Efficient Derivation of Numerical Dependencies

Paolo Ciaccia, Matteo Golfarelli, Stefano Rizzi*

DISI - Università di Bologna, Italy

Abstract

Numerical dependencies (NDs) are database constraints that limit the number of distinct Y-values that can appear together with any X-value, where both X and Y are sets of attributes in a relation schema. While it is known that NDs are not finitely axiomatizable, there is no study on how to efficiently derive NDs using a set of sound (yet necessarily incomplete) rules. In this paper, after proving that solving the entailment problem for NDs using the chase procedure has exponential space complexity, we show that, given a set of inference rules similar to those used for functional dependencies, the membership problem for NDs is NP-hard. We then provide a graph-based characterization of NDs, which is exploited to design an efficient branch & bound algorithm for ND derivation. Our algorithm adopts several optimization strategies that provide considerable speed-up over a naïve approach, as confirmed by the results of extensive tests we made for efficiency and effectiveness using six different datasets.

Keywords: numerical dependency, membership problem, branch and bound algorithm, projection cardinality estimation

1. Introduction

Reasoning with database constraints has several practical applications, including database design, query processing and optimization, schema matching, and data lineage and repair. Consequently, understanding the properties of a given type of constraints has always been a major topic in database theory. Central questions that arise with the study of any constraint type are whether a given constraint is logically implied by a set of constraints (the *entailment* problem), whether the constraint type is *axiomatizable* (which enables a syntactic derivation of constraints through rules), and what is the *complexity* of entailment and derivation.

In this paper we tackle the problem of efficiently reasoning with the so-called *numerical dependencies* (NDs), introduced by Grant and Minker [1]. Intuitively,

*Corresponding author

Email addresses: paolo.ciaccia@unibo.it (Paolo Ciaccia), matteo.golfarelli@unibo.it (Matteo Golfarelli), stefano.rizzi@unibo.it (Stefano Rizzi)

given two sets of attributes X and Y from a same relation schema, there is an ND from X to Y (denoted $X \xrightarrow{k} Y$) if each value of X can never be associated to more than k distinct values of Y (where $k \geq 1$ is called the *weight* of the ND). NDs are a natural generalization of functional dependencies, which are obtained when $k = 1$.

Reasoning with NDs has a number of applications in the database field, among which:

- Estimating the projection size of a relation, i.e., the number of distinct values over a subset of its attributes, is a frequent problem in database applications [2]. When data are not available (e.g., at design time), statistical techniques like those based on histograms [3] or sampling [4] cannot be used to this end, so probabilistic approaches must be followed. Such approaches rely on the assumption that the attributes are mutually independent; when this is not the case, NDs enable inter-attribute cardinality constraints to be captured, thus noticeably improving the accuracy in projection size estimation [5].

- Accurately estimating the cardinality of aggregate views has a crucial importance in the field of data warehousing, with reference to logical and physical design as well as query processing and optimization [6]. In particular, view materialization may significantly benefit from using NDs since the algorithms that select the best aggregate views to be materialized are based on view cardinality estimates [7, 5].

- In the Entity-Relationship model, cardinality ratio constraints are used to impose restrictions on the mappings between entities and relationships. In this context, reasoning with NDs allows designers to verify the soundness of schemata, e.g., to identify inconsistencies among cardinality constraints specified for a given relationship [8] or to ensure that no entity or relationship is compelled to be empty in all legal instances of a schema [9].

- The study of integrity constraints has also been recognized as one of the most important yet challenging areas of XML research. XML constraints have a wide range of applications ranging from schema design, query optimization, efficient storing and updating, data exchange and integration, to data cleaning [10]. Among XML constraints, NDs are a highly useful and natural class.

- NDs can also be used as a facility for incomplete specifications in design and planning, to represent indefinite or approximate information in relations [11, 12], as well as for efficient query processing in nondeterministic databases [13].

Unfortunately NDs are not finitely axiomatizable, thus no finite set of sound and complete inference rules exists for them [1]. Indeed, this negative result has prevented further research aiming to deepen the understanding and practical

use of NDs, which has left several issues related to this kind of dependencies unexplored.

Given a set of NDs Δ and two sets of attributes X and Y, the problem we face in this paper is to determine the minimum value of k, if there exists one, such that the $X \xrightarrow{k} Y$ holds. The main contributions we give to this end can be summarized as follows:

1. We first address the entailment problem for NDs; we adapt the classical chase procedure and show that in the worst case exponential space is required to solve the problem (Section 3.1).

2. This leads us to face the derivation problem in Sections 3.2 and 4. We first prove that, given a set of sound inference rules similar to those used for functional dependencies, the derivation problem for NDs is NP-hard. We also prove that such rules are complete except for the value of the minimal weight k, so they can always be used to find an upper bound of k, if it exists, which is necessary to successfully apply the chase. Finally, we provide a partial characterization of the cases in which our rules are guaranteed to find the minimum weight.

3. The NP-hardness of the derivation problem motivates the investigation of efficient algorithms that achieve considerable speed-up over a naïve derivation approach. To this end we introduce the concept of *tight closure* of a set of NDs, that intuitively includes, among the NDs that can be derived through the inference rules, only the "interesting" ones (Section 3.2.1). Then, in Section 5 we provide a graph-based characterization (called *ND-graph*) of the tight closure

4. Finally, we exploit the ND-graph characterization to design an efficient branch & bound algorithm, BBND, that adopts several optimization techniques (Section 6). In Section 7 we evaluate BBND for efficiency (how scalable it is and how effective the different optimization techniques and enumeration strategies are, even depending on the NDs topology) and effectiveness (how NDs improve the cardinality bound of projections).

The paper outline is completed by Section 2, that introduces the notational background, Section 8, that surveys the related literature and draws the conclusions, and by an Appendix, that gives the proofs of theorems and lemmas.

2. Background

In this section we review the basic concepts of relational databases and constraints that are needed for the paper.

We denote by $R(U)$ a relation schema, where R is the name of the relation and $U = \{A_1, \ldots, A_n\}$ is a set of attributes. Following a standard convention, uppercase letters from the beginning (ending) of the alphabet denote single

(respectively, sets of) attributes. We use concatenation for forming sets of attributes, thus writing ABC for $\{A, B, C\}$, and for denoting union, thus XY stands for $X \cup Y$. Each attribute $A_i \in U$ is associated with a countable domain, denoted with $dom(A_i)$. A tuple t over a schema $R(U)$ is a function that associates each $A_i \in U$ with a value of $dom(A_i)$. This also applies to sets of attributes, that is, for each $X \subseteq U$, $dom(X) = \prod_{A_i \in X} dom(A_i)$ denotes the Cartesian product of the domains of the attributes in X, and an X-value is any value from $dom(X)$. A relation r over a schema $R(U)$, also called an instance of $R(U)$, is a finite set of tuples over $R(U)$. The cardinality of r, denoted $|r|$, is the number of tuples in r. Given a tuple t over $R(U)$, the projection of t on $X \subseteq U$, denoted $t[X]$, is the tuple obtained from t by considering only the attributes in X. Similarly, $r[X]$ is the projection of r on X, i.e., $r[X] = \{t[X] | t \in r\}$.

Given a *constraint* δ over $R(U)$, $Sat(\delta)$ denotes the set of *legal* instances of $R(U)$ with respect to δ, i.e., those instances that satisfy δ. This naturally applies to a set of constraints Δ, i.e., $r \in Sat(\Delta)$ iff $r \in Sat(\delta)$ for each $\delta \in \Delta$. A set of constraints Δ *entails* a constraint δ, written $\Delta \models \delta$, if $r \in Sat(\Delta)$ implies $r \in Sat(\delta)$ for each r. Finally, the (semantic) closure Δ^+ of Δ is the set of all constraints entailed by Δ. Given a constraint δ and a set of constraints Δ, the *membership problem* consists in determining if $\delta \in \Delta^+$.

Given a set of inference rules I, δ is *derivable* from Δ using I, denoted $\Delta \vdash_I \delta$, if there exists a finite sequence of derivation steps such that at step i $(i = 1, \ldots, p)$ a constraint δ_i is generated using a rule in I and the constraints in $\Delta \cup \{\delta_j, j < i\}$, with $\delta_p = \delta$. The syntactic closure of Δ using I, denoted Δ_I^+, is the set of constraints that are derivable from Δ using I. The inference rules I are *sound* if $\Delta_I^+ \subseteq \Delta^+$ and *complete* if $\Delta^+ \subseteq \Delta_I^+$.

3. Numerical Dependencies

Functional dependencies (FDs) are among the most common types of relational database constraints. The FD $X \rightarrow Y$ (X determines Y) is satisfied by an instance r of $R(XYZ)$ if any two tuples with the same X-value also have the same Y-value, that is, each X-value present in r is associated with a single Y-value. It is remarkable that the membership problem for FDs can be easily solved in linear time. Numerical dependencies [1] are a generalization of FDs, in which a restriction is posed on the number of Y-values that can be associated with any X-value.

Definition 1 (Numerical Dependencies). *Given $R(XYZ)$ and a finite integer $k \geq 1$, we say an instance r of $R(XYZ)$ satisfies the* numerical dependency *(ND) $\delta : X \xrightarrow{k} Y$ if for any $k+1$ tuples $t_1, \ldots, t_{k+1}$ in r such that $t_1[X] = \ldots = t_{k+1}[X]$ there are at least two of these tuples, t_i and t_j, $i \neq j$, for which it is $t_i[Y] = t_j[Y]$. In this case we say that δ is an ND* from X to Y with weight k, *the latter also denoted as $w(\delta)$.*

Clearly, an FD is just a particular case of ND with $k = 1$, i.e., $X \xrightarrow{1} Y \equiv X \rightarrow Y$.

The connection between NDs and projection cardinalities is established by the following basic fact, which directly follows from Definition 1:

Lemma 1. *Let X and Y be two sets of attributes. If an instance r of $R(XYZ)$ satisfies the ND $X \xrightarrow{k} Y$, then it holds that $|r[XY]| \leq k \cdot |r[X]|$.*

In particular, *cardinality constraints* on sets of attributes can be specified using NDs of the form $\perp \xrightarrow{k} Y$, where $\perp$ denotes the empty set of attributes [1]. Conventionally assuming $|r[\perp]| = 1$, this is equivalent to asserting that the cardinality of the projection on Y can never exceed k: $|r[Y]| \leq k \ \forall r$.

Example 1. Let $U = ABCDE$. Two possible NDs over $R(U)$ are $A \xrightarrow{3} BC$ and $BC \xrightarrow{5} DE$. The first ND states that each value of A can never be associated to more than 3 distinct values of BC, the second that each BC-value can never appear with more than 5 distinct DE-values. The two NDs guarantee that in each legal instance r of $R(U)$ the cardinality of $r[ABC]$ will never be higher than 3 times that of $r[A]$ and that the cardinality of $r[BCDE]$ will not be higher than 5 times that of $r[BC]$, respectively. ■

3.1. Entailment of NDs

In this section we consider the problem of determining if an ND δ is entailed by a set of NDs Δ. To this end we adapt the classical chase procedure [14] to NDs, and show that exponential space is required in the worst case to solve the problem. We preliminarily observe that NDs are a particular case of *Disjunctive Equality-Generating Dependencies* (DEGDs) [15], since an ND $\delta : X \xrightarrow{k} Y$ on a schema $R(XYZ)$ can be written as:

$$\forall x, y_i, z_i : \quad \bigwedge_{i=1}^{k+1} R(x, y_i, z_i) \Longrightarrow \bigvee_{1 \leq p < q \leq k+1} y_p = y_q \tag{1}$$

where x, y_i, z_i are X-, Y-, and Z-values, respectively. In [15], an extension to the classical chase procedure suitable for DEGDs, called the *disjunctive chase*, was introduced in the context of data exchange problems. Below we detail how the disjunctive chase can be used to reason about NDs.

Let $V = \{v_1, v_2, \ldots\}$ be a countably infinite set of variables, and U a set of attributes. A *row* over U is a mapping that associates a variable in V to each $A_i \in U$. A (typed) *tableau* T over U is a set of rows over U, such that any variable v_q may appear only in one attribute A_i. Given two tableaux T_1 and T_2 over U, a homomorphism h from T_1 to T_2, $h : T_1 \to T_2$, is a mapping such that: (1) $h(v_q) = v_p$, $p \leq q$, for each variable v_q in T_1; (2) $h(T_1) \subseteq T_2$, i.e., for each row $t_1 \in T_1$ there exists a row $t_2 \in T_2$ such that it is $h(t_1[A_i]) = t_2[A_i]$, for each $A_i \in U$.

Next, we define chase steps and the disjunctive chase for NDs.

Definition 2 (Chase step for NDs).
Let T be a tableau over U and $\delta : X \xrightarrow{k} Y$ be an ND, with $XY \subseteq U$. Assume that T contains $k+1$ rows $t_1, \ldots, t_{k+1}$ which agree on X and are pairwise different on Y, in which case we say that T does not satisfy *δ. Let δ_j, $j = 1, \ldots, (k+1)k/2$, denote the j-th EGD obtainable from δ (i.e., the j-th disjunct in (1)). Let $h_j : T \to T_j$ be the homomorphism from T to T_j such that $h_j(t_{j,1}[Y]) = h_j(t_{j,2}[Y])$, $t_{j,1}, t_{j,2} \in \{t_1, \ldots, t_{k+1}\}$. Thus, h_j equates the Y-variables of rows $t_{j,1}$ and $t_{j,2}$ and is the identity elsewhere. An* (ND-)chase step *using δ transforms T into the set of tableaux $\{T_j, j = 1, \ldots, (k+1)k/2\}$.*

Definition 3 (Disjunctive chase for NDs).
Given a set of NDs Δ over $R(U)$, let T be a tableau over U. A chase tree of T using Δ *is a finite tree whose root is T and:*

- *If T_j is a node in the chase tree with children $\{T_{j,i}, i = 1, \ldots, (k_j+1)k_j/2\}$, then there exists $\delta_j : W_j \xrightarrow{k_j} Z_j \in \Delta$ such that T_j does not satisfy δ_j and the tableaux $\{T_{j,i}\}$ are obtained from T_j by applying homomorphisms $\{h_{j,i}\}$, each equating a distinct pair of variables of Z_j;*
- *If T_l is a leaf node, then all $\delta \in \Delta$ are satisfied by T_l.*

Chasing a tableaux T with a set of ND Δ yields a (finite) chase tree, whose set of leaves, $Chase_\Delta(T)$, consists of tableaux to which no further chase step can be applied (because all ND are satisfied).

The disjunctive chase can be applied to test whether $\Delta \vDash \delta : X \xrightarrow{k} Y$ in a way that generalizes the method used to test implication of FDs. To this end, let the *tableau for δ*, denoted T_δ, be the tableau with $k+1$ rows $t_1, \ldots, t_{k+1}$ which agree on X and have distinct variables elsewhere.

Theorem 1. *Given a set of NDs Δ over $R(U)$, let $\delta : X \xrightarrow{k} Y$ be an ND and T_δ be the tableau for δ. It is $\Delta \vDash \delta$ iff, for all tableaux in $Chase_\Delta(T_\delta)$, the number of distinct rows on Y is at most k, in which case we say that the chase* succeeds *on δ, otherwise it* fails.

Having proved that the entailment problem for NDs is decidable, next we detail how this result can be used in practice. From the definition of ND it follows that, for any sets of attributes X and Y and any k value, it is $\{X \xrightarrow{k} Y\} \vDash X \xrightarrow{k'} Y$, whenever $k' \geq k$. Let $k^\perp = k^\perp(X, Y)$ be the minimum weight of an ND from X to Y that is entailed by a set of NDs Δ, i.e., $X \xrightarrow{k^\perp} Y \in \Delta^+$ whereas $X \xrightarrow{k^\perp - 1} Y \notin \Delta^+$. In order to determine $k^\perp(X, Y)$ a possibility would be to use a chase-based binary search procedure, which is roughly described as follows. Let $k^\top$ be an upper bound of $k^\perp$ (in Section 4 we will show that $k^\top$ can be easily determined in time linear in the size of Δ). The first iteration sets $k = \lfloor k^\top/2 \rfloor$ and runs the chase with $\delta : X \xrightarrow{k} Y$. If the chase succeeds, then we know that $k^\perp \leq k$, otherwise it is $k^\perp > k$. Repeating this process $\Theta(\log k^\top)$

times yields the result. However, the following theorem shows that, provided an upper bound of $k^{\perp}$ is known, a single chase invocation suffices.

Theorem 2. *Given a set of NDs Δ over $R(U)$, let $\delta : X \xrightarrow{k} Y$ be an ND and T_δ be the tableau for δ. If the chase succeeds on δ, then the tableau in $Chase_\Delta(T_\delta)$ with the maximum number of distinct rows on Y has exactly $k^{\perp}(X, Y)$ distinct rows on Y.*

Although the above theorem provides a logarithmic speedup over the binary search approach, the next result shows that in the worst case the entailment of NDs through the chase has exponential space complexity.

Theorem 3. *For every set of attributes $U = \{A_1, \ldots, A_n\}$ there exists a set Δ of NDs over $R(U)$ and an ND δ entailed by Δ such that T_δ has $\Omega(2^n)$ rows.*

Proof. Let $\Delta = \{A_1 \xrightarrow{2} A_i | i = 2, \ldots, n\}$. Setting $X = \{A_1\}$ and $Y = \{A_2, \ldots, A_n\}$, it is easy to show that $k^{\perp}(X, Y) = 2^{n-1}$. Thus, for the chase to succeed it has to be run using a tableau with at least $k > 2^{n-1}$ rows. □

The set of NDs used in the above proof is in some sense "simple" to deal with (and this is confirmed by the rule-based approach in Section 3.2). However, there are more difficult cases for which it is $k^{\perp}(X, Y) \in \Omega(2^n)$, and determining a tight upper bound of $k^{\perp}(X, Y)$ is NP-hard (see Theorem 4 in Section 4). A further source of complexity arising from the chase-based approach is related to the size of the chase tree. In particular, it is not difficult to show that $Chase_\Delta(T_\delta)$ consists of exponentially many tableaux. Clearly, due to Theorem 2, it is conceivable that several optimizations might be developed to reduce the size of the chase tree so as to locate the tableau in $Chase_\Delta(T_\delta)$ with the maximum number of distinct rows on Y. However, for not too small weight values (which influence both the size of T_δ and the number of tableaux generable at each chase step) and sets of NDs, an application of the chase procedure in practical settings remains unaffordable. For this reason, in the following we introduce a set of (incomplete) inference rules for NDs and provide a partial characterization of which NDs such rules can derive.

3.2. Derivation of NDs

Unlike FDs and other types of database constraints, NDs do not admit a finite set of sound and complete inference rules [1]. In particular, even if only NDs with maximum weight 2 are considered, no p-ary complete axiomatization is possible for NDs unless a restriction is posed on the number of attributes in U.[1] Thus, specific rules with an increasing number of hypotheses can be added depending on (the number of attributes in) the schema, but these would not be enough for another schema with more attributes.

[1] A rule is p-ary if its antecedent contains exactly p distinct members. An axiomatization is p-ary if each of its rules is at most p-ary [16].

A set of sound inference rules, that generalize those valid for FDs and hold for any number of attributes in the schema and for any value of k and l ($k, l \geq 1$), is the following one [1]:

$$\begin{aligned}
\textit{Reflexivity}\,(R): &\quad \vdash X \to X \\
\textit{Transitivity}\,(T): &\quad X \xrightarrow{k} Y \wedge Y \xrightarrow{l} Z \vdash X \xrightarrow{k \cdot l} YZ \\
\textit{Union}\,(U): &\quad X \xrightarrow{k} Y \wedge X \xrightarrow{l} Z \vdash X \xrightarrow{k \cdot l} YZ \\
\textit{Decomposition}\,(D): &\quad X \xrightarrow{k} YZ \vdash X \xrightarrow{k} Y \\
\textit{Successor}\,(S): &\quad X \xrightarrow{k} Y \vdash X \xrightarrow{k+l} Y
\end{aligned}$$

For the purpose of this paper it is more convenient to deal with a different, yet equivalent, set of rules, which we call REDS, in which rules T and U are replaced by the single Extended transitivity (E) rule:

$$\textit{Extended transitivity}\,(E): \quad X \xrightarrow{k} YW \wedge Y \xrightarrow{l} Z \vdash X \xrightarrow{k \cdot l} YWZ$$

The two sets of rules are easily shown to be equivalent. Indeed, by rules R and D it is derived $YW \to Y$, and by rule T: $YW \xrightarrow{l} YZ$; a second application of rule T yields rule E. Going the other way, observe that Rule T is just a special case of rule E obtained when $W = \emptyset$. Rule U is obtained by first applying rule E to $X \xrightarrow{k} Y$ (which holds by hypothesis) and $X \to X$, obtaining $X \xrightarrow{k} XY$. Another application of rule E yields $X \xrightarrow{k \cdot l} XYZ$, from which the result follows by rule D.

Given the above REDS rules and a set of NDs Δ, we would like to know whether a given ND $\delta : X \xrightarrow{k} Y$ is REDS-derivable, $\Delta \vdash_{REDS} \delta$. Similarly to what observed in Section 3.1, it is clear that if $\Delta \vdash_{REDS} X \xrightarrow{k} Y$, then, due to Rule S, all the NDs $X \xrightarrow{k'} Y$, $k' > k$, are also derivable. To this end, let $k_I^{\perp} = k_I^{\perp}(X, Y)$ ($\geq k^{\perp}(X, Y)$) denote the minimum value of the weight of an ND from X to Y that can be derived from Δ using rules I. Then, it makes more sense to consider the optimization version of the problem, which can be precisely stated as follows:

Problem 1.

> ***Given*** *a set of NDs Δ over $R(U)$ and two sets of attributes X and Y, $XY \subseteq U$*
> ***Determine*** *$k^{\perp}_{REDS}(X, Y)$, i.e., the minimum value of k such that $X \xrightarrow{k} Y \in \Delta^{+}_{REDS}$, if there exists one.*

The following result is obvious.

Lemma 2. *The solution to Problem 1 does not change if only the RED rules (i.e., Reflexivity, Extended transitivity, and Decomposition) are used, i.e., $k^{\perp}_{REDS}(X, Y) = k^{\perp}_{RED}(X, Y)$, $\forall X, Y$.*

Due to Lemma 2, in the following we will only consider the RED rules and will call Δ^+_{RED} the *RED closure* of Δ.

3.2.1. The Tight Closure of a Set of Numerical Dependencies

Unlike the case of FDs, the RED closure of a set of NDs Δ has infinite size even when Δ consists of a single ND $X \xrightarrow{k} Y$. Indeed, from $X \xrightarrow{k} Y$ one can derive $X \xrightarrow{k^p} Y$, for any $p > 1$. Intuitively, for the purpose of solving the optimization Problem 1, such NDs are useless, as confirmed by the following fact.

Lemma 3. *Given a set of NDs Δ over $R(U)$, if $X \xrightarrow{l \cdot w(\delta_i)^p} Y \in \Delta^+_{RED}$, then $X \xrightarrow{l \cdot w(\delta_i)} Y \in \Delta^+_{RED}$ for each $\delta_i \in \Delta$, $X, Y \subset U$, and $l, p \geq 1$.*

In light of this, in the following we will only consider derivations that use at most once any ND in Δ, which implies that only a finite number of NDs can be derived. However, as the following example shows, these may still include several useless NDs.

Example 2. Let $U = ABCD$ and $\Delta = \{A \xrightarrow{k_1} B, B \xrightarrow{k_2} C, B \xrightarrow{k_3} D, D \xrightarrow{k_4} C\}$. All the following NDs with the same left- and right-hand sides are in Δ^+_{RED}:

$$\delta_1 : A \xrightarrow{k_1 k_2 k_3} BCD \qquad \delta_2 : A \xrightarrow{k_1 k_2 k_3 k_4} BCD \qquad \delta_3 : A \xrightarrow{k_1 k_3 k_4} BCD$$

For instance, δ_2 can be derived as follows:

1. $A \xrightarrow{k_1} B, B \xrightarrow{k_2} C \vdash A \xrightarrow{k_1 k_2} BC$ (rule E)
2. $B \xrightarrow{k_3} D, D \xrightarrow{k_4} C \vdash B \xrightarrow{k_3 k_4} CD$ (rule E)
3. $A \xrightarrow{k_1 k_2} BC, B \xrightarrow{k_3 k_4} CD \vdash A \xrightarrow{k_1 k_2 k_3 k_4} BCD$ (rule E)

While it is impossible to say which ND, between δ_1 and δ_3, has the lowest weight without knowing the values of the k_i's involved, δ_2 is clearly "loose". Indeed, it is always true that $w(\delta_2) \geq w(\delta_1)$ and $w(\delta_2) \geq w(\delta_3)$, *regardless of k_i's values.* ■

For an ND $\delta : X \xrightarrow{k_1 \cdot \ldots \cdot k_p} Y$, let $K(\delta)$ denote the set of k_i's in δ. It is then obvious that, given two NDs $\delta : X \xrightarrow{w(\delta)} Y$ and $\delta' : X \xrightarrow{w(\delta')} Y$, if $K(\delta) \subset K(\delta')$ then $w(\delta) \leq w(\delta')$. For instance, in Example 2 it is $K(\delta_3) \subset K(\delta_2)$ and $K(\delta_1) \subset K(\delta_2)$, whereas neither $K(\delta_1) \subset K(\delta_3)$ nor $K(\delta_3) \subset K(\delta_1)$ hold. This leads us to introduce the following definition, which characterizes the "interesting" part of the RED closure:

Definition 4 (Tight RED Closure). *An ND $\delta : X \xrightarrow{w(\delta)} Y \in \Delta^+_{RED}$ is* tight *if, for any other ND $\delta' : X \xrightarrow{w(\delta')} Y \in \Delta^+_{RED}$, it is not $K(\delta') \subset K(\delta)$, and is* loose *otherwise. The* tight (RED) closure *of Δ is the set Δ^*_{RED} of all the tight NDs in Δ^+_{RED}.*

Thus, of all the NDs that can be derived from Δ, the tight closure includes only those for which it is necessary to look at the values of the k_i's to determine which one has the lowest weight.

4. RED Rules: Complexity and Incompleteness

In this section we provide some results concerning the complexity of Problem 1 (minimum derivable weight for an ND from X to Y) and the relationship existing between entailed and derived NDs, the aim being to provide a partial characterization of the cases in which RED rules indeed yield the minimum valid weight, i.e., $k^{\perp}_{RED}(X,Y) = k^{\perp}(X,Y)$.

Our first result shows that, even using incomplete rules, derivation of NDs can be a complex task.

Theorem 4. *Given a set of NDs Δ over $R(U)$ and sets of attributes $X, Y \subseteq U$, determining if $\Delta \vdash_{RED} X \xrightarrow{k} Y$ is NP-hard.*

Noticeably, when all NDs can be applied from the start (i.e., all left-hand sides of the NDs are subsets of X) the derivation problem is equivalent to the *weighted set cover problem*[2] (WSC), where the set to be covered is Y and the collection of subsets is the collection of right-hand sides of the NDs. Although WSC is known to be NP-hard, in some specific cases it turns out to become tractable:

- If all subsets have cardinality 2, WSC becomes equivalent to the *edge cover problem*. So, when all NDs in Δ have two attributes in their right-hand side, the derivation problem can be solved in polynomial time [17].
- If the subsets fulfill an acyclicity property known from the concept of tree decompositions of graphs, the WSC is called *tree-like*. So, when the right-hand sides of the NDs are tree-like, the derivation problem is fixed-parameter tractable[3] in the maximum cardinality of the right-hand sides [19].

On the other extreme, when the NDs form a *chain*, $\Delta = \{\delta_1, \ldots, \delta_m\}$ with $\delta_i = W_i \xrightarrow{k_i} Z_i$, $W_1 \subseteq X$, $W_i \not\subseteq \cup_{j<i-1} Z_j \cup X$, and $W_i \subseteq \cup_{j<i} Z_j \cup X$ $(i = 2, \ldots, m)$, the derivation of an ND from X to Y becomes a linear process, since at each step only one ND can be applied.

Now we turn to consider issues that are relevant to characterize the relationship between derivation and entailment of NDs.

Similarly to FDs, the *ND-closure* of a set of attributes X given a set of NDs Δ, denoted $X^{+}_{ND}(\Delta)$ or simply X^{+}_{ND}, is defined as the set of those attributes A_i such that an ND from X to A_i can be derived using the RED rules:

$$X^{+}_{ND}(\Delta) = \{A_i | \exists k_i : X \xrightarrow{k_i} A_i \in \Delta^{+}_{RED}\}$$

[2] Given a set S and a collection C of subsets of S, each associated with a weight, find the subsets in C with minimum overall weight and whose union contains all the elements in S.

[3] A problem is *fixed-parameter tractable* with respect to parameter p if it has a solution running in $f(p) \times n^{\mathcal{O}(1)}$ time, where n is the input size and f is a function of p which is independent of n [18].

Let Δ_{FD} be the set of FDs obtained by setting all weights of the NDs in Δ to 1. Because RED rules generalize inference rules for FDs, and actual weight values of NDs are irrelevant to ND-closure computation, the following identity should not be surprising:

$$X^+_{ND}(\Delta) = X^+_{FD}(\Delta_{FD}) \qquad \forall X, \Delta \tag{2}$$

where X^+_{FD} is the set of all attributes A_i such that the FD $X \to A_i$ holds. Now, since the problem of computing the FD-closure of X can be done in time linear in the length of the given set of constraints [20], the same result also applies to NDs, which immediately yields the following result:

Lemma 4. *Given a set of NDs Δ over $R(U)$ and sets of attributes $X, Y \subseteq U$, let len(Δ) stand for the length of an encoding of Δ, i.e., the total number of non-distinct attribute symbols appearing in Δ. Then, determining if $k^{\perp}_{RED}(X, Y)$ exists requires $\mathcal{O}(len(\Delta))$ time.*

This lemma guarantees that an upper bound of $k^{\perp}(X, Y)$ can be easily found, if it exists, which is a necessary condition for successfully applying the chase (see Theorem 2). Note that this result is based on the simple observation that an ND from X to Y is derivable iff $Y \subseteq X^+_{ND}$.

A basic fact about RED rules is established by the following theorem, which says that such rules are actually complete except for the value of the minimal weight.

Theorem 5. *Given a set of NDs Δ over $R(U)$, for any sets of attributes $X, Y \subseteq U$ there exists an ND from X to Y iff an ND from X to Y is derivable using the RED rules.*

Theorem 5 provides a partial characterization of what kind of incompleteness one can expect from using the RED rules. If the value of $k^{\perp}(X, Y)$ is sought, then the RED rules either provide an upper bound $k^{\perp}_{RED}(X, Y)$ of $k^{\perp}(X, Y)$, if this exists, or, in $\mathcal{O}(\text{len}(\Delta))$ time, they can be used to determine that no ND from X to Y exists.

Before presenting some cases in which the RED rules are indeed complete, it is instructive to see an example in which this is not the case.

Example 3. Let $U = ABCD$ and $\Delta = \{A \xrightarrow{2} BC, A \xrightarrow{2} CD, A \xrightarrow{2} BD\}$. Although it is $k^{\perp}_{RED}(A, BCD) = 4$, there is no instance $r \in Sat(\Delta)$ where a value of A appears in more than 2 tuples. For instance, given the instance:

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_1

any attempt to add a third tuple with $A = a_1$, for instance:

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_1
a_1	b_2	c_2	d_1

will necessarily lead to violate an ND in Δ ($A \xrightarrow{2} BC$ in this case); the same holds for any other instance, thus $k^{\perp}(A, BCD) = 2 < k^{\perp}_{RED}(A, BCD) = 4$. ■

In an attempt to provide a characterization of when RED rules can indeed derive the minimum weight $k^{\perp}(X, Y)$, let us consider a scenario in which the NDs in Δ are "flat", i.e., all of them have the same left-hand side X. Note that, when $X = \perp$, this is tantamount to saying that Δ consists of a set of constraints on the cardinality of some projections, which is indeed a relevant case. Also observe that, in order to prove that $k^{\perp}_{RED}(X, Y) = k^{\perp}(X, Y)$, it is sufficient to exhibit an instance r with $k^{\perp}_{RED}(X, Y)$ tuples, all with the same value on X and distinct Y-values, and show that $r \in Sat(\Delta)$.

With flat NDs we can prove the following preliminary result, assuming without loss of generality that $X \cap Y = \emptyset$.

Lemma 5. *Let $\Delta = \{\delta_i : X \xrightarrow{k_i} Z_i, i = 1, \ldots, m\}$, $Y \subseteq \cup_i Z_i$, and $X \cap Y = \emptyset$. Let $\Delta^{\perp}$ be the subset of NDs used to derive the ND from X to Y with weight $k^{\perp}_{RED}(X, Y)$, i.e., $k^{\perp}_{RED}(X, Y) = \prod_{\delta_i \in \Delta^{\perp}} k_i$. Then there exists an instance $r^{\perp}$ with $k^{\perp}_{RED}(X, Y)$ tuples with the same value on X and distinct Y-values, such that $r^{\perp} \in Sat(\Delta^{\perp})$.*

To get an intuition of how $r^{\perp}$ is made, it is important to distinguish the attributes in U in "local","join", and "don't care" ones. An attribute A_i is: 1) local (to δ_i) if it belongs to Y and it only appears in the right-hand side of a single ND in $\Delta^{\perp}$; 2) join if it appears in more than one ND in $\Delta^{\perp}$; 3) don't care otherwise (i.e., it is neither a join attribute nor is in Y). Note that each ND $\delta_i \in \Delta^{\perp}$ has at least one local attribute, otherwise dropping δ_i from $\Delta^{\perp}$ would still yield a valid ND from X to Y.

The instance $r^{\perp}$ can be obtained as the join of a set of relations, each with k_i rows and attributes corresponding to the right-hand sides of the NDs in $\Delta^{\perp}$. For the join to have $k^{\perp}_{RED}(X, Y) = \prod_i k_i$ tuples with the same value on X and distinct Y-values, each join attribute must have the same single value wherever it appears (so the join reduces to a Cartesian product), whereas the values of the local attributes must guarantee that tuples are indeed distinct. Don't care attributes are given the same value on all tuples. For instance, let $U = ABCD$, $X = \perp$ and $Y = BCD$. With $\Delta^{\perp} = \{\perp \xrightarrow{2} BC, \perp \xrightarrow{2} CD\}$ the instance $r^{\perp}$ with $k^{\perp}_{RED}(\perp, BCD) = 4$ tuples would be:

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_1
a_1	b_1	c_1	d_2
a_1	b_2	c_1	d_2

where B and D are local attributes, C is a join attribute, and A is a don't care attribute.

Lemma 5 shows that only NDs *not* in $\Delta^{\perp}$ can be the cause of incompleteness. We are now in the position to specify a set of cases in which $k^{\perp}_{RED}(X, Y) = k^{\perp}(X, Y)$ is guaranteed to hold.

Theorem 6. *For each $\delta_i : X \xrightarrow{k_i} Z_i \in \Delta^\perp$ let $L_i \subseteq Z_i$ be the set of local attributes in δ_i and define $\Lambda = \Delta \setminus \Delta^\perp = \{\delta_j : X \xrightarrow{k_j} W_j\}$. If one of the following cases occurs, then it is $k^\perp_{RED}(X,Y) = k^\perp(X,Y)$:*

1. *For all $\delta_j \in \Lambda$ and all $\delta_i \in \Delta^\perp$ it is $W_j \cap L_i = \emptyset$;*
2. *Each ND in $\Delta^\perp$ has a single local attribute, $L_i = \{A_i\}$, and it is:*

$$\prod_{i:A_i \in W_j} k_i \leq k_j \qquad \forall \delta_j \in \Lambda \tag{3}$$

3. *For all $\delta_j \in \Lambda$ and all $\delta_i \in \Delta^\perp$ it is $|W_j \cap L_i| \leq 1$, i.e., $W_j \cap L_i = \{A_{i,j}\}$ or $W_j \cap L_i = \emptyset$, and there exists an instance $r^\perp$ such that the following system of inequalities admits solution:*

$$\prod_{A_p \in L_i} |r^\perp[A_p]| \geq k_i \qquad \forall \delta_i \in \Delta^\perp \tag{4}$$

$$\prod_{i:W_j \cap L_i \neq \emptyset} |r^\perp[A_{i,j}]| \leq k_j \qquad \forall \delta_j \in \Lambda \tag{5}$$

In the above theorem it is immediate to recognize that case 3 generalizes both cases 1 and 2 (which, on the other hand, are incomparable with each other). Any attempt to extend the above results to more general scenarios has to face the complex combinatorics arising when multiple NDs not in $\Delta^\perp$ provide a bound to the same, arbitrary, *set* of attributes (whereas case 3 only allows for singleton sets: $W_j \cap L_i = \{A_{i,j}\}$). Indeed, when no restrictions on the $W_j \cap L_i = S_{i,j}$ intersections are put, in place of inequalities (4) and (5) one should check if it is possible to obtain k_i distinct L_i-values given a set of upper bounds on the size of some projections $S_{i,j}$'s. Not surprisingly, this is *exactly* the ND entailment problem, now localized to the set of local attributes L_i introduced by an ND in $\Delta^\perp$. An interesting issue, which is however beyond the scope of the paper, would be to understand if more general completeness results can be obtained by (recursively) applying Theorem 6 to such more focused parts of the initial set of attributes. Finding similar characterizations for general (i.e., not flat) sets of NDs also remains an open problem.

5. A Graph-Based Characterization

The purpose of this section is to show that the tight closure of a set of NDs Δ can be precisely characterized in graph-theoretical terms. Without loss of generality we assume that Δ does not contain two NDs with the same left- and right-hand sides.[4] We represent a set Δ of NDs through a graph defined as follows:

[4] If both $X \xrightarrow{k} Y$ and $X \xrightarrow{k'} Y$ are in Δ, and $k \leq k'$, then the second ND can be removed.

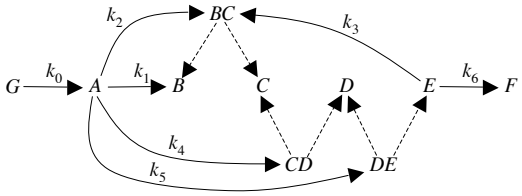

Figure 1: The ND-graph for the set of NDs in Example 4

Definition 5 (ND-graph). *Given a set Δ of NDs over $R(U)$, the* ND-graph $G_\Delta = (\mathcal{V}, \mathcal{E})$ induced by Δ *is the directed graph with nodes $\mathcal{V} \subseteq 2^U$, arcs $\mathcal{E} = \mathcal{E}^f \cup \mathcal{E}^d$ ($\mathcal{E}^f \cap \mathcal{E}^d = \emptyset$), and an arc labeling function $\omega : \mathcal{E} \to \mathbb{N}$ (weight) such that:*

1. *For every ND $\delta : X \xrightarrow{k} Y \in \Delta$ there are in $\mathcal{V}$ two nodes X and Y, and there is in $\mathcal{E}^f$ a* full *arc $\langle X, Y\rangle$ (oriented from X to Y) such that $\omega(\langle X, Y\rangle) = k$. Thus, $\omega(\langle X, Y\rangle) = w(\delta)$.*

2. *For every* compound *node $X \in \mathcal{V}$, $X = A_1, \ldots, A_r$, $r > 1$, there are r* simple *nodes $A_1, \ldots, A_r$ in $\mathcal{V}$ and r* dotted *arcs $\langle X, A_1\rangle, \ldots, \langle X, A_r\rangle$ in $\mathcal{E}^d$ with $\omega(\langle X, A_i\rangle) = 1$.*

3. *If the empty set of attributes $\perp$ is in $\mathcal{V}$, then for each simple node $A_i \in \mathcal{V}$ there is in $\mathcal{E}^d$ a dotted arc $\langle A_i, \perp\rangle$ with $\omega(\langle A_i, \perp\rangle) = 1$.*

In the particular case $\Delta = \emptyset$, it is conventionally assumed that, for any *choice of $X \subseteq U$, the graph including node X, and completed respecting the above rule 2 if X consists of more than one attribute, is also an ND-graph. With a slight abuse of terminology we say that this is the* ND-graph induced by X. *If $\Gamma \subseteq \Delta$, then G_Γ is also called an ND-subgraph of G_Δ.*

In the following, when necessary, we will use notation $\langle X, Y\rangle_k$ to emphasize that $\omega(\langle X, Y\rangle) = k$.

Example 4. Here we introduce the running example that will be used throughout the rest of the paper. Let $U = ABCDEFG$ and consider the set of NDs:

$$\Delta = \{G \xrightarrow{k_0} A, A \xrightarrow{k_1} B, A \xrightarrow{k_2} BC, E \xrightarrow{k_3} BC, A \xrightarrow{k_4} CD, A \xrightarrow{k_5} DE, E \xrightarrow{k_6} F\}$$

The corresponding ND-graph is shown in Figure 1. ■

The ND-graph represents in compact form all the relevant information needed to derive tight NDs. In particular, the attributes appearing in the nodes of a suitably defined ND-subgraph of G_Δ, called *ND-path*, precisely characterize which NDs can be derived using the RED rules. To this end, for any ND-subgraph G_Γ of G_Δ, let $Attr(G_\Gamma)$ denote the set of all the attributes that appear in the nodes of G_Γ.

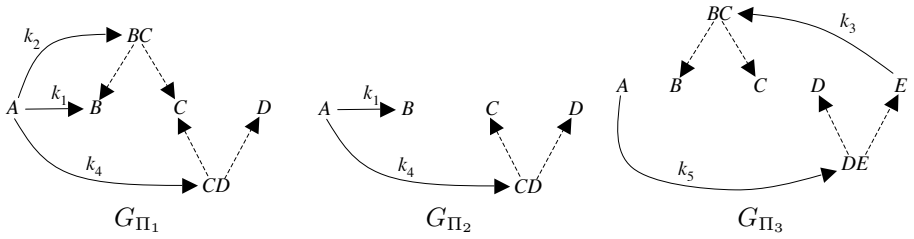

Figure 2: Examples of ND-paths from A obtained from the ND-graph in Figure 1

Definition 6 (ND-path from X). *Given the ND-graph $G_\Delta = (\mathcal{V}, \mathcal{E})$ and $X \in \mathcal{V}$, an* ND-path from X *is any ND-subgraph of G_Δ that can be inductively obtained as follows:*

1. *The ND-subgraph induced by X is an ND-path from X.*
2. *If G_Π is an ND-path from X induced by a set of NDs $\Pi \subset \Delta$, and $\langle W, Z\rangle_k \in \mathcal{E}^f$, with $W \subseteq Attr(G_\Pi)$ and $Z \not\subseteq Attr(G_\Pi)$, the ND-subgraph induced by $\Pi \cup \{W \xrightarrow{k} Z\}$ is also an ND-path from X.*
3. *No other ND-subgraph of G_Δ is an ND-path from X.*

In order to emphasize that an ND-subgraph G_Π is also an ND-path from X, the notation G_Π^X will also be used. The weight $\omega(G_\Pi^X)$ *of G_Π^X is the product of the weights in its* full *arcs; by definition, the weight of $G_\emptyset^X$, i.e., the ND-path from X with no full arcs, is 1.*

Note that, though any subset of NDs $\Gamma \subseteq \Delta$ induces an ND-subgraph, due to rule 2 in the above definition not all such ND-subgraphs are also ND-paths.

Definition 7 (ND-path from X to Y). *Given an ND-path G_Π^X from X and a set of attributes Y, if $Y \subseteq Attr(G_\Pi^X)$ then G_Π^X is also called an* ND-path from X to Y*, and we say that Y is* reachable *from X (in G_Π^X). The ND-path G_Π^X from X to Y is Y-*minimal *iff there is no other ND-path $G_{\Pi'}^X$ from X to Y such that $\Pi' \subset \Pi$.*

Example 5. With reference to the ND-graph introduced in Example 4, Figure 2 shows three ND-paths from A. It is $Attr(G_{\Pi_1}) = Attr(G_{\Pi_2}) = \{ABCD\}$ and $Attr(G_{\Pi_3}) = \{ABCDE\}$; the weights are $\omega(G_{\Pi_1}) = k_1k_2k_4$, $\omega(G_{\Pi_2}) = k_1k_4$, and $\omega(G_{\Pi_3}) = k_3k_5$, respectively. Both G_{Π_1} and G_{Π_2} are ND-paths to BC and BCD, but only G_{Π_2} is both BC- and BCD-minimal. Conversely, G_{Π_3} is BC-minimal but not E-minimal. Finally, Figure 3 shows an ND-subgraph that is not an ND-path; the reason is that, since there are two full arcs ending in BC, there is no way to incrementally build the graph in a way that respects rule 2 in Definition 6. ■

The building rules of ND-paths ensure that, for any G_Π^X, $\Pi \subseteq \Delta$, the set of attributes that are reachable from X in G_Π^X coincides with the ND-closure of X

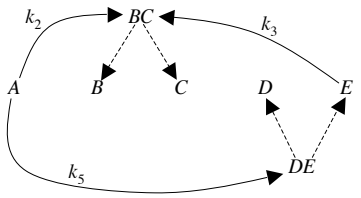

Figure 3: An ND-subgraph of the ND-graph in Figure 1 that is not an ND-path

computed using the NDs in Π, i.e., $Attr(G^X_\Pi) = X^+_{ND}(\Pi)$. This is made precise by the following lemma.

Lemma 6. *Let G^X_Π be an ND-path from X to Y. Then $X \xrightarrow{\omega(G^X_\Pi)} Y \in \Delta^+_{RED}$.*

Although ND-paths correspond to NDs in the RED closure of Δ, not all of them are necessarily tight. This leads us to the following main result, that precisely characterizes the tight RED closure of Δ in terms of *minimal* ND-paths.

Theorem 7. *Let G_Δ be the ND-graph induced by a set of NDs Δ, and let $X \in \mathcal{V}$. There exists in G_Δ a Y-minimal ND-path G^X_Π from X to Y having weight $\omega(G^X_\Pi)$ iff $\delta : X \xrightarrow{\omega(G^X_\Pi)} Y \in \Delta^*_{RED}$.*

The intuition behind Theorem 7 is that if δ is a tight ND, then *any* of its derivations will add at each step some new attribute, thus all of them will correspond to the same Y-minimal ND-path from X to Y. Conversely, if δ is loose then there exists at least one derivation in which at some step no new attribute is added, thus that step is redundant. This is also to say that an ND-path is minimal iff *all* the ways to build it succeed without violating rule 2 in Definition 6.

So far, we have used G_Δ to reason on NDs whose left-hand side is a node in G_Δ. In order to generalize to arbitrary NDs it is sufficient to properly extend G_Δ by adding the required node.

Definition 8 (Extended Δ-Graph). *Given the ND-graph $G_\Delta = (\mathcal{V}, \mathcal{E})$ and a set of attributes $X = A_1, \ldots, A_r$, we call* ND-graph extended to X *the graph $G_{\Delta,X}$ with nodes $\mathcal{V} \cup \{X, A_1, \ldots, A_r\}$ and arcs $\mathcal{E} \cup \{\langle X, A_i\rangle$ dotted $: A_i \in X, \omega(\langle X, A_i\rangle) = 1\}$.*

The following is immediate.

Corollary 1. *Let G_Δ be the ND-graph associated with a set Δ of NDs, and let $X \subseteq U$. There exists in $G_{\Delta,X}$ a Y-minimal ND-path G^X_Π with weight $\omega(G^X_\Pi)$ iff $\delta : X \xrightarrow{\omega(G^X_\Pi)} Y \in \Delta^*_{RED}$.*

Due to Corollary 1, in the following, without losing in generality and to avoid unnecessary complication, we will always consider that X is a node in G_Δ.

6. Finding the Minimal Derivable Weight

In this section we will lean on the theoretical results of Section 5 to show how, given an ND-graph G_Δ and two sets of attributes X and Y, the minimal derivable weight $k^\perp_{RED}(X,Y)$ can be efficiently determined, thus solving Problem 1.

A naïve method to determine $k^\perp_{RED}(X,Y)$ is to strictly exploit the definition of ND-path. The Naïve algorithm (Algorithm 1) starts with the ND-graph $G^X_\emptyset$ induced by X, which is progressively extended by iteratively adding full arcs, so that all possible ND-paths from X to Y, conveniently called *solutions*, are enumerated. Due to Theorem 7, these include all the ones corresponding to the tight NDs from X to Y, thus even the one(s) whose weight is $k^\perp_{RED}(X,Y)$.

Algorithm 1 The Naïve algorithm

Input: G_Δ, X, Y
Output: $k^\perp_{RED}(X,Y)$

1: $k^\perp_{RED}(X,Y) \leftarrow \infty$
2: $ActiveNDPaths \leftarrow \{G^X_\emptyset\}$ ▷ ND-paths to be extended
3: **while** $ActiveNDPaths \neq \emptyset$ **do**
4: $\quad G^X_\Pi \leftarrow \mathsf{Pop}(ActiveNDPaths)$
5: $\quad$ **for all** $G^X_{\Pi_i} \in \mathsf{AllExtensions}(G^X_\Pi)$ **do**
6: $\quad\quad$ **if** $Y \subseteq Attr(G^X_{\Pi_i})$ **then** ▷ found a solution...
7: $\quad\quad\quad$ **if** $\omega(G^X_{\Pi_i}) < k^\perp_{RED}(X,Y)$ **then** ▷ ... better than the current one
8: $\quad\quad\quad\quad k^\perp_{RED}(X,Y) \leftarrow \omega(G^X_{\Pi_i})$
9: $\quad\quad$ **else**
10: $\quad\quad\quad \mathsf{Push}(ActiveNDPaths, G^X_{\Pi_i})$ ▷ $G^X_{\Pi_i}$ must be further extended
11: **return** $k^\perp_{RED}(X,Y)$

Algorithm 2 The AllExtensions method

Input: G^X_Π
Output: $\{G^X_{\Pi_i}\}$

1: $ExtendedNDPaths \leftarrow \emptyset$
2: **for all** $\delta_i \in \Delta \setminus \Pi, \delta_i : W_i \xrightarrow{k_i} Z_i$ **do**
3: $\quad$ **if** $W_i \subseteq Attr(G^X_\Pi) \wedge Z_i \not\subseteq Attr(G^X_\Pi)$ **then**
4: $\quad\quad G^X_{\Pi_i} \leftarrow \mathsf{Extend}(G^X_\Pi, \delta_i)$
5: **return** $ExtendedNDPaths$

A brief description of the methods used by the algorithm follows:

- $\mathsf{Pop}(ActiveNDPaths)$ picks and removes from the $ActiveNDPaths$ queue the next ND-path to be extended.
- $\mathsf{Push}(ActiveNDPaths, G^X_{\Pi_i})$ adds the ND-path $G^X_{\Pi_i}$ to $ActiveNDPaths$.

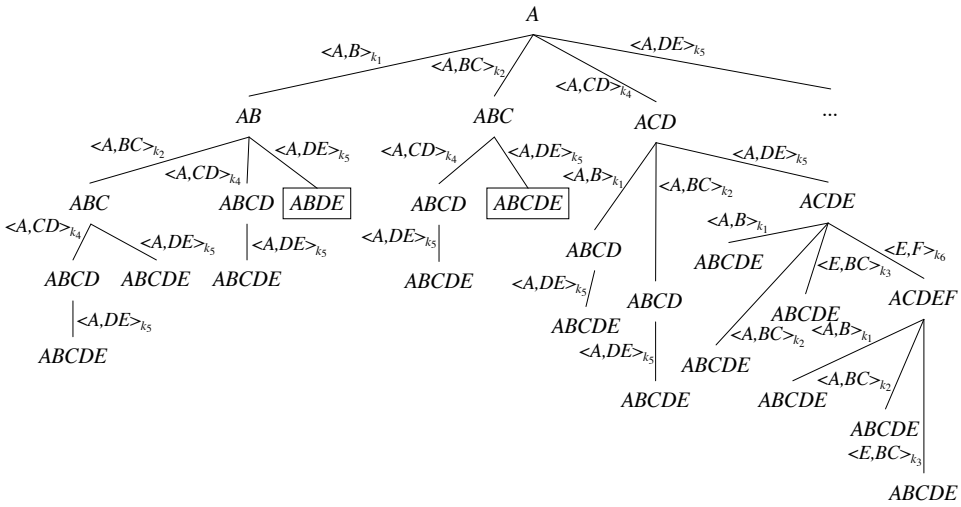

Figure 4: Part of the search space of the Naïve algorithm applied to the ND-graph in Figure 1 with $X \equiv A$ and $Y \equiv BE$; boxed nodes represent BE-minimal ND-paths

- AllExtensions(G^X_Π), described in Algorithm 2, returns the set of ND-paths obtained by extending G^X_Π with one full arc in all possible ways (rule 2 of Definition 6).
- Extend(G^X_Π, δ) returns the ND-path $G^X_{\Pi_i}$ that extends G^X_Π with the full arc corresponding to δ ($\Pi_i = \Pi \cup \{\delta\}$).

Example 6. Here we give an intuition on how the Naïve algorithm works on the ND-graph in Figure 1, assuming that $X \equiv A$ and $Y \equiv BE$. Figure 4 shows a tree representing part of the search space of the algorithm. Each node corresponds to an ND-path from A and is represented by its set of attributes; all leaves represent solutions. Each edge is labelled with the new full arc being added. The tree includes two BE-minimal ND-paths (boxed in Figure 4), whose weights are k_1k_5 and k_2k_5, respectively. At the beginning, $G^A_\emptyset = (\{A\}, \emptyset)$ is the only active ND-path ($\omega(G^A_\emptyset) = 1$). AllExtensions extends $G^A_\emptyset$ in four ways, by adding full arcs $\langle A, B\rangle_{k_1}$, $\langle A, BC\rangle_{k_2}$, $\langle A, CD\rangle_{k_4}$, and $\langle A, DE\rangle_{k_5}$. None of the obtained ND-paths is a solution, so all of them are pushed into $ActiveNDPaths$ to be further extended. If $ActiveNDPaths$ is managed as a first-in-first-out queue, the subsequent steps orderly add full arcs $\langle A, BC\rangle_{k_2}$, $\langle A, CD\rangle_{k_4}$, and $\langle A, DE\rangle_{k_5}$ (creating the leftmost path in the tree), plus simple nodes C, D and E and related dotted arcs. A first solution is thus found, with weight $k_1k_2k_4k_5$. All other solutions are found by progressively extending the partial solutions. ■

As Example 6 makes evident, the Naïve algorithm has several drawbacks: (i) it creates some solutions using arc $\langle E, F\rangle_{k_6}$, that obviously can never contribute

to reach BE; (ii) it generates twice or more the same ND-paths by adding the same set of full arcs in different orders; (iii) it extends an ND-path even if its weight is higher than that of the current solution; (iv) it does not detect non-minimal ND-paths, so it wastes time in extending them.

In the following we present a branch & bound algorithm, BBND (Algorithm 3), that overcomes these problems. Before describing in detail how BBND works,

Algorithm 3 The BBND algorithm

Input: G_Δ, X, Y
Output: $k^\perp_{RED}(X,Y)$

1: $k^\perp_{RED}(X,Y) \leftarrow \infty$
2: $ActiveNDPaths \leftarrow \{G^X_\emptyset\}$ ▷ ND-paths to be extended
3: **if** $Y \not\subseteq Attr(G_\Delta)$ **then return** $k^\perp_{RED}(X,Y)$ ▷ Y is not reachable from X
4: RemoveUselessNDs$(G_\Delta, X, Y)\}$
5: **while** $ActiveNDPaths \neq \emptyset$ **do**
6: $G^X_\Pi \leftarrow$ Pop$(ActiveNDPaths)$
7: **for all** $G^X_{\Pi_i} \in$ SmartExtensions(G^X_Π) **do**
8: **if** $Y \subseteq Attr(G^X_{\Pi_i})$ **then** ▷ found a solution...
9: **if** $\omega(G^X_{\Pi_i}) < k^\perp_{RED}(X,Y)$ **then** ▷ ...better than the current one
10: $k^\perp_{RED}(X,Y) \leftarrow \omega(G^X_{\Pi_i})$
11: **else if** $\omega(G^X_{\Pi_i}) < k^\perp_{RED}(X,Y) \wedge$ $\neg$IsDominated$(G^X_{\Pi_i}, ActiveNDPaths) \wedge$ IsMinimal$(G^X_{\Pi_i})$ **then**
12: Push$(ActiveNDPaths, G^X_{\Pi_i})$ ▷ $G^X_{\Pi_i}$ must be further extended
13: **return** $k^\perp_{RED}(X,Y)$

we provide a brief description of the methods used.

- RemoveUselessNDs(G_Δ, X, Y) removes from G_Δ the subset of full arcs that do not belong to any ND-path from X to Y (see Section 6.1).
- SmartExtensions(G^X_Π) returns the set of ND-paths obtained by extending the ND-path G^X_Π with one full arc by avoiding duplicates (see Section 6.2).
- IsMinimal$(G^X_{\Pi_i})$ returns true if $G^X_{\Pi_i}$ is $Attr(G^X_{\Pi_i})$-minimal (see Section 6.3), false otherwise.
- IsDominated$(G^X_{\Pi_i}, ActiveNDPaths)$ returns true if $G^X_{\Pi_i}$ is dominated (see Section 6.4) by at least one of the ND-paths in $ActiveNDPaths$, false otherwise. As a side effect, if $G^X_{\Pi_i}$ dominates some of the ND-paths in $ActiveNDPaths$, these are dropped from the queue.

Note that the behavior of the Pop method actually depends on the specific enumeration strategy of ND-paths adopted, as discussed in Section 6.5.

In the following subsections we will describe in detail the major ingredients of BBND.

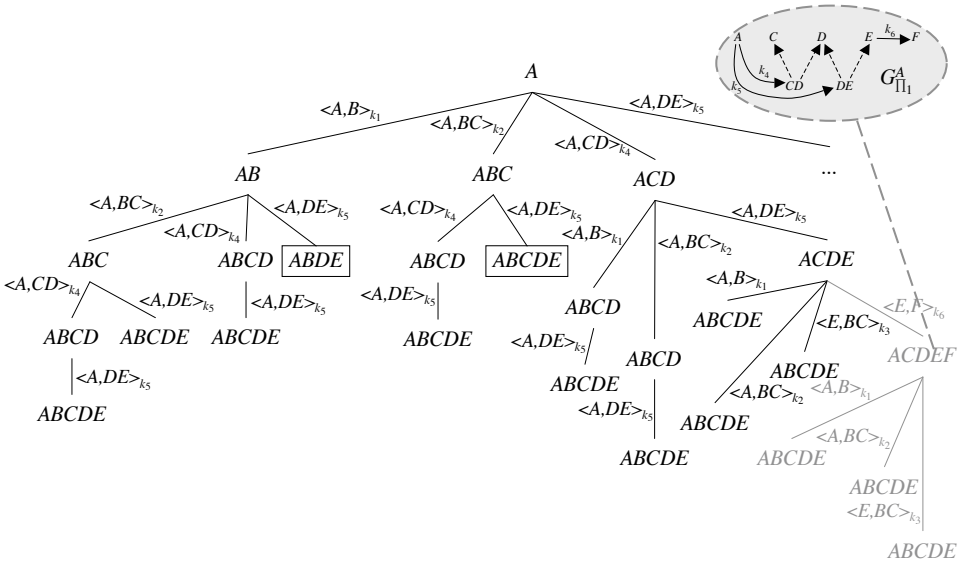

Figure 5: Reduced search space without useless NDs; $G^A_{\Pi_1}$ is an ND-path including a useless ND

6.1. Removing Useless NDs

The ND-graph G_Δ includes *all* the full arcs corresponding to NDs defined in the application domain, but only some of them might be relevant in determining $k^{\perp}_{RED}(X, Y)$. For instance, with reference to the ND-graph in Figure 1, full arcs $\langle G, A\rangle_{k_0}$ and $\langle E, F\rangle_{k_6}$ are obviously useless to compute $k^{\perp}_{RED}(A, BE)$. These arcs should be removed from G_Δ in advance, since they would determine an exponential number of extensions before BBND can discover that the so-obtained ND-paths are non-minimal.

Removal of useless NDs is carried out by the RemoveUselessNDs method, that works in two phases. In the first phase it navigates the ND-graph forward starting from X, so as to mark all the nodes that can be reached from X, and consequently a set of full arcs; in our example, all full arcs are marked except $\langle G, A\rangle_{k_0}$, that is removed. In the second phase, RemoveUselessNDs navigates the so-reduced ND-graph backward starting from the nodes corresponding to each $A_i \in Y$, and marks all the full arcs from which A_i can be reached; in our example, only the full arc $\langle E, F\rangle_{k_6}$ is not marked and consequently removed. Figure 5 shows in grey, with reference to the search space of Figure 4, the portion of the search space that is pruned if useless NDs are removed in advance.

6.2. Avoiding Repeated ND-Paths

Different sequences of extension steps might lead to ND-paths sharing the same set of full arcs. To avoid generating repeated ND-paths, it is necessary to be able to recognize if a given set of full arcs has already been obtained in the

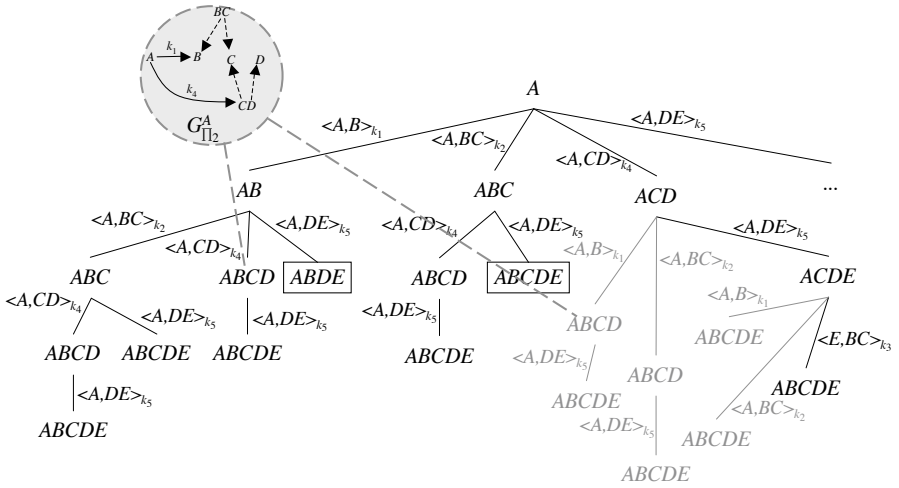

Figure 6: Reduced search space without repeated ND-paths; $G^A_{\Pi_2}$ is a repeated ND-path. Useless NDs are not shown for clarity

enumeration process. Clearly, keeping trace of all the sets of full arcs (whether corresponding to a solution or not) generated so far would require an exponential effort in terms of space. A simpler alternative is to impose an ordering criterion on the set of NDs, such as a lexicographically ordering of their right-hand sides. This can be easily implemented as follows.

Let $\Delta = \{\delta_1, \delta_2, \ldots, \delta_m\}$ be the given set of NDs, where subscripts now reflect the chosen ordering criterion. Consider an ND-path $G^X_{\Pi_i}$, $\Pi_i \subset \Delta$, that has been obtained by extending the ND-path G^X_Π with (the full arc corresponding to) δ_i. Furthermore, let $\Gamma \subseteq \Delta \setminus \Pi$ be the set of NDs that could have been used to extend G^X_Π (these are precisely all the NDs in $\Delta \setminus \Pi$ that satisfy rule 2 in Definition 6) and $\Gamma_i \subseteq \Delta \setminus (\Pi \cup \{\delta_i\})$ be similarly defined. Method AllExtensions used by the naïve approach would extend $G^X_{\Pi_i}$ by picking *all* the NDs in Γ_i. In order to guarantee that the same ND-path is not generated more than once, method SmartExtensions does *not* pick from Γ_i those NDs δ_j such that $\delta_j \in \Gamma$ and $j < i$, i.e., those NDs that lexicographically precede δ_i and that could have been added at the previous extension step.

Figure 6 shows in grey, with reference to the search space of Figure 5, the portion of the search space that is pruned using SmartExtensions.

6.3. Pruning Non-Minimal ND-Paths

The Naïve algorithm is likely to waste a lot of work in generating non-minimal ND-paths. For instance, it is apparent from Figure 7 that *any* solution obtained by extending the ND-path $G^A_{\Pi_3}$ is not BE-minimal. As the following result shows, this can be avoided.

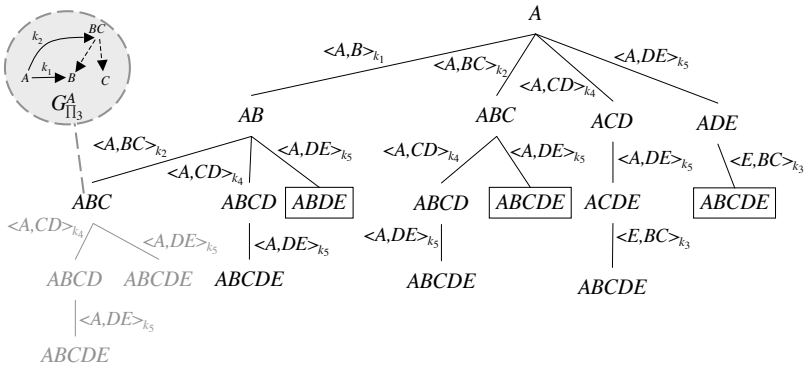

Figure 7: Reduced search space without non-minimal ND-paths; $G^A_{\Pi_3}$ is a non-minimal ND-path. Useless NDs and repeated ND-paths are not shown for clarity

Lemma 7. *Let G^X_Π and G^X_Γ be two ND-paths from X, with $\Gamma \subset \Pi$ and G^X_Π obtained by extending G^X_Γ. If G^X_Γ is not $Attr(G^X_\Gamma)$-minimal, then G^X_Π is not Y-minimal for all $Y \subseteq Attr(G^X_\Pi)$.*

Detection of non-minimal ND-paths is based on Definition 6. Given an ND-path G^X_Γ, if all the NDs included in Γ are removed one at a time and the corresponding ND-subgraphs G_{Γ_i} are generated, then if at least one of such G_{Γ_i}'s is indeed an ND-path from X with $Attr(G^X_{\Gamma_i}) = Attr(G^X_\Gamma)$ it follows that G^X_Γ is not minimal and can therefore be discarded. The worst-case complexity of method IsMinimal is $\mathcal{O}(m \text{ len}(\Gamma))$, where m is the number of NDs in Γ and $\text{len}(\Gamma)$ is the length of a representation of Γ. Note that the $\text{len}(\Gamma)$ complexity factor directly follows from Lemma 4, since the problem here is just to check that all the attributes in $Attr(G_{\Gamma_i})$ are indeed reachable from X.

Method IsMinimal adopts two complementary strategies for reducing the actual number of ND-subgraphs to be tested. The first of them is based on the concept of *essential* NDs. Intuitively, these are those NDs whose removal makes some attributes not reachable anymore from X, thus method IsMinimal should not try to remove them.

Definition 9 (Essential NDs). *Given an ND-path G^X_Π, the ND $\delta \in \Pi$, $\delta : W \xrightarrow{k} Z$, is said to be* essential *for G^X_Π if there exists $A \in Z \setminus X$ such that no other ND in Π includes A in its right-hand side.*

Lemma 8. *Let G^X_Π be an ND-path from X, and G_{Π_i} be the ND-subgraph induced by $\Pi_i = \Pi \setminus \delta$. If δ is essential for G^X_Π, then $Attr(G^X_{\Pi_i}) \subset Attr(G^X_\Pi)$.*

A second way to reduce the actual running time of the minimality check is based on the following lemma.

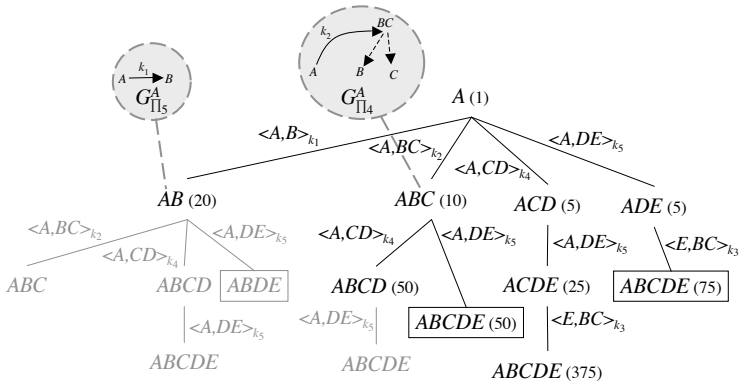

Figure 8: Reduced search space without dominated ND-paths if $k_1 = 20$, $k_2 = 10$, $k_3 = 15$, and $k_4 = k_5 = 5$. Useless NDs and repeated/non-minimal ND-paths are not shown for clarity

Lemma 9. *Let G^X_Γ be an ND-path from X, obtained by extending a minimal ND-path G^X_Π with the ND $\delta : W \xrightarrow{k} Z$. Let $\delta_i : W_i \xrightarrow{k_i} Z_i \in \Pi$ be a non-essential ND for G^X_Π. If $Z \cap Z_i = \emptyset$, then the ND-subgraph G_{Γ_i} induced by $\Gamma_i = \Gamma \setminus \delta_i$ is not an ND-path from X.*

Figure 7 shows in grey, with reference to the search space of Figure 6, the portion that is pruned using IsMinimal.

6.4. Exploiting Weights

A first, simple way to exploit the numerical values of weights for reducing the search space is to prune an ND-path if its weight is not lower than that of the best solution found so far. We call this technique *lazy domination* because it relies on a comparison with a solution.

A more effective way to exploit numerical values of weights is based on the concept of *eager domination* among ND-paths.

Definition 10 (Eager Domination). *Let G^X_Π and G^X_Γ be two ND-paths from X. G^X_Π is said to* eagerly dominate *G^X_Γ if $Attr(G^X_\Pi) \supseteq Attr(G^X_\Gamma)$ and $\omega(G^X_\Pi) \leq \omega(G^X_\Gamma)$.*

For instance, in Figure 8, if $k_2 \leq k_1$ then $G^A_{\Pi_4}$ eagerly dominates $G^A_{\Pi_5}$ since $Attr(G^A_{\Pi_4}) = ABC \supset AB = Attr(G^A_{\Pi_5})$. Note that lazy domination does not necessarily imply eager domination: though an ND-path is *not* a solution, it could include some attributes that are not included in the best solution found so far.

Lemma 10. *Let G^X_Π and G^X_Γ be two ND-paths from X. If G^X_Π eagerly dominates G^X_Γ then, for any ND-path $G^X_{\Gamma_i}$ obtained by extending G^X_Γ with an ND δ_i, either (1) G^X_Π eagerly dominates $G^X_{\Gamma_i}$, or (2) the ND-path $G^X_{\Pi_i}$ that extends G^X_Π with δ_i eagerly dominates $G^X_{\Gamma_i}$.*

Lemma 10 proves that dominated ND-paths can be safely discarded from $ActiveNDPaths$. Eager domination is highly effective in reducing the size of the search space because it can be applied to generic ND-paths, none of which needs to be a solution, and even because it works in both directions. Indeed, for each ND-path $G^X_{\Pi_i}$ being examined, method IsDominated not only checks if $G^X_{\Pi_i}$ is eagerly dominated by an ND-path in $ActiveNDPaths$, but it also verifies if some of the ND-paths in $ActiveNDPaths$ are eagerly dominated by $G^X_{\Pi_i}$, in which case they are removed from the queue.

Unlike the previous optimization strategies, the influence that both types of domination have on the search space depends on the specific strategy adopted for enumerating ND-paths.

6.5. Enumeration Strategies

Algorithm BBND assumes that a proper enumeration strategy is implemented by the Pop method. In this section we propose and discuss four possible strategies, namely depth-first (DF), breadth-first (BrF), best-first (BeF), and cheapest-first (ChF). All four strategies will be extensively evaluated and compared in Section 7.

The DF strategy minimizes the number of ND-paths simultaneously stored in the $ActiveNDPaths$ queue, thus minimizing memory requirements. Furthermore, DF quickly determines an initial solution, thus enabling lazy domination to be earlier applied.

The BrF strategy first generates all ND-paths from X with m full arcs before proceeding to generate ND-paths with $m+1$ full arcs. Clearly, in the worst case this leads to an exponential number of ND-paths to be simultaneously active. Although a solution is obtained later than with DF, the higher number of ND-paths available will make pruning based on eager domination more likely. Furthermore, it is straightforward to verify that, using the BrF strategy, for each non-minimal ND-path $G^X_{\Pi_i}$ generated, there is always in $ActiveNDPaths$ at least one ND-path that eagerly dominates $G^X_{\Pi_i}$. This means that the BrF strategy does not require the minimality check to be executed, since it is subsumed by the eager domination check.

The BeF strategy extends the "most promising" ND-path, i.e., the one chosen according to a heuristic function that attempts to predict how close the ND-path is to a solution. The specific heuristic we use extends the ND-path G^X_Π for which the cardinality of $Attr(G^X_\Pi) \cap Y$ is maximum, i.e., the ND-path that is "closest" to Y, first. In case of ties, the ND-path with the lowest weight is chosen. Figure 8 shows in grey, with reference to the search space of Figure 7, the portion of the search space that is pruned using domination with the BeF strategy if weights are as follows: $k_1 = 20$, $k_2 = 10$, $k_3 = 15$, and $k_4 = k_5 = 5$. Next to each node, the weight of the corresponding ND-path is reported. Among the first four nodes generated, ADE is the first to be extended since it includes the highest number of attributes of Y and it has the lowest weight. Its only possible extension is through arc $\langle E, BC\rangle_{k_3}$, which leads to the first solution with weight 75. The next node extended is ABC, which leads to a better

solution with weight 50. Note that nodes AB and $ABCD$ will never be extended because they are dominated by ABC and $ABCDE$, respectively.

Finally, the ChF strategy is inspired by the worst case for our derivation problem that, as discussed in Section 4, is equivalent to a weighted set cover problem. According to [21], a greedy algorithm that builds a set cover by choosing at each step the subset with minimal per-attribute weight is guaranteed to find a cover whose weight is not higher than $\ln(Card_{\max})$ times the weight of the optimal cover (where $Card_{\max}$ is the largest subset size). In our case, this means extending first, at each step, the ND-path for which the last added ND paid the lowest per-attribute weight. More precisely, let $\delta : W \xrightarrow{k} ZV$ be the last ND added to the ND-path G_{Π}^{X}, in which Z are the newly added attributes. This ND-path will be extended first if $k/|Z \cap Y|$ is minimal with respect to all the other active ND-paths. In case of ties, the ND-path with the lowest weight is chosen.

6.6. Complexity

In the worst case a branch & bound algorithm must explore every node in the search space [22], so its worst case computational complexity is the size of the search space (i.e., the same as the complexity of the brute force search). The search space for BBND has maximum size when (i) no ND in Δ is useless, and (ii) all NDs in Δ can be applied from the start (based on rule 2 of Definition 6, this means that the left-hand sides of all NDs are included in X and the right-hand sides of all NDs have non-empty intersection with Y). Considering that method SmartExtensions avoids duplicates, the search space in this case is the power set of Δ and has 2^m nodes, where m is the number of NDs. Conversely, the best case for BBND is when the NDs form a chain (e.g., $X \xrightarrow{k_1} V$, $V \xrightarrow{k_2} W$, etc.), because in this case there is no branching in the search space and the complexity decreases to $\mathcal{O}(m)$.

7. Experimental Results

We extensively tested the BBND algorithm for efficiency and effectiveness using six different datasets. Considering that a valuable application of NDs is the estimation of the cardinality of projections, we decided to push our tests in this direction. We recall that a bound k of the cardinality of the projection on a set of attributes Y can be established by an ND of the form $\perp \xrightarrow{k} Y$. Note that in the following we use the term *proper ND* to denote an ND whose weight is strictly higher than 1, i.e., an ND that is not also an FD.

So we defined a relation schema $R(U)$ including 24 attributes and a basic set Δ_0 of 24 proper NDs of the form $\perp \xrightarrow{k_i} A_i$ that put a cardinality constraint k_i on each attribute A_i of R. Note that, using only Δ_0, the cardinality bound of (the projection on) $Y \subseteq U$ is simply computed as the product of the cardinalities of the attributes in Y (*basic bound*). Then we defined six extended sets of NDs over R:

- Since deriving tight bounds is particularly crucial in physical design of multidimensional databases and data warehouses, to define the first three ND sets we took inspiration from a star schema including eight dimension tables with three attributes each. So we created three ND sets (Δ_1, Δ_2, and Δ_3) each including, besides the 24 proper NDs in Δ_0, 16 FDs that model multidimensional hierarchies (two for each hierarchy) plus 10, 25, and 50 randomly-generated additional proper NDs, respectively. For each additional ND $W_i \xrightarrow{k_i} Z_i$, the total number of attributes in W_iZ_i ranges between 2 and 8, and its weight k_i reduces the basic bound of W_iZ_i by a factor randomly ranging between 0.80 and 0.95.

- To investigate how efficiency and effectiveness depend on the topology of constraints for a generic database (no multidimensional structure), we introduced three more ND sets. To this end, we partitioned the 24 attributes of R into 3 groups, each including 8 attributes. Overall, 50 proper NDs and 16 FDs were introduced: in both Δ_4 and Δ'_4, only intra-group FDs were created; in Δ_4 even all proper NDs are intra-group, while in Δ'_4 they are freely defined over all attributes of R. In the last ND set, Δ_5, all NDs are freely defined over all attributes.

Finally, we randomly generated 300 subsets Y_j (each including 2 to 8 attributes); for each Y_j and Δ_m, we used the BBND algorithm to compute the minimal derivable weight $k^{\perp}_{RED}(\perp, Y_j)$. All tests were carried out on a Pentium 4, 3.4 GHz, 2 GB RAM; all execution times are in seconds.

The goals and results of our tests can be summarized as follows:

1. Measure how effective NDs are in improving the basic cardinality bounds; as shown in Subsection 7.1, even a solution that includes a very few NDs can drastically cut the basic bound by about 90%.

2. Compare the four enumeration strategies proposed in Section 6.5; it turns out that the BEF strategy is by far more efficient than the others (Subsection 7.2).

3. Evaluate to what extent the optimization techniques discussed in Section 6 do actually reduce the algorithm search space; the results point out that removing useless NDs highly improves efficiency and the most effective pruning technique is the one based on domination (Subsection 7.3).

4. Discuss how BBND scales with the problem size; remarkably, though obviously the execution time increases with the number of attributes in Y_j and with the number of NDs, good bounds are always found during the very early stages of search space exploration (Subsection 7.4).

5. Analyze how efficiency and effectiveness depend on the topology of NDs; we found that performances with clustered NDs progressively deteriorate when the FDs are defined on larger groups of attributes (Subsection 7.5).

Table 1: ND effectiveness

	Δ_1	Δ_2	Δ_3
additional NDs	10	25	50
% bound reduction	57%	89%	96%

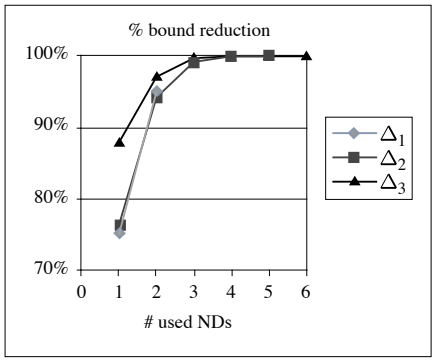

Figure 9: Bound reduction of BBND as a function of the number of additional proper NDs used in the optimal solution

7.1. Effectiveness

To evaluate the effectiveness of NDs, we measure how the basic bound of each Y_j is reduced using the different ND sets. The percentage average reductions are reported in Table 1 for increasing numbers of NDs. Besides, Figure 9 shows how the average reduction depends on the number of additional proper NDs used in the optimal solution. Clearly, NDs are highly effective in reducing cardinality bounds: even having a single additional proper ND in the optimal solution reduces the cardinality bound by about 80%. The percentage reduction is similar for the three sets of NDs; Δ_3 is slightly more effective since more NDs are available, so finding a lower weight —for the same number of used NDs— is more likely.

7.2. Comparison of Enumeration Strategies

The first set of tests aims at comparing the four strategies for exploring the search space: breadth-first (BrF), depth-first (DF), best-first (BeF), and cheapest-first (ChF). Table 2 reports, for each strategy, the execution time, the total number of search space nodes (i.e., the number of ND-paths generated), the number of search space leaves (i.e., the number of ND-paths that are not extended either because they are solutions or because of the application of some pruning technique), the percentage of leaves pruned by lazy/eager domination, and the maximum number of active ND-paths in the queue. Clearly, BeF is by far more efficient than the other strategies. In particular, DF generates a huge number of nodes (more than five millions) due to the lack of a smart

Table 2: Comparison of enumeration strategies using Δ_1 (all data are averaged on 300 optimization problems)

	BeF	DF	BrF	ChF
BBND execution time (secs.)	11.81	134.72	2 433.59	217.39
no. nodes	316 417	5 255 043	305 890	91 155
no. leaves	247 187	2 950 032	254 389	74 506
% lazily dominated leaves	62.63%	15.01%	68.98%	57.19%
% eagerly dominated leaves	24.02%	7.59%	30.03%	11.46%
max. active ND-paths	457	27	30 482	3 850

strategy for choosing the next ND for extension, which leads to an increase of the time to find a strict bound and makes numerical pruning less effective. This is confimed by the percentage of nodes pruned by lazy/eager domination, that is much lower than the one of BeF. As to BrF, its execution times are large (almost 2 500 secs.) despite the fact that minimality checks need not be carried out; indeed, BrF wastes most of the time in handling its huge queue of active ND-paths. Similar considerations hold for ChF, that generates a long queue thus suggesting that this enumeration strategy is quite ineffective. In light of the above, we adopt BeF for all subsequent tests.

7.3. Evaluation of Pruning Techniques

In order to verify how effective the proposed optimization techniques are in improving efficiency by pruning the search space, we compare BBND with three variants: $\mathsf{BBND}_{\mathsf{useless}}$ (where useless NDs are not initially removed), $\mathsf{BBND}_{\mathsf{minimal}}$ (where no minimality check is executed), and $\mathsf{BBND}_{\mathsf{eager}}$ (where no eager domination check is executed). The Naïve algorithm, where all pruning techniques are switched off, is not evaluated because its computational costs with our test sets would be prohibitive.

Table 3 summarizes the results. Besides the execution times, it reports the number of useless NDs, the total number of nodes in the search space, the number of leaves in the search space, and the percentages of leaves that correspond to solutions, to lazily/eagerly dominated ND-paths, to non-minimal ND-paths, and to repeated ND-paths, respectively. It is apparent from these results that: (1) although the problem is NP-hard, our algorithm finds the optimal solution within a reasonable time even for a very large problem size thanks to its optimization strategies; (2) removing useless NDs highly improves efficiency, because detecting these NDs requires a small computational effort and may drastically cut the search space; (3) checking ND-path minimality may significantly reduce the search space size, but for large sets of NDs the cost may overcome the benefit; (4) the most effective pruning technique is the one based on eager domination, in fact the execution times of $\mathsf{BBND}_{\mathsf{eager}}$ are one order of magnitude higher than those of BBND.

The previous results could be somehow misleading when comparatively evaluating pruning techniques, because their *actual* effectiveness is partially hidden

Table 3: Comparison of pruning techniques (all data are averaged on 300 optimization problems)

		Δ_1	Δ_2	Δ_3
execution time (secs.)	BBND	11.81	14.62	62.09
	$\text{BBND}_{\text{useless}}$	79.8	15.63	90.97
	$\text{BBND}_{\text{minimal}}$	15.54	15.57	39.7
	$\text{BBND}_{\text{eager}}$	152.85	593.27	–
no. useless NDs		6	2	5
no. nodes	BBND	316 417	572 061	1 952 367
	$\text{BBND}_{\text{useless}}$	807 787	573 485	2 005 973
	$\text{BBND}_{\text{minimal}}$	367 923	778 018	2 169 724
	$\text{BBND}_{\text{eager}}$	5 282 603	6 057 819	–
no. leaves	BBND	247 187	472 294	1 696 905
	$\text{BBND}_{\text{useless}}$	639 960	471 038	1 745 219
	$\text{BBND}_{\text{minimal}}$	271 244	638 476	1 883 072
	$\text{BBND}_{\text{eager}}$	3 984 943	5 137 935	–
% solution leaves	BBND	0.17%	0.46%	0.69%
	$\text{BBND}_{\text{useless}}$	0.12%	0.45%	0.63%
	$\text{BBND}_{\text{minimal}}$	0.19%	0.49%	0.69%
	$\text{BBND}_{\text{eager}}$	0.10%	0.27%	–
% lazily dominated leaves	BBND	62.63%	77.89%	86.56%
	$\text{BBND}_{\text{useless}}$	60.86%	77.64%	86.66%
	$\text{BBND}_{\text{minimal}}$	67.55%	81.96%	87.23%
	$\text{BBND}_{\text{eager}}$	81.12%	95.28%	–
% eagerly dominated leaves	BBND	24.02%	13.72%	10.64%
	$\text{BBND}_{\text{useless}}$	22.92%	14.30%	10.67%
	$\text{BBND}_{\text{minimal}}$	28.45%	14.29%	10.81%
	$\text{BBND}_{\text{eager}}$	0.00%	0.00%	–
% non-minimal leaves	BBND	10.57%	4.11%	0.87%
	$\text{BBND}_{\text{useless}}$	10.59%	3.37%	0.74%
	$\text{BBND}_{\text{minimal}}$	0.00%	0.00%	0.00%
	$\text{BBND}_{\text{eager}}$	16.24%	2.90%	–
% repeated leaves	BBND	2.62%	3.82%	1.24%
	$\text{BBND}_{\text{useless}}$	5.51%	4.25%	1.31%
	$\text{BBND}_{\text{minimal}}$	3.81%	3.26%	1.27%
	$\text{BBND}_{\text{eager}}$	2.54%	1.56%	–

Table 4: Order-independent comparison of pruning techniques for BBND

	Δ_1	Δ_2	Δ_3
% lazily dominated leaves	62.80%	78.35%	87.25%
% eagerly dominated leaves	25.05%	15.36%	13.40%
% non-minimal leaves	44.75%	18.81%	6.16%
% repeated leaves	5.90%	4.48%	1.53%

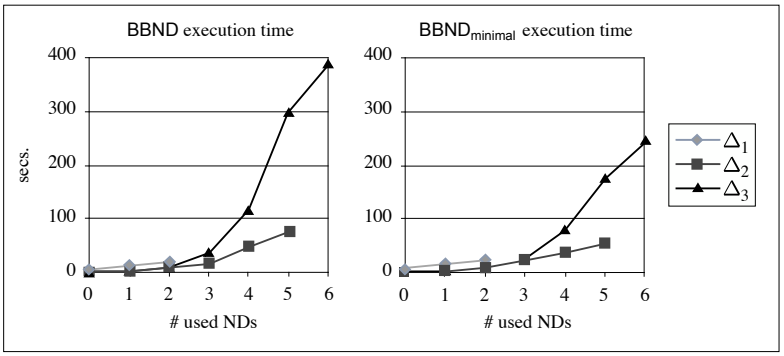

Figure 10: Efficiency of BBND and BBND$_{minimal}$ as a function of the number of additional proper NDs used in the optimal solution

behind their execution order. In our implementation, "low-cost" techniques are applied first; so, for instance, an ND-path that is both dominated and non-minimal will be pruned by domination since the domination check is less expensive than the minimality check. To gain further insight into this issue, Table 4 shows the actual percentages of nodes generated by BBND that would be pruned by each technique independently of their execution order (i.e., all checks were applied before actually pruning the node).[5] These results show that the pruning capability of the minimality check is actually higher than shown in Table 3. However, the order we adopt for carrying out pruning is easily justified by considering that a large part of non-minimal ND-paths are also dominated, and that minimality checks have higher costs.

7.4. Scalability

After the above discussion, it is clear that the two more efficient variants of the algorithm are BBND and BBND$_{minimal}$. A closer analysis reveals that their average execution times are strictly related to the number of proper NDs included in the optimal solution (see Figure 10). In both cases, the execution time increases much faster with the number of NDs for Δ_3 because the search

[5]The percentages do not sum to 100 because a node may satisfy two or more checks.

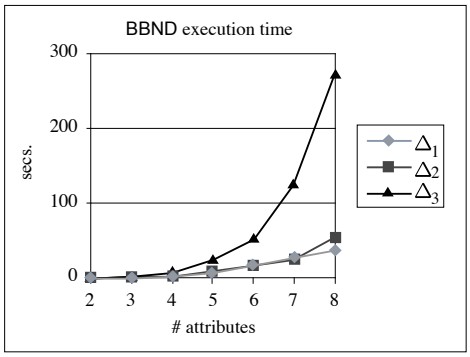

Figure 11: Efficiency of BBND as a function of the number of attributes in the bounded projection

space is larger. Besides, a comparison between the two parts of Figure 10 confirms that $\mathsf{BBND}_{\mathsf{minimal}}$ is more convenient than BBND for large problems due to the cost of minimality checks. Since there is no other notable difference between $\mathsf{BBND}_{\mathsf{minimal}}$ and BBND, in the following we will focus on the latter only.

There is a strong correlation between the number of proper NDs used in the optimal solution and the number of attributes in the projection Y_j whose cardinality is bounded. In fact, the number of NDs used turns out to be roughly proportional to the number of attributes in the bounded projection. As a consequence, the execution time scales with the number of attributes of Y_j in much the same way as it scales with the number of used NDs (see Figure 11).

Remarkably, regardless of the number of attributes in Y_j, good bounds are always found during the early stages of search space exploration. Figure 12 shows how the best bound found so far improves with elapsed time. After only 5 seconds, a bound that differs from the optimal one by less than 10% is found even for projections including 8 attributes. So, adopting a heuristic approach that returns the best bound found after a fixed amount of time would be highly effective.

7.5. Topology

Finally, we analyze how performance depends on the topology of NDs. An ND is useful to bound Y_j if it directly bounds some attributes in Y_j or if it can be used to build an ND-path to Y_j. The presence of chains of NDs positively affects the probability of building such ND-paths. In a database showing no multidimensional structure, the probability of having two or more FDs in a chain is higher if these are defined on a reduced group of attributes. This is confirmed by the tests made with Δ_3, Δ_4, and Δ_5: they include the same total numbers of FDs, but defined within increasingly larger groups of attributes. As shown in Table 5, effectiveness progressively decreases when FDs are defined over larger groups of attributes. A similar behavior is shown by proper NDs,

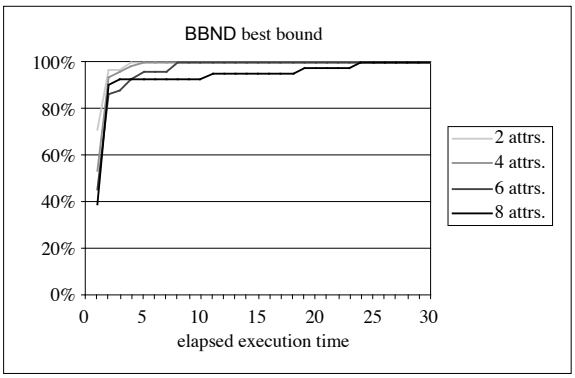

Figure 12: Best bound for BBND as a function of the elapsed time

as confirmed by comparing Δ_4 with Δ'_4 (in the latter, proper NDs are defined over a single larger group).

Overall, the effects of topology can be summarized as follows:

- The effectiveness in reducing the bound is much higher for Δ_3 than for the other ND sets, because in a multidimensional schema all the attributes in a hierarchy belong to the same chain of FDs (e.g., $A \to B$, $B \to C$) while in the other ND sets the attributes in each FD are randomly chosen within a single group of attributes (e.g., $AB \to DE$, $CE \to F$).

- The efficiency is maximized for Δ_4 and Δ'_4, since lazy and eager dominations avoid node proliferation and reduce non-minimality checks. On the other hand, though the number of nodes generated is higher in Δ_3 than in Δ_5, the efficiency over Δ_5 is lower due to a large number of minimality checks. This behavior is related to FD chains: in Δ_3 their presence determines a large number of ND-paths, while in Δ_5 their absence prevents from finding solutions or partial solutions capable of pruning the search space.

8. Related Work and Conclusions

In this paper we have considered the problem of reasoning with numerical dependencies (NDs). We have first shown that entailment of NDs can be decided using a variant of the classical chase procedure, which requires exponential space in the worst case. For the syntactic derivation of NDs, we have introduced a set of rules, called RED, which, although incomplete, can always determine if an ND from a set of attributes X to another set of attributes Y exists. We have also provided a partial characterization of the cases in which RED rules guarantee completeness. After characterizing the "interesting" derivable NDs in graph-theoretical terms, we have introduced an efficient branch & bound algorithm,

Table 5: Effectiveness and efficiency for ND sets with different topologies (all data are averaged on 300 optimization problems)

	Δ_3	Δ_4	Δ'_4	Δ_5
% bound reduction	96%	79%	73%	58%
execution time (secs.)	62.09	5.59	12.05	105.13
no. useless NDs	5	7	3	7
no. nodes	1 952 367	730 373	409 301	899 452
no. leaves	1 696 905	683 108	382 662	693 155
% solution leaves	0.69%	0.79%	0.44%	0.43%
% lazily dominated leaves	86.56%	94.08%	94.40%	63.35%
% eagerly dominated leaves	10.64%	4.99%	4.82%	31.35%
% non-minimal leaves	0.87%	0.06%	0.18%	2.19%
% repeated leaves	1.24%	0.09%	0.15%	0.68%

BBND, and evaluated its performance over several ND sets, demonstrating its efficiency with respect to a naïve derivation approach.

Two issues we left for future research are a deeper analysis of the cases in which RED rules can guarantee completeness, and the development of an optimized chase-based procedure.

The present work follows that from the same authors in [5], which focused on estimating the cardinality of aggregate views in multidimensional databases. The basic formal problem addressed by [5] was that of determining whether one set of NDs always provides a tighter bound of the cardinality of a view than another set of constraints. However, no algorithm for determining the best (least) upper bound was proposed.

Our ND-graph representation is largely inspired to the one proposed in [23] for FDs; the major difference with [23] is that, in the worst case,[6] we have to generate *all* possible minimal ND-paths from X to Y, which is not an issue for FDs.

Other works using NDs in different contexts are mentioned in the following.

Some works have considered cardinality constraints in the context of conceptual models. Since a cardinality constraint also imposes a lower bound, problems of consistency of a set of constraints arise, which is not an issue with NDs. For instance, in [8] an approach is presented for modeling n-ary relationships in conceptual schemata by unifying and extending the cardinality constraints expressed in a number of existing models. Rules for identifying inconsistencies among cardinality constraints are then defined by extending the work done in axiomatization of functional and NDs for relational databases.

More specifically, some works use dependencies of different types to express constraints in the Entity-Relationship model. For instance, in [9] the notion

[6] This arises if, depending on weights, no ND-path leading to a (non-optimal) Y-minimal ND-path is dominated.

of strong satisfiability is introduced to ensure that, given a set of cardinality ratio constraints that impose restrictions on the mappings between entities and relationships, no entity or relationship is compelled to be empty in all of the legal instances of the schema. In [24] the focus is on cardinality constraints, that impose restrictions on the number of relationships an object may be involved in. The entailment problem is faced, and combinatorial methods for reasoning about sets of cardinality constraints are proposed; besides, the interplay between cardinality constraints and FDs is discussed. Similarly, in [10] the satisfiability and implication problems for numerical constraints are faced with reference to the XML language. A numerical constraint is defined in terms of path expressions, and restricts the number of nodes that have the same values on some selected subnodes.

Other works use NDs in connection with conditional, nondeterministic, or approximate information. In [25] an extension of conditional FDs is proposed to uniformly express cardinality constraints, domain-specific conventions, and patterns of semantically-related attribute values. Intuitively, a conditional functional dependency specifies a pattern of values for two sets of attributes X and Y: each tuple that matches the pattern on X must also respect the pattern on Y, and an upper bound of the number of distinct values of Y related to each value of X is specified. Complexity bounds for the satisfiability and implication problems associated with conditional FDs are also established. In [13] a set of constraints expressed by NDs enable efficient query processing in nondeterministic databases. In [11], NDs enable management of indefinite information in relations, in particular they are used to mine a relation to see how well a given set of FDs is approximated.

References

[1] J. Grant, J. Minker, Numerical dependencies, in: H. Gallaire, J. Minker, J.-M. Nicolas (Eds.), Advances in Database Theory, Vol. II, Plenum Publ. Co., 1984.

[2] P. Ciaccia, D. Maio, Domains and active domains: What this distinction implies for the estimation of projection sizes in relational databases, IEEE Trans. on Knowledge and Data Engineering 7 (4) (1995) 641–655.

[3] M. Muralikrishna, D. DeWitt, Equi-depth histograms for estimating selectivity factors for multi-dimensional queries, in: Proc. ACM Sigmod Conf., Chicago, IL, 1988, pp. 28–36.

[4] W. Hou, G. Özsoyoglu, Statistical estimators for aggregate relational algebra queries, ACM Trans. on Database Systems 16 (4) (1991) 600–654.

[5] P. Ciaccia, M. Golfarelli, S. Rizzi, Bounding the cardinality of aggregate views through domain-derived constraints, Data and Knowledge Engineering 45 (2) (2003) 131–153.

[6] P. Vassiliadis, Gulliver in the land of data warehousing: practical experiences and observations of a researcher, in: Proc. DMDW'00, Stockholm, Sweden, 2000, pp. 12/1–12/16.

[7] D. Theodoratos, M. Bouzeghoub, A general framework for the view selection problem for data warehouse design and evolution, in: Proc. DOLAP 2000, Washington, DC, 2000, pp. 1–8.

[8] A. J. McAllister, Complete rules for n-ary relationship cardinality constraints, Data Knowl. Eng. 27 (3) (1998) 255–288.

[9] M. Lenzerini, P. Nobili, On the satisfiability of dependency constraints in entity-relationship schemata, in: Proc. VLDB, Brighton, England, 1987, pp. 147–154.

[10] S. Hartmann, S. Link, Numerical constraints for XML, in: Proc. WoLLIC, Rio de Janeiro, Brazil, 2007, pp. 203–217.

[11] E. Collopy, M. Levene, Resampling in an indefinite database to approximate functional dependencies, in: Proc. PKDD, Nantes, France, 1998, pp. 291–299.

[12] G. Piatetsky-Shapiro, C. J. Matheus, Measuring data dependencies in large databases, in: Proc. Knowledge Discovery in Databases Workshop, 1993, pp. 162–173.

[13] K. V. Vadaparty, S. A. Naqvi, Using constraints for efficient query processing in nondeterministic databases, IEEE Trans. Knowl. Data Eng. 7 (6) (1995) 850–864.

[14] D. Maier, A. Mendelzon, Y. Sagiv, Testing implications of data dependencies, ACM Trans. on Database Systems 4 (4) (1979) 455–469.

[15] R. Fagin, P. Kolaitis, R. Miller, L. Popa, Data exchange: Semantics and query answering, Theoretical Computer Science 336(1) (2005) 89–124.

[16] M. A. Casanova, R. Fagin, C. H. Papadimitriou, Inclusion dependencies and their interaction with functional dependencies, in: Proc. PODS, Los Angeles, California, 1982, pp. 171–176.

[17] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, 1979.

[18] R. G. Downey, M. R. Fellows, Fixed parameter tractability and completeness, in: Complexity Theory: Current Research, 1992, pp. 191–225.

[19] J. Guo, R. Niedermeier, Exact algorithms and applications for tree-like weighted set cover, J. Discrete Algorithms 4 (4) (2006) 608–622.

[20] C. Beeri, P. Bernstein, Computational problems related to the design of normal form relational schemas, ACM Trans. on Database Systems 4 (1) (1979) 30–59.

[21] V. Chvatal, A greedy heuristic for the set-covering problem, Mathematics of Operations Research 4 (3) (1979) 233–235.

[22] W. Zhang, Branch-and-bound search algorithms and their computational complexity, Tech. Rep. ISI/RR-96-443, USC/Information Sciences Institute (1996).

[23] G. Ausiello, A. D'Atri, D. Saccà, Graph algorithms for functional dependency manipulation, Journal of the ACM 30(4) (1983) 752–766.

[24] S. Hartmann, On the implication problem for cardinality constraints and functional dependencies, Ann. Math. Artif. Intell. 33 (2-4) (2001) 253–307.

[25] W. Chen, W. Fan, S. Ma, Incorporating cardinality constraints and synonym rules into conditional functional dependencies, Inf. Process. Lett. 109 (14) (2009) 783–789.

Appendix

Appendix A. Proofs of Formal Results

Theorem 1. *Given a set of NDs Δ over $R(U)$, let $\delta : X \xrightarrow{k} Y$ be an ND and T_δ be the tableau for δ. It is $\Delta \models \delta$ iff, for all tableaux in $Chase_\Delta(T_\delta)$, the number of distinct rows on Y is at most k, in which case we say that the chase* succeeds *on δ, otherwise it* fails.

Proof. ($\Rightarrow$) If there exists a tableau $T_l \in Chase_\Delta(T_\delta)$ with $k+1$ distinct rows on Y (i.e.,the chase fails), then any relation r with $k+1$ tuples obtained from a *valuation* ρ of T_l, $r = \rho(T_l)$, i.e., a homomorphism that transforms each variable of T_l into a value of the corresponding domain, and such that $k+1$ different Y-values are generated, would represent a violation of δ. On the other hand, since T_l satisfies all NDs in Δ, so does $r = \rho(T_l)$, from which it follows that $\Delta \not\models \delta$.
($\Leftarrow$) Consider a relation $r \in Sat(\Delta)$ and $k+1$ tuples $t'_1, \ldots, t'_{k+1}$ in r which agree on X. We have to prove that the number of distinct Y-values in such tuples is at most k. Let ρ be a valuation from T_δ to $\{t'_1, \ldots, t'_{k+1}\}$. We show that there exists a tableau $T_l \in Chase_\Delta(T_\delta)$ such that ρ is also a valuation from T_l to $\{t'_1, \ldots, t'_{k+1}\}$, from which the result follows since by hypothesis $\rho(T_l)$ cannot yield more than k distinct Y-values.

Let $T_1, \ldots, T_p$, with $T_1 \equiv T_\delta$ and $T_p \equiv T_l$ be a sequence of tableaux, where T_{j+1} is obtained from T_j through a chase step. We prove by induction on j

that, for each $j = 1, \ldots, p$, ρ is a valuation from T_j to $\{t'_1, \ldots, t'_{k+1}\}$. The basis, $j = 1$, trivially holds. Then, assume that the result holds for T_j, $j < p$. Since T_j is not a leaf of the chase tree, there exists $\delta_{i_j} : W_{i_j} \xrightarrow{k_{i_j}} Z_{i_j} \in \Delta$ that is not satisfied by T_j. Since $\{t'_1, \ldots, t'_{k+1}\}$ satisfies δ_{i_j} (because r does), there exists (at least) one tableaux, say $T_{j,s}$, obtained from T_j using δ_{i_j} and homomorphism h_s, such that $\{t'_1, \ldots, t'_{k+1}\}$ is obtained from a valuation of $T_{j,s}$. Assume that h_s is $h_s(t_{s,1}(Z_{i_j})) = h_s(t_{s,2}(Z_{i_j}))$, and the identity elsewhere. Thus, the only difference between T_j and $T_{j,s}$ is that all occurrences of the $Z_{i,j}$-variable appearing in $t_{s,2}(Z_{i_j})$ are replaced by occurrences of the $Z_{i,j}$-variable appearing in $t_{s,1}(Z_{i_j})$, or viceversa. It follows that ρ is also a valuation from $T_{j,s} \equiv T_{j+1}$ to $\{t'_1, \ldots, t'_{k+1}\}$, proving the assert. □

Theorem 2. *Given a set of NDs Δ over $R(U)$, let $\delta : X \xrightarrow{k} Y$ be an ND and T_δ be the tableau for δ. If the chase succeeds on δ, then the tableau in $Chase_\Delta(T_\delta)$ with the maximum number of distinct rows on Y has exactly $k^\perp(X, Y)$ distinct rows on Y.*

Proof. The result is a direct consequence of Theorem 1. Indeed, for any relation $r \in Sat(\Delta)$ with $k+1$ tuples $t'_1, \ldots, t'_{k+1}$ which agree on X, the proof of Theorem 1 shows that there is a tableau $T_l \in Chase_\Delta(T_\delta)$ and a valuation ρ such that $\rho(T_l) = \{t'_1, \ldots, t'_{k+1}\}$, from which the result immediately follows. □

Lemma 3. *Given a set of NDs Δ over $R(U)$, if $X \xrightarrow{l \cdot w(\delta_i)^p} Y \in \Delta^+_{RED}$, then $X \xrightarrow{l \cdot w(\delta_i)} Y \in \Delta^+_{RED}$ for each $\delta_i \in \Delta$, $X, Y \subset U$, and $l, p \geq 1$.*

Proof. We start by observing that case $p \geq 3$ can easily be led back to case $p = 2$ by induction. We also note that, since E is the only rule in RED that introduces new weight factors, then the repeated factor must necessarily have been introduced by rule E. Three cases are possible:

1. The repeated factor has been introduced by applying E as follows:

$$X \xrightarrow{l \cdot k_i} VW \wedge V \xrightarrow{k_i} Z \vdash X \xrightarrow{l \cdot k_i^2} VWZ$$

   where the second ND used in the premise is $\delta_i : V \xrightarrow{k_i} Z \in \Delta$. Since δ_i has already been used to derive $X \xrightarrow{l \cdot k_i} VW$, then $Z \subseteq VW$ and $VWZ \equiv VW$.[7] This means that no new ND has been derived by this application of E, which proves the assertion.

[7] If also rule D has been used to derive $X \xrightarrow{l \cdot k_i} VW$, it may not be $Z \subseteq VW$; however, if this is the case, an alternative derivation of $X \xrightarrow{l \cdot w(\delta_i)^p} Y$ can always be found where D is applied at the end, which ensures that $Z \subseteq VW$.

2. The repeated factor has been introduced by applying E as follows:

$$X \xrightarrow{l' \cdot k_i} VW \wedge V \xrightarrow{l'' \cdot k_i} Z \vdash X \xrightarrow{l' \cdot l'' \cdot k_i^2} VWZ$$

where the second ND used in the premise is not in Δ and has been derived using also δ_i, with $w(\delta_i) = k_i$. Since the RED rules preserve the left-hand sides of the NDs they are applied to, the derivation τ that brought to $V \xrightarrow{l'' \cdot k_i} Z$ started with an ND $\overline{\delta}$ with left-hand side V. Then an alternative derivation for $X \xrightarrow{l' \cdot l'' \cdot k_i^2} VWZ$ can be written that applies rule E to $X \xrightarrow{l' \cdot k_i} VW$ and $\overline{\delta}$ to derive $\overline{\delta}'$, and then orderly applies the derivation steps in τ using $\overline{\delta}'$ instead of $\overline{\delta}$ as a premise. By repeatedly applying this argument we go back to case 1, which proves the assertion.

3. The repeated factor has been introduced by applying E as follows:

$$X \xrightarrow{k_i} VW \wedge V \xrightarrow{l \cdot k_i} Z \vdash X \xrightarrow{l \cdot k_i^2} VWZ$$

where $\delta_i : X \xrightarrow{k_i} VW \in \Delta$. Since δ_i has already been used to derive $V \xrightarrow{l \cdot k_i} Z$, it is $VW \subseteq Z$ and $VWZ \equiv Z$, so the same argument used in case 1 proves the assertion.

□

Theorem 4. *Given a set of NDs Δ over $R(U)$ and sets of attributes $X, Y \subseteq U$, determining if $\Delta \vdash_{RED} X \xrightarrow{k} Y$ is NP-hard.*

Proof. The proof is by polynomial-time reduction from the MINIMUM COVER problem, whose NP-hardness is well known [17]. Given a finite set $S = \{s_1, \ldots, s_n\}$ and a collection $C = \{C_1, \ldots, C_m\}$ of subsets of S, a cover of S is a subset $C' \subseteq C$ such that each element of S belongs to at least one member of C'. The MINIMUM COVER problem is to determine if C contains a cover of S of size $|C'| \leq K$.

For each element $s_i \in S$ define an attribute A_i, $i = 1, \ldots, n$, and let U be the disjoint union of X and S, i.e., $U = XS$, $X \cap S = \emptyset$. For each C_j define the ND $\delta_j : X \xrightarrow{2} C_j$, and let $\Delta = \{\delta_j | j = 1, \ldots, m\}$ and $Y \equiv S$. We show that C contains a cover of S of size K iff $X \xrightarrow{2^K} Y \in \Delta^+_{RED}$.

($\Rightarrow$) Assume C contains a cover $C' = \{C_{i_1}, \ldots, C_{i_K}\}$ of size K. By applying $K - 1$ times the Union (U) rule (which is derivable from the RED rules) to the NDs $\delta_{i_1}, \ldots, \delta_{i_K}$, the ND $X \xrightarrow{2^K} Y$ is derived.

($\Leftarrow$) Going the other way, assume $\delta : X \xrightarrow{2^K} Y \in \Delta^+_{RED}$. Since all NDs have the same weight, it is obvious that δ has been derived using K NDs in Δ. The right-hand sides of such NDs are a cover of S of size K, as it can be immediately verified.

We conclude the proof with a remark on the Successor (S) rule. As said, this rule can be safely ignored if one aims to find the ND from X to Y with the minimum derivable weight, yet it is needed to show that the derivation problem for NDs is in NP, i.e., that the derivation of $\delta : X \xrightarrow{k} Y$ has polynomial length. For this, let $\delta^{\perp} : X \xrightarrow{k^{\perp}_{REDS}} Y$, which is in Δ^{+}_{REDS} by hypothesis. Since in the derivation of $\delta^{\perp}$ all NDs in Δ are used at most once (See Lemma 3), this also proves that there exists a derivation of δ of polynomial length, because a single application of the S rule suffices.[8] □

Theorem 5. *Given a set of NDs Δ over $R(U)$, for any sets of attributes $X, Y \subseteq U$ there exists an ND from X to Y iff an ND from X to Y is derivable using the RED rules.*

Proof. ($\Leftarrow$) Obvious, since RED rules are sound.
($\Rightarrow$) The proof is based on arguments that generalize those used to prove the completeness of Armstrong axioms for FDs. In particular, we exhibit, for each finite value of k, an instance $r_k \in Sat(\Delta)$ with $k+1$ tuples such that: 1) $\Delta \nvdash_{RED} X \xrightarrow{k} Y$ and 2) the ND $X \xrightarrow{k} Y$ does not hold in r_k. From this it follows that if no ND from X to Y can be derived, that no ND from X to Y is entailed by Δ.

For any value of k consider a relation r_k with $k+1$ tuples which agree on the attributes of X^{+}_{ND}, and have pairwise different values on all the other attributes. We claim that $r_k \in Sat(\Delta)$. By contradiction, assume that there exists an ND $\delta_i : W_i \xrightarrow{k_i} Z_i \in \Delta$ that is violated by r_k. For this, it has to be $W_i \subseteq X^{+}_{ND}$ and $W_i \not\subseteq X^{+}_{ND}$. But this would contradict the hypothesis that X^{+}_{ND} is the ND-closure of X, from which we conclude that r_k is a legal instance. Since $X \xrightarrow{k} Y$ is not derivable, it is $Y \not\subseteq X^{+}_{ND}$, thus $X \xrightarrow{k} Y$ is not entailed by Δ. Since this holds for any value of k, it follows that there is no ND from X to Y. □

Lemma 5. *Let $\Delta = \{\delta_i : X \xrightarrow{k_i} Z_i, i = 1, \ldots, m\}$, $Y \subseteq \cup_i Z_i$, and $X \cap Y = \emptyset$. There exists an instance $r^{\perp}$ with $k^{\perp}_{RED}(X, Y)$ tuples with the same value on X and distinct Y-values, such that $r^{\perp} \in Sat(\Delta^{\perp})$.*

Proof. We distinguish the attributes in U in "local","join", and "don't care" ones. An attribute A_i is: 1) *local* (to δ_i) if it belongs to Y and only appears in a single right-hand side of an ND $\delta_i \in \Delta^{\perp}$; 2) *join* if it appears in more than

[8] We note that the problem is *not* in NP if the S rule is replaced with the one: $X \xrightarrow{k} Y \vdash X \xrightarrow{k+1} Y$, as in the original paper by Grant and Minker [1], since in this case the length of a shortest derivation of δ would depend on the value of k.

one ND in $\Delta^\perp$; 3) *don't care* otherwise. Notice that a don't care attribute is not in Y and that each ND in $\Delta^\perp$ necessarily has at least one local attribute, otherwise dropping δ_i from $\Delta^\perp$ would still yield a valid ND from X to Y.

The instance $r^\perp$ has $k^\perp_{RED}(X,Y)$ tuples which agree on the values of all join and don't care attributes. Let $L_i \subseteq Z_i \cap Y$ be the set of local attributes of δ_i. For each $\delta_i \in \Delta^\perp$ we choose k_i distinct L_i-values from $dom(L_i)$, and generate all possible $\prod_i k_i = k^\perp_{RED}(X,Y)$ distinct combinations, each of which identifies a specific tuple of $r^\perp$. Notice that, since each join attribute has a single value in $r^\perp$, this construction is indeed possible.

By construction, for each $\delta_i : X \xrightarrow{k_i} Z_i \in \Delta^\perp$ there is a single X-value in $r^\perp$ (since attributes in X cannot be local) and there are exactly k_i distinct Z_i-values (since there are k_i distinct L_i-values, and each attribute in $Z_i \setminus L_i$ has a single value). Thus, $r^\perp$ satisfies all NDs in $\Delta^\perp$. □

Theorem 6. *For each $\delta_i : X \xrightarrow{k_i} Z_i \in \Delta^\perp$ let $L_i \subseteq Z_i$ be the set of local attributes in δ_i and define $\Lambda = \Delta \setminus \Delta^\perp = \{\delta_j : X \xrightarrow{k_j} W_j\}$. If one of the following cases occurs, then it is $k^\perp_{RED}(X,Y) = k^\perp(X,Y)$:*

1. *For all $\delta_j \in \Lambda$ and all $\delta_i \in \Delta^\perp$ it is $W_j \cap L_i = \emptyset$;*
2. *Each ND in $\Delta^\perp$ has a single local attribute, $L_i = \{A_i\}$, and it is:*

$$\prod_{i:A_i \in W_j} k_i \leq k_j \qquad \forall \delta_j \in \Lambda \tag{3}$$

3. *For all $\delta_j \in \Lambda$ and all $\delta_i \in \Delta^\perp$ it is $|W_j \cap L_i| \leq 1$, i.e., $W_j \cap L_i = \{A_{i,j}\}$ or $W_j \cap L_i = \emptyset$, and there exists an instance $r^\perp$ such that the following system of inequalities admits solution:*

$$\prod_{A_p \in L_i} |r^\perp[A_p]| \geq k_i \qquad \forall \delta_i \in \Delta^\perp \tag{4}$$

$$\prod_{i:W_j \cap L_i \neq \emptyset} |r^\perp[A_{i,j}]| \leq k_j \qquad \forall \delta_j \in \Lambda \tag{5}$$

Proof. Case 1: Any instance $r^\perp \in Sat(\Delta^\perp)$, as defined in Lemma 5, has exactly k_i distinct L_i-values and a single value on each other attribute, as the proof of the same lemma shows. Thus, no ND in Λ can influence the validity of $r^\perp$.

Case 2: An ND $\delta_j : X \xrightarrow{k_j} W_j \in \Lambda$ is violated by an instance $r^\perp \in Sat(\Delta^\perp)$ iff $r^\perp$ has more than k_j distinct W_j-values. Thus, when inequality (3) is satisfied, so it is δ_j.

Case 3: Inequalities (5) are a generalization of (3) to the case in which L_i can consist of more than one local attribute, thus what observed for case 2 applies also here. Since when an ND $\delta_i : X \xrightarrow{k_i} Z_i \in \Delta^\perp$ introduces more

than one local attribute there are several possibilities to obtain k_i distinct L_i-values, inequalities (4) are there to guarantee that each attribute $A_p \in L_i$ has a sufficient number of distinct values so that k_i distinct L_i-values can indeed be obtained. □

Lemma 6. *Let G^X_Π be an ND-path from X to Y. Then $X \xrightarrow{\omega(G^X_\Pi)} Y \in \Delta^+_{RED}$.*

Proof. It is sufficient to prove that $X \xrightarrow{\omega(G^X_\Pi)} Attr(G^X_\Pi) \in \Delta^+_{RED}$, the case $Y \subset Attr(G^X_\Pi)$ following by rule D. The proof is by induction on the number of steps needed to build the ND-path.

In the base case the ND-path is $G^X_\emptyset$, i.e., the ND-graph induced by X, for which it is $Attr(G^X_\emptyset) = X$ and $\omega(G^X_\emptyset) = 1$. This represents the trivial ND $X \to X \in \Delta^+_{RED}$ (rule R). Assume now that one has an ND-path $G^X_{\Pi'}$ from X, obtained by m applications of the 2nd rule in Definition 6. At step $m+1$ $G^X_{\Pi'}$ is extended with the full arc $\langle W, Z\rangle_k$, thus using the ND $W \xrightarrow{k} Z \in \Delta$, with $W \subseteq Attr(G^X_{\Pi'})$ and $Z \not\subseteq Attr(G^X_{\Pi'})$. From the inductive hypothesis, it is $X \xrightarrow{\omega(G^X_{\Pi'})} Attr(G^X_{\Pi'}) \in \Delta^+_{RED}$. Since $W \subseteq Attr(G^X_{\Pi'})$, the result follows by rule E. □

Theorem 7. *Let G_Δ be the ND-graph induced by a set of NDs Δ, and let $X \in \mathcal{V}$. There exists in G_Δ a Y-minimal ND-path G^X_Π from X to Y having weight $\omega(G^X_\Pi)$, iff $\delta : X \xrightarrow{\omega(G^X_\Pi)} Y \in \Delta^*_{RED}$.*

Proof. ($\Leftarrow$) The proof is by induction on the number of NDs of Δ used in a derivation of δ. In the base case no ND is used, thus it has necessarily to be $Y \subseteq X$. Since $X \in \mathcal{V}$ by hypothesis, the ND-graph induced by X is the required Y-minimal ND-path. For the inductive step, assume the result holds for all NDs whose derivation requires no more than $m-1$ NDs from Δ ($m \geq 1$), whereas δ can be derived by using m NDs. Since $\delta \in \Delta^*_{RED}$ by hypothesis, no proper subset of such NDs can derive it. The last application of rule E in a derivation of δ has therefore the form:

$$\delta' : X \xrightarrow{\omega(G^X_{\Pi'})} Y_{m-1},\ \delta_m : W_m \xrightarrow{k_m} Z_m \ \vdash X \xrightarrow{\omega(G^X_{\Pi'})\cdot k_m} Y_m$$

where $W_m \subseteq Y_{m-1}$, $\delta_m \in \Delta$, and $Y \subseteq Y_m = Y_{m-1}Z_m$. Since δ is tight, it is $Y \not\subseteq Y_{m-1}$.

By the inductive hypothesis, $\delta' \in \Delta^*_{RED}$ and there exists a Y_{m-1}-minimal ND-path $G^X_{\Pi'}$ from X to Y_{m-1} with weight $\omega(G^X_{\Pi'})$. The ND-path obtained by adding to $G^X_{\Pi'}$ node Z_m and the full arc $\langle W_m, Z_m\rangle_{k_m}$ is the required Y-minimal ND-path for δ.

($\Rightarrow$) Let $G^X_\Pi = (\mathcal{V}'', \mathcal{E}'')$. By Lemma 6 we have that $\delta : X \xrightarrow{\omega(G^X_\Pi)} Y \in \Delta^+_{RED}$.

It remains to show that δ is tight. By contradiction, assume δ is loose, i.e., there exists a tight ND $\delta' \in \Delta^*_{RED}$ with $K(\delta') \subset K(\delta)$. From the first part of the proof we know that there exists a Y-minimal ND-path $G^X_{\Pi'} = (\mathcal{V}', \mathcal{E}')$ for δ'. Since $K(\delta') \subset K(\delta)$ implies $\mathcal{E}' \subset \mathcal{E}''$, this contradicts the hypothesis that G^X_Π is Y-minimal. □

Lemma 7. *Let G^X_Π and G^X_Γ be two ND-paths from X, with $\Gamma \subset \Pi$ and G^X_Π obtained by extending G^X_Γ. If G^X_Γ is not $Attr(G^X_\Gamma)$-minimal, then G^X_Π is not Y-minimal for all $Y \subseteq Attr(G^X_\Pi)$.*

Proof. By contradiction, assume there exists Y such that G^X_Π is Y-minimal, yet G^X_Γ is not $Attr(G^X_\Gamma)$-minimal. Then, there exists $G^X_{\Gamma'}$, with $\Gamma' \subset \Gamma$, such that $Attr(G^X_{\Gamma'}) = Attr(G^X_\Gamma)$. Let $\Pi' = \Pi \setminus \Gamma \cup \Gamma'$. The ND-subgraph $G^X_{\Pi'}$ is an ND-path, since it can be obtained by extending $G^X_{\Gamma'}$ using the same sequence of steps as in the extension from G^X_Γ to G^X_Π. It follows that G^X_Π is not $Attr(G^X_\Pi)$-minimal. Since $Y \subseteq Attr(G^X_\Pi)$, G^X_Π cannot be Y-minimal, a contradiction. □

Lemma 9. *Let G^X_Γ be an ND-path from X, obtained by extending a minimal ND-path G^X_Π with the ND $\delta : W \xrightarrow{k} Z$. Let $\delta_i : W_i \xrightarrow{k_i} Z_i \in \Pi$ be a non-essential ND for G^X_Π. If $Z \cap Z_i = \emptyset$, then the ND-subgraph G_{Γ_i} induced by $\Gamma_i = \Gamma \setminus \delta_i$ is not an ND-path from X.*

Proof. Consider the ND-subgraph G_{Π_i} induced by $\Pi_i = \Pi \setminus \delta_i$. Clearly, it is $Attr(G_{\Pi_i}) = Attr(G^X_\Pi)$, since δ_i is not essential for G^X_Π. However, by hypothesis, G_{Π_i} is not an ND-path from X (otherwise G^X_Π would not be minimal). In turn, this implies that there exists at least one ND $\delta^* : W^* \xrightarrow{k^*} Z^*$ in Π_i such that the node W^* is not reachable from X in G_{Π_i}.

Since $\Gamma_i = \Pi_i \cup \{\delta\}$, and the ND-subgraph induced by a set of NDs is univocally determined, it follows that G_{Γ_i} equals G_{Π_i} plus the full arc corresponding to δ. Since $Z \cap Z_i = \emptyset$ by hypothesis, it follows that the node W^* is not reachable from X also in G_{Γ_i}, thus G_{Γ_i} is not an ND-path from X. □

Lemma 10. *Let G^X_Π and G^X_Γ be two ND-paths from X. If G^X_Π eagerly dominates G^X_Γ then, for any ND-path $G^X_{\Gamma_i}$ obtained by extending G^X_Γ with an ND δ_i, either (1) G^X_Π eagerly dominates $G^X_{\Gamma_i}$, or (2) the ND-path $G^X_{\Pi_i}$ that extends G^X_Π with δ_i eagerly dominates $G^X_{\Gamma_i}$.*

Proof. Let $\Gamma_i = \Gamma \cup \{\delta_i : W_i \xrightarrow{k_i} Z_i\}$, thus $Attr(G^X_{\Gamma_i}) = Attr(G^X_\Gamma) \cup Z_i$ and $\omega(G^X_{\Gamma_i}) = \omega(G^X_\Gamma) \cdot k_i$. If δ_i can be used to also extend G^X_Π, then the result immediately follows. If this is not the case, since $Z_i \subset Attr(G^X_\Pi)$, it is $Attr(G^X_{\Gamma_i}) \subseteq Attr(G^X_\Pi)$ and $\omega(G^X_{\Gamma_i}) \geq \omega(G^X_\Pi)$, thus G^X_Π eagerly dominates $G^X_{\Gamma_i}$. □