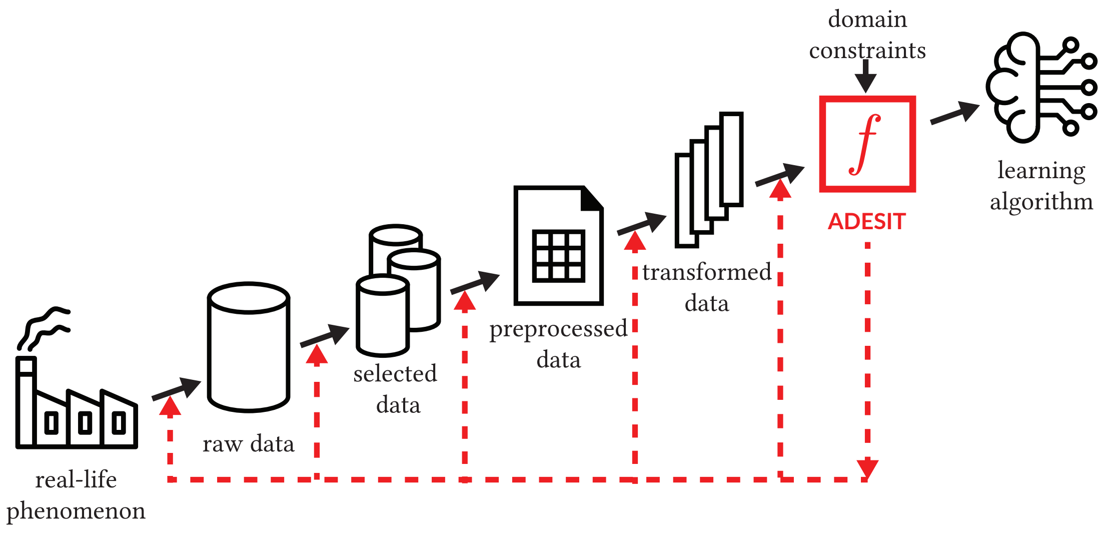


Figure 6.1 – *Place of* ADESIT *in the data processing pipeline for machine learning. Just before the model creation step,* ADESIT *allows the user to understand the limits of her dataset, thus taking part of an iterative data refinement process from data acquisition to preprocessing.*

one of our primary concerns while designing ADESIT was to involve domain experts in the process: the intuitive visualization of counterexamples proposed by ADESIT allows a quick understanding and explanation of the potential issues which may arise, promoting dialogue within all the actors of the data chain.

In summary, for a given dataset and a specific SL problem, ADESIT proposes the following contributions:

– We offer statistics derived from counterexamples analysis to characterize the ability of the dataset to perform well for the given problem. In particular, we propose the g_1, g_2 and g_3 indicators [KM95] with predicates for domain knowledge integration and use the efficient implementations of the FASTG3 library presented in Section 5. Notably, the g_3-error serves as an upper bound on the maximum accuracy obtainable with any model on a given dataset [LPS20].

– We propose several ways to visualize counterexamples in the dataset to better understand their distribution: table, scatter plot, heat map, attribute histograms. Notably, it allows to find and identify so-called "data regions" with good learning potential, but also subsets that could need refinement. It is also possible to select and analyze tuples individually by visualizing the conflict graph locally.

– Our objective is also to propose new visualizations to facilitate the dialogue between domain experts and data scientists. Thus, we worked on proposing an intuitive interface based on visualization with detailed explanations for each component.

The main results of this chapter have been published in [FGPSLG21].

It is important to emphasize that we are upstream from the model creation itself. The g_3 indicator acts as an upper-bound on the target accuracy but will not necessarily be the desired one as it might result in a lack of generalization. However, by making informed choices for the similarity measures (thresholds) and therefore assuming imperfections in the accuracy of the attributes, we produce a better reflection of the real-life phenomenon behind the data and therefore reduce the overall variance. ADESIT is used to identify contexts where creating a good model will be difficult or promising and provides the user with knowledge to explain why to potentially refine your data from acquisition (*e.g. Should we get new sensors or more precise ones?*) to preprocessing. Also note that ADESIT proposes intuitive indicators *in view of a given dataset* and is not meant to analyze the unpredictability of the underlying phenomenon itself which is notably linked to the Bayes error [Fuk13].

Chapter organization. First, we present the technical details behind ADESIT and its connection to FASTG3. Second, we introduce the three main areas of ADESIT along with a running example to motivate its real-life utility. Finally, we conclude and detail some of the current limitations.

6.2. Technical presentation

The computation in ADESIT is based on the FASTG3 library presented in Chapter 5. Thus, it incorporates both the classical equality and predicates with relative and absolute uncertainties as presented in formula 5.2.

In addition to the g_3-error, ADESIT also proposes two other indicators introduced by Kivinen and Mannila [KM95]: the g_1 and g_2 indicators. As for the g_3, those indicators are initially proposed for crisp FDs with classical equality. Nonetheless, they remain well-defined with the use of predicates. Let (R,Φ) be a relation scheme with predicates, r a relation over R and φ a FD. We consider the conflict graph $\mathsf{CG}(\varphi, r) = (r, E)$. The g_1 and g_2 indicators can be formulated as follows:

– g_1 is the proportion of counterexamples among all possible counterexamples, or:

$$g_1^{\Phi}(\varphi, r) = \frac{|\{(t,t') \mid t,t' \in r, (t,t') \not\models_{\Phi} \varphi\}|}{|r|^2} = \frac{|E|}{|r|^2}$$

– g_2 is the proportion of tuples involved in at least one counterexample, or:

$$g_2^{\Phi}(\varphi, r) = \frac{|\{t \mid t \in r, \exists t' \in r \text{ s.t. } (t,t') \not\models_{\Phi} \varphi\}|}{|r|} = \frac{|\{t \mid t \in r, \exists e \in E \text{ s.t. } t \in e\}|}{|r|}$$

For the computation of g_3, `EXA_MemOpt` (Algorithm 5, page 38) is used if only equality is defined for the attributes. If the user chooses to define uncertainties, it is then possible to choose between exact and approximate computation with the `EXA_WGYC` and `APPROX_GIC` algorithms respectively (see Section 5.2).

ADESIT is an open-source web-application under BSD 3-Clause License. Its source code is available publicly on GitHub (github.com/datavalor/ADESIT) and it is composed of about 4000 lines of code. It is implemented in Python using the Dash framework. We also use Pandas and Numpy. A beta version of ADESIT hosted on the LIRIS laboratory servers is available at adesit.liris.cnrs.fr.

6.3. ADESIT overview

In this section, we present the interface and various features of ADESIT. As a running example, we use the 20,000 randomly sampled rows of the `Hydroturbine` dataset presented in Chapter 5, Section 5.4.2 and explain how ADESIT can be used to assist the SL process. As shown in the screenshot available in Figure 6.2, the interface of ADESIT is divided into three main areas described in the following sections.

6.3.1. Supervised learning problem settings (area A)

This area is used to define the SL problem by setting the features and the target as well as their associated uncertainties. After uploading a dataset in *csv* or *excel* format, the user can then select attributes to define the features X and the target A of the prediction problem $f(X) \simeq A$. It is then possible to define the absolute and relative uncertainties for each attribute. If both are left at zero, equality is used for that attribute. If no uncertainty is defined on all implied attributes, polynomial algorithms will be applied to compute g_3. Finally, the user can choose the type of computation for g_3 (approximate or exact) and start analysis (counterexample enumeration and indicators computation). At the present time, unknown values are not explicitly handled by ADESIT and the corresponding tuples are simply removed as soon as the dataset is uploaded. Nonetheless, approaches such as [WL19] could be implemented in the future.

Example 10. *In the screenshot in Figure 6.2, the FD* "opening, position, elevation $\to$ power" *is defined. It corresponds to the problem of predicting the power from the 3 other attributes. The opening and position attributes have an absolute uncertainty of* 0.05*, the elevation an absolute uncertainty of* 0.01 *and the power a relative uncertainty of* 0.01*. These uncertainties have been chosen in interaction with domain experts from CNR.*

6.3.2. Counterexample indicators (area B)

In this second area, ADESIT displays the number of tuples involved in a counterexample (unnormalized g_2) along with the g_1, g_2 and g_3 indicators. Those indicators are inverted versions of the raw measures (*e.g.* g_1 becomes $1-g_1$) for readability: *the higher the better*. If the performances are not appropriate for the domain experts, we can intervene on the data collection process by exploring solutions such as increasing data accuracy to influence similarity measures (*e.g.* using better sensors), adding new features or refine the data processing pipeline (acquisition, denoising, gap filling...).

Example 11 (continued)**.** *In the screenshot in Figure 6.2, we have a very low value of g_1 ($1-g_1$ greater than 99.99%) and an average value of g_2 ($1-g_2$ is 66.54%). It means that a good amount of tuples are involved in a counterexample but that they are quite spread out through the dataset. The 84.7% corresponds to $1-g_3$. This value indicates that the function is quite verified in the data and thus seems likely to be learnable.*

6.3.3. Counterexamples exploration (area C)

This area allows to interactively explore the dataset in view of counterexamples. It can be divided in two main areas: C.1 gives an overview of the counterexample distribution at the dataset level while C.2 allows to obtain informations at the tuple granularity. These two areas follow the same color scheme presented in Table 6.1. All these colors along their meaning are reminded in legend visible in the upper part of in area C.1 of Figure 6.2. We now describe areas C.1 and C.2 independently.

Table 6.1 – TUPLE/NODE COLOR LEGEND OF ADESIT.

Color	Involved it at least on counterexample	Selected
blue 🟦	·	·
green 🟩	·	✓
red 🟥	✓	·
yellow 🟨	✓	✓

Global dataset visualization (area C.1) This area is used to get a global vision of the dataset in the perspective of counterexample analysis. These visualizations are meant to detect areas with high densities of counterexamples and to help to decide if the process of data selection for SL can continue or if new or better data is needed. It is divided in three tabs that we describe in the following:

- **Tabular view.** The tabular view shown in Figure 6.3 is one of the most ubiquitous way to browse tabular data. Columns are sorted by their role in the studied function (feature, target or not included) as well as their data type (numerical or categorical) and can be hidden for visualization convenience. Columns which are not included in the function are left with white background. If a tuple t is selected, tuples which form a counterexample with t are framed in bold. It can be also be used to select a tuple and export the current selection (square tool in 2D View).

- **2D View.** The 2D view presents the data along two attributes. By keeping the target on the ordinate, it is possible switch between the various features two understand their individual impact on the counterexample distribution w.r.t the target. The results of dimensionality reduction techniques such as PCA (principal component analysis) applied to the features X can also be used as an alternative axis. This is especially useful to compensate for the restrictions of 2D views and to get an overview of how well counterexamples are spread in the dataset. Associated with each axis, stacked histogram is also proposed to show the distributions of the data and counterexamples. The number of bins can be defined with a slider. In addition to zooming and panning, the user can also use the lasso or square tools to select a part of the dataset and display its statistics or export it as a table.
  In this setting, the first view is the usual **scatter plot** (shown in area C.1 of Figure 6.2). Each point on the graph represents a tuple, and counterexamples are highlighted according to the chosen parameters. By hovering over a point, it is possible to display its specific information. It is possible to select a tuple by clicking on it. If a tuple t is selected, tuples which form a counterexample with t are made bigger a framed in bold.
  It is also possible to display the heat map of the dataset (shown in Figure 6.4). This view is particularly useful for finding specific areas where the function will be difficult to learn and where the resulting model may be therefore less reliable.
- **Attributes histograms.** In this view, each attribute has its own stacked histogram to understand the distribution of data and counterexamples (shown in Figure 6.5). The currently selected tuple and its associated counterexamples appear as vertical lines in each attribute. Such view is particularly useful for understanding which attribute might be missing in the current FD in order to remove some counterexamples. Indeed, while a tuple is selected, it is possible to visually distinguish new potential feature attributes which may play a discriminative role regarding the target. These are characterized by a spreading of the vertical lines, meaning that the counterexamples are dissimilar to the selected tuple on the considered attribute. This view can also be used to filter the data using a range slider associated with each attribute.

In each tab, the user can choose to filter all tuples, counterexamples, or free tuples only thanks to the filter visible in Figure 6.2.

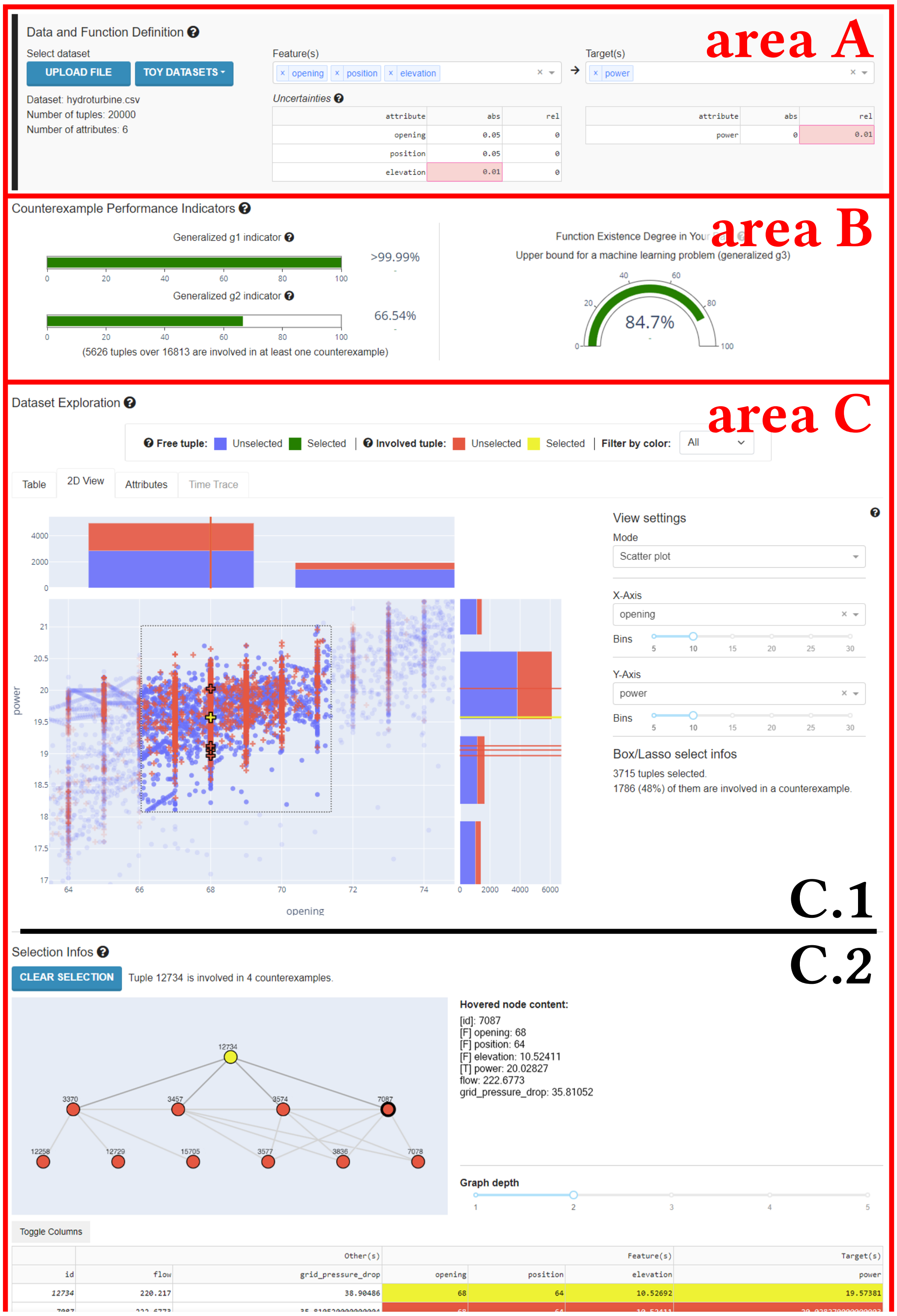
Tuple-wise analysis (area C.2) This last area displays information about the currently selected tuple. It shows the local conflict graph around the tuple with a customizable depth. Nodes can be hovered over to get more information about the corresponding tuples and clicked to be selected. A table summarizing the selected tuple and its direct neighborhood is also available.

Example 12 (continued)**.** *In the screenshots in Figures 6.2-6.5, we get different perspectives* *on the dataset and the selected tuple number 12,734. As visible in the histogram section, tuples with low flow have been filtered as they correspond to different production regimes. The vertical lines in the 2D View correspond to the discrete opening position (they correspond to an integer percentage). Points between line correspond to transitional opening states. We now make some observations.*

We can see on Figure 6.4 that a great proportion a counterexamples lies in the 65-70% opening range. This means that a SL model would propose less reliable results in that range unless a discriminative feature is added. Such feature can be found in the conflict graph in which we observe that the selected tuple and its neighborhood form a clique. We can see by highlighting other tuples or looking at the table below that the counterexamples have significantly different values on the grid_pressure_drop *attribute than the selected one. This is also visible on the histogram of* grid_pressure_drop *in Figure 6.5 where the red vertical lines are quite spread out around the selected yellow lines.*

6.4. Conclusion

In this chapter, we introduced ADESIT, a tool for analyzing a dataset in view of counterexamples in the frame of SL. The various views (table, scatter plot, heat map, histograms, conflict graph) allow to get a comprehensive picture of how counterexamples are distributed in the dataset and how they might impact the learning process. Furthermore, the computation is fast thanks to the optimizations presented in Chapter 5.



Data and Function Definition

Select dataset

UPLOAD FILE | TOY DATASETS

Dataset: hydroturbine.csv
Number of tuples: 20000
Number of attributes: 6

Feature(s): opening, position, elevation → Target(s): power

Uncertainties

attribute	abs	rel
opening	0.05	0
position	0.05	0
elevation	0.01	0

attribute	abs	rel
power	0	0.01

Counterexample Performance Indicators

Generalized g1 indicator >99.99%

Generalized g2 indicator 66.54%

(5626 tuples over 16813 are involved in at least one counterexample)

Function Existence Degree in Your
Upper bound for a machine learning problem (generalized g3)

84.7%

Dataset Exploration

Free tuple: 🟦 Unselected 🟩 Selected | Involved tuple: 🟥 Unselected 🟨 Selected | Filter by color: All

Table | 2D View | Attributes | Time Trace

View settings

Mode: Scatter plot

X-Axis: opening
Bins: 10

Y-Axis: power
Bins: 10

Box/Lasso select infos

3715 tuples selected.
1786 (48%) of them are involved in a counterexample.

Selection Infos

CLEAR SELECTION Tuple 12734 is involved in 4 counterexamples.

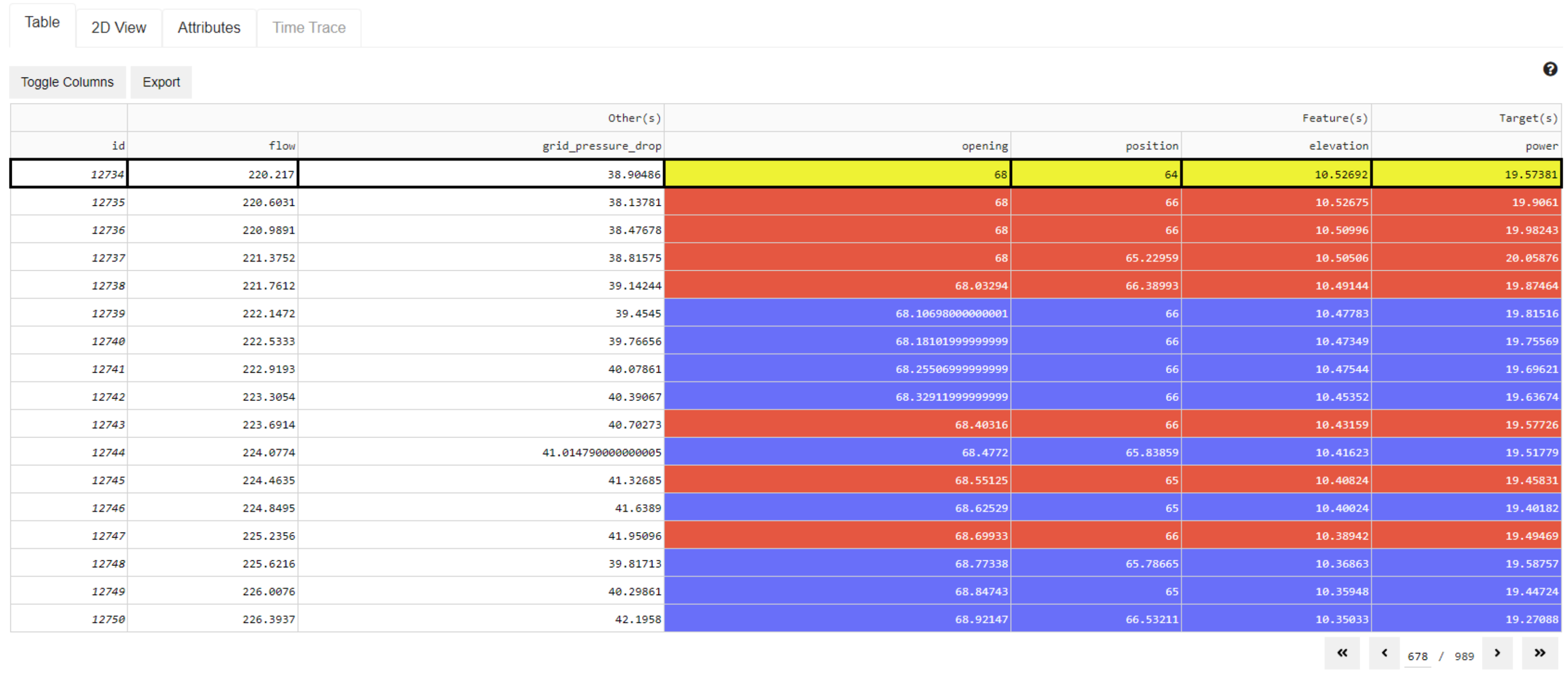
Hovered node content:
[id]: 7087
[F] opening: 68
[F] position: 64
[F] elevation: 10.52411
[T] power: 20.02827
flow: 222.6773
grid_pressure_drop: 35.81052

Graph depth: 2

Toggle Columns

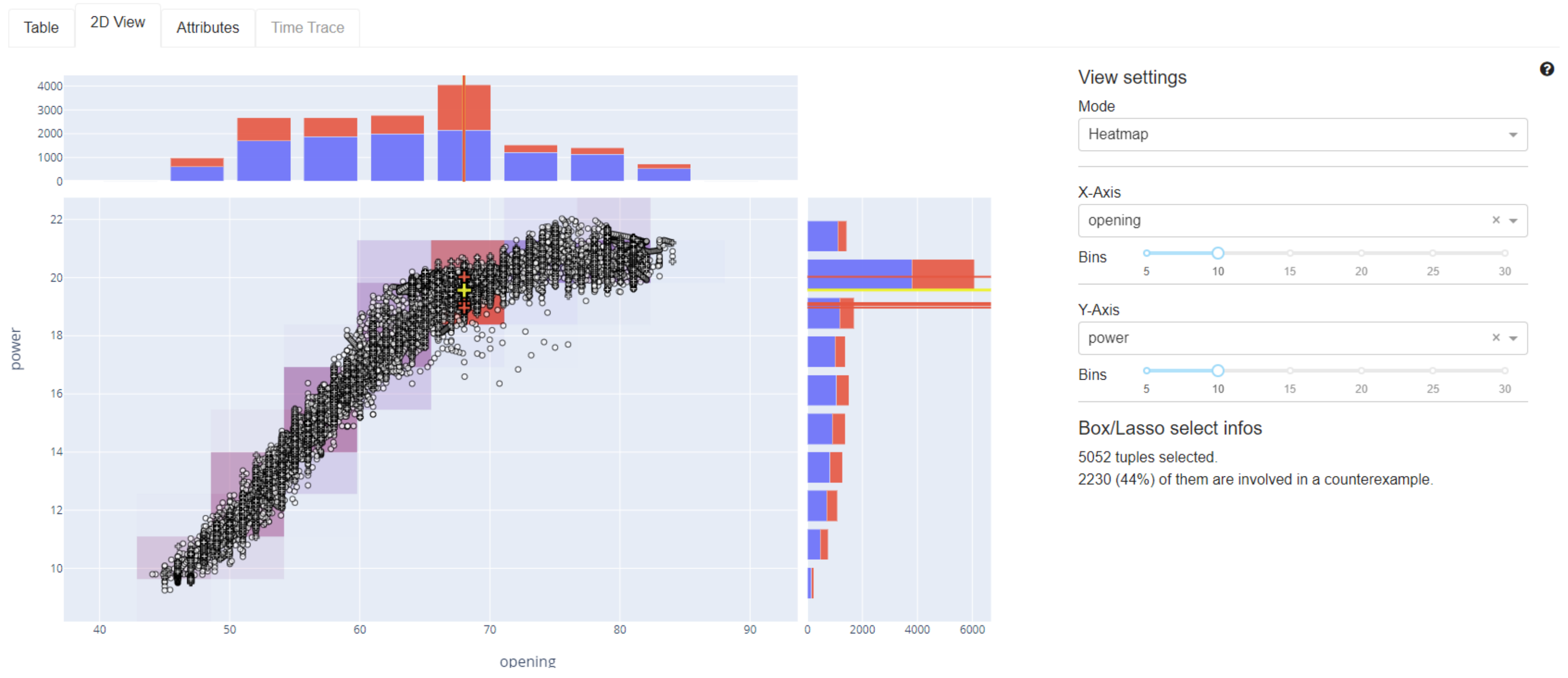
	Other(s)		Feature(s)			Target(s)
id	flow	grid_pressure_drop	opening	position	elevation	power
12734	220.217	38.90486	68	64	10.52692	19.57381
7087	222.6773	35.810520000000004	68	64	10.52411	20.028270000000003

Figure 6.2 – *Labeled screenshot of the ADESIT interface. A: Supervised learning problem settings as a functional dependency with predicates for each attribute. B: Counterexample indicators. C: Counterexamples exploration (C.1: dataset-wise ; C.2: tuple-wise).*



	Other(s)		Feature(s)			Target(s)
id	flow	grid_pressure_drop	opening	position	elevation	power
12734	220.217	38.90486	68	64	10.52692	19.57381
12735	220.6031	38.13781	68	66	10.52675	19.9061
12736	220.9891	38.47678	68	66	10.50996	19.98243
12737	221.3752	38.81575	68	65.22959	10.50506	20.05876
12738	221.7612	39.14244	68.03294	66.38993	10.49144	19.87464
12739	222.1472	39.4545	68.10698000000001	66	10.47783	19.81516
12740	222.5333	39.76656	68.18101999999999	66	10.47349	19.75569
12741	222.9193	40.07861	68.25506999999999	66	10.47544	19.69621
12742	223.3054	40.39067	68.32911999999999	66	10.45352	19.63674
12743	223.6914	40.70273	68.40316	66	10.43159	19.57726
12744	224.0774	41.014790000000005	68.4772	65.83859	10.41623	19.51779
12745	224.4635	41.32685	68.55125	65	10.40824	19.45831
12746	224.8495	41.6389	68.62529	65	10.40024	19.40182
12747	225.2356	41.95096	68.69933	66	10.38942	19.49469
12748	225.6216	39.81713	68.77338	65.78665	10.36863	19.58757
12749	226.0076	40.29861	68.84743	65	10.35948	19.44724
12750	226.3937	42.1958	68.92147	66.53211	10.35033	19.27088

Figure 6.3 – *View as tabular data.*



View settings

Mode

Heatmap

X-Axis

opening

Bins 5 10 15 20 25 30

Y-Axis

power

Bins 5 10 15 20 25 30

Box/Lasso select infos

5052 tuples selected.

2230 (44%) of them are involved in a counterexample.

Figure 6.4 – *Heat map view of the dataset.*

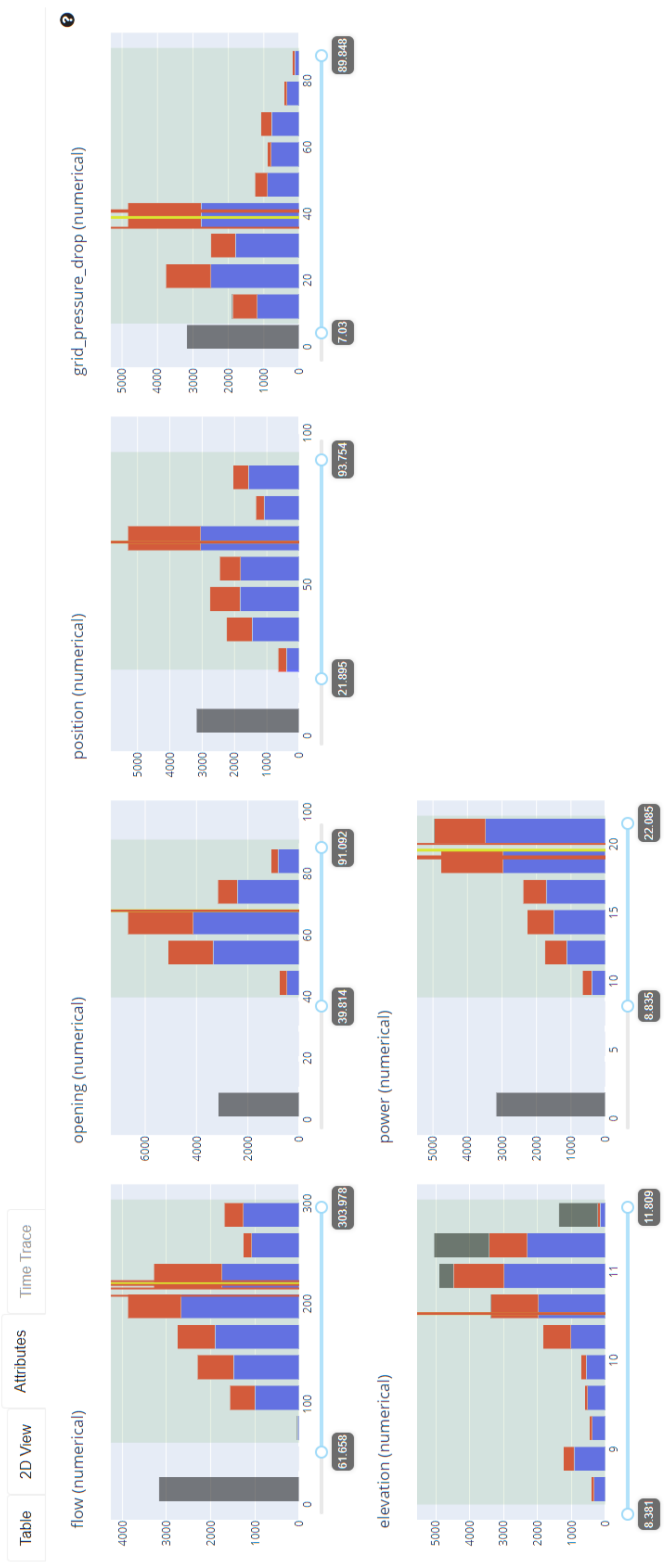


Figure 6.5 – *Histograms for each attribute.*

Chapter 7

Application to air gap monitoring in compact hydro-generators

7.1. Introduction

As mentioned in the introduction (Chapter 1), most of the production from the CNR comes from the exploitation of 19 hydroelectric power plants along the French side of the Rhône river. Most of these are run-of-the-river facilities with almost no reservoir. This means that the water turbines must disturb the natural flow of the river as little as possible, so as not to upset the balance of the ecosystem. This is why, when the first power plants were designed in the 1960s and given the novelty of the problem, the CNR commissioned the design of new, more compact turbines.

To ensure the compactness of the generating units, it was decided to significantly reduce the *air gap* value (distance between the rotor and the stator of the generator, see Figure 7.2). This air gap was originally designed to be about 7 mm under normal operating conditions for turbines of several meters in diameter. About a decade after the first generators were installed, a collision occurred in one of the generators, resulting in significant damage and repair costs. It was concluded from the incident that *it was the stator that underwent a progressive deformation* up to the collision and not the rotor, as it is usually the case in such turbines. These deformations are caused by the electromagnetic forces within the turbine, amplified by the smallness of the air gap.

Although it is possible to slow down the deformation of the stator, it cannot be completely avoided. However, it is possible to compensate for the reduction in air gap by tightening the stator mountings or removing of pole plates. The aim is therefore to anticipate such collisions in order to correct the machine before they occur. However, these operations involve significant maintenance costs and should ideally be carried out as late as possible. For this reason, the CNR started to develop predictive maintenance techniques by monitoring the air gap value. Due to the novelty of the problem, they experimented with several solutions and finally settled on a set of rotating remote sensors mounted on the rotor. In addition, a pair of proximity sensors is used to track the displacement of the rotor shaft to monitor its eccentricity. Currently, the sensor data is processed manually in a long and tedious process. In addition, the uncertainty of the result cannot be quantified and the assessment of its quality is left to the discretion of the analyst.

In this work, we benefit from a close collaboration between the CNR experts and data science researchers to design a tailored solution for this industrial data-centric problem. Our study is twofold:

– First, we validate the approach developed in this thesis to verify assumptions about this problem. In particular, we use ADESIT to check that the measured air gap is indeed a function of the rotor angle and that adding the rotor eccentricity improves the accuracy. We also verify that the temperature plays a significant role in the value of the air gap.

– Second, we propose an automatic solution for analyzing air gap data in the special case of stator deformation. The aim is to improve the quality and usability of the data, while remaining aware of its limitations. It also facilitates data storage by reducing its memory size. Our solution starts with angular resampling followed by synchronous averaging, two well-known signal processing techniques. In particular, *synchronous averaging* allows us to extract the meaningful part of the signal by reducing the noise. Moreover, we can also evaluate the strength of the resulting signal by computing higher synchronous moments. Then, we propose to geometrically correct the rotor eccentricity and to compute the critical air gap which is the actual closest distance between the rotor and the stator when the generator is operating. Finally, we propose appropriate visualizations designed to be understood by domain experts, even when they are not familiar with the processing techniques used.

The main results of this chapter have been published in [FGTS23].

Chapter organization. First, we describe the problem and state our assumptions. Second, we present the sensor sensor instrumentation and the resulting time series as well as the recorded data used in this chapter. Third, we show how to use counterexample analysis with ADESIT as a preliminary step to prepare the data science project. Then, we describe our solution, experiment with some sample of data from the CNR and discuss some parameters of the acquisition strategy. Finally, we conclude this study and present some future work.

7.2. The air gap monitoring problem

7.2.1. Problem description

A classical generator consists of an excited rotor rotating inside a stator. CNR's turbines have generators several meters in diameter as shown in the illustration in Figure 7.1. As their rotation speed is synchronized with the European grid ($\approx 50\,\text{Hz}$), a typical CNR generator consisting of 32 pairs of poles rotates at approximately $1.56\,\text{Hz}$.

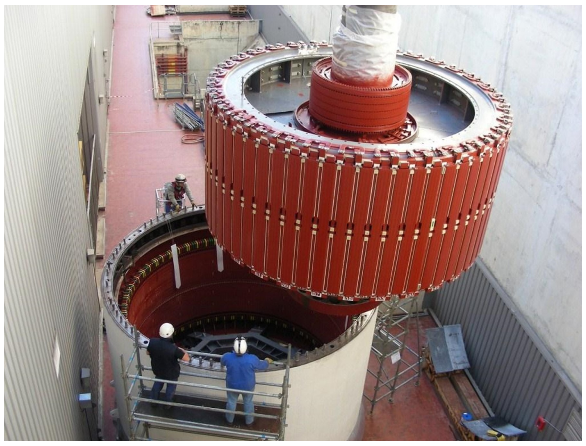


Figure 7.1 – *Photo of a turbine undergoing maintenance. The rotor is extracted from the stator in this operation. Source: CNR.*

As shown in the schematic presented in Figure 7.2, the air gap is the area of air separating the rotor from the stator. Due to the compact design of CNR run-of-river turbines, this air gap of about 7 mm is quite thin compared to the diameter of the generator. As explained in the introduction, a progressive decrease of the air gap distance has been observed on some turbines, leading to rotor-stator collisions.

The cause of such collision is the local reduction of the air gap distance at some point of the stator. This reduction is due to multiple factors. The main causes considered in this study are:

- **Stator deformation.** The electromagnetic interactions with the rotor gradually deform the various parts of the stator.
- **Rotor eccentricity.** Due to imperfect rotor mass distribution, stator deformation and electromagnetic interactions, the rotor center may not be aligned with the geometrical stator center (axis of rotation). This phenomenon is called *rotor eccentricity*. In particular, the rotor center is not only off-center but it also periodically moves with time: this is *dynamic* eccentricity (e.g. see [FEAT09]).
- **Turbine state parameters.** The various instantaneous characteristics of the generator (temperature, tension...) influence the value of the air gap. For example, the rise in

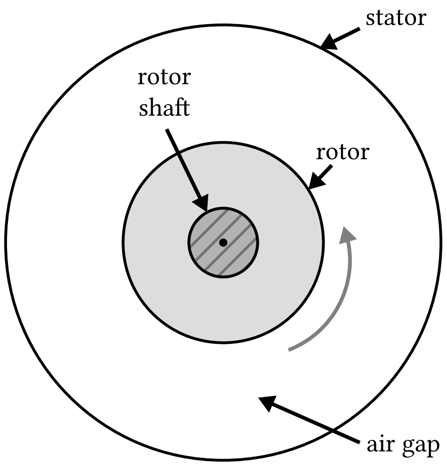


Figure 7.2 – *Schematic of an ideal turbine.*

temperature causes the expansion of the metal parts and the rise in voltage intensifies the electromagnetic interactions. These parameters have a great influence on the rotor eccentricity. It is therefore important to compare air gap values under fixed parameters.

These various factors are illustrated in Figure 7.3. A detailed analysis of the factors influencing the air gap value is proposed in [Nik02].

Following discussions with domain experts from CNR, this work is made under the following assumptions:

Assumption 1. The rotor shaft is non-deformable.

Assumption 2. The rotor shaft and the rotor body have uniform diameters.

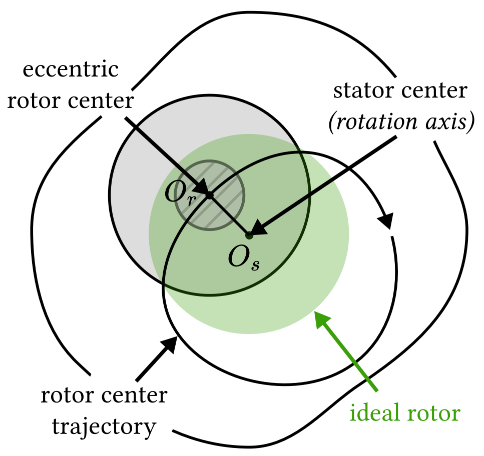
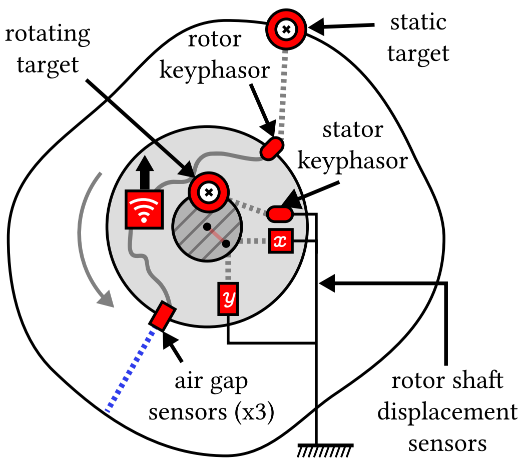
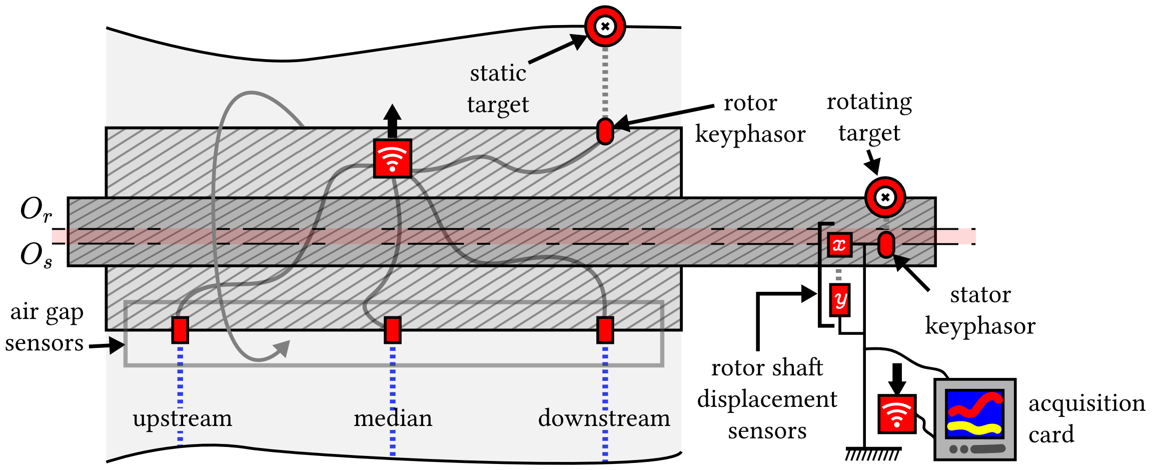


Figure 7.3 – *Model of real turbine used in this paper.*

7.2.2. Sensor Instrumentation



(a) *Front view*



(b) *Section view*

Figure 7.4 – *Schematic of the turbine instrumentation.*

A schematic of the instrumentation is presented in Figure 7.4. The sensors can be divided into two main categories:

- **Rotating sensors.** The air gap is measured by 3 rotating capacitive sensors attached along the rotor (upstream, median, downstream) to get a comprehensive depth profile. Those 3 sensors are associated to a unique rotating keyphasor. This keyphasor uses a metallic target attached to the stator to detect full rotor revolutions. The data collected by those 4 sensors is transmitted wirelessly to the acquisition card.
- **Static sensors.** The displacement of the rotor shaft is measured by 2 proximity sensors. These sensors are associated to a static keyphasor with a rotating target attached

to the rotor. The data is finally transmitted to the acquisition card by wire.

As the rotor data is transmitted wirelessly, its synchronicity with wired stator data can not be guaranteed. *This is why two keyphasors are required.*

7.2.3. Presentation of the time series

This chapter mainly uses 5 sensor time series. As we want the air gap to be constant at recording scale, the time series are required to be *first-order cyclo-stationary* (CS1) [GNP06]. This implies that the state parameters of the machine change negligibly during the recording as they affect the air gap value (see Section 7.2).

First, we consider an air gap sensor (e.g. median sensor) and its associated keyphasor, two rotating sensors:

- $\delta_m(t)$: measured air gap distance
  model: `MC-monitoring AGT-212 M4`
- $k_r(t)$: keyphasor signal associated to $\delta_m(t)$
  model: `CONTRINEX DW-AS-603-M30-002`

Second, we consider the rotor shaft displacement measured by static sensors. It is measured in Cartesian coordinates $(x(t), y(t))$ by two proximity sensors (model: `MEGGITT TQ 402`) and we convert it to polar coordinates to facilitate the analysis. These measures are associated to a keyphasor:

- $e(t)$: radial coordinate of the rotor shaft (radial eccentricity)
- $\theta(t)$: angular coordinate of the rotor shaft
- $k_s(t)$: keyphasor signal associated to $e(t)$ and $\theta(t)$
  model: `CONTRINEX DW-AS-623-M12-120`

The keyphasor signals will be used to recover the instantaneous rotor angle $\beta(t)$ thanks to an order tracking procedure that will be described in more detail in Section 7.4. These various time series are modeled geometrically in Figure 7.5.

Finally, a 6^{th} time series will be used occasionally. It corresponds to the temperature or the stator and is it is noted $\gamma(t)$ in degree Celsius.

7.2.4. Presentation of the recordings

Two distinct sets of recordings of the time series presented above will be used in this chapter:

- **Low frequency.** This first time series was recorded with the acquisition card currently used in CNR turbines. It comes from one of the turbines of the Avignon hydroelectric power plant and its frequency is 50 Hz. It lasts approximately 1 minute and 30 seconds.

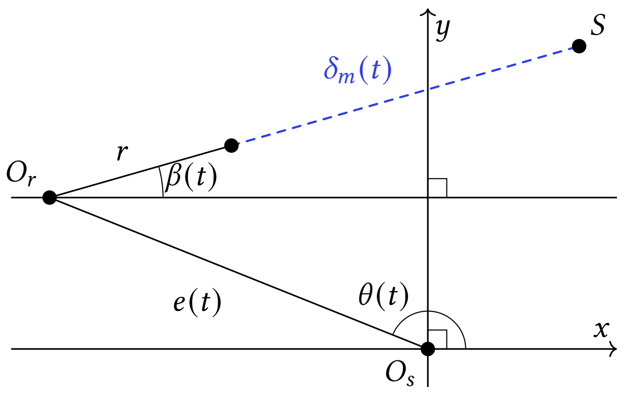


Figure 7.5 – *Geometric model of the generator. $\beta(t)$ is inferred from the two keyphasor signals.*

- **High frequency.** This second set was recorded with an external acquisition card at 2000 Hz at the Péage-de-Roussillon hydroelectric power plant. It consists of 5 recordings between 1 and 2 minutes, taken at regular intervals during the during the start-up of a turbine. Each recording is therefore accompanied by an increasing temperature, which is considered to be stable over the scale of the recording.

Remark 4. Currently, CNR records samples of 1 minute at 50 Hz every several hours such as the *low frequency* one. The recordings are then stored in a remote server to be manually exploited by an industry expert. Depending on their work load and due to the length and tediousness of the process, the analysis is generally done once a year or less for each turbine.

7.3. Validation with ADESIT

In this section we verify several assumptions that will form the basis of our forthcoming solution in the next section. First, we check that the rotor angle does indeed define the measured air gap, *i.e.* we measure in the dataset the veracity of the following FD:

$$\beta(t) \to \delta_m(t)$$

To define the associated predicates, we use the absolute/relative uncertainty predicate defined in formula 5.2, page 42. We know from the sensor's datasheet that the measured air gap δ_m has a relative uncertainty of 2%. We can define the following predicate: $\Phi_{\delta_m}(x,y) = \Phi^u(x,y;0,0.02)$. The absolute uncertainty of the instant rotor angle β can also be calculated according to the following formula:

$$\frac{360 \cdot 50 \text{ (network frequency)}}{32 \text{ (number of poles)} \cdot f \text{ (recording frequency)}}$$

For this test, we use the *low frequency* recording at 50 Hz. Thus, we have an absolute uncertainty of 11° and can use the following predicate: $\Phi_\beta(x,y) = \Phi^u(x,y;11,0)$. Through ADESIT, we obtain that g_3 is about 86% which is quite high and gives confidence in the construction of the model. In Figure 7.6, we can visualize the rotor eccentricity by setting the $x(t)$ and $y(t)$ axis on the scatter plot. After selecting a tuple involved in at least one counterexample and by looking at the histogram of the displacement on the x coordinate (on top of the scatter plot), the spreading of the red vertical line seems to confirm that it would be a discriminative feature. We also observe regions of data with a higher density of counterexamples. If we were to ignore the rotor eccentricity completely, our model might be less accurate in these regions.

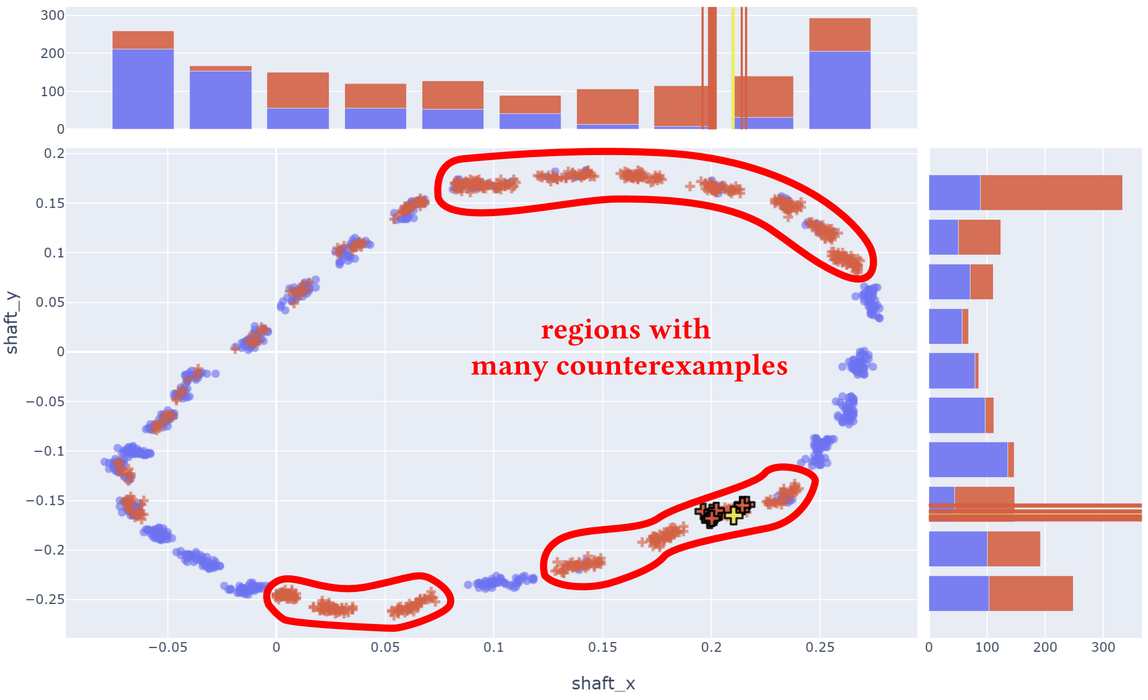


Figure 7.6 – *The FD $\beta(t) \to \delta_m(t)$ is analyzed in* ADESIT *with the low frequency dataset. Here is shown the scatter plot of $x(t)$ vs $y(t)$.*

To go further, we can restart the computation with ADESIT to verify the following FD:

$$\beta(t), x(t), y(t) \to \delta_m(t) \tag{7.1}$$

We use the displacement in Cartesian coordinates $(x(t), y(t))$ to directly use the uncertainties from the datasheet. The two sensors have an absolute uncertainty of 0.005 mm and a relative uncertainty of 5%. We can define the following predicates: $\Phi_x(x,y) = \Phi_y(x,y) = \Phi^u(x,y;0.005,0.05)$. The g_3 has improved considerably by exceeding 95%, confirming the usefulness of incorporating the rotor eccentricity. In Figure 7.6, we also see that most counterexamples have disappeared, except for the upper right region. Some hypotheses have been put forward by experts from CNR. *Does the placement of the receiving antenna cause higher noise in this position? Could the deformation of the stator cause more vibrations in this area?*

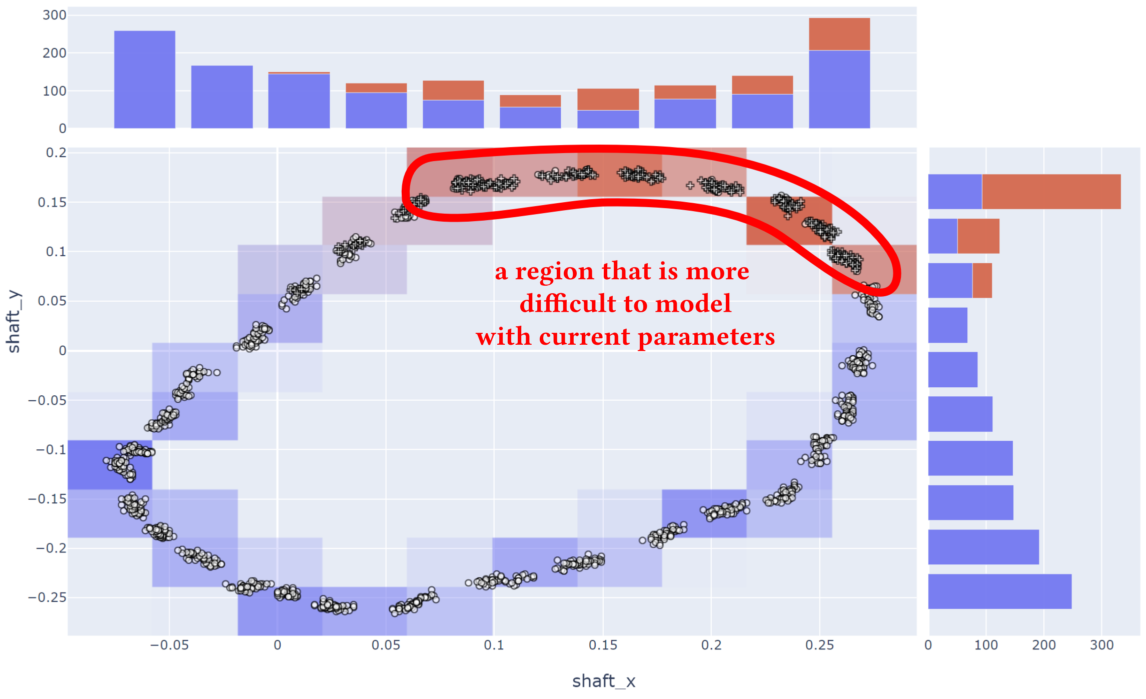


Figure 7.7 – *The FD $\beta(t), x(t), y(t) \rightarrow \delta_m(t)$ is analyzed in* ADESIT *with the low frequency dataset. Here is shown the scatter plot of $x(t)$ vs $y(t)$.*

To summarize, we confirmed that the measured value was a function of the rotor angle concluded that it is safe to assume that the rotor displacement was also a determining factor for the measured air gap.

Before presenting our solution for processing such data, it is also important to verify that temperature does affect the measured air gap. This would confirm that comparing recordings made at similar temperatures is a prerequisite for consistency. To do this, we use the *high frequency* set of recordings, as it consists of several recordings made at different temperatures. For this test, we aggregate all the recordings to obtain a single recording where several temperatures are mixed. Since the recordings were made at 50 Hz, we need to redefine the uncertainty for the instantaneous angle: $\Phi_\beta(x,y) = \Phi^u(x,y;0.30,0)$. In a first test, we do not include the temperature and analyze the FD 7.1 again. In Figure 7.8 we can observe about 6 temperature clusters corresponding to each of the aggregated recordings. In particular, we see that the selected tuple mainly forms counterexamples with tuples from other clusters. This strongly supports the hypothesis that the temperature feature is discriminative with respect to the air gap values. The lack of the temperature feature is supported by the low g_3 value of about 44% and the overwhelming presence of red tuples.

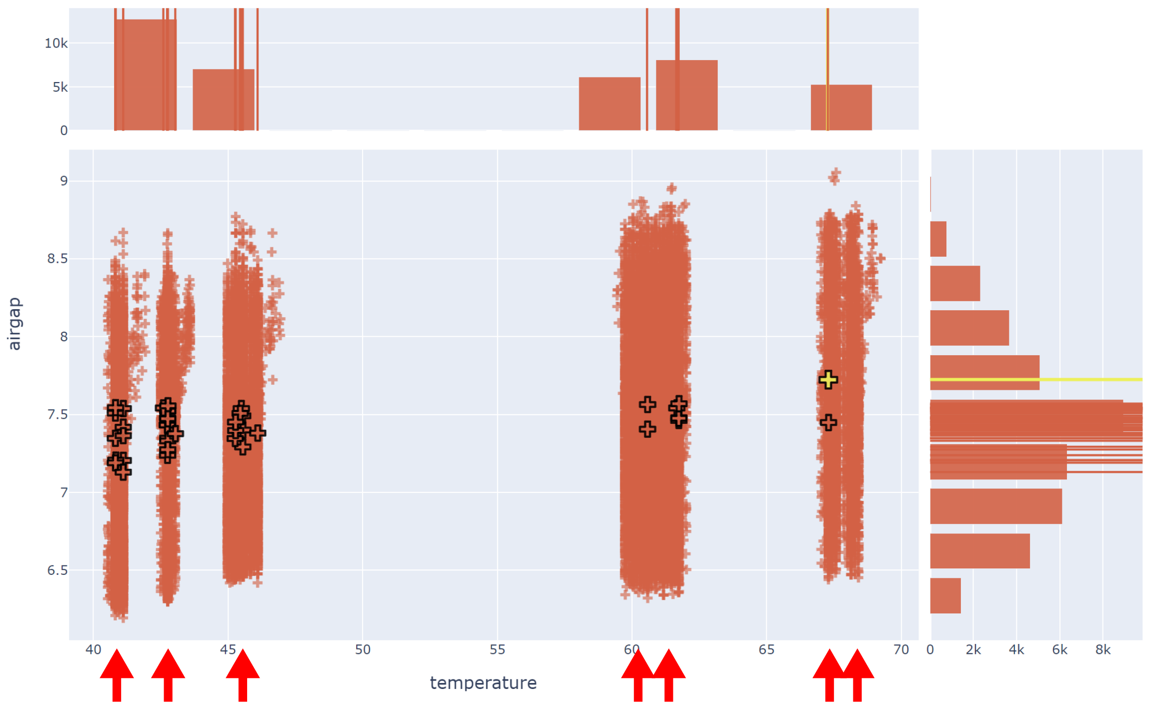


Figure 7.8 – *The FD $\beta(t), x(t), y(t) \rightarrow \delta_m(t)$ is analyzed in* ADESIT *with the high frequency dataset. Here is shown the scatter plot of $\gamma(t)$ vs $\delta_m(t)$. Arrows highlight the temperature clusters corresponding to each of the aggregated recordings.*

To confirm the importance of the temperature, we can add its corresponding attribute and analyze the following FD:

$$\beta(t), x(t), y(t), \gamma(t) \rightarrow \delta_m(t)$$

We know that the temperature sensor has an absolute uncertainty of $0.5°C$. We can therefore define the following predicate: $\Phi_\gamma(x,y) = \Phi^u(x,y;0.5,0)$. The resulting scatter plot is presented in Figure 7.9. For the selected tuple, we observe that all counterexamples coming from other clusters have disappeared, confirming that the temperature was a discriminative feature. The g_3 value has also risen significantly to reach 77%. The remaining counterexamples are due to the limitations of the data itself: *signal noise, sensor precision, missing feature, need of a more complex model....*

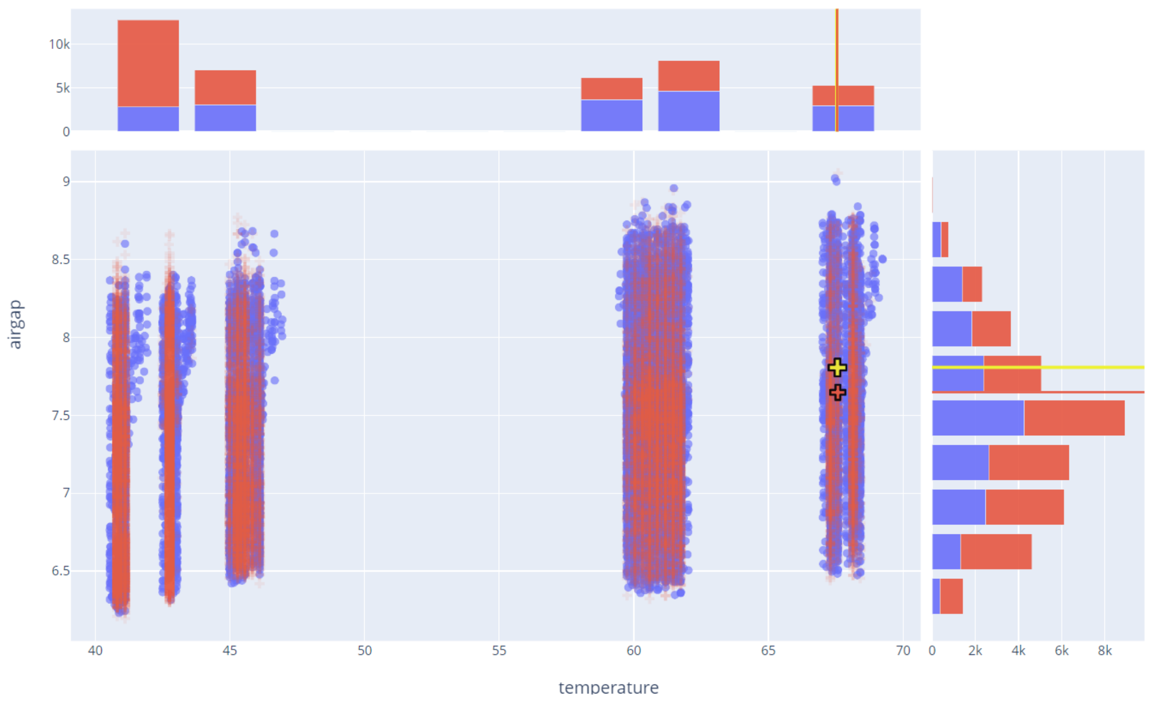


Figure 7.9 – *The FD $\beta(t), x(t), y(t), \gamma(t) \rightarrow \delta_m(t)$ is analyzed in* ADESIT *with the high frequency dataset. Here is shown the scatter plot of $\gamma(t)$ vs $\delta_m(t)$.*

7.4. Automatic processing of the air gap monitoring data

In this chapter, we describe our solution to exploit the sensor data from the instrumentation presented in Section 7.2.2. After providing an overview of our solution, we present the time series and describe each operation in turn. If we describe the solution for one air gap sensor, it can easily be applied to all sensors. In a second stage, we apply our solution to the low frequency recording and present our results. The main symbols used in this paper are presented in Table 7.1.

Table 7.1 – MAIN SYMBOLS USED IN THIS SECTION.

Symbol	**Definition**
$O_r(\theta, e)$	rotor center
r	rotor radius
β	rotor angle
$O_s(0,0)$	stator center
S	a point on the stator
δ_m	measured air gap
δ_a	absolute air gap
δ_c	critical air gap

7.4.1. Solution overview

To help the domain experts anticipate maintenance, the objective of our monitoring solution is two-fold: we want to obtain a clean stator profile to locate deformations but also to know the current smallest air gap. To this end, multiple operations are carried out on the raw sensor data.

First, we use *order tracking* to resample the time series with respect to the rotor angle, a more natural way to handle rotating machine data [Ran21]. Second, we apply *synchronous averaging* to extract the periodic component of the data. Notably, this helps to reduce noise which can be particularly important in such intense electromagnetic fields and with wireless transmission. Third, we geometrically correct the rotor eccentricity to reconstruct the *absolute air gap value*. This is to the distance between the rotor and the stator in the absence of eccentricity. The word *absolute* emphasizes the difference with the raw values *measured* by the air gap sensors which are relative to the rotor angle and eccentricity at the time of recording. Finally, the *critical air gap* is calculated by comparing the rotor movement with the absolute stator profile. This is crucial as the sensors can underestimate the critical air gap by an error of up to 2 times the maximum eccentricity amplitude [PF20]. The concepts of measured, absolute and critical air gaps are illustrated in Figure 7.10.

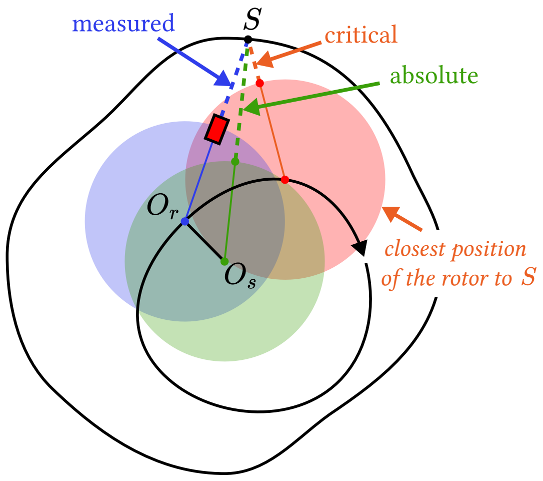


Figure 7.10 – *Absolute, relative and critical air gaps for a given point S on the stator.*

We describe each step in the following sections and the overall pipeline of our solution is schematized in Figure 7.11.

7.4.2. Detailed solution

In this section, we detail the full processing pipeline presented in Figure 7.11.

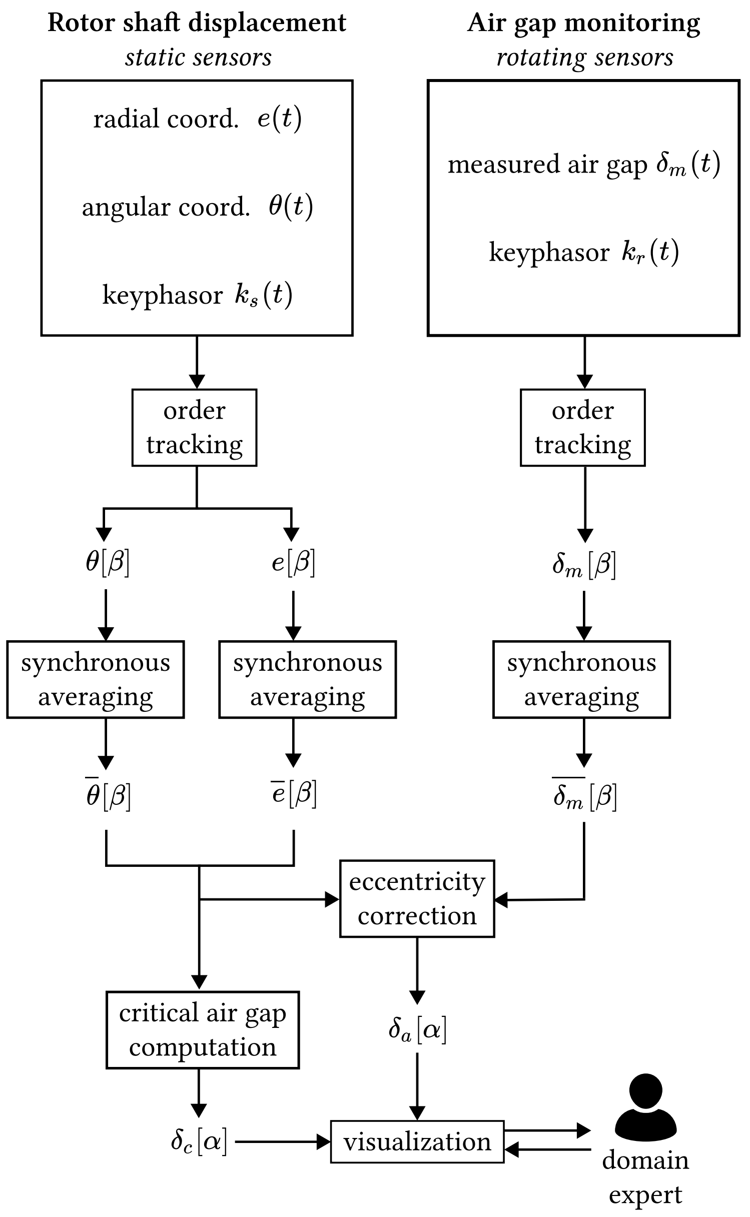


Figure 7.11 – *Schematic of the proposed data science pipeline.*

Order tracking

When dealing with rotating machines, it is common to use data in the angular domain instead of time domain as it is a natural model of the shaft rotation. When the series are not natively recorded indexed on angle (e.g. with an encoder), it is nevertheless possible to resample them using a method such as *computed order tracking* (survey in [FM97]). To perform such resampling, we keep track of full rotor revolutions by using a *rising edge detection* algorithm on the keyphasor series. As the position of the keyphasor target is known, it is thus possible to interpolate the missing angles.

After computed order tracking, all the series are thus index on discrete β increments thanks to their respective keyphasor series. We get: $\delta_m[\beta]$, $\theta[\beta]$ and $e[\beta]$. Note that we use brackets [] instead of parentheses () to highlight the discrete nature of the result. In this work, we use spline interpolation at order 1 and an angular resolution of $\frac{\pi}{16}$ (32 points total).

Synchronous averaging

Synchronous averaging is a common technique for separating the periodic part of some data (the CS1 component) from its background noise (electrical noise, asynchronous sensor vibrations...) [Ran21, BK09, Bon04]. It is done by averaging together a series of segments, each measuring one period. In our case, a period corresponds to a full rotor revolution or a β increment of 2π. Thus, for a signal $x[\beta]$ and K the total number of revolutions in x, its averaged counterpart $\overline{x}[\beta]$ can be expressed as $\overline{x}[\beta] = \frac{1}{K}\sum_{k=0}^{K} x[\beta + 2\pi k]$. An illustration of such process is proposed in Figure 7.12.

Before exploiting $\overline{x}[\beta]$, it is important to verify if the extracted CS1 component is prevalent enough and/or reliable. *I.e. is there too much sensor noise? could the study of other asynchronous components bring interesting information*? We propose to estimate the importance of the noise by computing the synchronous standard deviation (SD) [BAZEBG03, BK09]. If it is high, it means that the noise is important and that further investigation should be conducted to understand its source. It it is due to electrical noise for instance, using a longer recording could improve the results.

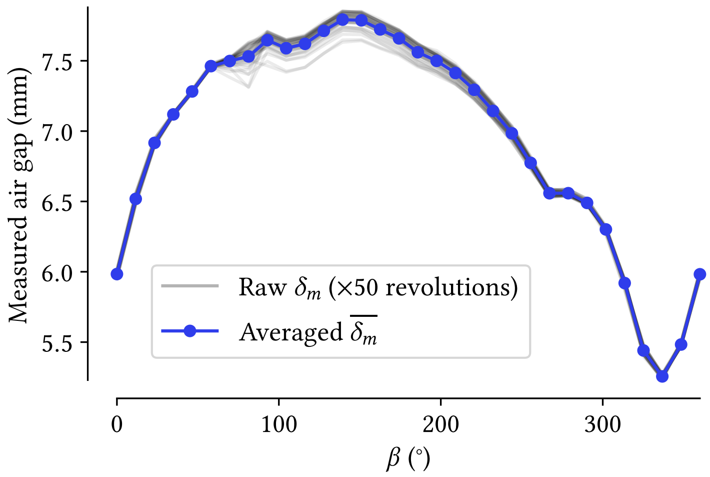


Figure 7.12 – *Illustration of synchronous averaging on a sample of air gap data. In gray, we observe 50 superposed rotor revolutions and in blue their averaged counterpart.*

All in all, this method is particularly interesting because it allows not only to extract the measurement of the denoised signal, but also to evaluate its reliability. It also allows to drastically reduce the storage size of the data, from thousands of points for each time series to a few tens of points after synchronous averaging, as shown in Figure 7.12. It is also quite a cheap operation. We keep an angular resolution of $\frac{\pi}{16}$ and obtain the following signals: $\overline{\delta_m}[\beta]$, $\overline{\theta}[\beta]$ and $\overline{e}[\beta]$.

Rotor eccentricity correction

Now that the signals are resampled in the angular domain and denoised, the next step is to correct rotor eccentricity. Indeed, the values measured by the air gap sensors are biased by the rotor displacement as shown in Figure 7.10 and the objective is to recover the absolute air gap values. As proposed in Figure 7.13, the geometric model can be refined: the series are now indexed on rotor angle and the model is augmented by two new variables (δ_a and α) designating the position of S from the stator center. Thus, δ_a corresponds to the air gap in the case of an ideal rotor with no eccentricity: *the absolute air gap*.

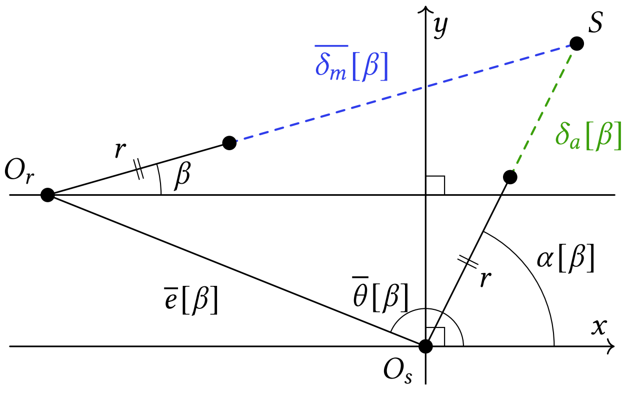


Figure 7.13 – *Geometric model after synchronous averaging. The objective is to compute the values of δ_a and α to construct the absolute air gap profile $\delta_a[\alpha]$.*

It is possible to derive the formulas for δ_a and α from known values as follows:

$$\delta_a[\beta] = \sqrt{\begin{array}{l}\overline{e}[\beta]^2 + (r+\overline{\delta_m}[\beta])^2 \\ \qquad + 2\cdot\overline{e}[\beta]\cdot(r+\overline{\delta_m}[\beta])\cdot\cos(\beta-\overline{\theta}[\beta])\end{array}} - r$$

$$\alpha[\beta] = \beta + \arccos\left(\frac{(r+\overline{\delta_m}[\beta])^2 + (r+\delta_a[\beta])^2 - \overline{e}[\beta]^2}{2\cdot(r+\overline{\delta_m}[\beta])\cdot(r+\delta_a[\beta])}\right)$$

These are consistent as in the absence of eccentricity ($e[\beta]=0$), we have $\overline{\delta_m}[\beta] = \delta_a[\beta]$ and $\alpha[\beta] = \beta$. Once this process is applied for all β values, we obtain a set of (δ_a, α) pairs. This corresponds to $\delta_a[\alpha]$, the angular series describing the absolute air gap values.

Critical air gap computation

Finally, once the absolute air gap profile is computed, it is crucial to evaluate the influence of rotor eccentricity on the air gap distance. To do so, for a fixed point $S(\alpha, r+\delta_a[\alpha])$ on the stator, the objective is to find the position of the rotor the closest to S. *This corresponds to the position where the rotor is the most likely to rub against the stator at point S.* As the trajectory of the rotor is known $(O_r(\overline{\theta}[\beta], \overline{e}[\beta]))$, the critical air gap $\delta_c[\alpha]$ can be expressed as follows:

$$\delta_c[\alpha] = \mathrm{min}_\beta \left(\left\| O_r(\overline{\theta}[\beta], \overline{e}[\beta]) - S(\alpha, r+\delta_a[\alpha]) \right\|_2 \right) - r$$

where $\| \|_2$ is the l_2 norm. This distance is modeled in Figure 7.14.

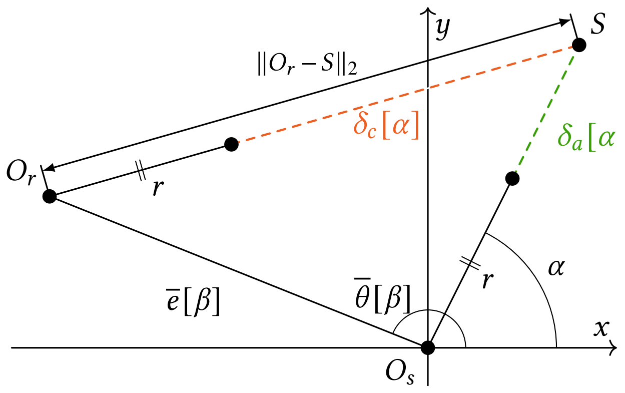


Figure 7.14 – *Geometric model after computing the absolute air gap profile. For a known absolute air gap value $\delta_a[\alpha]$, the new objective is to find the rotor center O_r the* closest *to S as possible. This allows to compute $\delta_c[\alpha]$, the critical air gap, i.e. the position where the rotor is the most likely to rub against the stator at point S.*

7.4.3. Experiments and visualization

In this section, we apply our solution to a 1 minute recording made in one of the generators from CNR. First, we describe the data and give some technical details. Then, we discuss the plotting of the rotor center trajectory. Afterwards, we propose visualizations for the measured, absolute and critical air gap profiles. Finally, we use the 3 air gap sensors altogether to provide a 3D representation of the absolute air gap showing the stator deformation.

Data description and implementation

In this chapter, we apply our solution to the *low frequency* recording described in Section 7.2.4. As the stator deformation evolves slowly, a single but sufficiently long recording is sufficient to build a faithful picture of the stator in the medium term. Nonetheless, comparing different recordings under the same state parameters could help detecting stator deformation evolutions but it is not within the current scope of this study.

All our analysis is made using Python. We use Pandas and Numpy for data processing and Scipy for usual signal processing operations. Plots are made using Matplotlib. All the operations described in this paper can be conducted in a few milliseconds on an average personal computer. Preliminary studies have been made with GeoGebra.

Rotor center trajectory

The rotor center trajectory after synchronous averaging is shown in Figure 7.15. This plot helps the domain expert analyze the eccentricity evolutions and amplitude. In that case, it has a maximum amplitude of roughly 0.28 mm, leading to errors in air gap estimation up to 0.56 mm. This range of error is quite important and should be taken into account to better anticipate future maintenance.

In addition, we can display the synchronous standard deviation for both the angular (θ) and radial (e) components with two perpendicular lines (in red in Figure 7.15). These lines are quite small in that plot, meaning that sensor noise is quite low and that the CS1 component contains supposedly reliable information.

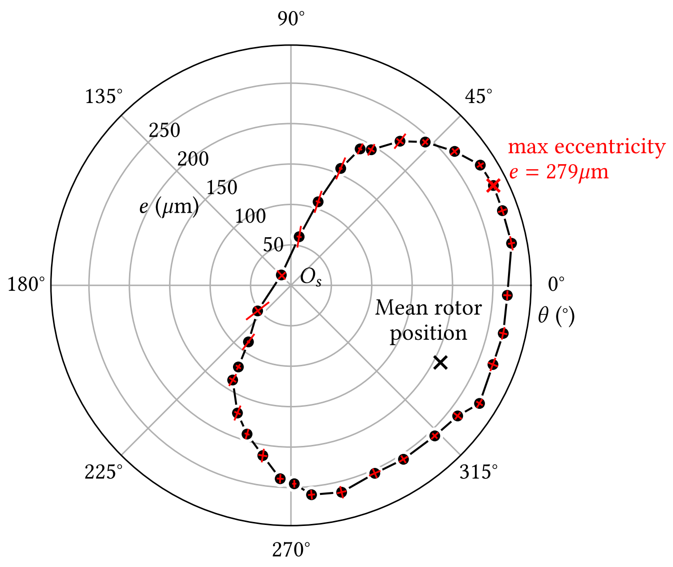


Figure 7.15 – *Rotor center O_r trajectory. Synchronous standard deviation for the angular and radial coordinates is displayed in red.*

Air gap profiles

In Figure 7.16, we display the measured ($\overline{\delta_m}[\beta]$), absolute ($\delta_a[\alpha]$) and critical ($\delta_c[\alpha]$) air gap profiles in a polar plot. The polar plot is particularly suitable because it provides a visual similar to a cross-sectional view of the stator.
For the air gap profile, we display the synchronous standard deviation as a colored area around the main plot. The presence of points with a larger standard deviation has led to interesting hypothesis with domain experts: *wireless cards positioning? local stator attachment vibrations?* Nonetheless, the standard deviation is quite reasonable and solutions for improving the results will be discussed in Section 7.4.3. In addition, the absolute air gap presents the closest representation of the stator sectional view (by definition). It already shows some interesting deviations from the the raw measured air gap. Finally, the critical air gap displays the closest distance between the stator and the rotor and accounts for the eccentricity: *those are the most important values to monitor to prevent collisions*. By comparing the measured and critical air gaps, we observe zones with differences up to about 0.6 mm. This confirms that the theoretical worst case explained before *may happen* in practice.

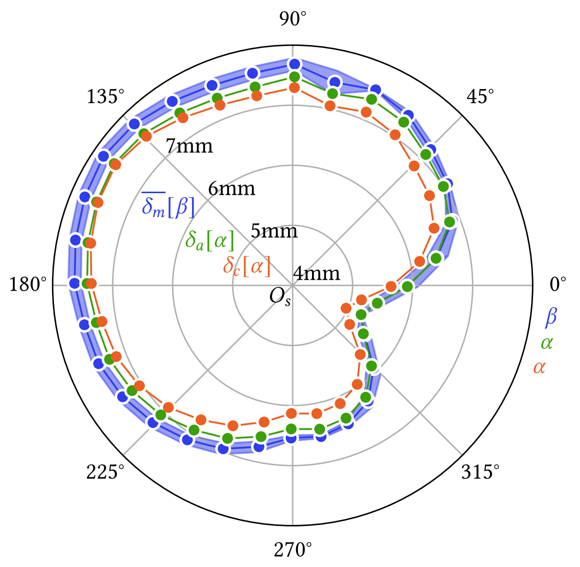


Figure 7.16 – *Polar plot of the measured ($\overline{\delta_m}[\beta]$), absolute ($\delta_a[\alpha]$), and critical ($\delta_c[\alpha]$) air gap profiles. Synchronous SD for $\delta_m[\beta]$ is displayed as the blue area.*

3D profile

In Figure 7.17, we apply our solution to the 3 air gap sensors (upstream, median and downstream) and obtain a 3D profile of the absolute air gap. This can be seen as an unfolding of the stator surface which easier to visualize than its tubular form on a 2D format. This allows to capture trends in the stator deformation to understand where maintenance should be carried out. In that case, we observe a tendency of the stator to contract in its upstream part with a quite low minimum absolute air gap. It is also possible to display the critical air gap instead but the representation becomes a less intuitive representation of the stator surface.

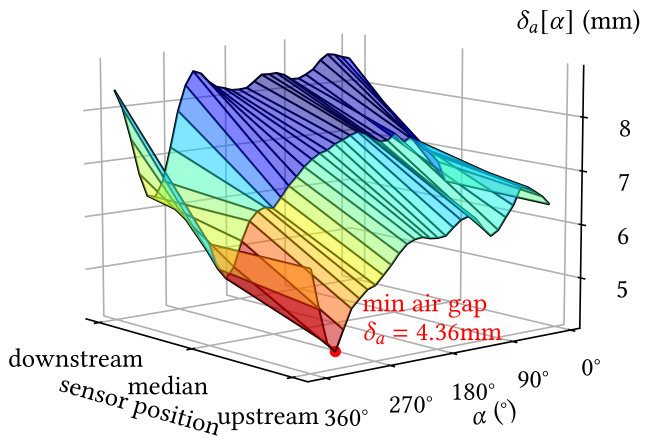


Figure 7.17 – *3d air gap profile allowing to capture the general trends in stator deformation. Can be displayed with critical air gap too.*

Discussion on data acquisition

Today, only a part of CNR's generators is equipped with this monitoring solution and CNR is willing to extend this instrumentation to the others. However, this operation is really expensive, and a trade-off between costs and quality must be found. On this purpose, CNR would like to refine its data acquisition strategy. In this section, we discuss two important parameters controlling data acquisition and having an impact on data quality: *the frequency and the length of the recordings*.

Recording frequency. As recording at higher frequencies is expensive, finding the optimum balance to meet domain requirements is critical to optimizing costs. According to domain experts, 32 spatial points are sufficient to obtain a meaningful stator profile. As the rotor rotates at a maximum frequency of 1.5625 Hz (see Section 1.3), recording at $1.5625 \cdot 32 = 50$ Hz is the absolute minimum and leaves no room for variations in the rotation frequency.

Another operation depends on the recording frequency: *order tracking*. Indeed, order tracking uses the keyphasor to detect full rotor revolutions with a *rising edge detection* algorithm. As shown in Figure 7.18, the keyphasor produces a peak whose beginning is identified as a rising edge when it exceeds a given threshold. This rising edge is then associated with the known reference angle of the target. For such an algorithm to work, at least one point must be recorded at each peak summit. In addition, the more points that are recorded for each peak, the more accurately we can identify the position of the reference angle (see [FM97] for more). With the data used in the experiments at 50 Hz, we can barely see all the peaks, requiring a more complex rising edge detection algorithm. Thus, further experiments conducted with the *high frequency* recordings together with geometrical considerations have shown that sampling at 2000Hz allows to record approximately 14 points on the considered generator. This is why in order to obtain 2 or 3 points for each peak, sampling at 300 Hz should be sufficient.
All in all, we propose to use a recording frequency of at least 300 Hz but it would be beneficial to conduct additional tests on different generators.

Remark 5. Note that we cannot use an alternative method based frequency analysis [FM97] (e.g. Fourier or Hilbert transforms) instead of the keyphasors. Indeed, we need to be able to get an angular reference, especially for spatially matching the wired and wireless data.

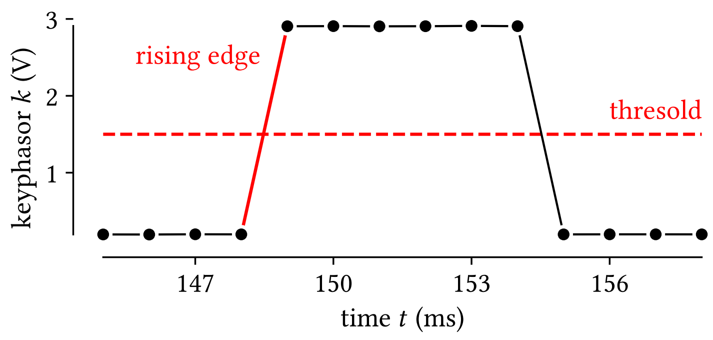


Figure 7.18 – *Illustration of the rising edge detection algorithm on a 1000Hz keyphasor sample recorded at the Péage-de-Roussillon power plant.*

Recording length. Using our solution, the data can be processed either after or before storage. In this latter case, the memory usage can be drastically reduced by only storing the air gap profiles (this data is sufficient to answer the given problem). Indeed, as explained in Section 7.4.2, synchronous averaging produces a short summary of the original data whose size depends on the chosen angular resolution (32 points in the experiments). If we also store the synchronous SD and the absolute and critical air gaps, it is still radically cheaper than storing all the initial sensor time series (117 times less in our experiments). In that case, we can even record longer time series which will strengthen the result of the synchronous averaging process by condensing more rotor revolutions. As these series would only be stored temporarily, they would not increase storage costs.

7.5. Conclusion

In this chapter, we studied the automatic processing of air gap monitoring data in compact hydro-generators, more specifically in the case of stator deformation with rotating remote sensors and dynamic rotor eccentricity. We used counterexample analysis as a preliminary step to show how the various attributes affected the air gap and verify our primary assumptions. Then, we presented a data science pipeline to automatize and improve the analysis of air gap monitoring data at the CNR.

Our method allows to improve the quality and interpretability of the raw data by extracting the periodic component and removing the uninformative noise. In addition to significantly reducing the memory size of the data, it also allows to evaluate the strength of the extracted information, thus detecting cases were data is not exploitable. Moreover, we also show how to recover the stator profile and the critical air gap values by correcting rotor eccentricity. Paired with the visualizations presented in Section 5.4, the results provided by this study provided a basis for the improvement of the air gap monitoring system of CNR. Notably, it assists the development of a more effective predictive maintenance system.

In the future, it would be useful to verify that the sensor data streams are CS1 at the order of the shaft, i.e. *that the air gap and rotor eccentricity does not change during the recording*. While visualization can provide reliable insights into this question, more robust statistical tests could be performed [BAZEBG03, AB19]. An other future challenge would be to exploit historical data to not only analyze the current air gap values, but also *predict future stator deformation*. Such information would considerably help optimizing maintenance costs and agenda.

Chapter 8

Related work, conclusion and perspective

This chapter concludes the manuscript. First, we mention some related literature. We question our choice to study the g_3-error as opposed to other coverage measures. We also review the current knowledge about the complexity of computing g_3 and mention previous algorithmic studies for its computation. We also compare our data science solution for processing air gap monitoring data with previous solutions from the literature. Second, we wrap up this dissertation and mention some research perspectives.

8.1. Related work

First, it is interesting to mention alternatives to g_3 and justify its use in this work. Indeed, the g_3 indicator is not the only known coverage measure. For example, purity dependencies [SCC02] are based on an impurity measure, soft FDs [IMH+04] and probabilistic FDs [WDS+09] use a probabilistic approach when the alternative of a compression-based measure is preferred in partial determinations [PK95]. An overview of various coverage measures is presented by Caruccio et al. [CDP15]. Nonetheless, the g_3 indicator is undoubtedly among the most widely used coverage measures: Approximate FDs [KM95], approximate DDs [SC11] or approximate comparable dependencies [SCP13] are examples of FDs using the g_3 indicator as their coverage measure and many mining algorithms also use the g_3 indicator to find FDs which almost hold in a relation [KM95, WSC+17, HKPT99, CDP16]. This ubiquity of the g_3 indicator is notably due to its intuitive interpretation and its flexibility. When the g_3 only requires the definition of FD satisfiability (which is at the core of any FD definition), the equality relaxation generalized by FDs with predicates is harder to capture with coverage measures such as the probabilistic ones mentioned above where the need to group values together struggles with the loss of transitivity.

We now mention previous complexity results for the computation of g_3 for crisp FDs and their extension to predicates. First, it is known that the g_3-error can be computed in polynomial time for crisp FDs [HKPT98]. Then, Song et al. show that the EVPP is NP-complete for comparable dependencies [SCP13] but do not study predicate properties as we do in this work. However, comparable dependencies happen to be reflexive (`ref`) and symmetric (`sym`) predicates, which coincides with Theorem 2. This allows to determine the hardness of the EVPP for differential [SC11], matching [Fan08], metric [KSSV09], neighborhood [BW01], and comparable dependencies [SCP13]. For some of these dependencies, predicates can be defined over sets of attributes. Using one predicate per attribute and taking their conjunction is a particular case of predicate on attribute sets.

In the following, we mention previous studies of algorithmic solutions for the computation of g_3. Kivinen and Mannila introduce the g_3-error for crisp FDs for the first time in the literature [KM95] but only the computation process is described and no detailed algorithm for its exact computation is considered. Huhtala et al. present a simple algorithm to compute the g_3-error [HKPT99]. Concerning random sampling, an approach is proposed in [KM95] for the EVPP and is mentioned in Section 5.3.3. Cormode et al. explore the computation of the confidence of conditional FDs by proposing streaming algorithms based on advanced sampling schemes [CGF+09]. One of their propositions has been implemented (`APPROX_SRS`), improved (`APPROX_SRSI`) and compared to other alternatives.
g_3 has been used extensively in the context of FD mining, but little work has been done on algorithms for it extension to general predicates. Sond and Chen established for the first time an equivalence between the error (or confidence) and the MVC (or MIS) in the case of DDs [SC11]. A well-known 2-approximation algorithm [PS98] is used to solve the MVC in both [SC11] and [CDP16]. This algorithm corresponds to `APPROX_2Approx` in this paper and has been shown to give average results in practice. The alternative of sublinear algorithms has been slightly studied and a simple algorithm by Parnas and Ron is proposed [PR07]. Significant improvements in terms of complexity and approximation guarantees have been proposed in this area and [NO08, YYI09, ORRR12] (we study [YYI09, ORRR12] in this thesis).

We now focus on the construction of conflict-graphs and the CEE. Despite some studies on algorithms for computing g_3 with predicates, the problem of converting the input from a relation to a graph problem is not considered by [SC11] or [CDP16]. However, this is a computationally intensive process, which in our experiments proved to be a bottleneck for some of the very efficient approximation algorithms used to solve the MVC/MIS. The CEE approach used to achieve this conversion is a very interesting problem at the intersection of record linkage and similarity joins. While we have tried to cover most of the relevant literature, the case of unordered metric space has been deliberately omitted for brevity, although it is often useful for comparing strings with metrics such as the Levenshtein distance. This operation is similar to a *range self-similarity join in a metric space* and has been extensively studied [ZADB06] through the use of indexing such as tree data structures (see [HS03] for a survey) or divide-and-conquer algorithms such as QuickJoin [JS08]. In any case, pairwise tuple enumeration is known to be a hard problem with no silver bullet. Similarly, although efficient approximation algorithms exist, solving MVC exactly in the most general case remains a major computational challenge.

Finally, we mention some previous literature on air gap monitoring solution. Talas and Toom exploit a fiber-optic instrumentation attached to the rotor to monitor the air gap of large hydro-generators [TT83]. In a study by Pollock and Tyles, the sensors are mounted on the stator to monitor the rotor, the opposite of our study [PL92]. Nikhil uses capacitive sensors attached to the stator and propose a detailed flow chart before data treatment [Nik02]. A study by Xuan et al. presents a new approach based on measurement coils instead of capacitive sensors for a cheaper instrumentation [XSWK06]. Finally, Adamowski et al. offer a new alternative by attaching ultrasonic sensors to the stator [ASP+13]. While these works propose extensive studies on sensor instrumentation and data acquisition, often in the case of static air gap sensors attached to the stator, they offer little or no discussion of the data processing pipeline. To summarize, to the best of our knowledge, no data science study specific to the *exploitation* of air gap monitoring data can be found in the literature. Our contribution is a first step in this direction. Furthermore, the data science tools used in this dissertation are common which makes our pipeline easily accessible. We redirect the reader to [Ran21, Blo03, Ant09, Bon04] for more details on the analysis of rotating machinery and cyclostationary processes. More details on order tracking and synchronous averaging are also available in [FM97, BLAT05] and [BK09] respectively.

8.2. Research summary and perspectives

In this dissertation, we investigated several aspects of counterexample analysis of FDs with predicates, focusing in particular on the g_3-error. First, we studied the impact of four common properties (reflexivity, transitivity, antisymmetry, and symmetry) on the computational complexity of g_3 and produced the hierarchy proposed in Figure 4.2. Second, we studied the algorithmic pipeline to compute the g_3-error accurately and approximately. All algorithms are proposed in an open-source Python library: FASTG3. In the general **NP**-hard case, efficient approximation algorithms can be used to reduce the complexity of computing an exact MVC. In this case, the construction of the conflict-graph using the CEE becomes the bottleneck of the operation, and sublinear algorithms offer a good alternative. In the polynomial case, where the predicates are restricted to be at least transitive and symmetric, the algorithms have proved to be quite scalable. The problem can also be approximated quite accurately by stratified sampling approaches. We also present ADESIT, a web application for performing a counterexample analysis of a dataset with respect to a function. We have described the various features of ADESIT and given an example of its use. Finally, we applied counterexample analysis to the problem of air gap monitoring as a go/no-go step before constructing the processing pipeline itself. We now highlight some research directions for future work.

In a recent work [LKR20], Livshits et al. study the problem of computing optimal repairs in a relation with respect to a set of FDs. A repair is a collection of tuples that does not violate a given set of FDs. It is optimal if it is of maximal size among all possible repairs. Henceforth, there is a strong connection between the problem of computing repairs and computing the g_3-error with respect to a collection of FDs. In their work, the authors give a dichotomy between tractable and intractable cases based on the structure of the FDs. In particular, they use previous results from Gribkoff et al. [GVdBS14] to show that the problem is already **NP**-complete for 2 FDs in general. In the case where computing an optimal repair can be done in polynomial time, it would be interesting to use our approach and relax equality with predicates in order to study the tractability of computing the g_3-error on a collection of FDs with relaxed equality.

Despite the optimizations proposed in Chapter 5, the computation of the g_3-error is still expensive on large datasets, especially in the general **NP**-hard case. First, for approximation algorithms, progressive sampling techniques could be considered to speed up the algorithms. Instead of sampling a fixed number of tuples, one would sample tuples *progressively* until the g_3 estimate stabilizes. Second, we observed that the bottleneck of the computation lies in the construction of the conflict-graph using the CEE. Sublinear algorithms offer significant time savings, but they all simulate `APPROX_2Approx`, which is known to provide average approximations in practice. So it might be interesting as future work to adapt a practically better approximation algorithm (*e.g.* degree-based heuristics) which should give better results. It already seems to us that some algorithms such as `APPROX_GIC` [HR97] have an approach that we consider to be too global with respect to the graph to allow for local decisions. However, the Sorted List Right (SLR) algorithm [DL08] offers very good practical performance [DL10] and seems to be adaptable in sublinear time. This dichotomy between *adaptable* and *non-adaptable* algorithms as well as this adaptation of the algorithm SLR itself seems interesting.

Finally, the link between the g_3-error and the Bayes error rate could be examined in more detail. Indeed, in the context of supervised learning, the former describes the error associated with a dataset [LPS20], while the latter is associated with the error of the process itself [JWHT13]. In Appendix D, page xxix, we prove that if the data set is i.i.d. (independent and identically distributed), then the classical g_3-error with crisp FDs corresponds to the Bayes error rate when the number of tuples to tends to infinity. However, this connection is not well identified when using predicates instead of equality. Is it possible to approximate the Bayes error rate by incorporating domain knowledge? Such a connection could be of interest to the community.

Publications

The results of this thesis have led to multiple publications in conferences. First, here is a list of the international conference publications accepted at the time of writing:

- **Functional Dependencies with Predicates:**
  What Makes the g_3-error Easy to Compute?
  Simon Vilmin, Pierre Faure--Giovagnoli, Jean-Marc Petit, Vasile-Marian Scuturici
  International Conference on Conceptual Structures, 2023, 14 pages
- **Assessing the Existence of a Function in your Dataset with the g_3 Indicator**
  Pierre Faure--Giovagnoli, Jean-Marc Petit, Vasile-Marian Scuturici
  IEEE International Conference on Data Engineering, 2022, 14 pages
- **ADESIT: Visualize the Limits of your Data in a Machine Learning Process**
  Pierre Faure--Giovagnoli, Marie Le Guilly, Jean-Marc Petit, Vasile-Marian Scuturici
  International Conference on Very Large Data Bases, 2021, 4 pages

One other publication accepted on abstract in a international domain conference:

- **Automatic Processing of Air Gap Monitoring Signals in Hydro-Generators**
  Pierre Faure--Giovagnoli, Christophe Turbidi and Vasile-Marian Scuturici
  SURVISHNO (SURveillance, VIbration SHocks and NOise), 2023, 3 pages

Two softwares were also registered:

- **Fastg3 - A Python library for computing the g_3 indicator efficiently**
  Pierre Faure--Giovagnoli, Jean-Marc Petit, Vasile-Marian Scuturici
  2023
- **ADESIT - An application for visualizing the limits of a dataset in supervised learning**
  Pierre Faure--Giovagnoli, Marie Le Guilly, Jean-Marc Petit, Vasile-Marian Scuturici
  2023

Finally, a journal publication named **"Assessing the Existence of a Function in a Dataset: Complexity and Algorithmics"** written in conjunction with Simon Vilmin, Jean-Marc Petit and Vasile-Marian Scuturici is currently under preparation.

Bibliography

[AB19] Jérôme Antoni and Pietro Borghesani. A statistical methodology for the design of condition indicators. *Mechanical Systems and Signal Processing*, 114:290–327, 2019.

[Ant09] Jérôme Antoni. Cyclostationarity by examples. *Mechanical Systems and Signal Processing*, 23(4):987–1036, 2009.

[Arm74] William Ward Armstrong. Dependency structures of data base relationships. In *IFIP congress*, volume 74, pages 580–583. Geneva, Switzerland, 1974.

[ASM22] Andrea Ahlemeyer-Stubbe and Agnes Müller. Why domain knowledge is essential for data scientists in marketing. *Applied Marketing Analytics*, 7(4):362–373, 2022.

[ASP+13] Julio C Adamowski, Alan T Souza, Nicolás Pérez, Allan A Lima, Paulo D Oda, and Hamilton H Tiba. Ultrasonic dynamic air-gap monitoring system for large hydro-generators. In *2013 IEEE International Ultrasonics Symposium (IUS)*, pages 1311–1314. IEEE, 2013.

[BAZEBG03] Frédéric Bonnardot, A Al Zohbi, Mohamed El Badaoui, and François Guillet. Aide à l'interprétation des signaux cyclostationnaires. In *CNR IUT 2003*, 2003.

[BBC+11] Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2):31–39, 2011.

[BBFL05] Leopoldo Bertossi, Loreto Bravo, Enrico Franconi, and Andrei Lopatenko. Complexity and approximation of fixing numerical attributes in databases under integrity constraints. In *International Workshop on Database Programming Languages*, pages 262–278. Springer, 2005.

[Ber73] Claude Berge. Graphs and hypergraphs. *North-Holland Pub. Co.*, 1973.

[Ber11] Leopoldo Bertossi. Database repairing and consistent query answering. *Synthesis Lectures on Data Management*, 3(5):1–121, 2011.

[BFG$^+$07] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. Conditional functional dependencies for data cleaning. In *2007 IEEE 23rd international conference on data engineering*, pages 746–755. IEEE, 2007.

[BK09] Eric Bechhoefer and Michael Kingsley. A review of time synchronous average algorithms. In *Annual Conference of the PHM society*, volume 1, 2009.

[BKN13] Jaume Baixeries, Mehdi Kaytoue, and Amedeo Napoli. Computing similarity dependencies with pattern structures. In *The Tenth International Conference on Concept Lattices and their Applications-CLA 2013,*, pages 33–44, 2013.

[BLAT05] Anders Brandt, Thomas Lago, Kjell Ahlin, and Jiri Tuma. Main principles and limitations of current order tracking methods. *Sound and Vibration*, 39(3):19–22, 2005.

[Blo03] Jason R Blough. A survey of dsp methods for rotating machinery analysis, what is needed, what is available. *Journal of sound and vibration*, 262(3):707–720, 2003.

[Bon04] Frédéric Bonnardot. *Comparaison entre les analyses angulaire et temporelle des signaux vibratoires de machines tournantes. Etude du concept de cyclostationnarité floue*. PhD thesis, Institut National Polytechnique de Grenoble-INPG, 2004.

[BSEK06] Piero P Bonissone, Raj Subbu, Neil Eklund, and Thomas R Kiehl. Evolutionary algorithms+ domain knowledge= real-world evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 10(3):256–280, 2006.

[BW01] Renaud Bassée and Jef Wijsen. Neighborhood dependencies for prediction. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 562–567. Springer, 2001.

[CC13] Yunus Cengel and John Cimbala. *Ebook: Fluid mechanics fundamentals and applications (si units)*. McGraw Hill, 2013.

[CCPP17] Brice Chardin, Emmanuel Coquery, Marie Pailloux, and Jean-Marc Petit. Rql: a query language for rule discovery in databases. *Theoretical Computer Science*, 658:357–374, 2017.

[CDP15] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. Relaxed functional dependencies—a survey of approaches. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):147–165, 2015.

[CDP16] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. On the discovery of relaxed functional dependencies. In *Proceedings of the 20th International Database Engineering & Applications Symposium*, pages 53–61, 2016.

[CGF$^+$09] Graham Cormode, Lukasz Golab, Korn Flip, Andrew McGregor, Divesh Srivastava, and Xi Zhang. Estimating the confidence of conditional functional dependencies. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, page 469–482, New York, NY, USA, 2009. Association for Computing Machinery.

[Chr11] Peter Christen. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE transactions on knowledge and data engineering*, 24(9):1537–1555, 2011.

[CIP13] Xu Chu, Ihab F Ilyas, and Paolo Papotti. Discovering denial constraints. *Proceedings of the VLDB Endowment*, 6(13):1498–1509, 2013.

[CKX06] Jianer Chen, Iyad A Kanj, and Ge Xia. Improved parameterized upper bounds for vertex cover. In *International symposium on mathematical foundations of computer science*, pages 238–249. Springer, 2006.

[CSLS13] Shaowei Cai, Kaile Su, Chuan Luo, and Abdul Sattar. Numvc: An efficient local search algorithm for minimum vertex cover. *Journal of Artificial Intelligence Research*, 46:687–716, 2013.

[DCH97] Surnjani Djoko, Diane J. Cook, and Lawrence B. Holder. An empirical study of domain knowledge and its benefits to substructure discovery. *IEEE Transactions on Knowledge and Data Engineering*, 9(4):575–586, 1997.

[DF12] Rodney G Downey and Michael Ralph Fellows. *Parameterized complexity*. Springer Science & Business Media, 2012.

[DGZ03] Vlastislav Dohnal, Claudio Gennaro, and Pavel Zezula. Similarity join in metric spaces using ed-index. In *International Conference on Database and Expert Systems Applications*, pages 484–493. Springer, 2003.

[DH82] Jirun Dong and Richard Hull. Applying approximate order dependency to reduce indexing space. In *Proceedings of the 1982 ACM SIGMOD international conference on Management of data*, pages 119–127, 1982.

[DL08] François Delbot and Christian Laforest. A better list heuristic for vertex cover. *Information Processing Letters*, 107(3-4):125–127, 2008.

[DL10] François Delbot and Christian Laforest. Analytical and experimental comparison of six algorithms for the vertex cover problem. *Journal of Experimental Algorithmics (JEA)*, 15:1–1, 2010.

[DP02] Brian A Davey and Hilary A Priestley. *Introduction to lattices and order*. Cambridge university press, 2002.

[DP12] Thomas H Davenport and DJ Patil. Data scientist. *Harvard business review*, 90(5):70–76, 2012.

[dR22] Compagnie Nationale du Rhône. https://www.cnr.tm.fr/wp-content/uploads/2022/07/essentiel-2022-bat2.pdf, 2022.

[DS05] Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of mathematics*, pages 439–485, 2005.

[Fan08] Wenfei Fan. Dependencies revisited for improving data quality. In *Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 159–170, 2008.

[FEAT09] Jawad Faiz, Bashir Mahdi Ebrahimi, Bilal Akin, and Hamid A Toliyat. Comprehensive eccentricity fault diagnosis in induction motors using finite element method. *IEEE Transactions on Magnetics*, 45(3):1764–1767, 2009.

[FGPS22] Pierre Faure-Giovagnoli, Jean-Marc Petit, and Vasile-Marian Scuturici. Assessing the existence of a function in a dataset with the g3 indicator. In *IEEE International Conference on Data Engineering*, 2022.

[FGPSLG21] Pierre Faure-Giovagnoli, Jean-Marc Petit, Vasile-Marian Scuturici, and Marie Le Guilly. Adesit: Vizualize the limits of your data in a machine learning process. In *International Conference on Very Large Data Bases*, pages 2679–2682, Copenhaguen, Denmark, August 2021.

[FGTS23] Pierre Faure-Giovagnoli, Christophe Turbidi, and Vaile-Marian Scurutici. Automatic processing of air gap monitoring signals in hydro-generators. *Book of Proceedings Survishno 2023*, 2023.

[FM97] KR Fyfe and EDS Munck. Analysis of computed order tracking. *Mechanical systems and signal processing*, 11(2):187–205, 1997.

[Fuk13] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[GH83] Seymour Ginsburg and Richard Hull. Order dependency in the relational model. *Theoretical computer science*, 26(1-2):149–195, 1983.

[GJ79] Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

[GKK+08] Lukasz Golab, Howard Karloff, Flip Korn, Divesh Srivastava, and Bei Yu. On generating near-optimal tableaux for conditional functional dependencies. *Proceedings of the VLDB Endowment*, 1(1):376–390, 2008.

[GKK+09] Lukasz Golab, Howard Karloff, Flip Korn, Avishek Saha, and Divesh Srivastava. Sequential dependencies. *Proceedings of the VLDB Endowment*, 2(1):574–585, 2009.

[GNP06] William A Gardner, Antonio Napolitano, and Luigi Paura. Cyclostationarity: Half a century of research. *Signal processing*, 86(4):639–697, 2006.

[Gol04] Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*. Elsevier, 2004.

[Gol10] Oded Goldreich. *P, NP, and NP-Completeness: The basics of computational complexity*. Cambridge University Press, 2010.

[GRT97] Vassilis Giakoumakis, Florian Roussel, and Henri Thuillier. On p_4-tidy graphs. *Discrete Mathematics and Theoretical Computer Science*, 1:17–41, 1997.

[GVdBS14] Eric Gribkoff, Guy Van den Broeck, and Dan Suciu. The most probable database problem. In *Proceedings of the First international workshop on Big Uncertain Data (BUDA)*, pages 1–7, 2014.

[HKPT98] Ykä Huhtala, Juha Karkkainen, Pasi Porkka, and Hannu Toivonen. Efficient discovery of functional and approximate dependencies using partitions. In *Proceedings 14th International Conference on Data Engineering*, pages 392–401. IEEE, 1998.

[HKPT99] Yka Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *The computer journal*, 42(2):100–111, 1999.

[HLSS20] Demian Hespe, Sebastian Lamm, Christian Schulz, and Darren Strash. Wegotyoucovered: The winning solver from the pace 2019 challenge, vertex cover track. In *2020 Proceedings of the SIAM Workshop on Combinatorial Scientific Computing*, pages 1–11. SIAM, 2020.

[HR97] Magnús M Halldórsson and Jaikumar Radhakrishnan. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. *Algorithmica*, 18(1):145–163, 1997.

[HS03] Gisli R Hjaltason and Hanan Samet. Index-driven similarity search in metric spaces (survey article). *ACM Transactions on Database Systems (TODS)*, 28(4):517–580, 2003.

[IMH$^+$04] Ihab F Ilyas, Volker Markl, Peter Haas, Paul Brown, and Ashraf Aboulnaga. Cords: Automatic discovery of correlations and soft functional dependencies. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 647–658, 2004.

[JPN$^+$20] Abhinav Jain, Hima Patel, Lokesh Nagalapatti, Nitin Gupta, Sameep Mehta, Shanmukha Guttula, Shashank Mujumdar, Shazia Afzal, Ruhi Sharma Mittal, and Vitobha Munigala. Overview and importance of data quality for machine learning tasks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3561–3562, 2020.

[JS08] Edwin H Jacox and Hanan Samet. Metric space similarity joins. *ACM Transactions on Database Systems (TODS)*, 33(2):1–38, 2008.

[JWHT13] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[Kar09] George Karakostas. A better approximation ratio for the vertex cover problem. *ACM Transactions on Algorithms (TALG)*, 5(4):1–8, 2009.

[Kit14] Rob Kitchin. Big data, new epistemologies and paradigm shifts. *Big data & society*, 1(1):2053951714528481, 2014.

[KM95] Jyrki Kivinen and Heikki Mannila. Approximate inference of functional dependencies from relations. *Theoretical Computer Science*, 149(1):129–149, 1995.

[KSSV09] Nick Koudas, Avishek Saha, Divesh Srivastava, and Suresh Venkatasubramanian. Metric functional dependencies. In *2009 IEEE 25th International Conference on Data Engineering*, pages 1275–1278. IEEE, 2009.

[Li94] Kim-Hung Li. Reservoir-sampling algorithms of time complexity $\mathcal{O}(n \cdot (1 + \log(\frac{N}{n})))$. *ACM Transactions on Mathematical Software (TOMS)*, 20(4):481–493, 1994.

[LKR20] Ester Livshits, Benny Kimelfeld, and Sudeepa Roy. Computing optimal repairs for functional dependencies. *ACM Transactions on Database Systems (TODS)*, 45(1):1–46, 2020.

[LL12] Mark Levene and George Loizou. *A guided tour of relational databases and beyond*. Springer Science & Business Media, 2012.

[LPS20] Marie Le Guilly, Jean-Marc Petit, and Vasile-Marian Scuturici. Evaluating classification feasibility using functional dependencies. *Trans. Large Scale Data Knowl. Centered Syst.*, 44:132–159, 2020.

[MAEA05] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *International conference on database theory*, pages 398–412. Springer, 2005.

[MSL$^+$15] Ingo Müller, Peter Sanders, Arnaud Lacurie, Wolfgang Lehner, and Franz Färber. Cache-efficient aggregation: Hashing is sorting. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1123–1136, 2015.

[MV80] Silvio Micali and Vijay V Vazirani. An $\mathcal{O}(\sqrt{|v|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*, pages 17–27. IEEE, 1980.

[NC01] Noel Novelli and Rosine Cicchetti. Functional and embedded dependency inference: a data mining point of view. *Information Systems*, 26(7):477–506, 2001.

[Ng01] Wilfred Ng. An extension of the relational data model to incorporate ordered domains. *ACM Transactions on Database Systems (TODS)*, 26(3):344–383, 2001.

[Nik02] Kumar Nikhil. *Air Gap Monitoring of Hydropower Unit Generator to Advance Vibration Diagnostic Procedure*. PhD thesis, Riga Technical University, 2002.

[NK04] Ullas Nambiar and Subbarao Kambhampati. Mining approximate functional dependencies and concept similarities to answer imprecise queries. In *Proceedings of the 7th International Workshop on the Web and Databases: Colocated with ACM SIGMOD/PODS 2004*, pages 73–78, 2004.

[NO08] Huy N Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 327–336. IEEE, 2008.

[ORRR12] Krzysztof Onak, Dana Ron, Michal Rosen, and Ronitt Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1123–1131. SIAM, 2012.

[PF20] J. Panchot and Di Nolfo F. Conception du système de surveillance d'entrefer : matériels, modalités d'installation, d'exploitation et de maintenance. *Compagnie Natioale du Rhône*, 2020.

[PK95] Bernhard Pfahringer and Stefan Kramer. Compression-based evaluation of partial determinations. In *Knowledge Discovery and Data Mining*, pages 234–239, 1995.

[PL92] GB Pollock and JF Lyles. Vertical hydraulic generators experience with dynamic air gap monitoring. *IEEE Transactions on Energy Conversion*, 7(4):660–668, 1992.

[Pol74] Svatopluk Poljak. A note on stable sets and colorings of graphs. *Commentationes Mathematicae Universitatis Carolinae*, 15(2):307–309, 1974.

[PR07] Michal Parnas and Dana Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theoretical Computer Science*, 381(1-3):183–196, 2007.

[PS98] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[PSTP20] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. Blocking and filtering techniques for entity resolution: A survey. *ACM Computing Surveys (CSUR)*, 53(2):1–42, 2020.

[Ran21] Robert Bond Randall. *Vibration-based condition monitoring: industrial, automotive and aerospace applications*. John Wiley & Sons, 2021.

[SC11] Shaoxu Song and Lei Chen. Differential dependencies: Reasoning and discovery. *ACM Transactions on Database Systems (TODS)*, 36(3):1–41, 2011.

[SCC02] Dan A Simovici, Dana Cristofor, and Laurentiu Cristofor. Impurity measures in databases. *Acta Informatica*, 38(5):307–324, 2002.

[SCP13] Shaoxu Song, Lei Chen, and S Yu Philip. Comparable dependencies over heterogeneous data. *The VLDB journal*, 22(2):253–274, 2013.

[Ser74] Robert J Serfling. Probability inequalities for the sum in sampling without replacement. *The Annals of Statistics*, pages 39–48, 1974.

[SGHW20] Shaoxu Song, Fei Gao, Ruihong Huang, and Chaokun Wang. Data dependencies extended for variety and veracity: A family tree. *IEEE Transactions on Knowledge and Data Engineering*, 10, 2020.

[SS13] Seref Sagiroglu and Duygu Sinanc. Big data: A review. In *2013 international conference on collaboration technologies and systems (CTS)*, pages 42–47. IEEE, 2013.

[SVSF14] Rebecca C Steorts, Samuel L Ventura, Mauricio Sadinle, and Stephen E Fienberg. A comparison of blocking methods for record linkage. In *International conference on privacy in statistical databases*, pages 253–268. Springer, 2014.

[TT83] P Talas and P Toom. Dynamic measurement and analysis of air gap variations in large hydroelectric generators. *IEEE Transactions on Power Apparatus and Systems*, (9):3098–3106, 1983.

[VFGPS23] Simon Vilmin, Pierre Faure-Giovagnoli, Jean-Marc Petit, and Vasile-Marian Scuturici. Functional dependencies with predicates: what makes the g3-error easy to compute? In *International Conference on Conceptual Structures*, 2023.

[Via13] Stijn Viaene. Data scientists aren't domain experts. *IT Professional*, 15(6):12–17, 2013.

[Vit85] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

[WDS+09] Daisy Zhe Wang, Xin Luna Dong, Anish Das Sarma, Michael J Franklin, and Alon Y Halevy. Functional dependency generation and applications in pay-as-you-go data integration systems. In *WebDB*, 2009.

[Wil11] William Willis. Bounds for the independence number of a graph. *Virginia Commonwealth University*, 2011.

[WL19] Ziheng Wei and Sebastian Link. Embedded functional dependencies and data-completeness tailored database design. *Proceedings of the VLDB Endowment*, 12(11):1458–1470, 2019.

[WMCS84] Calvin Cropper Warnick, Howard A Mayo, James L Carson, and Lee H Sheldon. *Hydropower engineering*. Number BOOK. Prentice-Hall Englewood Cliffs, NJ, 1984.

[WSC+17] Yihan Wang, Shaoxu Song, Lei Chen, Jeffrey Xu Yu, and Hong Cheng. Discovering conditional matching rules. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 11(4):1–38, 2017.

[XSWK06] M Tu Xuan, J-J Simond, R Wetter, and S Keller. A novel air-gap monitoring system for large low speed hydro-generators. In *2006 IEEE Power Engineering Society General Meeting*. IEEE, 2006.

[YHPM99] Suk-Chung Yoon, Lawrence J Henschen, EK Park, and Sam Makki. Using domain knowledge in knowledge discovery. In *Proceedings of the eighth international conference on Information and knowledge management*, pages 243–250, 1999.

[YYI09] Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. An improved constant-time approximation algorithm for maximum matchings. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 225–234, 2009.

[ZADB06] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity search: the metric space approach*, volume 32. Springer Science & Business Media, 2006.

Appendices

Appendix A

Proof of Theorem 2

THEOREM (2). *The problem* EVPP *is* **NP***-complete even when the predicates are symmetric (*`sym`*) and reflexive (*`ref`*).*

Proof. We first show that the EVPPbelongs to NP. Let (R, Φ) be a relation scheme with predicates, r a relation over R, $X \rightarrow A$ a functional dependency over R and $k \in \mathbb{R}$. We have that $g_3^{\Phi}(X \rightarrow A, r) \leq k$ if and only if there exists a subrelation s in r satisfying $s \models_{\Phi} X \rightarrow A$ and $1 - \frac{|s|}{|r|} \leq k$, or $|s| \geq (1-k) \cdot |r|$ equivalently. Therefore, a certificate for the EVPP is a subrelation s containing at least $(1-k) \cdot |r|$ tuples and satisfying $X \rightarrow A$ (with respect to Φ). Since predicates can be computed in polynomial time by assumption, it takes polynomial time to check that $s \models_{\Phi} X \rightarrow A$. Thus, the EVPP belongs to NP.

To show NP-completeness, it is convenient to use a reduction from Maximum Clique (MC) rather than the MIS, even though the problems are polynomially equivalent:

MAXIMUM CLIQUE (MC)

Input:	A graph $G = (V, E)$, $k \in \mathbb{N}$.
Output:	`yes` if G has a clique with at least k vertices, `no` otherwise.

Let $G = (V, E)$ be a graph with $V = \{u_1, \dots, u_n\}$ for some $n \in \mathbb{N}$, and $E = \{e_1, \dots, e_m\}$ for some $m \in \mathbb{N}$. Let k be an integer such that $k \leq |V|$. We construct an instance of the EVPP. We begin with a relation scheme with predicates (R, Φ) where $R = \{B_1, \dots, B_m, A\}$, $\Phi = \{\phi_1, \dots, \phi_m, \phi_A\}$, and:

- for each $1 \leq i \leq m$, $\mathsf{dom}(B_i) = \{0, 1, 2\}$ and ϕ_i is defined as follows:

$$\phi_i(x, y) = \begin{cases} \text{TRUE} & \text{if } x = y \text{ or } x + y < 3 \\ \text{FALSE} & \text{otherwise.} \end{cases}$$

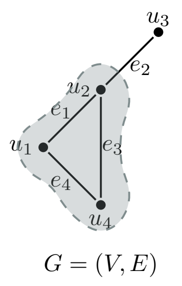
  Observe that ϕ_i is reflexive and symmetric.

– $\mathsf{dom}(A) = \{1, \ldots, n\}$, and the predicate ϕ_A for A is defined by $\phi_A(x,y) = \texttt{true}$ if and only if $x = y$. Thus, ϕ_A is reflexive and symmetric.

Observe that the predicates can be computed in polynomial time in the size of their input. Now, we build a relation $r = \{t_1, \ldots, t_n\}$ (one tuple per vertex in G) over R. For each $1 \le i \le n$, we put $t_i[A] = i$ and for each $1 \le j \le m$:

$$t_i[B_j] = \begin{cases} 0 & \text{if } u_i \notin e_j \\ 1 & \text{if } e_j = (u_i, u_\ell) \text{ and } i < \ell \\ 2 & \text{if } e_j = (u_\ell, u_i) \text{ and } \ell < i \end{cases}$$

Finally, let $k' = 1 - \frac{k}{n}$, and consider the functional dependency $X \longrightarrow A$ where $X = \{B_1, \ldots, B_m\}$. We obtain an instance of the EVPP which can be constructed in polynomial time in the size of G. The reduction is illustrated on an example in Figure A.1.



r	B_1	B_2	B_3	B_4	A
t_1	1	0	0	1	1
t_2	2	1	1	0	2
t_3	0	2	0	0	3
t_4	0	0	2	2	4

Figure A.1 – *Illustration of the reduction of Theorem 2. In gray, a clique and its associated subrelation satisfying $X \longrightarrow A$.*

To conclude the proof, we have to prove that G contains a clique K such that $|K| \ge k$ if and only if $g_3^\Phi(X \longrightarrow A, r) \le k'$. To do so, we show that for every distinct tuples t_i, t_j of r, $\bigwedge_{1 \le \ell \le m} \phi_\ell(t_i[B_\ell], t_j[B_\ell]) = \text{TRUE}$ if and only if (u_i, u_j) is not an edge of G.

We begin with the only if part. Hence, assume that for each $1 \le \ell \le m$, we have $\phi_\ell(t_i[B_\ell], t_j[B_\ell]) = \text{TRUE}$. By definition of ϕ_ℓ, we have two cases:

– $t_i[B_\ell] = t_j[B_\ell]$. By construction of r, it follows that $t_i[B_\ell] = 0$. Hence, neither u_i nor u_j belongs to e_ℓ.

– $t_i[B_\ell] + t_j[B_\ell] < 3$. It follows that either $t_i[B_\ell] = 0$ or $t_j[B_\ell] = 0$. Without loss of generality, assume that $t_i[B_\ell] = 0$. Then, again by construction of r, we deduce that $u_i \notin e_\ell$.

Thus, (u_i, u_j) is not an edge of G.

We move to the if part. We use contrapositive. Hence, assume there exists some B_ℓ, $1 \le \ell \le m$, such that $\phi_\ell(t_i[B_\ell], t_j[B_\ell]) = \text{FALSE}$. By definition of ϕ_ℓ, we deduce that $t_i[B_\ell] \neq t_j[B_\ell]$ and $t_i[B_\ell] + t_j[B_\ell] \ge 3$. Without loss of generality, we obtain $t_i[B_\ell] = 1$ and $t_j[B_\ell] = 2$.

Therefore, by construction of r, $u_i \in e_\ell$ and $u_j \in e_\ell$ must hold. As $t_i[B_\ell] \neq t_j[B_\ell]$, we deduce that $(u_i, u_j) = e_\ell$, concluding this part of the proof.

Consequently, a subset K of V is a clique in G if and only if the corresponding set of tuples $s(K)$ is a subrelation of r which satisfies $X \rightarrow A$. Therefore, G contains a clique K such that $|K| \geq k$ if and only if $g_3^\Phi(X \rightarrow A, r) \leq k'$ holds, which concludes the proof. ■

Appendix B

Proof of Theorem 3

THEOREM (3). *The problem* EVPP *is* **NP***-complete even when the predicates are transitive (*`tra`*), reflexive (*`ref`*), and antisymmetric (*`asym`*).*

Proof. The fact that the EVPP belongs to NPhas been shown in Theorem 2.

To show NP-completeness, we use a reduction from MIS in 2-subdivision graphs, in which MIS remains NP-complete [Pol74]. Let $G = (V, E)$ be an (undirected) graph where $V = \{u_1, \ldots, u_n\}$ and $E = \{e_1, \ldots, e_m\}$. Without loss of generality, we assume that G is loopless and that each vertex belongs to at least one edge. Let $V_2 = V \cup \{v_k^i \mid 1 \leq k \leq m, 1 \leq i \leq n \text{ and } u_i \in e_k\}$ be a new set of vertices. We construct a set E_2 of edges. It is obtained from E by replacing each edge $e_k = (u_i, u_j)$ by a path made of three edges $\{(u_i, v_k^i), (v_k^i, v_k^j), (v_k^j, u_j)\}$. The graph $G_2 = (V_2, E_2)$ is the 2-subdivision of G. Every 2-subdivision graph is the 2-subdivision of some graph.

Now we construct an instance of the EVPP. Let $\{a_1, \ldots, a_n\}$ be a set of characters. We build a relation scheme with predicates (R, Φ) where $R = \{B, A\}$, $\Phi = \{\phi_B, \phi_A\}$, and:

– $\mathsf{dom}(B)$ is the set of pairs of symbols associated to $\{a_1, \ldots, a_n\} \cdot \{a_1, \ldots, a_n\}$. We add a predicate ϕ_B as follows:

$$\phi_B(x, y) = \begin{cases} \text{TRUE} & \text{if } x = y \\ \text{TRUE} & \text{if } x \neq y \text{ and } x[1] = x[2] \text{ and } x[1] \in \{y[1], y[2]\} \\ \text{FALSE} & \text{otherwise.} \end{cases}$$

The predicate is *reflexive* by definition. We prove that it is *transitive*. Let $x, y, z \in \mathsf{dom}(B)$ and assume that $\phi_B(x, y) = \phi_B(y, z) = \text{TRUE}$. If $x = y = z$, we readily have $\phi_B(x, z) = \text{TRUE}$. Since $x \neq z$ implies $x \neq y$ or $y \neq z$, it is sufficient to show that $\phi(x, z) = \text{TRUE}$ in these two cases. Assume first that $x \neq y$. Then $\phi_B(x, y) = \text{TRUE}$ if and only if $x = a_i a_i$ and $y \in \{a_i a_j, a_j a_i\}$ for $1 \leq i, j \leq n$, $i \neq j$. It follows that $\phi_B(y, z)$ holds if and only if $z = y$. Thus, $\phi_B(x, z) = \phi_B(x, y) = \text{TRUE}$. Let us assume now that $y \neq z$. Then, $\phi_B(y, z) = \text{TRUE}$ implies that $y = a_i a_i$ for some $1 \leq i \leq n$, by definition of ϕ_B. Therefore, $\phi_B(x,y) = \text{TRUE}$ entails $x = y$. We deduce $\phi_B(x,z) = \text{TRUE}$. Consequently, ϕ_B is transitive. At last, assume that $\phi_B(x,y) = \text{TRUE}$ with $x \neq y$. Hence, $y[1] \neq y[2]$ and $\phi_B(y,x)$ cannot be TRUE. Therefore, $\phi_B(x,y) = \phi_B(y,x) = \text{TRUE}$ entails $x = y$. Thus, ϕ_B is also *antisymmetric*.

- $\mathsf{dom}(A) = \{1,\ldots,n\}$ and $\phi_A(x,y) = \text{TRUE}$ if and only if $x = y$. In other words, ϕ_A is the usual equality. Hence, it enjoys both reflexivity, transitivity and antisymmetry.

Observe that all predicates can be computed in polynomial time in the size of their input.

Now we construct a relation $r = \{t_1,\ldots,t_n\} \cup \{t_k^i \mid 1 \leq k \leq m, 1 \leq i \leq n, v_k^i \in V_2\}$ (one tuple per vertex in G_2) over R:

- for each $1 \leq i \leq n$, $t_i[B] = a_i a_i$ and $t_i[A] = i$,
- for each $1 \leq k \leq m$ and each $1 \leq i \leq n$ such that $v_k^i \in V_2$, let $e_k = (u_i, u_j)$, $1 \leq j \leq n$, be the corresponding edge of G. Then, we put $t_k^i[B] = a_i a_j$ if $i < j$ and $a_j a_i$ otherwise. As for A, we define $t_k^i[A] = j$.

Finally, we consider the functional dependency $B \rightarrow A$. The whole reduction can be computed in polynomial time in the size of G. It is illustrated on an example in Figure B.1. Intuitively, ϕ_B guarantees that two tuples representing adjacent vertices of G_2 will agree on B in (R,Φ). However, the transitivity of ϕ_B will produce pairs of tuples which agree on B even though they are not adjacent in G_2. More precisely, ϕ_B returns TRUE in two cases:

- when it compares t_i to t_k^i and t_k^j for each edge e_k of G to which u_i belongs, and
- when it compares t_k^i to t_k^j for each edge e_k of G.

The role of Φ_A is then to assert that non-adjacent tuples cannot produce counterexamples.

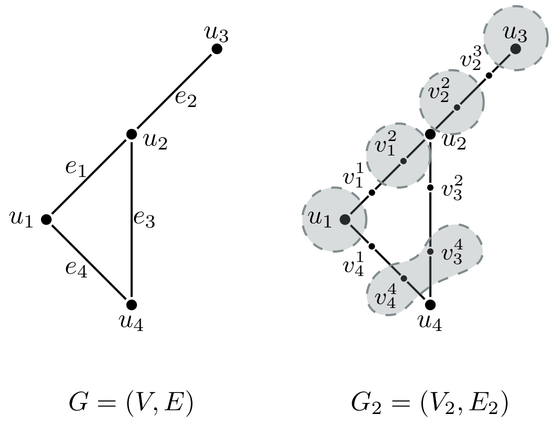
We show that two tuples of r do not satisfy $B \rightarrow A$ if and only if the associated vertices of V_2 are connected in G_2.

We begin with the if part. Consider two (distinct) vertices of V_2 that are adjacent in G_2. Because G_2 is the 2-subdivision of G, we have the following cases:

- u_i, v_k^i for some $1 \leq i \leq n$ and $1 \leq k \leq m$. For B, we have $t_i[B] = a_i a_i$ and $t_k^i[B] = a_i a_j$ (or $a_j a_i$) for some $1 \leq j \leq n$. Therefore, $\phi_B(t_i[B], t_k^i[B]) = \text{TRUE}$ holds. However, $t_i[A] \neq t_k^i[A]$ also by definition of r. Thus, $\{t_i, t_k^i\} \not\models_\Phi B \rightarrow A$.
- v_k^i, v_k^j for some $1 \leq i < j \leq n$ (without loss of generality) and $1 \leq k \leq m$. Then, $t_k^i[B] = t_k^j[B]$, $t_k^i[A] = j$, and $t_k^j[A] = i$. It follows that $\{t_k^j, t_k^i\} \not\models_\Phi B \rightarrow A$, by definition of ϕ_B and ϕ_A.

Thus, if two vertices are connected in G_2, the corresponding tuples in r does not satisfy the functional dependency $B \rightarrow A$, concluding this part of the proof.

We show the only if part using contrapositive. Consider two (distinct) vertices of V_2 that are not connected in G_2. We have four cases:



r	B_1	B_2	A
t_1	a_1b_0	a_1a_1	1
t_2	a_2b_0	a_2a_2	2
t_3	a_3b_0	a_3a_3	3
t_4	a_4b_0	a_4a_4	4
t_1^1	a_1b_1	a_1a_2	2
t_1^2	a_2b_1	a_1a_2	1
t_2^2	a_2b_2	a_2a_3	3
t_2^3	a_3b_2	a_2a_3	2
t_3^2	a_2b_3	a_2a_4	4
t_3^4	a_4b_3	a_2a_4	2
t_4^1	a_1b_4	a_1a_4	4
t_4^4	a_4b_4	a_1a_4	1

Figure B.1 – *Illustration of the reduction of Theorem 3. In gray, an independent set and its associated subrelation satisfying $B \longrightarrow A$.*

- u_i, u_j for some $1 \le i < j \le n$. By definition of r, we have $t_i[B] = a_ia_i$ and $t_j[B] = a_ja_j$. Thus, $\phi_B(t_i[B], t_j[B]) = \phi_B(t_j[B], t_i[B]) =$ FALSE, and $\{t_i, t_j\} \models_\Phi B \longrightarrow A$ holds.
- v_k^i, v_ℓ^j for some $1 \le k, \ell \le m$ and $1 \le i, j \le n$. Then, $t_k^i[B][1] \ne t_k^i[B][2]$ and $t_\ell^j[B][1] \ne t_\ell^j[B][2]$. According to G_2, v_k^i and v_ℓ^j are not connected if and only if $k \ne \ell$. Consequently, $t_k^i[B] \ne t_\ell^j[B]$ by definition of r. Hence, $\phi_B(t_k^i[B], t_\ell^j[B]) = \phi_B(t_\ell^j[B], t_k^i[B]) =$ FALSE. We deduce that $\{t_k^i, t_\ell^j\} \models_\Phi B \longrightarrow A$.
- u_i, v_k^j for some $1 \le i, j \le n$, $1 \le k \le m$ and $u_i \notin e_k$ in G. Then, $t_i[B] = a_ia_i$ and since $u_i \notin e_k$, we have $t_k^j[B] = a_ja_\ell$ (or $a_\ell a_j$) for some $1 \le \ell \le n$ and $i \ne j, \ell$. By definition of ϕ_B, we deduce that $\phi_B(t_i[B], t_k^j[B]) = \phi_B(t_k^j[B], t_i[B]) =$ FALSE must hold. Therefore, $\{t_i, t_j^k\} \models_\Phi B \longrightarrow A$ is TRUE too.
- u_i, v_k^j for some $1 \le i, j \le n$, $1 \le k \le m$ and $u_i \in e_k$ in G. Then, necessarily $i \ne j$ by construction of G_2. Consequently, we must have $t_i[A] = t_j^k[A] = i$ by definition of r. Therefore, $\phi_A(t_i[A], t_k^j[A]) =$ TRUE and $\{t_i, t_k^j\} \models_\Phi B \longrightarrow A$ holds.

Thus, whenever two vertices of G_2 are disconnected, the corresponding set of tuples of r satisfies $B \longrightarrow A$. This concludes the proof of the equivalence.

Consequently, G_2 has an independent set of size k if and only if there exists a subrelation s of r of size k which satisfies $B \longrightarrow A$, concluding the proof. ■

Appendix C

Proof of Theorem 5

THEOREM (5). *Let (R,Φ) be a relation scheme with predicates satisfying* `ref`, `tra`, `sym` *and* `asym`. *Let r be a relation and φ a functional dependency over (R,Φ). Let $\mathtt{A}((R,\Phi),\varphi,r)$ be an algorithm for computing $g_3^\Phi(\varphi,r)$ in $T_{\mathtt{A}}(|r|)$ time. For an error ϵ and a confidence δ,* $\underline{\mathtt{APPROX\_URS}}((R,\Phi),\varphi,r,\mathtt{A},\epsilon,\delta)$ *computes an estimate $\hat{g}_3^\Phi$ of $g_3^\Phi(\varphi,r)$ such that $p(|\hat{g}_3^\Phi - g_3^\Phi| \leq \epsilon) \geq \delta$. Its time complexity is $\mathcal{O}(T_{\mathcal{S}}(m,|r|) + T_{\mathtt{A}}(m))$ with $m = \mathsf{min}\left(|r|, \left\lceil \frac{1}{2\cdot\epsilon^2}\cdot \ln(\frac{2}{1-\delta})\right\rceil\right)$ and $T_{\mathcal{S}}(m,|r|)$ is the complexity of sampling m tuples in r.*

Proof. Let (R,Φ) be a relation scheme with predicates satisfying `ref`, `tra`, `sym` and `asym`. Let r be a relation and $\varphi : X \rightarrow A$ a functional dependency over (R,Φ). Let $\mathtt{A}((R,\Phi),\varphi,r)$ be an algorithm for computing $g_3^\Phi(\varphi,r)$.

The $g_3^\Phi(\varphi,r_i)$ indicator of a given equivalence class $r_i \in \Omega_X(r)$ is the sum of the frequencies of all elements in the domain of A (a.k.a. $\mathsf{dom}(A)$) except the most frequent one:

$$g_3^\Phi(\varphi,r_i) = 1 - \frac{\mathsf{max}_{a\in\mathsf{dom}(A)}(|\sigma_{A=a}(r_i)|)}{|r_i|} \tag{C.1}$$

By doing a weighted average of all the $g_3^\Phi(\varphi,r_i)$ values for each $r_i \in \Omega_X(r)$, we can compute compute the total g_3 for r:

$$g_3^\Phi(\varphi,r) = \sum_{r_i\in\Omega_X(r)} \frac{|r_i|}{|r|}\cdot g_3^\Phi(\varphi,r_i)$$

Remark 6. Note that in formula C.1, the domain of attribute A (a.k.a. $\mathsf{dom}(A)$) could be replaced by the active domain (a.k.a. $\mathsf{adom}(A)$) or more restrictively the projection of the equivalence class on A (a.k.a. $r_i[A]$). We keep the domain for the genericity of the notations.

Let $s \subset r$ be a sample of r of size $m = \mathsf{min}\left(|r|, \left\lceil \frac{1}{2\cdot\epsilon^2}\cdot \ln(\frac{2}{1-\delta})\right\rceil\right)$ drawn uniformly and without replacement. Observe that, since $s \subseteq r$, each s_i is contained in a unique r_j. For clarity, we assume $s_i \subseteq r_i$. In each equivalence class $s_i \in \Omega_X(s)$ and for each tuple $t_{i,j} \in s_i$, we associate a Bernoulli random variable $X_{i,j}$ equal to 0 when $t_{i,j}[A]$ is the most frequent element in s_i and 1 otherwise, or more formally:

$$X_{i,j} = \begin{cases} 0 \text{ if } t_{i,j}[A] = \mathsf{argmax}_{a\in\mathsf{dom}(A)}(|\sigma_{A=a}(r_i)|), \\ 1 \text{ otherwise} \end{cases}$$

We consider the following estimator $\hat{g}_3^{\Phi}$ of g_3^{Φ}, result of algorithm $\mathtt{A}$:

$$\begin{aligned} \hat{g}_3^{\Phi}(\varphi, r) &= \frac{1}{|s|} \sum_{s_i\in\Omega_X(s)} \sum_{t_{i,j}\in s_i} X_{i,j} \\ &= \sum_{s_i\in\Omega_X(s)} \frac{|s_i|}{|s|} \cdot \left(1 - \frac{\mathsf{max}_{a\in\mathsf{dom}(A)}(|\sigma_{A=a}(s_i)|)}{|s_i|}\right) \\ &= \sum_{s_i\in\Omega_X(s)} \frac{|s_i|}{|s|} \cdot g_3^{\Phi}(\varphi, s_i) = g_3^{\Phi}(\varphi, s) = \mathtt{A}(s) \end{aligned}$$

We prove that $\hat{g}_3^{\Phi}$ is an unbiased estimator of g_3^{Φ}:

$$\begin{aligned} \mathbb{E}(\hat{g}_3^{\Phi}(\varphi, r)) &= \mathbb{E}\left(\sum_{s_i\in\Omega_X(s)} \frac{|s_i|}{|s|} \cdot g_3^{\Phi}(\varphi, s_i)\right) \\ &= \sum_{r_i\in\Omega_X(r)} \frac{|r_i|}{|r|} \cdot \mathbb{E}(g_3^{\Phi}(\varphi, s_i)) \\ &= \sum_{r_i\in\Omega_X(r)} \frac{|r_i|}{|r|} \cdot g_3^{\Phi}(\varphi, r_i) = g_3^{\Phi}(\varphi, r) \end{aligned}$$

with

$$\begin{aligned} \mathbb{E}(g_3^{\Phi}(\varphi, s_i)) &= 1 - \mathbb{E}\left(\frac{\mathsf{max}_{a\in\mathsf{dom}(A)}(|\sigma_{A=a}(s_i)|)}{|s_i|}\right) \\ &= 1 - \mathsf{max}_{a\in\mathsf{dom}(A)}(P(a \in s_i[A])) \\ &= 1 - \frac{\mathsf{max}_{a\in\mathsf{dom}(A)}(|\sigma_{A=a}(r_i)|)}{|r_i|} = g_3^{\Phi}(\varphi, r_i) \end{aligned}$$

By applying the Hoeffding's inequality, we obtain: $p(|\hat{g}_3^{\Phi} - g_3^{\Phi}| \le \epsilon) \ge 2 \cdot e^{-2m\epsilon^2} = \delta$ ■

Appendix D

Equivalence between the g_3-error and the Bayes error

Consider a target A and a set of feature X. The Bayes error rate (or irreducible error) is the lowest possible error rate for any classifier predicting A from X. We use the form presented in [JWHT13], it corresponds to:

$$BE = 1 - \mathbb{E}(\max_{a \in \mathrm{dom}(A)}(P(A = a \mid X)))$$

Classic relational algebra operators are considered to be known by the reader [LL12]. Consider a relation r defined over a schema R such that $X \cup A \subseteq R$. The tuples from this relation are i.i.d.. In the following, we prove that $g_3(X \to A, r)$ with crisp equality is equivalent to the Bayes error when the size of r tends to infinity. For completeness, we recall the formula for $g_3(X \to A, r)$ [KM95]:

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

First, we have:

$$\begin{aligned}
\max\{|s| \mid s \subseteq r, s \models X \to A\} &= \sum_{r_i \in \Omega_X(r)} \max\{|s| \mid s \subseteq r_i, s \models X \to A\} \\
&= \sum_{r_i \in \Omega_X(r)} \max_{a \in \mathrm{dom}(A)} (|\sigma_{A=a}(r_i)|)
\end{aligned}$$

Thus we have:

$$\begin{aligned}
g_3(X \rightarrow A, r) &= 1 - \sum_{r_i \in \Omega_X(r)} \frac{\mathsf{max}_{a \in \mathsf{dom}(A)} \left(|\sigma_{A=a}(r_i)|\right)}{|r|} \\
&= 1 - \sum_{r_i \in \Omega_X(r)} \frac{|r_i|}{|r|} \cdot \mathsf{max}_{a \in \mathsf{dom}(A)} \left(\frac{|\sigma_{A=a}(r_i)|}{|r_i|}\right) \\
&= 1 - \sum_{r_i \in \Omega_X(r)} P(X = x) \cdot \mathsf{max}_{a \in \mathsf{dom}(A)} \left(P(A = a \mid X = x)\right) \\
&= 1 - \mathbb{E}(\mathsf{max}_{a \in \mathsf{dom}(A)}(P(A = a \mid X))) \\
&= BE
\end{aligned}$$