A minimal base or a direct base? That is the question!

Jaume Baixeries
Computer Science Department.
Universitat Politècnica de Catalunya
Jordi Girona, 1-3, Barcelona, Catalonia
Amedeo Napoli
Université de Lorraine, CNRS, LORIA
54000 Nancy
France

April 19, 2024

Abstract

In this paper we revisit the problem of computing the closure of a set of attributes, given a set of Armstrong dependencies. This problem is of main interest in logics, in the relational database model, in lattice theory and in Formal Concept Analysis as well. We consider here three main closure algorithms, namely Closure, LinClosure and WildClosure, which are combined with implication bases which may have different characteristics, among which being "minimal", e.g., the Duquenne-Guigues Basis, and being "direct", e.g., the Canonical-Direct Unit Basis and the D-basis. The impacts of minimality and directness on the closure algorithms are then deeply studied also experimentally. The results are extensively analyzed and propose a different and fresh look at computing the closure of a set of attributes.

This paper has been submitted to the *International Journal of Approximate Reasoning*.

Keywords: Functional dependencies, Implications, Horn Clauses, Dependency Covers, Closure.

0.1 Introduction

In this paper, we are interested in analyzing different covers or bases of dependencies, their characteristics, the way they are computed and the related efficiency. A dependency $X \to Y$ can be read as X *implies* Y and follows the so-called Armstrong axioms [5]. Dependencies are "first class citizens" in different fields of Computer Science, e.g., Horn clauses in logics, functional dependencies in the relational database model, implications in Formal Concept Analysis (FCA).

This paper is a follow-up of [8] where we studied three different covers, namely the minimal cover in relational database theory [24], the Canonical-Direct Unit Basis in lattice theory [12], and the Duquenne-Guigues Basis aka canonical basis in FCA [19]. These covers are introduced and characterized in many different textbooks, e.g., in database theory [24, 25, 1], in logics [13], in lattice theory [12], and in FCA [18, 17]. Moreover, Marcel Wild in [31] proposes an extensive and major study about implication bases and the relations existing between the different fields in which they are used.

The Duquenne-Guigues Basis has become the implication basis of reference in FCA while the canonical direct basis is of first importance in database theory [26]. In particular, authors in [9, 10] are interested in the computation of the Duquenne-Guigues Basis w.r.t. three closure algorithms, namely Closure, LinClosure, and WildClosure. In this paper we follow these tracks and we extend this seminal work in several directions, as we analyze not only the Duquenne-Guigues Basis but as well the Canonical-Direct Unit Basis and the D-basis [3]. In particular, we try to characterize the behaviors of several combinations of algorithms and to evaluate the importance for a cover of being minimal or direct.

As this will be made more precise farther, the construction of a cover depends on computing the closure $\text{closure}_\Sigma(X)$ of a set of attributes X w.r.t. a set of dependencies Σ thanks to the Armstrong axioms. Moreover, given a set of dependencies Σ, there may exist different sets of dependencies that are *equivalent* modulo Armstrong axioms. Then two extreme cases for covers can be considered, (i) a cover is *minimal* when it contains a minimal number of dependencies, i.e., minimal in order to maintain the equivalence modulo Armstrong axioms, (ii) a cover is *direct* if only one pass over the set Σ is sufficient to compute $\text{closure}_\Sigma(X)$ for any set of attributes X. For example, the Duquenne-Guigues Basis is minimal while the Canonical-Direct Unit Basis and the D-basis are direct.

To decide what should be the characteristics of the set of dependencies to be used to perform the computation of $\mathrm{closure}_\Sigma(X)$ for a set of attributes X remains an important problem because the number of dependencies that may hold in a relatively small dataset can be huge, and because costly operations are applied to sets of dependencies. Then the debate can be stated in the following terms: is it better to have a cover with a smaller set of dependencies that may require more than one pass to compute a closure, or to have a larger cover ensuring that only one pass is required to compute the closure? To be complete, the question of the algorithm computing $\mathrm{closure}_\Sigma(X)$ should also be raised.

The first well-known algorithm to compute the closure of a set of attributes w.r.t. a set of dependencies is the Closure Algorithm, which has a quadratic cost w.r.t. the size of the input, i.e., Σ. The LinClosure Algorithm is an improvement of Closure whose cost is not quadratic but lineal. Finally, the WildClosure Algorithm is a subsequent improvement of Closure Algorithm with the same complexity.

Since the asymptotic complexity of LinClosure is lineal w.r.t. size of the input set of dependencies, it would be obvious that using a minimal basis would be always the more efficient choice in terms of runtime. However, in practical terms, in some experiments such as those presented in [9, 10], LinClosure does not outperform Closure in a systematic way. In addition, the question of checking whether it is better to use a direct basis (e.g., Canonical-Direct Unit Basis) or a minimal basis (e.g., Duquenne-Guigues Basis) has not yet been fully explored. For example, the minimality of an implication basis has an effective impact on a process such as attribute exploration and its application to knowledge engineering, see e.g, [7, 6, 29, 27]. In addition, the fact that an implication basis is direct received a lot of attention in lattice theory [12, 3, 2] and in FCA [18, 17, 23], while this characteristic is ignored in database theory even if the Canonical-Direct Unit Basis is the implication basis of reference. Accordingly, the question that we address and discuss in this paper is the following: regardless of the hypothetical reasons why a direct basis is preferred in database theory instead of a minimal basis, what can be the best choice to effectively compute $\mathrm{closure}_\Sigma(X)$?

The remaining of this paper is organized as follows. In Section 0.2 we introduce the basic definitions useful in this paper. In Section 0.3 we make precise and detail three algorithms for computing a closure, namely Closure, LinClosure, and WildClosure. Then in Section 0.4 we present the characteristics of bases of dependencies while in Section 0.5 we analyze the possible impacts of using a direct basis when computing a closure. Finally, we propose a series of experiments in Section 0.6 and we discuss the results in Section 0.6.5 which are not necessarily the ones that could be expected.

0.2 Definitions

In this section we introduce the definitions used in this paper. Although in most of the cases we provide a single reference, namely [14], these definitions can be found as well in many different textbooks and papers related to the database theory, logics, and FCA. All along this paper, we consider a tabular dataset whose column labels form the *set of attributes* $\mathcal{U}$, which is the set of interest in the following. The row labels of the dataset determine the transactions or the objects whose descriptions are given by the columns.

Given $X, Y \subseteq \mathcal{U}$, the fact that a dependency $X \to Y$ is *valid* or *true* depends on the kind of dependency at hand. For example, an instance in which a Horn clause is true is a set of models, while an instance in which a functional dependency is valid or holds is a set of rows in a many-valued tabular dataset. Moreover, an instance where an implication is true in a formal context is a given set of objects. Since in this paper we only focus on the reasoning based on the Armstrong axioms, the context of the dependencies is not relevant.

Then, the dependency $X \to Y$ *holds* should be understood as $X \to Y$ holds for all the instances where it is valid or true. In addition, "If $X \to Y$ holds, then $XZ \to YZ$ holds" can be rephrased as "In any instance in which $X \to Y$ is valid, the dependency $XZ \to YZ$ is valid as well".

Definition 0.2.1 ([14]) *Given set of attributes $\mathcal{U}$, for any $X, Y, Z \subseteq \mathcal{U}$, the Armstrong axioms are:*

1. ***Reflexivity**: If $Y \subseteq X$, then $X \to Y$ holds.*
2. ***Augmentation**. If $X \to Y$ holds, then $XZ \to YZ$ holds.*
3. ***Transitivity**. If $X \to Y$ and $Y \to Z$ hold, then $X \to Z$ holds.*

The Armstrong axioms allow us to define the closure of a set of dependencies as the iterative application of these axioms over a set of dependencies.

Definition 0.2.2 ([14]) *Σ^+ denotes the closure of a set of dependencies Σ and can be constructed thanks to the iterative application of the Armstrong axioms over Σ.*

This iterative application terminates when no new dependency can be added, and it is finite. Therefore, Σ^+ contains the largest set of dependencies that hold in all instances in which all the dependencies in Σ hold.

The closure of a set of dependencies induces the definition of the cover of such a set of dependencies.

Definition 0.2.3 ([14]) *The* ***cover*** *or* ***basis*** *of a set of dependencies Σ is any set Σ' such that $\Sigma'^+ = \Sigma^+$.*

We define now the *closure* of a set of attributes $X \subseteq \mathcal{U}$ with respect to a set of dependencies Σ.

Definition 0.2.4 ([14]) *The closure of X with respect to a set of dependencies Σ is*

$$\mathrm{closure}_\Sigma(X) = X \cup \{\, Y \mid X \to Y \in \Sigma^+ \,\}$$

i.e., $\mathrm{closure}_\Sigma(X)$ *is the largest set of attributes Y such that $X \to Y$ can be derived by the iterative application of the Armstrong axioms over the set Σ.*

The closure operation returns the largest set of attributes such that $\Sigma \models X \to \mathrm{closure}_\Sigma(X)$. Therefore, the implication problem $\Sigma \models X \to Y$ boils down to testing whether $Y \subseteq \mathrm{closure}_\Sigma(X)$ (see Section 4 in [11]).

Now we introduce two main characteristics of a cover, being **direct** and being **minimal**. Recall that a main debate in this paper is to check the performance of a direct basis compared to the performance of a minimal basis when computing $\mathrm{closure}_\Sigma(X)$. The definition of a minimal cover is independent of how $\mathrm{closure}_\Sigma(X)$ is computed:

Definition 0.2.5 *Let Σ be a set of dependencies. We say that Σ_{min} is a* ***minimal basis*** *of Σ iff:*

1. *$\Sigma^+ = \Sigma^+_{min}$.*

2. *Σ_{min} does not include any smaller basis verifying the above property.*

We now give an alternative definition of the closure of a set of attributes, contrasting Definition 0.2.4. Actually, the main reason is that we need a definition allowing to reason on the way the three different algorithms presented in the next section compute such closure.

Definition 0.2.6 ([12]) *Let Σ be a set of dependencies and let $X \subseteq \mathcal{U}$ be a set of attributes. A pass over X w.r.t. Σ is defined as:*

$$\Pi_\Sigma(X) = X \cup \{\, b \mid A \subseteq X \text{ and } A \rightarrow b \in \Sigma \,\}$$

Then, the closure of a set of attributes X can be defined as follows:

Definition 0.2.7 ([12]) *Let Σ be a set of dependencies.*

$$\text{closure}_\Sigma(X) = \Pi_\Sigma(X) \cup \Pi_\Sigma^2(X) \cup \cdots \cup \Pi_\Sigma^{i-1}(X)$$

where $\Pi_\Sigma^i(X) = \Pi_\Sigma(\Pi_\Sigma^{i-1}(X))$.

Thus the computing of $\text{closure}_\Sigma(X)$ relies first on computing $\Pi_\Sigma(X)$, and then, computing $\Pi_\Sigma(\Pi_\Sigma(X))$, and so on, until a fixed point $\Pi_\Sigma^i(X) = \Pi_\Sigma^{i-1}(X)$ is reached. We can proceed now to define a direct basis:

Definition 0.2.8 ([12]) *Let Σ be a set of dependencies. Σ is a* ***direct basis*** *if for all $X \subseteq \mathcal{U}$:*

$$\text{closure}_\Sigma(X) = \Pi_\Sigma(X)$$

Then if we go back to Definition 0.2.6, it comes that Σ is a direct basis if, for all $X \subseteq \mathcal{U}$: $\text{closure}_\Sigma(X) = X \cup \{\, b \mid A \subseteq X \text{ and } A \rightarrow b \in \Sigma \,\}$. This means that only **one single pass** need to be performed over Σ, collecting the set $\{\, b \mid A \subseteq X \text{ and } A \rightarrow b \,\}$ and then, joining it to X in order to compute $\text{closure}_\Sigma(X)$.

The notion of direct basis seems to be original to lattice theory and FCA, but seems to be completely alien to the DB community. We can find references to a *direct basis* in [12] and, earlier, in [18].

0.3 Algorithms Computing the Closure of a Set of Attributes

In this section, we focus on the most well-known algorithms computing the closure of a set of attributes X, namely the Closure, LinClosure, and Wild Closure algorithms.

Function Closure(X, Σ)

Input : A set of attributes $X \subseteq \mathcal{U}$ and a set of implications Σ

Output: closure$_\Sigma(X)$

```
1  stable ← false
2  while not stable do                            // Outer loop
3      stable ← true
4      forall A → B ∈ Σ do                        // Inner loop
5          if A ⊆ X then                          // deps
6              X ← X ∪ B
7              stable ← false
8              Σ ← Σ \ { A → B }
9          end
10     end
11 end
12 return X
```

0.3.1 The Closure Algorithm

Closure is the *classical* algorithm computing closure$_\Sigma(X)$, which is detailed in many textbooks, e.g., in [24, 1]. Here we adapt the version proposed in [17] (Algorithm 14, page 93). In the Closure algorithm, the computing of a given $\Pi_\Sigma(X)$ is performed in lines $4 - 10$, and it iterates the loop in line $2 - 11$ until a fixed point is found. Once a dependency has been processed in lines $5 - 8$, it is removed in line 8.

The complexity of this algorithm is discussed in the related references, and the general consensus is that it is **quadratic** w.r.t. the input (see [8] for more details).

0.3.2 The LinClosure Algorithm

An improved version of Closure is the LinClosure algorithm [11]. This algorithm consists of two parts: a `preparation` part in which the necessary data structures are computed, and the `computation` part in which the computing of closure$_\Sigma(X)$ is performed. In `preparation` two data structures are constructed, the role of which is to ensure that only the dependencies necessary to compute the closure are considered while the other are ignored:

(i) for each attribute say x, the first structure records a pointer to all the dependencies $X \to Y$ such that x appears in the left-hand side X,

(ii) for each dependency $X \to Y$, the second structure includes a counter that records the number of attributes of X already visited during the computing part.

The general idea of the LinClosure algorithm can be checked in examining the two loops in lines $11 - 22$. During the execution of the outer loop, X contains the part of its closure that has been computed so far, i.e., $\Pi^i_\Sigma(X)$. Then, for each attribute in $x \in X$, we decrease the counter of all the dependencies $A \to B$ such that $x \in A$, i.e., $counter[A \to B]$. When line 16 tests positive, it means that $A \subseteq X$ for that particular dependency $A \to B$, and, therefore, B can added to X as part of its closure. In particular, this means that dependencies not containing a subset of X are not "used" as they will always test negative in line 16.

There is a general consensus about the complexity of LinClosure, which is of order $\mathcal{O}(|\Sigma|)$ for both the preparation part and the computation part [11].

Here the complexity of the preparation part is not discussed, which is assumed to be of the same complexity as the rest of the algorithm. One explanation of this fact appears in the pioneering paper [11], page 47 in the second paragraph (this paragraph is adapted to fit names in Algorithm LinClosure):

> *For each attribute in [update], the [outer] loop follows a constant number of steps for each occurrence of that attribute on the left side of an FD in Σ. Similarly, each right side of an FD in Σ is visited at most once in [the outer loop]. Thus [the outer loop] is also $\mathcal{O}(|\Sigma|)$ as is the entire Algorithm.*

Previously, the authors have concluded that the complexity of the preparation part is of order $\mathcal{O}(|\Sigma|)$, as well as the second part, hence the end of the last sentence *"is also $\mathcal{O}(|\Sigma|)$ the entire Algorithm"*.

0.3.3 The WildClosure Algorithm

Below we present a slightly more compact form of the WildClosure algorithm borrowed from [10]. The WildClosure Algorithm [30] aims at ensuring that inside each outer loop all the dependencies $A \to B$ fulfilling the condition $A \subseteq X$ are selected. The algorithm starts with one of the data structures also present in LinClosure: for each attribute say x there is a list recording

Function LinClosure(X, Σ)

Input : A set of attributes $X \subseteq \mathcal{U}$ and a set of implications Σ

Output: closure$_\Sigma(X)$

1 **forall** $A \to B \in \Sigma$ **do** `// Preparation`
2     $count[A \to B] \leftarrow \mid A \mid$
3   **if** $\mid A \mid = 0$ **then**
4         $X \leftarrow X \cup B$
5   **end**
6   **forall** $a \in A$ **do**
7         $list[a] \leftarrow list[a] \cup \{ A \to B \}$
8   **end**
9 **end**
10 $update \leftarrow X$
11 **while** $update \neq \emptyset$ **do** `// Outer loop`
12   choose $m \in update$
13     $update \leftarrow update \setminus \{ m \}$
14   **forall** $A \to B \in list[m]$ **do** `// Inner loop`
15         $count[A \to B] \leftarrow count[A \to B] - 1$
16   **if** $count[A \to B] = 0$ **then** `// deps`
17             $add \leftarrow B \setminus X$
18             $X \leftarrow X \cup add$
19             $update \leftarrow update \cup add$
20   **end**
21   **end**
22 **end**
23 **return** X

all the dependencies $A \to B$ such that x is contained in A. Then, is selects all dependencies $A \to B$ such that $A \subseteq X$ to be processed in the loop in lines $12 - 15$.

The most noticeable and relevant operation of the algorithm is performed in line 11, where it selects all the dependencies $A \to B$ such that $A \subseteq X$. We can check that there is no test in WildClosure algorithm in order to process a dependency: line 12 is a loop over all the dependencies in $\Sigma \setminus \Sigma_1$ without any conditional, unlike line 5 in Closure and line 16 in LinClosure.

This also means that, at each loop between lines 9 and 17, WildClosure algorithm computes $\Pi_\Sigma(X)$. We will see in Section 0.5.3 that, as in the case of LinClosure, this implies some relevant consequences.

Regarding the complexity of the algorithm, the author underlines in [30] that:

> *Algorithm 1 [Wild Closure] has complexity $\mathcal{O}(|\Sigma||\mathcal{U}|^2)$, which is actually the same as the complexity of Algorithm 0 [Closure]. Yet in practice Algorithm 1 [Wild Closure] takes a fraction of the time of Algorithm 0 [Closure] and also of LinClosure. Philosophy: Doing few set operations with big sets is better than doing many set operations with small sets.*

This apparent paradox between the asymptotic complexity of an algorithm and its real performance is of interest and will be more deeply discussed in Section 0.6.

0.4 Three Bases of Dependencies

In this section we briefly present three bases which will be processed by the three algorithms explained in Section 0.3. Here we consider two direct bases, namely the Canonical-Direct Unit Basis and the D-basis, and one minimal basis, namely the Duquenne-Guigues Basis.

0.4.1 The Canonical-Direct Unit Basis

The Canonical-Direct Unit Basis is deeply studied in [12] where different equivalent characterizations are examined. This basis can be characterized as follows:

1. All the dependencies in Σ must have one single attribute in the right-hand side ("unit basis").

2. Σ is left-reduced.

   A dependency $X \to y$ is left-reduced if, for all $X_i \subseteq X$, the dependencies $X_i \to y$ do not hold. Stated differently, all the left-hand sides of the dependencies lying in Σ are minimal.

Function WildClosure(X, Σ)

```
Input  : A set of attributes X ⊆ 𝒰 and a set of implications Σ
Output: closure_Σ(X)
forall m ∈ 𝒰 do                                   // Preparation
    forall A → B ∈ Σ do
        if m ∈ A then
            list[m] = list[m] ∪ { A → B }
        end
    end
end
stable ← false
while not stable do                               // Outer loop
    stable ← true
    Σ_1 ← ⋃_{m∈𝒰∖X} list[m]
    forall A → B ∈ Σ ∖ Σ_1 do                     // Inner loop / deps
        X ← X ∪ B
        stable ← false
    end
    Σ ← Σ_1
end
return X
```

The Canonical-Direct Unit Basis may contain some redundancy. For example, while the basis $\Sigma = \{ a \to b, b \to c, a \to c \}$ is left-reduced, the dependency $a \to c$ is redundant because $\Sigma^+ = (\Sigma \setminus \{ a \to c \})^+$. The Canonical-Direct Unit Basis is not necessarily minimal, but it is **direct** (as per Definition 0.2.8).

0.4.2 The D-basis

The D-basis is introduced in [3] as a subset of the Canonical-Direct Unit Basis. Actually, this basis can be constructed by removing some dependencies from a Canonical-Direct Unit Basis. The formal definition of a D-basis is based on two properties of a cover, namely (i) the *proper cover* of an attribute $x \in \mathcal{U}$, and (ii) the *minimality* of a cover. The definitions used hereafter in this subsection are borrowed from [22].

Let (M, φ) be a closure system, which in our case corresponds to $(\mathcal{U}, \text{closure}_\Sigma)$. Let us introduce the operator $\varphi^*(X) = \bigcup_{x \in X} \varphi(x)$.

Actually, the $\varphi^*(X)$ operator joins all the closures of elements $x \in X$. It can be checked that $\varphi^*(X)$ is a closure operator and that $\varphi^*(X) \subseteq \varphi(X)$, deriving from the fact that a closure operator is increasing.

Definition 0.4.1 *A set $X \subseteq \mathcal{U}$ is a **proper cover** of $x \in \mathcal{U}$ if $x \in \varphi(X) \setminus \varphi^*(X)$.*

Definition 0.4.1 allows to define a minimality relation between all proper covers of $x \in \mathcal{U}$.

Definition 0.4.2 *A proper cover Y for x is minimal if for any other proper cover Z for x, $Z \subseteq \varphi^*(Y)$ implies $Y \subseteq Z$.*

Based on this definition of minimality, a D-basis can be defined as follows:

Definition 0.4.3 *A D-basis is formed by the following two sets of dependencies:*

1. $\{\, y \to x \mid x \in \varphi(y) \setminus y$ *and* $y \in \mathcal{U} \,\}$,
2. $\{\, X \to y \mid X$ *is a minimal proper cover for* $x \,\}$.

Is the D-basis a direct basis? The authors write in [3]: *While the D-basis is not direct in this meaning of this term [this refers to Definition 0.2.8], the closures can still be computed in a single iteration of the basis, provided the basis was put in a specific order prior to computation .*

In particular, this is why the D-basis is called "*ordered* direct implication basis". Contrasting the Canonical-Direct Unit Basis, here the order is relevant (see for example [31]).

0.4.3 The Duquenne-Guigues Basis

The Duquenne-Guigues Basis [19, 18], also called the *Canonical Basis* in the FCA community, is the basis relying on pseudo-closed sets [18, 17]. This basis is also presented in [24], where it is called the *Minimum Cover*. Below we first recall the definition of a pseudo-closed set of attributes and then the definition of the Duquenne-Guigues Basis.

Definition 0.4.4 *Let Σ be a set of dependencies, and $\mathcal{U}$ the related set of attributes. $X \subseteq \mathcal{U}$ is* ***pseudo-closed*** *if:*

1. *$X \neq \text{closure}_\Sigma(X)$, i.e., X is not closed.*

2. *If $Y \subset X$ is a proper subset of X and is pseudo-closed, then $\text{closure}_\Sigma(Y) \subseteq X$.*

Definition 0.4.5 *The* ***Duquenne-Guigues Basis*** *of a set of dependencies Σ is defined as:*

$$\{\, X \to \text{closure}_\Sigma(X) \mid X \subseteq \mathcal{U} \textit{ and } X \textit{ pseudo-closed}\}$$

The Duquenne-Guigues Basis is not direct, but it is **minimal** and non-redundant.

0.5 Impact of a Direct Basis on the Three Algorithms

In this section we discuss the impact of a direct basis on the three algorithms computing a closure presented in Section 0.3. By *impact* we mean the possibility of improving the performance of those algorithms by taking advantage of the fact that a basis is direct. We explain, for each algorithm, what changes can be performed depending on Σ being a Canonical-Direct Unit Basis or a D-basis.

The discussion in this section is centered about the cases in which we can safely perform one single outer pass in the three previous algorithms. In the following Subsection 0.5.1, we discuss how two different kinds of direct bases appeared (see Definition 0.2.8), and the possible improvements in the three algorithms. In principle, a direct base contains enough information to ensure that one single pass is needed for computing a closure. Then the idea is to ensure that the outer loop of all three algorithms is performed just once. and prevent it from performing a second pass that would not modify the closure already calculated. One could argue that this potential improvement is not necessary since the basis is direct and at most two passes of the outer loop are necessary: one to effectively compute $\text{closure}_\Sigma(X)$ and a second pass to check that no more dependencies are needed to be processed. This is true, but yet, we find relevant to avoid this second loop in all cases, whenever possible.

0.5.1 Impact on Closure

How can we optimize Closure when the input is a Canonical-Direct Unit Basis? We present the algorithm Optimized Closure:

Function OptimizedClosure(X, Σ)

Input : A set of attributes $X \subseteq \mathcal{U}$ and a Canonical-Direct Unit Basis Σ

Output: $\mathrm{closure}_\Sigma(X)$

1 $\mathit{result} \leftarrow \emptyset$
2 **forall** $A \rightarrow B \in \Sigma$ **do**
3   **if** $A \subseteq X$ **then**
4         $\mathit{result} \leftarrow \mathit{result} \cup B$
5         $\Sigma \leftarrow \Sigma \setminus \{ A \rightarrow B \}$
6   **end**
7 **end**
8 **return** $X \cup \mathit{result}$

This algorithm differs from Closure in two things: (1) it only performs one pass over Σ, and this is why the outer loop has been removed, and (2) it accumulates the result in the variable *result*, and it does not add anything to X every time a dependency is processed. This last step is necessary in order to prevent the processing of unnecessary dependencies, as the following simple example shows:

Example 0.5.1 *Let us suppose that we have the following Canonical-Direct Unit Basis:* $\Sigma = \{ a \rightarrow b, b \rightarrow c, a \rightarrow c \}$. *If we want to compute* $\mathrm{closure}_\Sigma(a)$, *algorithm Closure would first start with* $X = a$. *Then, in line* 6 *it would execute* $X = X \cup b$ *(because of* $a \rightarrow b$*), thus* $X = ab$. *Because of* $b \rightarrow c$ *it would add* c *to* X, *and, finally, because of* $a \rightarrow c$ *it would also add* c *to* X. *This means that Closure has used all dependencies in* Σ.

However, Optimized Closure would also start with $X = a$, *but then, it would process* $a \rightarrow b$ *and accumulate* b *to the variable* result, *it would* ***not*** *process* $b \rightarrow c$ *and, finally, it would process* $a \rightarrow c$ *and add* c *to* result. *Finally, it would return* $X \cup \mathit{result} = abc = clo(a)$, *but only processing 2 dependencies instead of 3.*

It is straightforward to check that the loop of Optimized Closure between lines 2 and 7 computes $\mathit{result} = \bigcup \{ B \mid A \rightarrow B \in \Sigma \text{ and } A \subseteq X \}$, and that in line 8 $X \cup \bigcup\{\, B \mid A \to B \in \Sigma \text{ and } A \subseteq X \,\}$ is returned, which is the definition of a direct basis as per Definition 0.2.8.

Does Optimized Closure compute correctly closure_Σ if Σ is a D-basis? The answer is no and we present a counterexample.

Example 0.5.2 *Let us suppose that we have the following reduced and clarified formal context:*

$\mathbb{K}$	a	b	c	d
o_1	×		×	
o_2			×	×
o_3	×			
o_4		×		

The D-basis for this context is: $\Sigma = \{\, d \to c, bc \to a, ad \to b, ab \to c, bc \to d, ab \to d \,\}$*. defined on the set of attributes* $\mathcal{U} = \{\, a, b, c, d \,\}$*. Let us suppose that we want to compute* $\text{closure}_\Sigma(bd)$ *with Optimized Closure. The algorithm will end in line* 7 *with* $result = \{\, a, b, c \,\}$*, which is not the right result. This is because the variable* X *is not updated every time the test in line* 3 *is true. In fact, this disadvantage appears also when we try to improve LinClosure and WildClosure, and it does not appear when processing the Canonical-Direct Unit Basis because the latter also contains the dependency* $bd \to a$*.*

Therefore, **we cannot use Optimized Closure when Σ is a D-basis**. However, according to [3]: *In contrast [to the Duquenne-Guigues Basis], the computation of the closure of any input set, by the D-basis or canonical direct unit basis [Canonical-Direct Unit Basis], is done simply in one loop of this algorithm [Closure].* This means that Closure can be *optimized* not by performing the two improvements implemented in Optimized Closure but just the first one: ensuring that only one pass of Σ is performed. We do this by simply adding the line *stable* $\leftarrow$ *true* between lines 10 and 11 in the original Closure algorithm.

0.5.2 Impact on LinClosure

Can we apply the same two optimizations implemented in Optimized Closure to LinClosure? Compared to Closure, the outer loop of LinClosure scans not per dependency but per attribute: once the left-hand side of a dependency is checked as a subset of X, then the right-hand side is added to $\text{closure}_\Sigma(X)$. This means that performing just one single outer pass may not yield the correct computation of closure_Σ. But the second improvement, i.e., not accumulating the result in a variable different from X in the inner loop, may be implemented, as we will show it here after. Recall that one outer loop of Closure is equivalent to the computing of $\Pi^i_\Sigma(X)$, i.e., when Σ is a direct base, the computing may stop after one outer loop. By contrast, this is not the case in LinClosure, because at the end of the outer loop (line 22), the computing of $\Pi^i_\Sigma(X)$ may not be finished. We do this by **removing** A way to speed up LinClosure when Σ is a direct basis is to **remove** line 19, i.e., `update` $\leftarrow$ `update` $\cup$ `add`. The idea is, when Σ is a Canonical-Direct Unit Basis, to ensure that only those dependencies $A \to B$ such that $A \subseteq X$ test positive in line 16, and the removal of line 19 ensures this –as we will see– and prevents to potentially process dependencies whose left-hand side are not included in X but are included in $\text{closure}_\Sigma(X)$. Actually line 19 adds attributes to variable `update` that belong to some right-hand sides of dependencies processed in lines 16 to 20 that are not in X (line 17). But when Σ is a Canonical-Direct Unit Basis, only the dependencies whose left-hand side is contained in X should be processed, i.e., $X \bigcup \{\, B \mid A \subseteq X \text{ and } A \to B \in \Sigma\,\}$, but not the dependencies lying in $X \cup \Pi^i_\Sigma$.

We now prove that the removal of line 19 in LinClosure when Σ is a Canonical-Direct Unit Basis effectively computes $\text{closure}_\Sigma(X)$.

Proposition 0.5.1 *If LinClosure is modified by removing line* 19 *and if* Σ *is a Canonical-Direct Unit Basis, then LinClosure effectively computes* $\text{closure}_\Sigma(X)$.

Proof 0.5.1 *We should ensure that the right-hand sides of all the dependencies whose left-hand side is **contained** in X are added to* $\text{closure}_\Sigma(X)$.

Firstly, we check that all dependencies whose left-hand side is a subset of X are processed in lines 16 *to* 20*. Let us consider a dependency $A \to B$ in Σ such that $A \subseteq X$. The variable update contains all the attributes of X as indicated in line* 10*. In line* 12 *and* 13 *an attribute is picked in update (i.e., X) and then removed from update. It should be noticed that update is modified only in line* 13 *since line* 19 *is supposed to be removed. The outer loop in lines* $11-22$ *ensures that all the attributes in update (X) are processed one by one at each loop. In the inner loop, lines* $14-21$*, LinClosure marks all dependencies whose left-hand side contains at least one attribute* *in update (line 15). Since all attributes in X are processed in the outer loop and since $A \subseteq X$, this means that count$[A \to B]$ goes necessarily to 0, and therefore, line 17 is executed, i.e., the right-hand side of $A \to B$ is added to* $\text{closure}_\Sigma(X)$.

At the end of the algorithm, variable X contains $X \cup \bigcup\{\, B \mid A \subseteq X \text{ and } A \to B \in \Sigma\,\}$. Since Σ is a Canonical-Direct Unit Basis, Definition 0.2.8 concludes this proof.

However, if Σ is a D-basis, then, LinClosure may not yield a correct result, as shown in the next counterexample.

Example 0.5.3 *We continue with Example 0.5.2, where the D-basis $\Sigma = \{\, d \to c, bc \to a, ad \to b, ab \to c, bc \to d, ab \to d\,\}$. The computation of* $\text{closure}_\Sigma(bd)$ *goes as follows: in line 12, $m = b$, and in the first pass of the inner loop (lines $14-21$) the counters of $bc \to a, ab \to c, bc \to d$ and $ab \to d$ are decremented to 1, but the test in line 16 is negative in all these cases. In the second loop of the outer loop we have that $m = d$, and in the inner loop the counter of $d \to c$ is decremented to 0, which means that the attribute c is added to the variable update –recall that line 19 is assumed to be* ***removed****– and the counter of $ad \to b$ is decremented to 1. The returned value would be, then, abc, which is not the correct answer for* $\text{closure}_\Sigma(bd)$.

This means that we can remove line 19 from LinClosure when Σ is a Canonical-Direct Unit Basis, but **this is not possible when Σ is a D-basis**.

0.5.3 Impact on the WildClosure Algorithm

The structures of LinClosure and WildClosure are very similar. The drawback that WildClosure tries to solve w.r.t. LinClosure is to ensure that at each pass of the outer loop all the dependencies $A \to B$ such that $A \subseteq X$ are *directly* processed, i.e., it is not necessary to perform the test line 16 in LinClosure or the containment test line 5 in Closure.

As previously, we want to ensure that WildClosure algorithm performs only one single pass of the outer loop. Contrasting LinClosure, the outer loop in WildClosure is equivalent to the outer loop of Closure, making things easier. Then the improvement consists in adding the instruction `stable` $\leftarrow$ `true` between lines 16 and 17.

Proposition 0.5.2 *If the instruction* `stable` $\leftarrow$ `true` *is added between lines* 16 *and* 17 *in WildClosure algorithm and if* Σ *is a Canonical-Direct Unit Basis, then, WildClosure computes* $\mathrm{closure}_\Sigma(X)$*.*

Proof 0.5.2 *The key line of WildClosure algorithm is line* 11*, where are selected all the dependencies whose left-hand side contains an attribute not present in* X*. Actually, if there is an attribute* $a \in A$ *in* $A \to B$ *such that* $a \notin X$*, then it is impossible that* $A \subseteq X$*. Therefore, line* 11 *of WildClosure ensures that all the dependencies used in the inner loop in lines* $12 - 15$ *are such that* $\{ B \mid A \subseteq X \text{ and } A \to B \in \Sigma \}$*. Consequently, the right-hand sides of these dependencies are added to* X *in line* 13 *and thus WildClosure algorithm computes* $X \cup \bigcup \{ B \mid A \subseteq X \text{ and } A \to B \in \Sigma \}$*. Since* Σ *is assumed to be a Canonical-Direct Unit Basis, Definition 0.2.8 concludes this proof.*

The answer to the question "what happens if the base of dependencies is a D-basis?" is again negative as in the case of LinClosure. The counterexample presented for Closure and LinClosure can be reused here.

Example 0.5.4 *Let us consider the same set of dependencies as in Example 0.5.2, i.e.,* $\Sigma = \{ d \to c, bc \to a, ad \to b, ab \to c, bc \to d, ab \to d \}$*, and let us compute* $\mathrm{closure}_\Sigma(bd)$ *with WildClosure algorithm.*

Let us compute $\mathrm{closure}_\Sigma(bd)$*, which is abcd, assuming that only one pass of the outer loop is necessary. In line* 11*,* Σ_1 *contains the following dependencies:* $\Sigma_1 = \{ bc \to a, ad \to b, ab \to c, bc \to d, ab \to d \}$ *implying that* $\Sigma \setminus \Sigma_1 = \{ d \to c \}$*. Then, at the end of the first outer loop, it comes that* $X = \{ b, c, d \} \neq \mathrm{closure}_\Sigma(X) = \{ a, b, c, d \}$*.*

As in the case of LinClosure, we can improve WildClosure if Σ is a Canonical-Direct Unit Basis, but **it is not possible if Σ is a D-basis**.

0.6 Experiments

In this section, we first explain in Section 0.6.1 the previous experiments related to the comparison of the different bases and algorithms used to compute $\mathrm{closure}_\Sigma$. In Section 0.6.2 we make clear the goals of our experiments and how they generalize that previous work. In Sections 0.6.3 and 0.6.4 we present the analyzed datasets and some technicalities. Finally, in Section 0.6.5 and the followings we show and comment the obtained results.

0.6.1 Experiments: Previous Work

Although many papers and textbooks discuss both Closure and LinClosure algorithms, we were not able to find much work devoted to the comparison of the evaluation of their performance. We guess that this is related to the consensus stating that LinClosure being a linear and Closure being a quadratic algorithm, this implies that the former is preferable in all cases. Some papers compare the performance of both algorithms indirectly, as in [15], where the authors compare different algorithms for eliminating redundancy in sets of functional dependencies with different algorithms combining both Closure and LinClosure. We have also realized that although there are alternatives to the three algorithms that we compare here, they have not managed to become as popular as Closure and, in fact, we also should say that WildClosure algorithm has not become a popular alternative. Other alternatives computing closure_Σ are proposed in [28] (see Algorithm 3.2 that is based on an attribute-fd graph), and in [4]. In the latter an original algorithm is based on a set of axioms different of Armstrong's. Authors also performed an empirical comparison of their approach which outperforms LinClosure w.r.t. computation time by a significant factor in the majority of cases.

In the FCA community there are many different papers that are related indirectly to the computation of closure_Σ and, hence, to the performance of Closure, LinClosure and WildClosure Algorithms. These papers mostly deal with the computation of the Duquenne-Guigues Basis with Closure, or improved versions, e.g., [20], [21], [17], and [23].

Finally, two other papers have directly tested and compared the three algorithms dealing with the computation of closure_Σ using different bases, namely [10] and [3]. Below we review the experiments performed in these two papers. as they are close to the present experiments.

The first set of experiments in [10] compares the performance of Closure, LinClosure and WildClosure for the computation of closure_Σ with the Duquenne-Guigues Basis. The results show that *Algorithm 1 [Closure] was the fastest and Algorithm 2 [LinClosure] was the slowest*, which could be explained by the cost of the initialization step of LinClosure. WildClosure ranks between Closure and LinClosure considering both synthetic and real datasets.

Two different data sources are used: random formal contexts and real datasets from the UCI repository [1]. In both cases, they extract the Duquenne-

[1] `https://archive.ics.uci.edu/`

Guigues Basis, which is used as an input to compute closure$_\Sigma$ with all three algorithms. According to the authors, the results show that *Algorithm 1 [Closure] was the fastest and Algorithm 2 [LinClosure] was the slowest, even though it has the best asymptotic complexity*. WildClosure ranks between Closure and LinClosure in both synthetic and real datasets. The authors explain that the reason why Closure outperforms LinClosure may be *partly explained by the large overhead of the initialization step*.

In another set of experiments, the authors fix a given number of dependencies (1000) and compute closure$_\Sigma(X)$ with random X, where the size of the set of attributes varies from $5,000$ to $100,000$. In this case again, the execution time of Closure remains practically constant w.r.t. an increasing number of attributes, whereas the time grows linearly in both LinClosure and WildClosure. The authors argue that *[T]he reason is that Algorithm 1 [Closure] is quadratic in the number of implications, which is constant in this experiment*.

Here a comment is of order: the asymptotic complexity of Closure is quadratic w.r.t. the size of Σ, but also is multiplied by the size of the attribute set $\mathcal{U}$ (see [8] Section 4.1 for a more detailed explanation). For instance, in [17] this complexity is $min(|\mathcal{U}| \times |\Sigma|, |\Sigma|^2)$.

In [3], authors perform two types of experiments. The first one consists in testing the performance of Closure, *forward chaining algorithm* –an algorithm used in Logics to check the satisfiability of Horn formulas [16]–, and WildClosure. They generate different D-basis including 5 to 8 attributes, and compare the *execution time* of each algorithm. It appears that Closure outperformed WildClosure in all these tests with a small number of attributes, but the results also show that the difference in performance between both algorithms decreased when the number of attributes increases. One important remark is that the authors ensured that Closure performed only one single pass of Σ. In another experiment authors generate different random closure systems and then compute the Duquenne-Guigues Basis and the D-basis, and compare the performance of both bases when computing closure$_\Sigma$ using Closure.

The results show that D-basis checks less dependencies than Duquenne-Guigues Basis on the average in experiments where the number of attributes is 6 and 7.

0.6.2 Goals of the Present Experiments

We take as a departure point the experiments performed in both [10] and [3]. Due to their specific objectives, these papers do not perform a full comparison of the three algorithms w.r.t. the three bases, about execution time and number of processed dependencies. In addition, from our standpoint, there is a metric that is relevant and that should be taken into account, namely, the cost of the attribute operations. Thus, this paper aims at generalizing these former experiments and proposes the following novelties:

1. Comparing the performance of all three possible combinations of the three algorithms computing closure_Σ, i.e., Closure, LinClosure and WildClosure, with the three different bases, i.e., Canonical-Direct Unit Basis, D-basis and Duquenne-Guigues Basis.

2. Comparing the three involved algorithms when Σ is a direct basis w.r.t. the improvements discussed in Section 0.5.

3. Analyzing the results w.r.t. different metrics, i.e., execution time, number for processed dependencies, and number of attribute operations.

4. Performing experiments over a large set of real data, and as well synthetic datasets.

0.6.3 Datasets

We divide the datasets that are analyzed into three different categories.

Real datasets (`real`). We have analyzed a group of 19 datasets from the *UC Irvine Machine Learning Repository* [2]. These datasets (Table 1) have been processed in order to obtain, for each of them, a reduced and clarified formal context. For all these datasets, we have computed the closure of all possible sets of attributes, i.e., $2^{|\mathcal{U}|}$ sets of attributes.

Big Real Datasets (`big`). From the same UCI repository we have analyzed 5 datasets, also processed into reduced and clarified formal contexts (Table 2). The difference with the previous datasets relies on the large number of attributes and of objects. We have not been able to compute the closure of all possible sets of attributes. Instead, for each dataset we have computed the closure of a range of attribute sets, as explained in .1.

[2]https://archive.ics.uci.edu/

Dataset	$\|G\|$	$\|M\|$	$\|\Sigma_{cdb}\|$	$\|\Sigma_{dBasis}\|$	$\|\Sigma_{DG}\|$	Dataset	$\|G\|$	$\|M\|$	$\|\Sigma_{cdb}\|$	$\|\Sigma_{dBasis}\|$	$\|\Sigma_{DG}\|$
abalone	240	9	137	137	100	house-votes-84	25523	17	53	53	53
adult	9553	14	46	46	46	letter	119607	17	61	61	61
breast-cancer-wisconsin	837	11	46	46	43	mushroom	19655	22	3583	3583	1721
bridges	643	12	126	125	88	page-blocks	202	11	135	135	69
congress	25523	17	53	53	53	pen-recognition	22126	17	30463	30463	15885
echocardiogram	291	12	526	526	269	tic-tac-toe	1002	10	18	18	18
ecoli	71	8	46	46	46	waveform	592	22	24002	24002	24002
flights 20 500k	281	12	69	51	49	wine	113	14	1374	1374	1106
glass	104	10	160	160	120	zoo	1119	18	284	283	163
hepatitis	6071	20	8250	8250	2730						

Table 1: Group of datasets **real** from the UCI Repository with the number of objects $|G|$ and attributes $|M|$ of their reduced and clarified formal contexts. $|\Sigma_{cdb}|$: size of the Canonical-Direct Unit Basis. $|\Sigma_{dBasis}|$: size of the D-basis. $|\Sigma_{DG}|$: size of the Duquenne-Guigues Basis.

Dataset	$\|G\|$	$\|M\|$	$\|\Sigma_{cdb}\|$	$\|\Sigma_{dBasis}\|$	$\|\Sigma_{DG}\|$	Dataset	$\|G\|$	$\|M\|$	$\|\Sigma_{cdb}\|$	$\|\Sigma_{dBasis}\|$	$\|\Sigma_{DG}\|$
automobile	2767	26	4176	4040	1848	flight	1856	19	2473	1533	889
fd-reduced-1k	26	26	7483	5551	5551	soybean	826	21	4606	3752	585
fd-reduced-30	349	26	54363	35445	35445						

Table 2: Big datasets from the UCI Repository with the number of objects $|G|$ and attributes $|M|$ of their reduced and clarified formal contexts. $|\Sigma_{cdb}|$: size of the Canonical-Direct Unit Basis. $|\Sigma_{dBasis}|$: size of the D-basis. $|\Sigma_{DG}|$: size of the Duquenne-Guigues Basis.

Synthetic Datasets (`synthetic`). We have also analyzed a group of synthetic formal contexts that have been computed with the combination of all possible values of the parameters shown in Table 3.

Attribute	Range	Step
Objects	8 - 14	1
Attributes	10 - 26	1
Frequency	0.2 - 0.8	0.1

Table 3: Parameters for the computation of **synthetic**. Frequency: parameter of the Bernouilli distribution used to compute 0's and 1's.

Afterwards, all formal contexts have been clarified and reduced, which could, eventually, imply a reduction in their dimensions. For all these datasets, we have computed the closure of all possible sets of attributes, i.e., $2^{|\mathcal{U}|}$ sets of attributes.

0.6.4 Methodology

	Closure	LinClosure	WildClosure
Canonical Direct	Optimized Closure	Improved by removing line 19	Improved by adding `stable` $\leftarrow$ `true`
D-basis	Improved by ensuring one outer loop	No changes	No changes
DG-Basis	No changes	No changes	No changes

Table 4: Combinations of basis and algorithms used in the experiments.

We have used a custom algorithm in order to compute the Canonical-Direct Unit Basis and the Duquenne-Guigues Basis for each dataset. The computation of the D-basis has been performed with the `npar/dbasis` algorithm [3]. The combinations of bases plus algorithms that were tested are given in Table 4. Finally, we added the following counters to all the algorithms (which are also shown in the pseudocodes):

1. **deps** counts the number of times a dependency is *processed*, i.e., is used to compute $\text{closure}_\Sigma(X)$ (line 6 in Closure, line 16 in LinClosure). In WildClosure this counter is equivalent to **inner**.

2. **attributes** counts the number of attributes involved in the different computations performed in each algorithm. In Closure this is the concern of lines 5 and 6, in LinClosure lines $17, 18$ and 19, and in WildClosure line 13. In these cases we exactly count the number of attributes lying in each set involved.

3. **time** counts the number of milliseconds spent in the computation of $\text{closure}_\Sigma(X)$. It should be noticed that this counter **only** counts the milliseconds strictly used for computing closure_Σ each time this function is called.

We have not counted the preparation part of LinClosure in lines $1 - 9$, nor the preparation part in WildClosure in lines $2 - 7$ (which needs to be performed just once). In Closure, line 8, i.e., $\Sigma \leftarrow \Sigma \setminus \{ A \to B \}$, in which a dependency is removed after it has been used, has been implemented with a bitvector indicating whether a dependency has been used or not. Obviously,

[3] `https://gitlab.com/npar/dbasis`

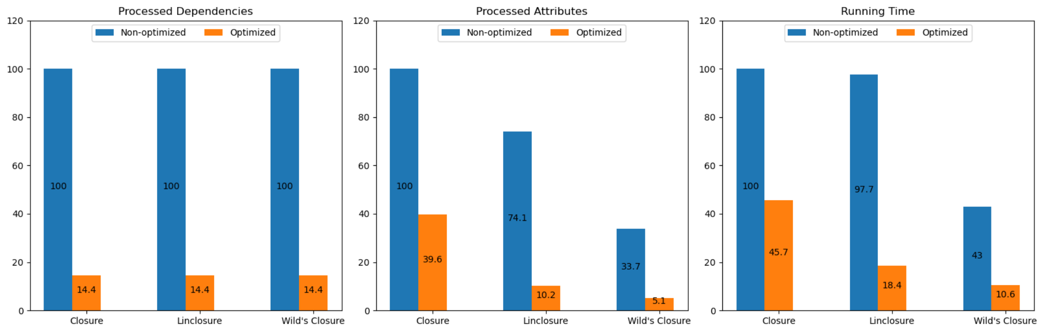

Figure 1: Comparison of the performance of each algorithm w.r.t. their optimized versions when processing the Canonical-Direct Unit Basis in **`real` datasets**. The values have been normalized to the interval (0,100).

after each call to Closure this vector needs to be reset to *true* in all of its values. This has not been counted in the execution time of Closure.

All these decisions were taken in order to be accurate on the counting of execution time for both algorithms.

All tests were executed in the cluster facilities at the High Performance Computing at the UPC [4], which ensures that each execution is performed in an isolated environment with a dedicated CPU and memory. For each dataset, a single program has computed the closures of **all** the combinations <Basis, Algorithm> analyzed here, providing a guarantee that all combinations are computed in the same conditions.

0.6.5 Results

First of all, we consider the following question: *how relevant are the improvements performed on Optimized Closure, LinClosure and WildClosure?* Figures 1, 2 and 3 show that the difference between Closure and Optimized Closure is salient, with a difference of different orders of magnitude in all cases.

In the rest of the experiments, for each group of datasets (**`real`**, **`big`** and **`synthetic`**), we have summed up all the results of each metric, i.e., processed dependencies, processed attributes, and running time, and for each

[4] `https://rdlab.cs.upc.edu/hpc/`

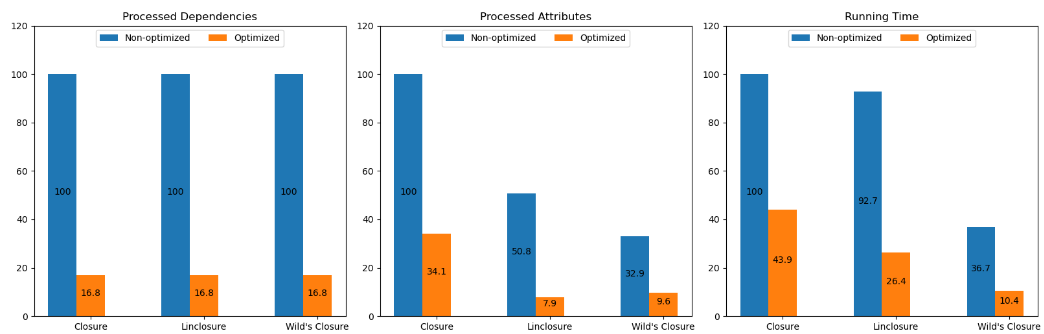

Figure 2: Comparison of the performance of each algorithm w.r.t. their optimized versions when processing the Canonical-Direct Unit Basis in **`big` datasets**. The values have been normalized to the interval (0,100).

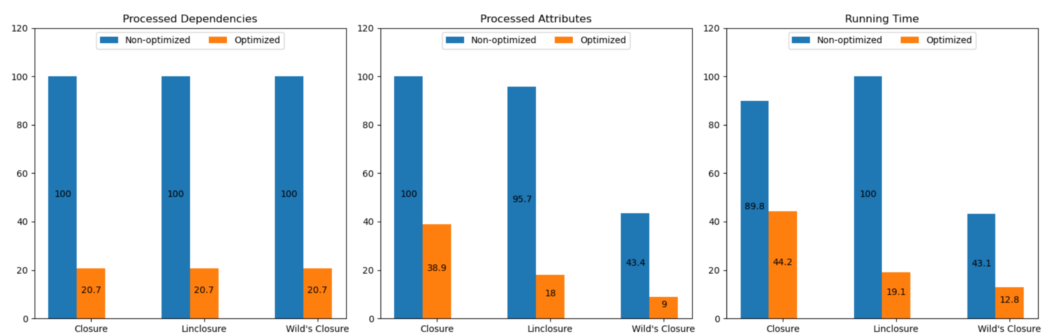

Figure 3: Comparison of the performance of each algorithm w.r.t. their optimized versions when processing the Canonical-Direct Unit Basis in **`synthetic` datasets**. The values have been normalized to the interval (0,100).

combination $<$Basis, Algorithm$>$, and we have plotted the results in Figures 4, 6, and 8.

Let us explain the contents of these plots in assuming that we are calculating closure_Σ with one combination $<$Basis, Algorithm$>$, and that we are processing **`real`**. For each dataset in **`real`** $= \{ D_1, D_2, \ldots, D_{19} \}$, we computed the closure $\text{closure}_\Sigma(X)$ for all $X \in 2^{\mathcal{U}}$, and we summed all the processed dependencies, i.e., $\text{deps}(D_i) = \sum_{X \in 2^{\mathcal{U}}} \text{deps}(\text{closure}_\Sigma(X))$, where $\text{deps}(\text{closure}_\Sigma(X))$ denotes the number of processed dependencies when computing $\text{closure}_\Sigma(X)$. Obviously, here Σ is the base of the dependencies that hold in D_i. Finally, we summed all $\sum_{D_i \in \mathbf{real}} \mathrm{deps}(D_i)$. We did it for all combinations of basis and algorithm, leading to a *grand total* for each of the nine combinations of $<$Basis, Algorithm$>$. We normalized these grand totals to the interval $(0, 100)$ and plotted it.

We computed also the evolution of these metrics w.r.t. the number of attributes. We grouped all the datasets with the same number of attributes and computed the average for each metric. We plotted the results in Figures 5, 7 and 9. Here we only compare the most performing combinations $<$Basis, Algorithm$>$ for each basis.

We also computed a ranking table recording how many times each combination $<$Basis, Algorithm$>$ was the best performer in the computation of each metric. These results are presented in Tables 5, 6 and 7. In particular, let us consider Table 5. Each column is a combination of $<$Basis, Algorithm$>$, and each row is one of the computed metrics. For example, the score of the metric *Processed Dependencies* (first row) and the first combination (column $<$Canonical-Direct Unit Basis,Closure $>$) is 19. This means that the combination $<$Canonical-Direct Unit Basis,Closure $>$ was the best performer when computing **Processed Attributes** in 19 **`real`** datasets. Since the total number of datasets in **`real`** is 19, each row must sum, at least, 19, but it may be bigger, since there can be more than one *winning* combination.

For the sake of completeness we present all numerical results in different tables in .2, .3 and .4.

0.6.6 Results on Real Datasets

Firstly, in the whole set **`real`**, the average size of the D-basis and the Duquenne-Guigues Basis w.r.t. the Canonical-Direct Unit Basis are, respectively, 99% and 67%, i.e., the sizes of the Duquenne-Guigues Basis are, on average, the 67% of the sizes of the Canonical-Direct Unit Basis. In fact, in six datasets, all three bases have the same size. Secondly, it should be noticed that all the algorithms computing a Canonical-Direct Unit Basis are *optimized*, giving an *a priori* advantage to combinations involving Canonical-Direct Unit Basis.

Figure 4 shows the totals for **`real`** datasets. Regarding the number of *processed dependencies*, we remark that all the combinations involving the Canonical-Direct Unit Basis clearly benefit from the optimizations performed. On the other hand, the number of *processed attributes* shows that

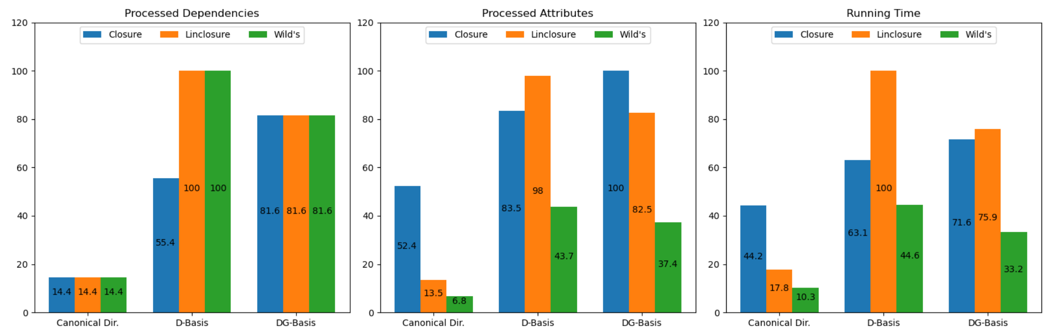

Figure 4: Totals for the analyzed measures for each combination (Base $\times$ Algorithm) in **`real` datasets**. The values have been normalized to the interval (0,100).

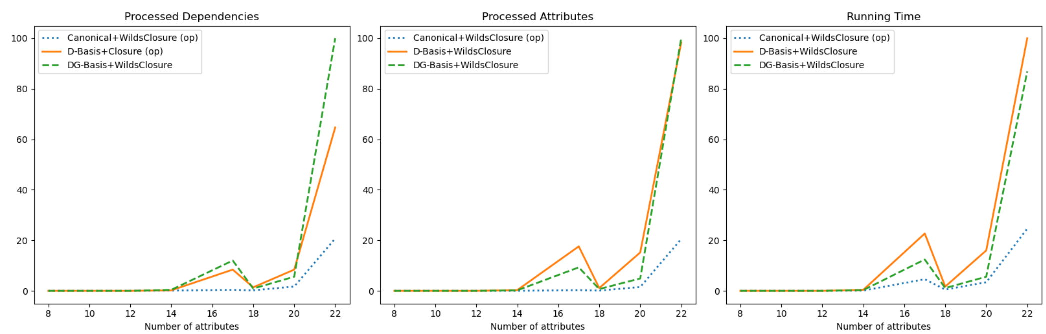

Figure 5: Performance of the best combinations of (Base $\times$ Algorithm) for the analyzed metrics w.r.t. the number of attributes in **`real` datasets**. The values have been normalized to the interval (0,100).

WildClosure is, by far, the less consuming option, followed by LinClosure. The fact that Closure performs less attribute operations than LinClosure when processing the D-basis can be explained by the fact that in that particular case, Closure processes less dependencies than LinClosure. The *execution time* also shows that the combinations with WildClosure are the most performing in all cases. In the rest of the cases, the running time seems to be more correlated to the processed attributes than to the processed dependencies. This may suggest that the number of attribute operations is a metric to be considered when explaining the performance of these algorithms.

	Canonical			D-Basis			DG-basis		
Attribute	CLO	LIN	WILD	CLO	LIN	WILD	CLO	LIN	WILD
Processed Dependencies	19	19	19	1	1	1	1	1	1
Processed Attributes	0	0	19	0	0	1	0	0	1
Running Time	0	0	17	0	0	1	0	0	1

Table 5: Best performance in the **real datasets** for each pair base+algorithm for the 5 metrics. The total number of databases is 19 (for each metric there can be more than one minimal combination).

Figure 5 shows that <Canonical-Direct Unit Basis, WildClosure > remains steady for datasets up to 20 attributes, in comparison to the two other combinations, which, in turn, show a more substantial increase from 20 attributes on.

These results are coherent with Table 5, showing that in most cases, <Canonical-Direct Unit Basis, WildClosure > is the most performing combination. The only exceptions are three cases for the running time, in which <D-basis, WildClosure > is the best combination.

0.6.7 Results on Big Datasets

Firstly, we remark that the average sizes of the D-basis and the Duquenne-Guigues Basis w.r.t. the Canonical-Direct Unit Basis are, respectively, 89% and 41%. This means that processing with the Duquenne-Guigues Basis may be more beneficial, while for the D-basis the difference is not so significant.

The results on **big** are shown in Figure 6. Regarding *processed dependencies* and *running time*, the results are similar to the ones explained for **real**, with the combination <Canonical-Direct Unit Basis, WildClosure > being still the best performer. A slight difference appears for *processed attributes* in the combinations <Canonical-Direct Unit Basis,LinClosure > and <Canonical-Direct Unit Basis, WildClosure >, where LinClosure outperforms WildClosure.

This tendency can also be observed in Table 6, where one combination of <Duquenne-Guigues Basis,WildClosure > is the most performing. It involves the dataset *soy-bean-small*, where the proportion of the size of the Canonical-Direct Unit Basis versus the Duquenne-Guigues Basis is 12%, i.e., the largest by far in **big**.

It should also be noticed that the performance regarding the number of attributes presented in Figure 7 shows a steady increase of <Canonical-Direct

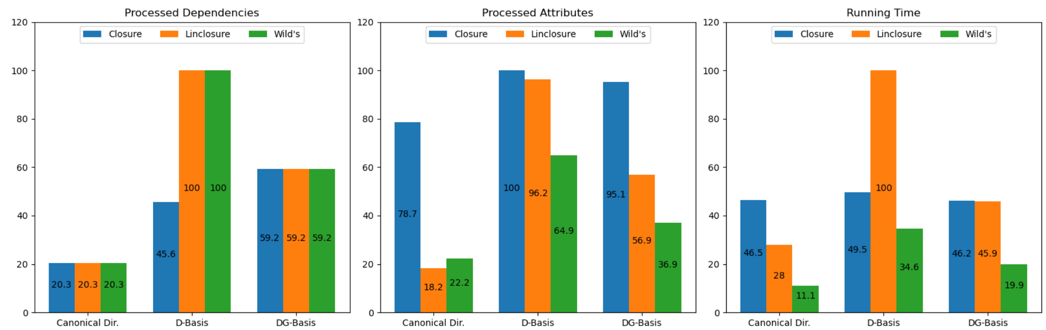

Figure 6: Totals for the analyzed measures for each combination (Base × Algorithm) in **big datasets**. The values have been normalized to the interval (0,100).

	Canonical			D-Basis			DG-basis		
Attribute	CLO	LIN	WILD	CLO	LIN	WILD	CLO	LIN	WILD
Processed Dependencies	4	4	4	0	0	0	1	1	1
Processed Attributes	0	2	2	0	0	0	0	1	0
Running Time	0	0	4	0	0	0	0	0	1

Table 6: Best performance in the **big datasets** for each pair base+algorithm for the 5 metrics. The total number of databases is 5 (for each metric there can be more than one minimal combination).

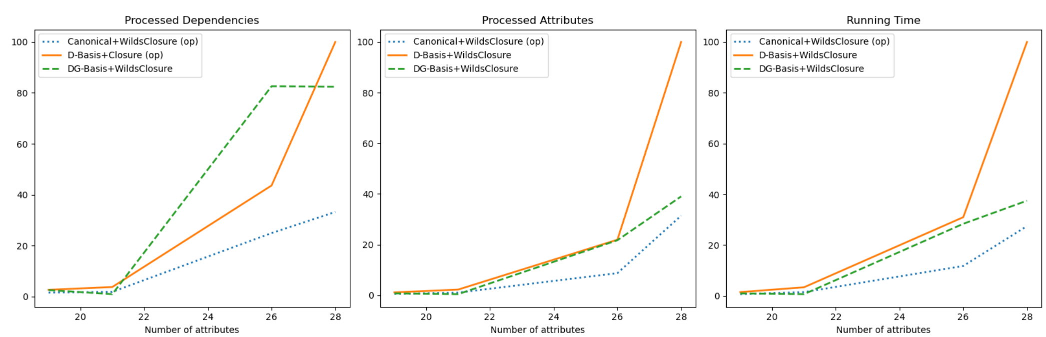

Figure 7: Performance of the best combinations of (Base × Algorithm) for the analyzed metrics w.r.t. the number of attributes in **big datasets**. The values have been normalized to the interval (0,100).

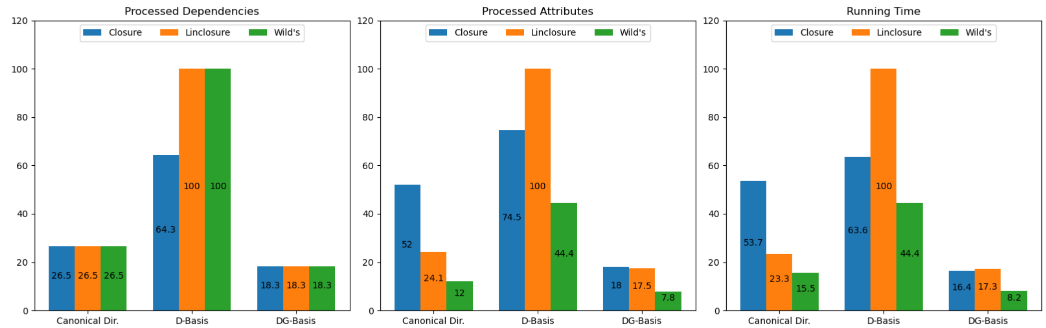

Figure 8: Totals for the analyzed measures for each combination (Base $\times$ Algorithm) in **`synthetic` datasets**. The values have been normalized to the interval (0,100).

	Canonical			D-Basis			DG-basis		
Attribute	CLO	LIN	WILD	CLO	LIN	WILD	CLO	LIN	WILD
Processed Dependencies	409	409	409	10	0	0	176	176	176
Processed Attributes	0	0	410	0	0	1	0	0	184
Running Time	0	0	336	0	0	0	0	0	259

Table 7: Best performance in the **`synthetic` datasets** for each pair base+algorithm for the 5 metrics. The total number of databases is 595 (for each metric there can be more than one minimal combination).

Unit Basis,WildClosure $>$ and $<$Duquenne-Guigues Basis,WildClosure $>$ w.r.t. the rest of the combinations.

0.6.8 Results on Synthetic Datasets

Here, the average sizes of the D-basis and the Duquenne-Guigues Basis w.r.t. the Canonical-Direct Unit Basis are, respectively, 77% and 55%. Considering **`synthetic`**, we can check in Figure 8 that the combinations involving the Duquenne-Guigues Basis and all the algorithms are now, in total, the most performing in all three metrics. However, in Table 7, the majority of winning combinations are still related to $<$Canonical-Direct Unit Basis,WildClosure $>$. This indicates that in some cases the combinations with Duquenne-Guigues Basis outperforms by a large margin those with Canonical-Direct Unit Basis, and that in the opposite cases the difference is not so large.

One could argue that when the Duquenne-Guigues Basis is substantially

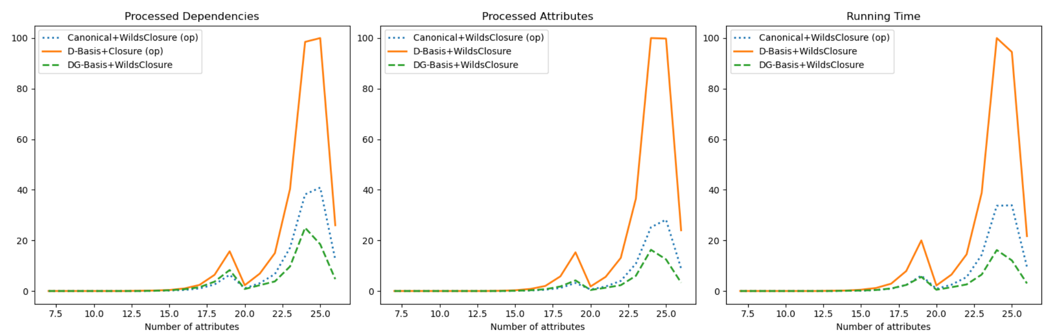

Figure 9: Performance of the best combinations of (Base $\times$ Algorithm) for the analyzed metrics w.r.t. the number of attributes in **`synthetic` datasets**. The values have been normalized to the interval (0,100).

smaller than a Canonical-Direct Unit Basis, then, it is expected that the former performs better than the latter. Then the question is in which proportion? When the proportion p, i.e., the size of Canonical-Direct Unit Basis divided by the size of Duquenne-Guigues Basis, is $8 \leq p$ Duquenne-Guigues Basis outperforms Canonical-Direct Unit Basis in all cases. When $3 \leq p \leq 8$, then p only explains around 60% of the cases. Yet, there are cases with an inferior proportion where the performance of the Duquenne-Guigues Basis is still better. This suggests that even if this proportion may explain some of these cases, it is not the only variable to be involved.

Figure 9 shows that for $|\Sigma| \leq 17$ the performance of all combinations is similar. Afterwards Duquenne-Guigues Basis and Canonical-Direct Unit Basis have a similar behaviour whereas D-basis performance increases dramatically. We may notice that the growth from $|\Sigma| \geq 20$ seems to be exponential, while it decays when $|\Sigma| \geq 26$.

0.7 Discussion

We have performed exhaustive experiments over different datasets in order to answer different questions. The first is: *Is it better to use a direct basis or a a minimal basis to compute* closure$_\Sigma$? In general terms, the results show that the Canonical-Direct Unit Basis *with optimizations* is the best option in **`real`** and **`big`**, whereas in **`synthetic`** a Duquenne-Guigues Basis shows the better performance. The variable which better explains this behaviour is, obviously, the proportion between the size of both basis, but this is not the only explanation. Here, the fact that the Canonical-Direct Unit Basis is combined with optimized algorithms is crucial, otherwise the best options would be in all cases the Duquenne-Guigues Basis. This can be clearly seen in Figures 1, 2 and 3, where the non-optimized versions would be outperformed by the combinations with the Duquenne-Guigues Basis. This makes the Duquenne-Guigues Basis a very valuable alternative to the Canonical-Direct Unit Basis in different applications. Meanwhile the D-basis was not favored for two reasons, (i) it could not be computed by improved versions of the algorithms, and, (ii) the difference in size was not big enough to outperform any other combination. To sum up, the D-basis did not enjoy the same benefits of being direct as Canonical-Direct Unit Basis, nor enjoy the benefits of being smaller as Duquenne-Guigues Basis.

Regarding the algorithms, WildClosure–improved or not– is the most performing (virtually) in all combinations. It can be argued that the fact that we are using very specific basis may influence this performance, this is, if instead of using Canonical-Direct Unit Basis, D-basis or Duquenne-Guigues Basis we were using some other (random?) basis, the outcome would have been different. We can't answer this question. Firstly, both LinClosure and WildClosure have shown the best behavior in terms of the number of processed dependencies (obviously expected). Secondly, the performance of LinClosure w.r.t. number of attributes processed is worst than that of WildClosure. This two elements may explain the systematic difference in the execution time of both algorithms. Actually, this fact validates the comment of the author of WildClosure which is transcribed at the end of Section 0.3.3.

The classical algorithm Closure is competitive when it is optimized (Optimized Closure) or semi-optimized, as when combined with D-basis. For instance, it shows an overall better running time than LinClosure when processing the D-basis and the Duquenne-Guigues Basis.

In fact, as we have previously mentioned, it seems that the execution time seems to be more correlated to the attribute operations than to the number of processed dependencies. It also can be argued that the execution time is very sensitive to the implementation, with which we fully agree. We have tried to be fair with all algorithms, and implement them using the same data structures, but it does not mean that our implementation of LinClosure may not be improved. We only can reason on the evidence provided by our results, which show that the total number of processed dependencies is the same for both LinClosure and WildClosure, and that the divergence seems to appear in processed attributes.

To sum up, we may remark that, (1) the improvements performed when processing a Canonical-Direct Unit Basis make the choice of this basis preferable in some instances, but not in all of them, (2) the amount of attribute operations may be relevant w.r.t. the running time of the algorithm, (3) the Duquenne-Guigues Basis may be a suitable and efficient alternative to the Canonical-Direct Unit Basis in some setups, but this needs to be further investigated, and (4) the peculiar structure of D-basis does not allow to perform many improvements, implying that the performance stays far behind both Canonical-Direct Unit Basis and Duquenne-Guigues Basis.

0.8 Conclusions

The notion of being *direct* for a cover seems to be foreign to the DB community, but it is clearly present in lattice theory and in FCA. This difference somehow parallels that of the most common basis in each community: whereas in the DB community all state-of-the-art algorithms mining functional dependencies are computing the Canonical-Direct Unit Basis, the Duquenne-Guigues Basis is central in the FCA community. Each basis enjoys different –and somehow contradictory– properties: the Canonical-Direct Unit Basis is direct and the Duquenne-Guigues Basis is minimal. In this paper, we discussed which one of these two properties may be more decisive when computing closure$_\Sigma$. We compared the performance of these two bases in combining three of the most well-known algorithms computing a closure. To take into account the fact of being direct and to be consistent in the comparison of the full potential of both bases, we improved these three algorithms when the input is the Canonical-Direct Unit Basis. We also compared these two bases to the D-basis, which is not minimal and enjoys the property of being direct.

Our results have shown that the Canonical-Direct Unit Basis may compete with the Duquenne-Guigues Basis thanks to the improvements brought in to the algorithms computing closure$_\Sigma$, and that although the number of processed dependencies has been the *de facto* standard to discuss, the complexity of these algorithms, the number of operations on attributes appeared also as a relevant factor to be considered. We also realized that the D-basis is not an alternative in any case, maybe due to the fact that we were not able to find examples where the size of the D-basis was considerably smaller than the size of the Canonical-Direct Unit Basis.

These results bring up the following questions: (i) can we determine with precision what are the relevant metrics that may decide when a Duquenne-Guigues Basis will be more performing that a Canonical-Direct Unit Basis? We have mentioned that the size is one of them, but this does not explain all the cases and, (ii) can we explain more precisely the influence of the operations on attributes in order to understand the actual performance of all three algorithms? Although this paper tries and partially answers some of these questions, we still think that the study of the performance of these bases should continue.

0.9 Acknowledgements

Jaume Baixeries is supported by a recognition 2021SGR-Cat (01266 LQMC) from AGAUR (Generalitat de Catalunya) and the grants AGRUPS-2022 and AGRUPS-2023 from Universitat Politècnica de Catalunya. Amedeo Napoli is carrying out this research work as part of the French ANR-21-CE23-0023 SmartFCA Research Project.

Bibliography

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] Kira V. Adaricheva and James B. Nation. Discovery of the D-basis in binary tables based on hypergraph dualization. *Theoretical Computer Science*, 658:307–315, 2017.

[3] Kira V. Adaricheva, James B. Nation, and Robert Rand. Ordered direct implicational basis of a finite closure system. *Discrete Applied Mathematics*, 161(6):707–723, 2013.

[4] A.Mora, P. Cordero, M. Enciso, I. Fortes, and G. Aguilera. Closure via functional dependence simplification. *International Journal of Computer Mathematics*, 89(4):510–526, 2012.

[5] William W. Armstrong. Dependency structures of data base relationships. In *IFIP congress*, volume 74, pages 580–583, 1974.

[6] Franz Baader and Felix Distel. A Finite Basis for the Set of EL-Implications Holding in a Finite Model. In Raoul Medina and Sergei A. Obiedkov, editors, *Proceedings of the 6th International Conference on Formal Concept Analysis (ICFCA)*, volume 4933 of *Lecture Notes in Computer Science*, pages 46–61. Springer, 2008.

[7] Franz Baader, Bernhard Ganter, Baris Sertkaya, and Ulrike Sattler. Completing Description Logic Knowledge Bases Using Formal Concept Analysis. In Manuela M. Veloso, editor, *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 230–235, 2007.

[8] Jaume Baixeries, Víctor Codocedo, Mehdi Kaytoue, and Amedeo Napoli. Three Views on Dependency Covers from an FCA Perspective. In Dominik Dürrschnabel and Domingo López-Rodríguez, editors, *Proceedings of the 17th International Conference on Formal Concept Analysis (ICFCA)*, volume 13934 of *Lecture Notes in Computer Science 13934*, pages 78–94. Springer, 2023.

[9] Konstantin Bazhanov and Sergei A. Obiedkov. Comparing Performance of Algorithms for Generating the Duquenne-Guigues Basis. In Amedeo Napoli and Vilém Vychodil, editors, *Proceedings of The Eighth International Conference on Concept Lattices and Their Applications (CLA)*, CEUR Workshop Proceedings 959, pages 43–57, 2011.

[10] Konstantin Bazhanov and Sergei A. Obiedkov. Optimizations in computing the Duquenne-Guigues basis of implications. *Annals of Mathematics and Artificial Intelligence*, 70(1-2):5–24, 2014.

[11] Catriel Beeri and Philip A. Bernstein. Computational Problems Related to the Design of Normal Form Relational Schemas. *ACM Transactions on Database Systems*, 4(1):30–59, 1979.

[12] Karell Bertet and Bernard Monjardet. The multiple facets of the canonical direct unit implicational basis. *Theoretical Computer Science*, 411(22-24):2155–2166, 2010.

[13] Yves Crama and Peter L. Hammer. *Boolean Functions - Theory, Algorithms, and Applications*, volume 142 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2011.

[14] Christopher J. Date. *An Introduction to Database Systems (7th Ed.)*. Addison-Wesley-Longman, 2000.

[15] Jim Diederich and Jack Milton. New Methods and Fast Algorithms for Database Normalization. *ACM Transactions on Database Systems.*, 13(3):339–365, 1988.

[16] William F. Dowling and Jean H. Gallier. Linear-Time Algorithms for Testing the Satisfiability of Propositional Horn Formulae. *Journal of Logic Programming*, 1(3):267–284, 1984.

[17] Bernhard Ganter and Sergei A. Obiedkov. *Conceptual Exploration*. Springer, 2016.

[18] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis - Mathematical Foundations*. Springer, 1999.

[19] Jean-Louis Guigues and Vincent Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences Humaines*, 95:5–18, 1986.

[20] Radek Janostik, Jan Konecny, and Petr Krajca. LinCbO: Fast algorithm for computation of the Duquenne-Guigues basis. *Information Sciences*, 572:223–240, 2021.

[21] Radek Janostik, Jan Konecny, and Petr Krajca. Pruning techniques in LinCbO for the computation of the Duquenne-Guigues basis. *Information Sciences*, 616:182–203, 2022.

[22] Estrella Rodríguez Lorenzo, Kira V. Adaricheva, Pablo Cordero, Manuel Enciso, and Ángel Mora. From an Implicational System to its Corresponding D-basis. In Sadok Ben Yahia and Jan Konecny, editors, *Proceedings of the Twelfth International Conference on Concept Lattices and Their Applications (CLA)*, CEUR Workshop Proceedings 1466, pages 217–228, 2015.

[23] Estrella Rodríguez Lorenzo, Karell Bertet, Pablo Cordero, Manuel Enciso, and Ángel Mora. Direct-optimal basis computation by means of the fusion of simplification rules. *Discrete Applied Mathematics*, 249:106–119, 2018.

[24] David Maier. *The Theory of Relational Databases*. Computer Science Press, 1983.

[25] Heikki Mannila and Kari-Jouko Räihä. *Design of Relational Databases*. Addison-Wesley, 1992.

[26] Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schönberg, Jakob Zwiener, and Felix Naumann. Functional Dependency Discovery: An Experimental Evaluation of Seven Algorithms. *Proc. VLDB Endow.*, 8(10):1082–1093, 2015.

[27] Uwe Ryssel, Felix Distel, and Daniel Borchmann. Fast algorithms for implication bases and attribute exploration using proper premises. *Annals of Mathematics and Artificla Intelligence*, 70(1-2):25–53, 2014.

[28] Solveig Torgerson. Automatic design of relational databases. Technical report, Cornell University, 1989.

[29] Johanna Völker and Sebastian Rudolph. Lexico-Logical Acquisition of OWL DL Axioms. In Raoul Medina and Sergei A. Obiedkov, editors, *Proceedings of the 6th International Conference on Formal Concept Analysis (ICFCA)*, volume 4933 of *Lecture Notes in Computer Science*, pages 62–77. Springer, 2008.

[30] Marcel Wild. Computations with Finite Closure Systems and Implications. In Ding-Zhu Du and Ming Li, editors, *Proceedings of the First Annual International Conference on Computing and Combinatorics (COCOON)*, Lecture Notes in Computer Science 959, pages 111–120. Springer, 1995.

[31] Marcel Wild. The joy of implications, aka pure Horn formulas: Mainly a survey. *Theoretical Computer Science*, 658:264–292, 2017.

.1 Computation of Closures for the Big Datasets

For each dataset in Table 2 we have computed the closure with all combinations of a number of attribute sets $X \subseteq \mathcal{U}$ with a given frequency (this is: the probability of having a given attribute in that set is 0.1). The list of number of sets and frequencies is in Table 8. For instance, this table says that we have computed $10,000$ sets with a probability 0.1, etc.

Sets	10K	20K	30K	40K	50K	40K	30K	20K	10K
Frequency	0.1	0.2	0.3	0.4	0.5	0.4	0.3	0.2	0.1

Table 8: Number of closures and their frequencies computed for each **big dataset**.

.2 Experiments with Real Datasets

Processed Dependencies		Canonical				D-Basis				DG-basis			
DB	$\lvert\mathcal{U}\rvert$	$\lvert\Sigma\rvert$	CLO	LIN	WILD	$\lvert\Sigma\rvert$	CLO	LIN	WILD	$\lvert\Sigma\rvert$	CLO	LIN	WILD
abalone	9	137	**4,256**	**4,256**	**4,256**	137	13,376	30,576	30,576	100	22,362	22,362	22,362
adult	14	46	**25,600**	**25,600**	**25,600**	46	27,620	28,556	28,556	46	28,556	28,556	28,556
breast-cancer-wisconsin	11	46	**9,760**	**9,760**	**9,760**	46	13,924	14,532	14,532	43	13,996	13,996	13,996
bridges	12	126	**45,184**	**45,184**	**45,184**	125	130,996	210,090	210,090	88	149,363	149,363	149,363
congress-votes	17	53	**5,248**	**5,248**	**5,248**	53	6,033	6,147	6,147	53	6,147	6,147	6,147
echocardiogram	12	526	**220,928**	**220,928**	**220,928**	526	844,550	1,701,742	1,701,742	269	871,771	871,771	871,771
ecoli	8	46	**2,080**	**2,080**	**2,080**	46	2,906	4,322	4,322	46	4,322	4,322	4,322
flights 20 500k	12	69	**45,824**	**45,824**	**45,824**	51	59,952	73,329	73,329	49	70,825	70,825	70,825
glass	10	160	**25,664**	**25,664**	**25,664**	160	64,899	104,270	104,270	120	79,975	79,975	79,975
hepatitis	20	8250	**40,146,301**	**40,146,301**	**40,146,301**	8250	197,937,704	398,620,818	398,620,818	2730	131,918,652	131,918,652	131,918,652
house-votes-84	17	53	**5,248**	**5,248**	**5,248**	53	6,033	6,147	6,147	53	6,147	6,147	6,147
letter	17	61	**2,240**	**2,240**	**2,240**	61	**2,240**	**2,240**	**2,240**	61	**2,240**	**2,240**	**2,240**
mushroom	22	3583	**42,401,713**	**42,401,713**	**42,401,713**	3583	92,114,537	120,482,269	120,482,269	1721	66,096,065	66,096,065	66,096,065
page-blocks	10	135	**15,680**	**15,680**	**15,680**	135	38,932	72,423	72,423	69	40,037	40,037	40,037
pen-recognition	17	30463	**37,626,368**	**37,626,368**	**37,626,368**	30463	797,945,288	2,161,555,214	2,161,555,214	15885	1,137,152,337	1,137,152,337	1,137,152,337
tic-tac-toe	10	18	**72**	**72**	**72**	18	288	360	360	18	360	360	360
waveform	22	24002	**935,838,999**	**935,838,999**	**935,838,999**	24002	2,967,583,156	4,663,660,606	4,663,660,606	24002	4,663,660,606	4,663,660,606	4,663,660,606
wine	14	1374	**3,430,400**	**3,430,400**	**3,430,400**	1374	11,224,253	22,290,574	22,290,574	1106	17,942,810	17,942,810	17,942,810
zoo	18	284	**4,775,936**	**4,775,936**	**4,775,936**	283	34,160,220	35,243,954	35,243,954	163	21,244,708	21,244,708	21,244,708
Average	14.32	3,654.32	56,033,026.37	56,033,026.37	56,033,026.37	3,653.26	215,904,047.74	389,689,903.63	389,689,903.63	2,453.79	317,858,488.37	317,858,488.37	317,858,488.37

Table 9: Totals of the measure **Processed Dependencies** for all **`real` datasets**. In bold are the minimal values. The last line contains the average of each measure: the sum of all values for each pair (Base $\times$ Algorithm) divided by the number of datasets.

Processed Attributes		Canonical				D-Basis				DG-basis			
DB	$\lvert\mathcal{U}\rvert$	$\lvert\Sigma\rvert$	CLO	LIN	WILD	$\lvert\Sigma\rvert$	CLO	LIN	WILD	$\lvert\Sigma\rvert$	CLO	LIN	WILD
abalone	9	137	642,670	71,385	**35,301**	137	792,213	568,872	269,429	100	861,921	410,714	197,705
adult	14	46	9,552,523	518,556	**257,284**	46	9,691,885	648,524	286,404	46	12,954,265	647,118	288,123
breast-cancer-wisconsin	11	46	939,720	158,555	**78,421**	46	1,047,207	297,267	126,909	43	1,672,144	258,271	117,285
bridges	12	126	5,545,480	933,384	**461,612**	125	7,457,890	5,282,520	2,381,510	88	9,347,703	3,668,299	1,668,181
congress-votes	17	53	132,655,295	147,001	**73,085**	53	132,699,290	176,512	86,601	53	134,617,601	176,512	86,601
echocardiogram	12	526	22,173,693	5,077,471	**2,529,480**	526	35,159,262	44,913,136	19,958,094	269	32,141,145	21,920,919	10,267,876
ecoli	8	46	90,361	27,272	**13,458**	46	107,749	67,674	30,366	46	208,737	67,077	30,332
flights 20 500k	12	69	2,873,916	850,186	**420,518**	51	2,754,047	1,535,590	683,523	49	5,194,400	1,499,148	658,414
glass	10	160	1,471,992	473,734	**235,115**	160	2,188,987	2,306,507	973,695	120	2,914,050	1,720,244	758,641
hepatitis	20	8250	7,808,101,115	1,575,946,487	**787,788,973**	8250	12,966,658,256	16,414,801,259	7,874,351,588	2730	6,808,745,482	5,330,144,435	2,564,982,670
house-votes-84	17	53	132,655,295	147,001	**73,085**	53	132,700,470	176,512	86,601	53	134,617,601	176,512	86,601
letter	17	61	163,021,707	66,277	**32,997**	61	163,033,940	67,584	**32,997**	61	163,866,019	67,584	**32,997**
mushroom	22	3583	14,176,252,115	1,427,400,693	**713,497,559**	3583	16,140,189,328	4,237,779,900	2,039,463,765	1721	18,063,471,147	2,311,701,537	1,114,924,873
page-blocks	10	135	1,264,898	265,707	**131,529**	135	1,712,409	1,379,829	655,551	69	1,630,397	752,878	353,967
pen-recognition	17	30463	61,745,162,923	1,253,902,086	**626,648,511**	30463	84,552,185,155	77,829,875,411	36,491,350,032	15885	85,241,002,493	39,289,014,325	19,195,092,139
tic-tac-toe	10	18	240,192	1,392	**682**	18	242,872	7,542	3,554	18	246,236	7,542	3,562
waveform	22	24002	90,957,167,084	41,155,276,963	**20,576,799,833**	24002	164,964,021,905	229,669,366,112	99,926,362,986	24002	224,034,789,490	229,678,147,337	102,585,149,913
wine	14	1374	252,229,902	94,758,079	**47,322,596**	1374	453,572,029	690,859,413	303,173,233	1106	590,204,163	553,496,891	250,300,845
zoo	18	284	1,116,100,136	135,645,229	**67,190,818**	283	2,001,499,400	1,305,688,423	594,172,388	163	1,846,038,241	783,835,415	348,561,050
Average	14.32	3,654.32	9,290,954,790.37	2,402,719,339.89	1,201,241,624.05	3,653.26	14,819,353,383.89	17,379,252,557.21	7,750,234,169.79	2,453.79	17,741,290,696.58	14,630,405,934.63	6,635,450,619.74

Table 10: Totals of the measure **Processed Attributes** for all **`real` datasets**. In bold are the minimal values. The last line contains the average of each measure: the sum of all values for each pair (Base $\times$ Algorithm) divided by the number of datasets.

Running Time		Canonical				D-Basis				DG-basis			
DB	$\lvert\mathcal{U}\rvert$	$\lvert\Sigma\rvert$	CLO	LIN	WILD	$\lvert\Sigma\rvert$	CLO	LIN	WILD	$\lvert\Sigma\rvert$	CLO	LIN	WILD
abalone	9	137	7.50	2.10	**1.54**	137	9.46	11.11	4.99	100	10.24	8.19	3.77
adult	14	46	80.58	25.46	**20.07**	46	84.84	28.81	23.81	46	105.08	29.69	24.10
breast-cancer-wisconsin	11	46	15.76	5.41	**4.07**	46	16.74	9.00	6.36	43	23.94	9.04	6.48
bridges	12	126	62.48	19.14	**13.68**	125	79.11	80.36	40.06	88	92.24	59.16	30.66
congress-votes	17	53	800.03	330.94	202.08	53	806.35	330.22	204.36	53	810.24	326.76	**201.30**
echocardiogram	12	526	378.84	111.64	**76.64**	526	537.54	853.83	359.86	269	462.87	444.84	193.56
ecoli	8	46	2.93	1.01	**0.70**	46	2.58	2.46	1.41	46	4.41	2.56	1.44
flights 20 500k	12	69	49.48	41.95	**13.91**	51	80.62	40.48	25.30	49	74.47	48.15	24.50
glass	10	160	34.31	17.36	**8.02**	160	87.46	78.18	73.43	120	75.03	51.48	38.99
hepatitis	20	8250	37,390.10	17,917.10	**9,891.58**	8250	57,813.80	109,878.00	47,157.80	2730	30,538.70	36,709.70	16,425.80
house-votes-84	17	53	895.67	383.03	**214.75**	53	820.83	421.79	247.84	53	862.46	509.13	218.58
letter	17	61	901.40	453.94	257.06	61	839.77	424.34	**218.53**	61	873.96	485.88	233.40
mushroom	22	3583	69,862.80	27,857.80	**12,572.50**	3583	75,676.00	55,107.20	34,264.20	1721	81,469.60	27,773.50	17,165.90
page-blocks	10	135	25.94	9.07	**6.00**	135	32.00	38.73	19.78	69	30.94	20.92	10.98
pen-recognition	17	30463	302,374.00	105,994.00	**53,093.10**	30463	399,765.00	647,315.00	267,251.00	15885	387,928.00	336,843.00	145,346.00
tic-tac-toe	10	18	4.49	1.57	**1.26**	18	4.64	1.87	1.34	18	4.65	1.76	1.29
waveform	22	24002	483,737.00	208,834.00	**132,619.00**	24002	742,195.00	1,217,360.00	556,247.00	24002	950,059.00	1,140,670.00	495,799.00
wine	14	1374	2,139.31	776.18	**507.31**	1374	3,139.92	5,667.09	2,386.24	1106	3,758.11	4,482.14	1,863.88
zoo	18	284	6,860.40	2,059.09	**1,335.50**	283	10,468.90	10,071.40	5,032.12	163	9,770.72	6,194.12	3,163.35
Average	14.32	3,654.32	47,664.37	19,202.15	11,096.78	3,653.26	68,024.24	107,774.73	48,082.39	2,453.79	77,208.14	81,824.74	35,829.10

Table 11: Totals of the measure **Running Time** for all **`real` datasets**. In bold are the minimal values. The last line contains the average of each measure: the sum of all values for each pair (Base × Algorithm) divided by the number of datasets.

DB	Base	Algorithm	deps	attrib	outer	inner	time (ms)	$\|\Sigma\|$	$\|\mathcal{U}\|$
tic-tac-toe	Canonical	Closure (op)	**72**	240,192	**1,023**	18,414	4.49	18	10
		Linclosure (op)	**72**	1,392	5,120	73,728	1.57	18	10
		WildsClosure (op)	**72**	**682**	**1,023**	**72**	**1.26**	18	10
	D-Basis	Closure (op)	288	242,872	**1,023**	18,414	4.64	18	10
		Linclosure	360	7,542	5,148	74,016	1.87	18	10
		WildsClosure	360	3,554	1,061	360	1.34	18	10
	DG-Basis	Closure (op)	360	246,236	1,059	18,558	4.65	18	10
		Linclosure	360	7,542	5,148	74,016	1.76	18	10
		WildsClosure	360	3,562	1,061	360	1.29	18	10
ecoli	Canonical	Closure (op)	**2,080**	90,361	**255**	11,730	2.93	46	8
		Linclosure (op)	**2,080**	27,272	1,024	15,232	1.01	46	8
		WildsClosure (op)	**2,080**	**13,458**	**255**	**2,080**	**0.70**	46	8
	D-Basis	Closure (op)	2,906	107,749	**255**	11,730	2.58	46	8
		Linclosure	4,322	67,674	1,380	18,060	2.46	46	8
		WildsClosure	4,322	30,366	613	4,322	1.41	46	8
	DG-Basis	Closure (op)	4,322	208,737	587	20,876	4.41	46	8
		Linclosure	4,322	67,077	1,380	25,708	2.56	46	8
		WildsClosure	4,322	30,332	683	4,322	1.44	46	8
adult	Canonical	Closure (op)	**25,600**	9,552,523	**16,383**	753,618	80.58	46	14
		Linclosure (op)	**25,600**	518,556	114,688	2,007,040	25.46	46	14
		WildsClosure (op)	**25,600**	**257,284**	**16,383**	**25,600**	**20.07**	46	14
	D-Basis	Closure (op)	27,620	9,691,885	**16,383**	753,618	84.84	46	14
		Linclosure	28,556	648,524	118,676	2,024,544	28.81	46	14
		WildsClosure	28,556	286,404	22,092	28,556	23.81	46	14
	DG-Basis	Closure (op)	28,556	12,954,265	21,817	965,088	105.08	46	14
		Linclosure	28,556	647,118	118,676	2,094,554	29.69	46	14
		WildsClosure	28,556	288,123	22,342	28,556	24.10	46	14
congress-votes	Canonical	Closure (op)	**5,248**	132,655,295	**131,071**	6,946,763	800.03	53	17
		Linclosure (op)	**5,248**	147,001	1,114,112	36,765,696	330.94	53	17
		WildsClosure (op)	**5,248**	**73,085**	**131,071**	**5,248**	202.08	53	17
	D-Basis	Closure (op)	6,033	132,699,290	**131,071**	6,946,763	806.35	53	17
		Linclosure	6,147	176,512	1,114,943	36,775,066	330.22	53	17
		WildsClosure	6,147	86,601	132,809	6,147	204.36	53	17
	DG-Basis	Closure (op)	6,147	134,617,601	132,717	7,027,343	810.24	53	17
		Linclosure	6,147	176,512	1,114,943	36,775,066	326.76	53	17
		WildsClosure	6,147	86,601	132,809	6,147	**201.30**	53	17
house-votes-84	Canonical	Closure (op)	**5,248**	132,655,295	**131,071**	6,946,763	895.67	53	17
		Linclosure (op)	**5,248**	147,001	1,114,112	36,765,696	383.03	53	17
		WildsClosure (op)	**5,248**	**73,085**	**131,071**	**5,248**	**214.75**	53	17
	D-Basis	Closure (op)	6,033	132,700,470	**131,071**	6,946,763	820.83	53	17
		Linclosure	6,147	176,512	1,114,943	36,775,066	421.79	53	17
		WildsClosure	6,147	86,601	132,809	6,147	247.84	53	17
	DG-Basis	Closure (op)	6,147	134,617,601	132,717	7,027,343	862.46	53	17
		Linclosure	6,147	176,512	1,114,943	36,775,066	509.13	53	17
		WildsClosure	6,147	86,601	132,809	6,147	218.58	53	17
letter	Canonical	Closure (op)	**2,240**	163,021,707	**131,071**	7,995,331	901.40	61	17
		Linclosure (op)	**2,240**	66,277	1,114,112	47,513,600	453.94	61	17
		WildsClosure (op)	**2,240**	**32,997**	**131,071**	**2,240**	257.06	61	17
	D-Basis	Closure (op)	**2,240**	163,033,940	**131,071**	7,995,331	839.77	61	17
		Linclosure	**2,240**	67,584	1,114,395	47,513,600	424.34	61	17
		WildsClosure	**2,240**	**32,997**	131,637	**2,240**	**218.53**	61	17
	DG-Basis	Closure (op)	**2,240**	163,866,019	131,637	8,027,617	873.96	61	17
		Linclosure	**2,240**	67,584	1,114,395	47,513,600	485.88	61	17
		WildsClosure	**2,240**	**32,997**	131,637	**2,240**	233.40	61	17

Table 12: Total values of **`real` datasets** per all analyzed measures: number of dependencies processed, number of operations on attributes, outer loops, inner loops and computation time in miliseconds. $|\Sigma|$: size of the base. $|\mathcal{U}|$: number of attributes

DB	Base	Algorithm	deps	attrib	outer	inner	time (ms)	$\lvert\Sigma\rvert$	$\lvert\mathcal{U}\rvert$
breast-cancer-wisconsin	Canonical	Closure (op)	**9,760**	939,720	**2,047**	94,162	15.76	46	11
		Linclosure (op)	**9,760**	158,555	11,264	172,032	5.41	46	11
		WildsClosure (op)	**9,760**	**78,421**	**2,047**	**9,760**	**4.07**	46	11
	D-Basis	Closure (op)	13,924	1,047,207	**2,047**	94,162	16.74	46	11
		Linclosure	14,532	297,267	12,977	184,256	9.00	46	11
		WildsClosure	14,532	126,909	3,853	14,532	6.36	46	11
	DG-Basis	Closure (op)	13,996	1,672,144	3,853	145,387	23.94	43	11
		Linclosure	13,996	258,271	12,977	203,424	9.04	43	11
		WildsClosure	13,996	117,285	4,365	13,996	6.48	43	11
flights 20 500k	Canonical	Closure (op)	**45,824**	2,873,916	**4,095**	282,555	49.48	69	12
		Linclosure (op)	**45,824**	850,186	24,576	401,408	41.95	69	12
		WildsClosure (op)	**45,824**	**420,518**	**4,095**	**45,824**	**13.91**	69	12
	D-Basis	Closure (op)	59,952	2,754,047	**4,095**	208,845	80.62	51	12
		Linclosure	73,329	1,535,590	33,726	363,886	40.48	51	12
		WildsClosure	73,329	683,523	11,746	73,329	25.30	51	12
	DG-Basis	Closure (op)	70,825	5,194,400	10,825	415,104	74.47	49	12
		Linclosure	70,825	1,499,148	33,726	378,520	48.15	49	12
		WildsClosure	70,825	658,414	12,034	70,825	24.50	49	12
bridges	Canonical	Closure (op)	**45,184**	5,545,480	**4,095**	515,970	62.48	126	12
		Linclosure (op)	**45,184**	933,384	24,576	1,032,192	19.14	126	12
		WildsClosure (op)	**45,184**	**461,612**	**4,095**	**45,184**	**13.68**	126	12
	D-Basis	Closure (op)	130,996	7,457,890	**4,095**	511,875	79.11	125	12
		Linclosure	210,090	5,282,520	34,736	1,404,976	80.36	125	12
		WildsClosure	210,090	2,381,510	9,713	210,090	40.06	125	12
	DG-Basis	Closure (op)	149,363	9,347,703	9,507	601,150	92.24	88	12
		Linclosure	149,363	3,668,299	34,736	1,158,680	59.16	88	12
		WildsClosure	149,363	1,668,181	11,509	149,363	30.66	88	12
page-blocks	Canonical	Closure (op)	**15,680**	1,264,898	**1,023**	138,105	25.94	135	10
		Linclosure (op)	**15,680**	265,707	5,120	228,352	9.07	135	10
		WildsClosure (op)	**15,680**	**131,529**	**1,023**	**15,680**	**6.00**	135	10
	D-Basis	Closure (op)	38,932	1,712,409	**1,023**	138,105	32.00	135	10
		Linclosure	72,423	1,379,829	7,769	343,183	38.73	135	10
		WildsClosure	72,423	655,551	2,845	72,423	19.78	135	10
	DG-Basis	Closure (op)	40,037	1,630,397	2,792	129,232	30.94	69	10
		Linclosure	40,037	752,878	7,769	150,210	20.92	69	10
		WildsClosure	40,037	353,967	3,036	40,037	10.98	69	10
abalone	Canonical	Closure (op)	**4,256**	642,670	**511**	70,007	7.50	137	9
		Linclosure (op)	**4,256**	71,385	2,304	147,968	2.10	137	9
		WildsClosure (op)	**4,256**	**35,301**	**511**	**4,256**	**1.54**	137	9
	D-Basis	Closure (op)	13,376	792,213	**511**	70,007	9.46	137	9
		Linclosure	30,576	568,872	3,087	195,815	11.11	137	9
		WildsClosure	30,576	269,429	1,035	30,576	4.99	137	9
	DG-Basis	Closure (op)	22,362	861,921	961	63,988	10.24	100	9
		Linclosure	22,362	410,714	3,087	142,283	8.19	100	9
		WildsClosure	22,362	197,705	1,079	22,362	3.77	100	9
glass	Canonical	Closure (op)	**25,664**	1,471,992	**1,023**	163,680	34.31	160	10
		Linclosure (op)	**25,664**	473,734	5,120	233,472	17.36	160	10
		WildsClosure (op)	**25,664**	**235,115**	**1,023**	**25,664**	**8.02**	160	10
	D-Basis	Closure (op)	64,899	2,188,987	**1,023**	163,680	87.46	160	10
		Linclosure	104,270	2,306,507	8,624	376,972	78.18	160	10
		WildsClosure	104,270	973,695	2,788	104,270	73.43	160	10
	DG-Basis	Closure (op)	79,975	2,914,050	2,746	194,477	75.03	120	10
		Linclosure	79,975	1,720,244	8,624	310,169	51.48	120	10
		WildsClosure	79,975	758,641	2,924	79,975	38.99	120	10

Table 13: Total values of `real` **datasets** per all analyzed measures: number of dependencies processed, number of operations on attributes, outer loops, inner loops and computation time in miliseconds. $\lvert\Sigma\rvert$: size of the base. $\lvert\mathcal{U}\rvert$: number of attributes

DB	Base	Algorithm	deps	attrib	outer	inner	time (ms)	$\|\Sigma\|$	$\|\mathcal{U}\|$
zoo	Canonical	Closure (op)	**4,775,936**	1,116,100,136	**262,143**	74,448,612	6,860.40	284	18
		Linclosure (op)	**4,775,936**	135,645,229	2,359,296	196,083,712	2,059.09	284	18
		WildsClosure (op)	**4,775,936**	**67,190,818**	**262,143**	**4,775,936**	**1,335.50**	284	18
	D-Basis	Closure (op)	34,160,220	2,001,499,400	**262,143**	74,186,469	10,468.90	283	18
		Linclosure	35,243,954	1,305,688,423	3,622,889	304,467,236	10,071.40	283	18
		WildsClosure	35,243,954	594,172,388	712,305	35,243,954	5,032.12	283	18
	DG-Basis	Closure (op)	21,244,708	1,846,038,241	684,009	84,963,633	9,770.72	163	18
		Linclosure	21,244,708	783,835,415	3,622,889	172,339,639	6,194.12	163	18
		WildsClosure	21,244,708	348,561,050	756,007	21,244,708	3,163.35	163	18
echocardiogram	Canonical	Closure (op)	**220,928**	22,173,693	**4,095**	2,153,970	378.84	526	12
		Linclosure (op)	**220,928**	5,077,471	24,576	3,676,160	111.64	526	12
		WildsClosure (op)	**220,928**	**2,529,480**	**4,095**	**220,928**	**76.64**	526	12
	D-Basis	Closure (op)	844,550	35,159,262	**4,095**	2,153,970	537.54	526	12
		Linclosure	1,701,742	44,913,136	43,087	6,369,798	853.83	526	12
		WildsClosure	1,701,742	19,958,094	11,456	1,701,742	359.86	526	12
	DG-Basis	Closure (op)	871,771	32,141,145	11,211	1,587,134	462.87	269	12
		Linclosure	871,771	21,920,919	43,087	4,038,451	444.84	269	12
		WildsClosure	871,771	10,267,876	13,244	871,771	193.56	269	12
wine	Canonical	Closure (op)	**3,430,400**	252,229,902	**16,383**	22,510,242	2,139.31	1374	14
		Linclosure (op)	**3,430,400**	94,758,079	114,688	31,342,592	776.18	1374	14
		WildsClosure (op)	**3,430,400**	**47,322,596**	**16,383**	**3,430,400**	**507.31**	1374	14
	D-Basis	Closure (op)	11,224,253	453,572,029	**16,383**	22,510,242	3,139.92	1374	14
		Linclosure	22,290,574	690,859,413	227,575	62,180,227	5,667.09	1374	14
		WildsClosure	22,290,574	303,173,233	48,918	22,290,574	2,386.24	1374	14
	DG-Basis	Closure (op)	17,942,810	590,204,163	47,887	19,656,652	3,758.11	1106	14
		Linclosure	17,942,810	553,496,891	227,575	57,775,494	4,482.14	1106	14
		WildsClosure	17,942,810	250,300,845	49,084	17,942,810	1,863.88	1106	14
mushroom	Canonical	Closure (op)	**42,401,713**	14,176,252,115	**194,303**	696,187,649	69,862.80	3583	22
		Linclosure (op)	**42,401,713**	1,427,400,693	2,597,369	2,571,680,313	27,857.80	3583	22
		WildsClosure (op)	**42,401,713**	**713,497,559**	**194,303**	**42,401,713**	**12,572.50**	3583	22
	D-Basis	Closure (op)	92,114,537	16,140,189,328	**194,303**	696,187,649	75,676.00	3583	22
		Linclosure	120,482,269	4,237,779,900	3,002,944	3,067,841,700	55,107.20	3583	22
		WildsClosure	120,482,269	2,039,463,765	560,134	120,482,269	34,264.20	3583	22
	DG-Basis	Closure (op)	66,096,065	18,063,471,147	532,670	802,630,637	81,469.60	1721	22
		Linclosure	66,096,065	2,311,701,537	3,002,944	1,429,868,871	27,773.50	1721	22
		WildsClosure	66,096,065	1,114,924,873	579,394	66,096,065	17,165.90	1721	22
hepatitis	Canonical	Closure (op)	**40,146,301**	7,808,101,115	**48,575**	400,743,750	37,390.10	8250	20
		Linclosure (op)	**40,146,301**	1,575,946,487	600,761	1,480,968,379	17,917.10	8250	20
		WildsClosure (op)	**40,146,301**	**787,788,973**	**48,575**	**40,146,301**	**9,891.58**	8250	20
	D-Basis	Closure (op)	197,937,704	12,966,658,256	**48,575**	400,743,750	57,813.80	8250	20
		Linclosure	398,620,818	16,414,801,259	969,302	2,257,167,229	109,878.00	8250	20
		WildsClosure	398,620,818	7,874,351,588	145,708	398,620,818	47,157.80	8250	20
	DG-Basis	Closure (op)	131,918,652	6,808,745,482	144,673	163,888,402	30,538.70	2730	20
		Linclosure	131,918,652	5,330,144,435	969,302	826,125,786	36,709.70	2730	20
		WildsClosure	131,918,652	2,564,982,670	193,463	131,918,652	16,425.80	2730	20
waveform	Canonical	Closure (op)	**935,838,999**	90,957,167,084	**194,303**	4,663,660,606	483,737.00	24002	22
		Linclosure (op)	**935,838,999**	41,155,276,963	2,597,369	8,453,500,560	208,834.00	24002	22
		WildsClosure (op)	**935,838,999**	**20,576,799,833**	**194,303**	**935,838,999**	**132,619.00**	24002	22
	D-Basis	Closure (op)	2,967,583,156	164,964,021,905	**194,303**	4,663,660,606	742,195.00	24002	22
		Linclosure	4,663,660,606	229,669,366,112	4,274,666	14,200,829,058	1,217,360.00	24002	22
		WildsClosure	4,663,660,606	99,926,362,986	582,908	4,663,660,606	556,247.00	24002	22
	DG-Basis	Closure (op)	4,663,660,606	224,034,789,490	388,606	4,663,660,606	950,059.00	24002	22
		Linclosure	4,663,660,606	229,678,147,337	4,274,666	14,200,829,058	1,140,670.00	24002	22
		WildsClosure	4,663,660,606	102,585,149,913	582,908	4,663,660,606	495,799.00	24002	22

Table 14: Total values of **real datasets** per all analyzed measures: number of dependencies processed, number of operations on attributes, outer loops, inner loops and computation time in miliseconds. $|\Sigma|$: size of the base. $|\mathcal{U}|$: number of attributes

DB	Base	Algorithm	deps	attrib	outer	inner	time (ms)	$\lvert\Sigma\rvert$	$\lvert\mathcal{U}\rvert$
pen-recognition	Canonical	Closure (op)	**37,626,368**	61,745,162,923	**131,071**	3,992,815,873	302,374.00	30463	17
		Linclosure (op)	**37,626,368**	1,253,902,086	1,114,112	13,662,224,384	105,994.00	30463	17
		WildsClosure (op)	**37,626,368**	**626,648,511**	**131,071**	**37,626,368**	**53,093.10**	30463	17
	D-Basis	Closure (op)	797,945,288	84,552,185,155	**131,071**	3,992,815,873	399,765.00	30463	17
		Linclosure	2,161,555,214	77,829,875,411	1,719,176	20,766,485,149	647,315.00	30463	17
		WildsClosure	2,161,555,214	36,491,350,032	336,291	2,161,555,214	267,251.00	30463	17
	DG-Basis	Closure (op)	1,137,152,337	85,241,002,493	334,992	3,350,711,286	387,928.00	15885	17
		Linclosure	1,137,152,337	39,289,014,325	1,719,176	11,889,063,541	336,843.00	15885	17
		WildsClosure	1,137,152,337	19,195,092,139	375,747	1,137,152,337	145,346.00	15885	17

Table 15: Total values of **`real` datasets** per all analyzed measures: number of dependencies processed, number of operations on attributes, outer loops, inner loops and computation time in miliseconds. $|\Sigma|$: size of the base. $|\mathcal{U}|$: number of attributes

.3 Experiments with Big Datasets

Processed Dependencies		Canonical				D-Basis				DG-basis			
DB	$\lvert\mathcal{U}\rvert$	$\lvert\Sigma\rvert$	CLO	LIN	WILD	$\lvert\Sigma\rvert$	CLO	LIN	WILD	$\lvert\Sigma\rvert$	CLO	LIN	WILD
automobile	26	4176	**146,338,203**	**146,338,203**	**146,338,203**	4040	275,630,561	484,182,775	484,182,775	1848	241,209,353	241,209,353	241,209,353
fd-reduced-30	26	54363	**2,513,337,795**	**2,513,337,795**	**2,513,337,795**	35445	4,392,161,319	8,598,892,056	8,598,892,056	35445	8,598,892,056	8,598,892,056	8,598,892,056
flight 1k 30c-sub	19	2473	**78,497,133**	**78,497,133**	**78,497,133**	1533	136,876,994	230,797,163	230,797,163	889	136,272,183	136,272,183	136,272,183
horse	28	128726	**1,777,335,359**	**1,777,335,359**	**1,777,335,359**	128726	5,349,880,831	13,103,894,345	13,103,894,345	40969	4,406,604,018	4,406,604,018	4,406,604,018
soybean-small	21	4606	98,068,246	98,068,246	98,068,246	3752	198,915,220	273,652,299	273,652,299	585	**48,919,716**	**48,919,716**	**48,919,716**
Average	24.00	38,868.80	922,715,347.20	922,715,347.20	922,715,347.20	34,699.20	2,070,692,985.00	4,538,283,727.60	4,538,283,727.60	15,947.20	2,686,379,465.20	2,686,379,465.20	2,686,379,465.20

Table 16: Totals of the measure **Processed Dependencies** for all **`big` datasets**. In bold are the minimal values. The last line contains the average of each measure: the sum of all values for each pair (Base $\times$ Algorithm) divided by the number of datasets.

Processed Attributes		Canonical				D-Basis				DG-basis			
DB	$\lvert\mathcal{U}\rvert$	$\lvert\Sigma\rvert$	CLO	LIN	WILD	$\lvert\Sigma\rvert$	CLO	LIN	WILD	$\lvert\Sigma\rvert$	CLO	LIN	WILD
automobile	26	4176	20,751,907,433	7,012,286,575	**6,485,511,016**	4040	27,761,426,778	24,423,087,681	17,520,620,895	1848	28,218,557,975	11,884,575,773	9,985,088,787
fd-reduced-30	26	54363	266,858,354,116	130,628,000,139	**94,775,417,559**	35445	306,399,080,378	494,505,663,710	237,920,818,017	35445	488,145,904,155	494,422,888,141	243,095,983,862
flight 1k 30c-sub	19	2473	9,699,721,505	**2,913,711,424**	3,177,277,727	1533	9,186,719,212	9,409,096,637	6,144,580,877	889	10,888,660,602	5,218,195,239	4,343,105,671
horse	28	128726	722,948,915,645	**95,737,616,809**	183,342,486,997	128726	956,723,760,792	730,825,820,743	583,831,000,283	40969	721,629,052,556	239,046,260,694	227,521,918,246
soybean-small	21	4606	20,252,151,381	3,915,010,918	5,936,668,041	3752	21,554,884,878	11,983,512,765	12,941,572,827	585	7,439,811,864	**1,981,127,453**	2,362,347,768
Average	24.00	38,868.80	208,102,210,016.00	48,041,325,173.00	58,743,472,268.00	34,699.20	264,325,174,407.60	254,229,436,307.20	171,671,718,579.80	15,947.20	251,264,397,430.40	150,510,609,460.00	97,461,688,866.80

Table 17: Totals of the measure **Processed Attributes** for all **`big` datasets**. In bold are the minimal values. The last line contains the average of each measure: the sum of all values for each pair (Base $\times$ Algorithm) divided by the number of datasets.

Running Time		Canonical				D-Basis				DG-basis			
DB	$\lvert\mathcal{U}\rvert$	$\lvert\Sigma\rvert$	CLO	LIN	WILD	$\lvert\Sigma\rvert$	CLO	LIN	WILD	$\lvert\Sigma\rvert$	CLO	LIN	WILD
automobile	26	4176	130,315.84	49,165.77	**33,975.41**	4040	154,351.41	162,763.93	93,454.99	1848	139,327.49	87,023.36	50,922.78
fd-reduced-30	26	54363	1,871,959.80	807,552.77	**485,580.11**	35445	1,675,370.80	3,561,460.30	1,274,132.10	35445	2,430,707.90	2,909,001.50	1,202,128.60
flight 1k 30c-sub	19	2473	64,336.54	23,621.68	**15,806.26**	1533	51,933.07	67,070.25	34,146.21	889	53,860.44	42,195.91	22,146.86
horse	28	128726	2,746,040.00	2,044,557.00	**608,416.50**	128726	3,249,807.00	6,730,818.10	2,208,924.40	40969	2,250,584.90	1,831,404.17	828,612.60
soybean-small	21	4606	138,485.57	53,499.65	33,963.21	3752	133,929.09	122,048.52	74,087.06	585	42,873.50	21,113.99	**14,833.41**
Average	24.00	38,868.80	990,227.55	595,679.37	235,548.30	34,699.20	1,053,078.27	2,128,832.22	736,948.95	15,947.20	983,470.85	978,147.79	423,728.85

Table 18: Totals of the measure **Running Time** for all **`big` datasets**. In bold are the minimal values. The last line contains the average of each measure: the sum of all values for each pair (Base $\times$ Algorithm) divided by the number of datasets.

DB	Base	Algorithm	deps	attrib	outer	inner	time (ms)	$\|\Sigma\|$	$\|\mathcal{U}\|$
automobile	Canonical	Closure (op)	**146,338,203**	20,751,907,433	**250,000**	1,044,000,000	130,315.84	4176	26
		Linclosure (op)	**146,338,203**	7,012,286,575	3,249,034	2,112,128,825	49,165.77	4176	26
		WildsClosure (op)	**146,338,203**	**6,485,511,016**	**250,000**	**146,338,203**	**33,975.41**	4176	26
	D-Basis	Closure (op)	275,630,561	27,761,426,778	**250,000**	1,010,000,000	154,351.41	4040	26
		Linclosure	484,182,775	24,423,087,681	5,006,083	2,967,106,679	162,763.93	4040	26
		WildsClosure	484,182,775	17,520,620,895	729,932	484,182,775	93,454.99	4040	26
	DG-Basis	Closure (op)	241,209,353	28,218,557,975	750,069	929,919,023	139,327.49	1848	26
		Linclosure	241,209,353	11,884,575,773	5,006,083	2,346,340,130	87,023.36	1848	26
		WildsClosure	241,209,353	9,985,088,787	1,000,983	241,209,353	50,922.78	1848	26
fd-reduced-30	Canonical	Closure (op)	**2,513,337,795**	266,858,354,116	**250,000**	13,590,750,000	1,871,959.80	54363	26
		Linclosure (op)	**2,513,337,795**	130,628,000,139	3,248,008	20,372,603,062	807,552.77	54363	26
		WildsClosure (op)	**2,513,337,795**	**94,775,417,559**	**250,000**	**2,513,337,795**	**485,580.11**	54363	26
	D-Basis	Closure (op)	4,392,161,319	306,399,080,378	**250,000**	8,861,250,000	1,675,370.80	35445	26
		Linclosure	8,598,892,056	494,505,663,710	6,322,353	25,847,006,545	3,561,460.30	35445	26
		WildsClosure	8,598,892,056	237,920,818,017	740,023	8,598,892,056	1,274,132.10	35445	26
	DG-Basis	Closure (op)	8,598,892,056	488,145,904,155	691,275	9,422,196,309	2,430,707.90	35445	26
		Linclosure	8,598,892,056	494,422,888,141	6,322,353	25,873,833,433	2,909,001.50	35445	26
		WildsClosure	8,598,892,056	243,095,983,862	740,044	8,598,892,056	1,202,128.60	35445	26
flight 1k 30c-sub	Canonical	Closure (op)	**78,497,133**	9,699,721,505	**250,000**	618,250,000	64,336.54	2473	19
		Linclosure (op)	**78,497,133**	**2,913,711,424**	2,375,729	1,300,942,602	23,621.68	2473	19
		WildsClosure (op)	**78,497,133**	3,177,277,727	**250,000**	**78,497,133**	**15,806.26**	2473	19
	D-Basis	Closure (op)	136,876,994	9,186,719,212	**250,000**	383,250,000	51,933.07	1533	19
		Linclosure	230,797,163	9,409,096,637	3,821,047	1,225,304,532	67,070.25	1533	19
		WildsClosure	230,797,163	6,144,580,877	734,482	230,797,163	34,146.21	1533	19
	DG-Basis	Closure (op)	136,272,183	10,888,660,602	711,857	409,047,989	53,860.44	889	19
		Linclosure	136,272,183	5,218,195,239	3,821,047	1,220,706,377	42,195.91	889	19
		WildsClosure	136,272,183	4,343,105,671	922,688	136,272,183	22,146.86	889	19
horse	Canonical	Closure (op)	**1,777,335,359**	722,948,915,645	**250,000**	32,181,500,000	2,746,040.00	128726	28
		Linclosure (op)	**1,777,335,359**	**95,737,616,809**	3,500,841	114,643,191,658	2,044,557.00	128726	28
		WildsClosure (op)	**1,777,335,359**	183,342,486,997	**250,000**	**1,777,335,359**	**608,416.50**	128726	28
	D-Basis	Closure (op)	5,349,880,831	956,723,760,792	**250,000**	32,181,500,000	3,249,807.00	128726	28
		Linclosure	13,103,894,345	730,825,820,743	5,393,860	175,140,719,609	6,730,818.10	128726	28
		WildsClosure	13,103,894,345	583,831,000,283	702,255	13,103,894,345	2,208,924.40	128726	28
	DG-Basis	Closure (op)	4,406,604,018	721,629,052,556	696,855	20,750,799,614	2,250,584.90	40969	28
		Linclosure	4,406,604,018	239,046,260,694	5,393,860	71,665,462,000	1,831,404.17	40969	28
		WildsClosure	4,406,604,018	227,521,918,246	911,633	4,406,604,018	828,612.60	40969	28
soybean-small	Canonical	Closure (op)	98,068,246	20,252,151,381	**250,000**	1,151,500,000	138,485.57	4606	21
		Linclosure (op)	98,068,246	3,915,010,918	2,625,217	3,204,601,264	53,499.65	4606	21
		WildsClosure (op)	98,068,246	5,936,668,041	**250,000**	98,068,246	33,963.21	4606	21
	D-Basis	Closure (op)	198,915,220	21,554,884,878	**250,000**	938,000,000	133,929.09	3752	21
		Linclosure	273,652,299	11,983,512,765	3,618,799	3,337,284,572	122,048.52	3752	21
		WildsClosure	273,652,299	12,941,572,827	719,874	273,652,299	74,087.06	3752	21
	DG-Basis	Closure (op)	**48,919,716**	7,439,811,864	702,207	327,457,529	42,873.50	585	21
		Linclosure	**48,919,716**	**1,981,127,453**	3,618,799	635,608,309	21,113.99	585	21
		WildsClosure	**48,919,716**	2,362,347,768	903,216	**48,919,716**	**14,833.41**	585	21

Table 19: Total values of **big datasets** per all analyzed measures: number of dependencies processed, number of operations on attributes, outer loops, inner loops and computation time in miliseconds. $|\Sigma|$: size of the base. $|\mathcal{U}|$: number of attributes

.4 Experiments with Synthetic Datasets

Processed Dependencies		Canonical				D-Basis				DG-basis			
DB	$\lvert\mathcal{U}\rvert$	$\lvert\Sigma\rvert$	CLO	LIN	WILD	$\lvert\Sigma\rvert$	CLO	LIN	WILD	$\lvert\Sigma\rvert$	CLO	LIN	WILD
freq-20	8	94	**5,212,799,106**	**5,212,799,106**	**5,212,799,106**	73	10,401,808,819	15,546,041,657	15,546,041,657	70	8,015,214,361	8,015,214,361	8,015,214,361
freq-30	9	97	9,891,631,345	9,891,631,345	9,891,631,345	89	21,908,681,341	32,423,482,207	32,423,482,207	60	**7,937,914,879**	**7,937,914,879**	**7,937,914,879**
freq-40	9	83	15,029,411,207	15,029,411,207	15,029,411,207	59	36,902,580,328	57,555,592,285	57,555,592,285	45	**9,249,194,332**	**9,249,194,332**	**9,249,194,332**
freq-50	10	96	13,087,833,980	13,087,833,980	13,087,833,980	65	33,597,965,565	52,368,712,857	52,368,712,857	36	**6,300,888,371**	**6,300,888,371**	**6,300,888,371**
freq-60	9	83	11,405,747,857	11,405,747,857	11,405,747,857	62	29,120,747,130	47,391,539,025	47,391,539,025	34	**5,838,397,926**	**5,838,397,926**	**5,838,397,926**
freq-70	10	46	4,881,897,956	4,881,897,956	4,881,897,956	37	13,134,647,361	20,565,631,711	20,565,631,711	28	**3,360,502,662**	**3,360,502,662**	**3,360,502,662**
freq-80	9	7	**1,031,814,069**	**1,031,814,069**	**1,031,814,069**	7	1,733,387,643	2,571,312,784	2,571,312,784	7	1,045,011,165	1,045,011,165	1,045,011,165
Average	9.14	72.29	8,648,733,645.71	8,648,733,645.71	8,648,733,645.71	56.00	20,971,402,598.14	32,631,758,932.29	32,631,758,932.29	40.00	5,963,874,813.71	5,963,874,813.71	5,963,874,813.71

Table 20: Totals of the measure **Processed Dependencies** for all **`synthetic` datasets**. In bold are the minimal values. The last line contains the average of each measure: the sum of all values for each pair (Base $\times$ Algorithm) divided by the number of datasets.

Processed Attributes		Canonical				D-Basis				DG-basis			
DB	$\lvert\mathcal{U}\rvert$	$\lvert\Sigma\rvert$	CLO	LIN	WILD	$\lvert\Sigma\rvert$	CLO	LIN	WILD	$\lvert\Sigma\rvert$	CLO	LIN	WILD
freq-20	8	94	359,187,495,023	224,481,835,032	**147,537,030,145**	73	525,728,453,710	749,890,638,698	347,411,681,585	70	364,684,847,087	382,745,032,146	180,654,708,036
freq-30	9	97	805,373,304,005	445,764,672,798	300,594,032,465	89	1,187,725,800,419	1,624,334,889,699	766,682,485,051	60	376,496,589,311	390,946,866,597	**185,389,205,775**
freq-40	9	83	1,367,129,516,657	684,812,209,279	482,059,526,380	59	2,086,884,002,945	2,875,262,340,711	1,381,768,113,430	45	446,521,171,237	451,305,943,635	**217,788,155,643**
freq-50	10	96	1,341,919,564,966	610,168,404,249	442,539,057,582	65	1,982,075,056,002	2,652,744,304,735	1,297,922,612,193	36	314,720,728,052	303,763,838,808	**151,858,388,563**
freq-60	9	83	1,315,681,986,990	526,300,944,900	408,842,511,054	62	1,783,087,145,002	2,382,128,252,077	1,189,020,362,964	34	299,632,667,233	272,419,088,073	**141,923,352,880**
freq-70	10	46	631,490,842,023	219,141,302,079	183,812,027,247	37	826,110,409,012	1,018,549,191,066	530,354,436,506	28	186,870,975,924	154,353,838,383	**84,566,497,823**
freq-80	9	7	119,547,382,552	41,284,120,229	35,461,561,625	7	108,683,234,073	113,064,200,651	65,849,100,911	7	66,363,837,720	42,486,991,737	**25,935,545,769**
Average	9.14	72.29	848,618,584,602.29	393,136,212,652.29	285,835,106,642.57	56.00	1,214,327,728,737.57	1,630,853,402,519.57	797,001,256,091.43	40.00	293,612,973,794.86	285,431,657,054.14	141,159,407,784.14

Table 21: Totals of the measure **Processed Attributes** for all **`synthetic` datasets**. In bold are the minimal values. The last line contains the average of each measure: the sum of all values for each pair (Base $\times$ Algorithm) divided by the number of datasets.

Running Time		Canonical				D-Basis				DG-basis			
DB	$\lvert\mathcal{U}\rvert$	$\lvert\Sigma\rvert$	CLO	LIN	WILD	$\lvert\Sigma\rvert$	CLO	LIN	WILD	$\lvert\Sigma\rvert$	CLO	LIN	WILD
freq-20	8	94	2,271,764.54	943,877.34	**653,738.83**	73	2,653,558.74	3,686,859.71	1,687,783.04	70	1,778,722.39	1,834,836.12	821,769.40
freq-30	9	97	4,714,365.25	2,095,123.18	1,400,932.87	89	5,566,623.20	8,478,313.82	3,864,571.61	60	1,805,848.36	1,959,385.85	**917,928.39**
freq-40	9	83	7,281,439.73	3,197,068.20	2,119,279.66	59	9,067,399.45	14,590,305.71	6,452,559.01	45	2,045,320.30	2,199,257.24	**1,043,838.34**
freq-50	10	96	6,850,468.95	3,034,666.16	1,970,513.29	65	8,538,367.96	13,717,193.64	5,928,738.60	36	1,468,953.57	1,561,211.67	**746,062.45**
freq-60	9	83	6,802,694.42	2,865,914.20	1,926,675.01	62	7,735,828.56	12,476,680.80	5,514,010.59	34	1,425,048.03	1,487,465.73	**733,641.72**
freq-70	10	46	3,184,726.06	1,331,646.39	923,214.90	37	3,498,613.55	5,433,222.89	2,467,409.12	28	876,832.28	874,062.93	**450,919.64**
freq-80	9	7	624,769.76	262,888.33	171,354.61	7	503,048.26	670,720.93	326,988.77	7	311,473.50	278,028.99	**139,858.06**
Average	9.14	72.29	4,532,889.82	1,961,597.69	1,309,387.03	56.00	5,366,205.67	8,436,185.36	3,748,865.82	40.00	1,387,456.92	1,456,321.22	693,431.14

Table 22: Totals of the measure **Running Time** for all **`synthetic` datasets**. In bold are the minimal values. The last line contains the average of each measure: the sum of all values for each pair (Base $\times$ Algorithm) divided by the number of datasets.

DB	Base	Algorithm	deps	attrib	outer	inner	time (ms)	$\|\Sigma\|$	$\|\mathcal{U}\|$
freq-20	Canonical	Closure (op)	**5,212,799,106**	359,187,495,023	**7,466,795**	19,851,314,800	2,271,764.54	94	8
		Linclosure (op)	**5,212,799,106**	224,481,835,032	78,270,450	26,307,220,814	943,877.34	94	8
		WildsClosure (op)	**5,212,799,106**	**147,537,030,145**	**7,466,795**	**5,212,799,106**	**653,738.83**	94	8
	D-Basis	Closure (op)	10,401,808,819	525,728,453,710	**7,466,795**	15,557,583,456	2,653,558.74	73	8
		Linclosure	15,546,041,657	749,890,638,698	145,459,682	37,381,193,721	3,686,859.71	73	8
		WildsClosure	15,546,041,657	347,411,681,585	22,441,247	15,546,041,657	1,687,783.04	73	8
	DG-Basis	Closure (op)	8,015,214,361	364,684,847,087	20,233,952	8,485,697,523	1,778,722.39	70	8
		Linclosure	8,015,214,361	382,745,032,146	145,459,682	17,050,158,455	1,834,836.12	70	8
		WildsClosure	8,015,214,361	180,654,708,036	22,747,745	8,015,214,361	821,769.40	70	8
freq-30	Canonical	Closure (op)	9,891,631,345	805,373,304,005	**10,685,291**	41,174,839,100	4,714,365.25	97	9
		Linclosure (op)	9,891,631,345	445,764,672,798	122,853,482	65,472,750,909	2,095,123.18	97	9
		WildsClosure (op)	9,891,631,345	300,594,032,465	**10,685,291**	9,891,631,345	1,400,932.87	97	9
	D-Basis	Closure (op)	21,908,681,341	1,187,725,800,419	**10,685,291**	32,477,776,381	5,566,623.20	89	9
		Linclosure	32,423,482,207	1,624,334,889,699	218,427,276	88,847,339,853	8,478,313.82	89	9
		WildsClosure	32,423,482,207	766,682,485,051	32,127,653	32,423,482,207	3,864,571.61	89	9
	DG-Basis	Closure (op)	**7,937,914,879**	376,496,589,311	28,359,788	8,509,809,701	1,805,848.36	60	9
		Linclosure	**7,937,914,879**	390,946,866,597	218,427,276	17,954,010,728	1,959,385.85	60	9
		WildsClosure	**7,937,914,879**	**185,389,205,775**	33,254,314	**7,937,914,879**	**917,928.39**	60	9
freq-40	Canonical	Closure (op)	15,029,411,207	1,367,129,516,657	**15,801,131**	69,717,808,385	7,281,439.73	83	9
		Linclosure (op)	15,029,411,207	684,812,209,279	184,435,058	121,029,394,838	3,197,068.20	83	9
		WildsClosure (op)	15,029,411,207	482,059,526,380	**15,801,131**	15,029,411,207	2,119,279.66	83	9
	D-Basis	Closure (op)	36,902,580,328	2,086,884,002,945	**15,801,131**	57,774,124,924	9,067,399.45	59	9
		Linclosure	57,555,592,285	2,875,262,340,711	327,507,245	171,941,582,414	14,590,305.71	59	9
		WildsClosure	57,555,592,285	1,381,768,113,430	47,575,664	57,555,592,285	6,452,559.01	59	9
	DG-Basis	Closure (op)	**9,249,194,332**	446,521,171,237	43,922,403	10,239,543,580	2,045,320.30	45	9
		Linclosure	**9,249,194,332**	451,305,943,635	327,507,245	22,374,436,212	2,199,257.24	45	9
		WildsClosure	**9,249,194,332**	**217,788,155,643**	51,732,610	**9,249,194,332**	**1,043,838.34**	45	9
freq-50	Canonical	Closure (op)	13,087,833,980	1,341,919,564,966	**12,961,963**	65,425,420,845	6,850,468.95	96	10
		Linclosure (op)	13,087,833,980	610,168,404,249	159,713,580	128,296,212,267	3,034,666.16	96	10
		WildsClosure (op)	13,087,833,980	442,539,057,582	**12,961,963**	13,087,833,980	1,970,513.29	96	10
	D-Basis	Closure (op)	33,597,965,565	1,982,075,056,002	**12,961,963**	52,849,340,835	8,538,367.96	65	10
		Linclosure	52,368,712,857	2,652,744,304,735	274,280,989	172,777,536,899	13,717,193.64	65	10
		WildsClosure	52,368,712,857	1,297,922,612,193	39,111,159	52,368,712,857	5,928,738.60	65	10
	DG-Basis	Closure (op)	**6,300,888,371**	314,720,728,052	37,104,654	7,237,165,395	1,468,953.57	36	10
		Linclosure	**6,300,888,371**	303,763,838,808	274,280,989	17,910,084,098	1,561,211.67	36	10
		WildsClosure	**6,300,888,371**	**151,858,388,563**	45,680,682	**6,300,888,371**	**746,062.45**	36	10
freq-60	Canonical	Closure (op)	11,405,747,857	1,315,681,986,990	**16,681,579**	65,214,649,092	6,802,694.42	83	9
		Linclosure (op)	11,405,747,857	526,300,944,900	202,654,289	135,592,084,930	2,865,914.20	83	9
		WildsClosure (op)	11,405,747,857	408,842,511,054	**16,681,579**	11,405,747,857	1,926,675.01	83	9
	D-Basis	Closure (op)	29,120,747,130	1,783,087,145,002	**16,681,579**	48,396,878,299	7,735,828.56	62	9
		Linclosure	47,391,539,025	2,382,128,252,077	349,306,575	168,212,549,694	12,476,680.80	62	9
		WildsClosure	47,391,539,025	1,189,020,362,964	51,255,239	47,391,539,025	5,514,010.59	62	9
	DG-Basis	Closure (op)	**5,838,397,926**	299,632,667,233	49,492,755	7,188,486,390	1,425,048.03	34	9
		Linclosure	**5,838,397,926**	272,419,088,073	349,306,575	19,346,354,085	1,487,465.73	34	9
		WildsClosure	**5,838,397,926**	**141,923,352,880**	64,932,810	**5,838,397,926**	**733,641.72**	34	9
freq-70	Canonical	Closure (op)	4,881,897,956	631,490,842,023	**13,367,915**	31,271,743,958	3,184,726.06	46	10
		Linclosure (op)	4,881,897,956	219,141,302,079	161,508,945	70,384,656,875	1,331,646.39	46	10
		WildsClosure (op)	4,881,897,956	183,812,027,247	**13,367,915**	4,881,897,956	923,214.90	46	10
	D-Basis	Closure (op)	13,134,647,361	826,110,409,012	**13,367,915**	22,224,621,507	3,498,613.55	37	10
		Linclosure	20,565,631,711	1,018,549,191,066	268,462,834	82,094,195,538	5,433,222.89	37	10
		WildsClosure	20,565,631,711	530,354,436,506	42,518,890	20,565,631,711	2,467,409.12	37	10
	DG-Basis	Closure (op)	**3,360,502,662**	186,870,975,924	40,183,894	4,835,014,164	876,832.28	28	10
		Linclosure	**3,360,502,662**	154,353,838,383	268,462,834	13,480,844,928	874,062.93	28	10
		WildsClosure	**3,360,502,662**	**84,566,497,823**	53,804,682	**3,360,502,662**	**450,919.64**	28	10

Table 23: Total values of **synthetic datasets** per all analyzed measures: number of dependencies processed, number of operations on attributes, outer loops, inner loops and computation time in miliseconds. $|\Sigma|$: size of the base. $|\mathcal{U}|$: number of attributes

DB	Base	Algorithm	deps	attrib	outer	inner	time (ms)	$\|\Sigma\|$	$\|\mathcal{U}\|$
freq-80	Canonical	Closure (op)	**1,031,814,069**	119,547,382,552	**10,167,211**	6,283,975,037	624,769.76	7	9
		Linclosure (op)	**1,031,814,069**	41,284,120,229	114,968,946	12,981,549,441	262,888.33	7	9
		WildsClosure (op)	**1,031,814,069**	35,461,561,625	**10,167,211**	**1,031,814,069**	171,354.61	7	9
	D-Basis	Closure (op)	1,733,387,643	108,683,234,073	**10,167,211**	3,530,986,829	503,048.26	7	9
		Linclosure	2,571,312,784	113,064,200,651	180,917,881	10,773,796,644	670,720.93	7	9
		WildsClosure	2,571,312,784	65,849,100,911	33,885,956	2,571,312,784	326,988.77	7	9
	DG-Basis	Closure (op)	1,045,011,165	66,363,837,720	30,572,033	2,178,372,067	311,473.50	7	9
		Linclosure	1,045,011,165	42,486,991,737	180,917,881	4,721,590,468	278,028.99	7	9
		WildsClosure	1,045,011,165	**25,935,545,769**	40,169,486	1,045,011,165	**139,858.06**	7	9

Table 24: Total values of **`synthetic` datasets** per all analyzed measures: number of dependencies processed, number of operations on attributes, outer loops, inner loops and computation time in miliseconds. $|\Sigma|$: size of the base. $|\mathcal{U}|$: number of attributes