Published in Journal of Computer Security (JCS), Volume 20, Issue 5, Pag 463-508, 2012.

An OBDD approach to enforce confidentiality and visibility constraints in data publishing*

Valentina Ciriani, Sabrina De Capitani di Vimercati, Sara Foresti,
Giovanni Livraga, Pierangela Samarati

DTI - Università degli Studi di Milano - 26013 Crema, Italy
firstname.lastname@unimi.it

Corresponding author: Pierangela Samarati
DTI - Università degli Studi di Milano
Via Bramante 65 - 26013 Crema, Italy
pierangela.samarati@unimi.it
phone: +39-0373-898061, *fax:* +39-0373-898074

Abstract

With the growing needs for data sharing and dissemination, privacy-preserving data publishing is becoming an important issue that still requires further investigation. In this paper, we make a step towards private data publication by proposing a solution based on the release of vertical views (*fragments*) over a relational table that satisfy confidentiality and visibility constraints expressing requirements for information protection and release, respectively. We translate the problem of computing a fragmentation composed of the minimum number of fragments into the problem of computing a maximum weighted clique over a fragmentation graph. The fragmentation graph models fragments, efficiently computed using Ordered Binary Decision Diagrams (OBDDs), that satisfy all the confidentiality constraints and a subset of the visibility constraints defined in the system. We then show an exact and a heuristic algorithm for computing a minimal and a locally minimal fragmentation, respectively. Finally, we provide experimental results comparing the execution time and the fragmentations returned by the exact and heuristic algorithms. The experiments show that the heuristic algorithm has low computation cost and computes a fragmentation close to optimum.

keywords: Privacy, fragmentation, confidentiality and visibility constraints, OBDDs, maximum weighted clique

*A preliminary version of this paper appeared under the title "Enforcing Confidentiality and Data Visibility Constraints: An OBDD Approach," in *Proc. of the 25th Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy* (DBSec 2011), Richmond, VA, USA, July 2011 [12].

1 Introduction

Information sharing and dissemination are typically selective processes. While on one side, there is a need - or demand - for making certain information available to others, there is on the other side an equally strong need to ensure proper protection of sensitive information. It is therefore important to provide data holders with means to express and enforce possible constraints over their data, modeling the need for information of the data recipients (*visibility* constraints) and the need for protecting confidential information from an improper disclosure (*confidentiality* constraints).

Recent proposals considering confidentiality and visibility constraints have put forward the idea of computing vertical fragments over the original data structure (typically a relation) so that all the constraints are satisfied [1, 10, 11, 14]. While such proposals have been introduced as a way of departing from data encryption when relying on external storage services, data fragmentation can result appealing also in data publication scenarios. In fact, data fragments can be seen as different (vertical) views that a data holder can release to external parties to satisfy their demand for information, while at the same time guaranteeing that confidential information is not disclosed. The problem of computing data views taking into consideration both privacy needs and visibility requirements makes however the data fragmentation problem far from trivial. In particular, ensuring some meaningful form of minimality of the fragments to be released (to the aim of avoiding unnecessary fragmentation of attributes), makes the problem NP-hard [14].

In this paper, we propose a new modeling of the fragmentation problem that exploits the representation of confidentiality and visibility constraints as Boolean formulas, and of fragments as truth assignments over Boolean variables corresponding to attributes in the original relation. In this way, the computation of a fragmentation that satisfies the given constraints relies on the efficiency with which Boolean formulas are represented and manipulated. Since the classical methods for operating on Boolean formulas are impractical for large-scale problems, we adopt reduced Ordered Binary Decision Diagrams (OBDDs), which are a canonical form for representing and efficiently manipulating Boolean formulas [20]. OBDDs are used in practical applications more often than other classical representations of Boolean formulas because they have a canonical form that uniquely characterizes a given function, and because operations on Boolean formulas can be performed quite efficiently in time and space [18]. The size of an OBDD does not directly depend on the size of the corresponding formula, and even though, in the worst case, it could be exponential in the number of variables in the formula, the majority of Boolean formulas can be represented by compact OBDDs. Our approach then transforms all the inputs of the fragmentation problem into Boolean formulas, and takes advantage of their representation through OBDDs to process different constraints simultaneously and to easily check whether a fragmentation satisfies all the given confidentiality and visibility constraints. In [12], we presented an early version of our

proposal that is here extended by introducing a graph modeling of the fragmentation problem that permits to reformulate it as the (NP-hard) problem of computing a maximum weighted clique. Based on this modeling, we then define an exact and a heuristic algorithm for computing a fragmentation composed of the minimum number of fragments. In addition, we formally analyze the correctness and computational complexity of both our exact and heuristic algorithms and present a set of experiments for assessing their efficiency (in terms of computational time) and the effectiveness of the heuristics (in terms of number of fragments of the computed fragmentation). The experimental results prove that our heuristics, while providing faster computational time, well approximates the minimal fragmentations computed by the exact algorithm.

The remainder of this paper is organized as follows. Section 2 introduces confidentiality and visibility constraints, and describes the fragmentation problem. Section 3 presents our modeling of the problem, defining OBDDs corresponding to constraints, and illustrating how the truth assignments that satisfy the constraints can be composed for computing a solution to the fragmentation problem. Section 4 uses the truth assignments extracted from OBDDs and their relationships to reformulate the fragmentation problem in terms of the maximum weighted clique problem over a fragmentation graph. Section 5 describes an exact algorithm for computing a minimal fragmentation, based on the graph modeling of the problem. Section 6 illustrates a heuristic approach that computes a locally minimal fragmentation by iteratively building a clique. Section 7 presents the experimental results comparing the exact and heuristic algorithms. Section 8 discusses related work. Finally, Section 9 reports our conclusions. The paper is complemented by the material in the Appendix that reports the proofs of the theorems in Sections 5 and 6.

2 Preliminary Concepts

We consider a scenario where, consistently with other proposals (e.g., [1, 11, 14, 25]), the data undergoing possible external release are represented with a single relation r over a relation schema $R(a_1, \ldots, a_n)$, and there are no dependencies among the attributes in R. We use standard notations of relational database theory and, when clear from the context, we use R to denote either the relation schema R or the set $\{a_1, \ldots, a_n\}$ of attributes in R. We consider two kinds of constraints on data: *confidentiality constraints*, imposing restrictions on the (joint) visibility of values of attributes in R, and *visibility constraints*, expressing requirements on data views [11, 14].

Definition 2.1 (Confidentiality constraint) *Given a relation schema $R(a_1, \ldots, a_n)$, a* confidentiality constraint *c over R is a subset of $\{a_1, \ldots, a_n\}$.*

Confidentiality constraints state that the values of an attribute (*singleton constraint*) or the associations among the values of a given set of attributes (*association constraint*) are sensitive and should not be visible. More precisely, a singleton constraint $\{a\}$ states that the values of attribute a should not be visible. An association constraint $\{a_{i_1}, \ldots, a_{i_m}\}$ states that the values of attributes $a_{i_1}, \ldots, a_{i_m}$ should not be visible in association. For instance, Figure 1(b) illustrates one singleton (c_1) and four association ($c_2, \ldots, c_5$) constraints for relation PATIENTS in Figure 1(a). The satisfaction of a confidentiality constraint c_i clearly implies the satisfaction of any confidentiality constraint c_j such that $c_i \subseteq c_j$, making c_j redundant. A set $\mathcal{C}$ of confidentiality constraints is *well defined* if $\forall c_i, c_j \in \mathcal{C}$, $i \neq j$, $c_i \not\subseteq c_j$, that is, $\mathcal{C}$ does not contain redundant constraints. Note that, while previous approaches assume that a pre-processing phase removes redundant constraints from $\mathcal{C}$, the solution proposed in this paper implicitly transforms $\mathcal{C}$ into a well defined set of confidentiality constraints (see Section 3).

Visibility constraints are defined as follows.

Definition 2.2 (Visibility constraint) *Given a relation schema $R(a_1, \ldots, a_n)$, a* visibility constraint *v over R is a monotonic Boolean formula over attributes in R.*

Intuitively, a visibility constraint imposes the release of an attribute or the joint release of a set of attributes. Visibility constraint $v = a$ states that the values of attribute a must be visible. Visibility constraint $v = v_i \wedge v_j$ states that v_i and v_j must be jointly visible (e.g., constraint v_2 in Figure 1(c) requires the joint release of attributes `Job` and `InsRate` since the associations between their values must be visible). Visibility constraint $v = v_i \vee v_j$ states that at least one between v_i and v_j must be visible (e.g., constraint v_1 in Figure 1(c) requires that the values of attribute `Name` or the association between the values of attributes `Birth` and `ZIP` be released). Note that negations are not used in the definition of visibility constraints since they model requirements of non-visibility, which are already captured by confidentiality constraints.

Confidentiality and visibility constraints can be enforced by partitioning (fragmenting) attributes in R in different sets (*fragments*). A fragmentation of relation R is a set of fragments, as formally captured by the following definition.

Definition 2.3 (Fragmentation) *Given a relation schema $R(a_1, \ldots, a_n)$, a* fragmentation *$\mathcal{F}$ of R is a set $\{F_1, \ldots, F_l\}$ of fragments, where each fragment F_i, $i = 1, \ldots, l$, is a subset of $\{a_1, \ldots, a_n\}$.*

Consistently with the proposal in [14], a fragmentation is not required to be complete, that is, it does not need to include all the attributes of the original relation. If the data holder is interested in releasing all the (non sensitive) attributes in R [1, 10], it is sufficient to include an additional visibility constraint $v = a$ for each attribute $a \in R$ such that there does not exist a constraint $c \in \mathcal{C}$ with $c = \{a\}$. Given a relation R, a set $\mathcal{C}$ of confidentiality constraints, and a set $\mathcal{V}$ of visibility constraints, a fragmentation $\mathcal{F}$ of R is *correct* if it

satisfies: *i)* all the confidentiality constraints in $\mathcal{C}$, and *ii)* all the visibility constraints in $\mathcal{V}$. Formally, a correct fragmentation is defined as follows.

Definition 2.4 (Correct fragmentation) *Given a relation schema $R(a_1, \ldots, a_n)$, a set $\mathcal{C}$ of confidentiality constraints over R, and a set $\mathcal{V}$ of visibility constraints over R, a fragmentation $\mathcal{F}$ of R is* correct *with respect to $\mathcal{C}$ and $\mathcal{V}$ iff:*

1. $\forall c \in \mathcal{C}$, $\forall F \in \mathcal{F}$: $c \not\subseteq F$ (*confidentiality*);
2. $\forall v \in \mathcal{V}$, $\exists F \in \mathcal{F}$: F *satisfies* v (*visibility*);
3. $\forall F_i, F_j \in \mathcal{F}$, $i \neq j$: $F_i \cap F_j = \emptyset$ (*unlinkability*).

Condition 1 ensures that neither sensitive attributes nor sensitive associations are visible in a fragment. Condition 2 ensures that all the visibility constraints are satisfied. Condition 3 ensures that fragments do not have common attributes and therefore that association constraints cannot be violated by joining fragments. We note that singleton constraints can be satisfied only by not releasing the involved sensitive attributes. Association constraints can be satisfied either by not releasing at least one of the attributes in each constraint, or by distributing the attributes among different (unlinkable) fragments. Visibility constraints are satisfied by ensuring that each constraint is satisfied by at least one fragment. Figure 2 illustrates an example of correct fragmentation of relation PATIENTS in Figure 1(a) with respect to the confidentiality and visibility constraints in Figure 1(b) and in Figure 1(c), respectively.

Given a set of confidentiality and visibility constraints, we are interested in a fragmentation that does not split attributes among fragments when it is not necessary for satisfying confidentiality constraints. The rationale is that maintaining a set of attributes in the same fragment releases, besides their values, also their associations. The utility of released data for final recipients is higher when releasing a fragmentation composed of fewer fragments, since they also have visibility of the associations among the attributes. Our goal is then to compute a *minimal* fragmentation, that is, a fragmentation with the minimum number of fragments. Formally, the problem of computing a minimal fragmentation is defined as follows.

Problem 2.5 (Minimal fragmentation) *Given a relation schema $R(a_1, \ldots, a_n)$, a set $\mathcal{C}$ of confidentiality constraints over R, and a set $\mathcal{V}$ of visibility constraints over R, determine (if it exists) a fragmentation $\mathcal{F}=\{F_1, \ldots, F_l\}$ of R such that:*

1. *$\mathcal{F}$ is a correct fragmentation of R with respect to $\mathcal{C}$ and $\mathcal{V}$ (Definition 2.4);*
2. *$\nexists \mathcal{F}'$ such that:* i) *$\mathcal{F}'$ is a correct fragmentation of R with respect to $\mathcal{C}$ and $\mathcal{V}$, and* ii) *$\mathcal{F}'$ is composed of fewer fragments than $\mathcal{F}$.*

The problem of computing a minimal fragmentation is NP-hard, since the minimum hypergraph coloring problem reduces to it in polynomial time [14]. We therefore adopt a definition of *locally minimal* fragmentation, which can be computed with an efficient heuristic. Such a definition is based on the following dominance relationship between the fragmentations of relation R.

Definition 2.6 (Dominance relationship) *Given a relation schema $R(a_1, \ldots, a_n)$ and two fragmentations $\mathcal{F}_i$ and $\mathcal{F}_j$ of R with $\bigcup_{F \in \mathcal{F}_i} F = \bigcup_{F \in \mathcal{F}_j} F$, $\mathcal{F}_i$* dominates *$\mathcal{F}_j$, denoted $\mathcal{F}_i \succ \mathcal{F}_j$, iff $\mathcal{F}_i \neq \mathcal{F}_j$, and $\forall F_j \in \mathcal{F}_j$, $\exists F_i \in \mathcal{F}_i$ such that $F_j \subseteq F_i$, and $\forall F_i \in \mathcal{F}_i$, $\exists \{F_{j_h}, \ldots, F_{j_l}\} \in \mathcal{F}_j$ such that $F_{j_h} \cup \ldots \cup F_{j_l} = F_i$.*

Definition 2.6 states that given two fragmentations $\mathcal{F}_i$ and $\mathcal{F}_j$ defined on the same set of attributes, $\mathcal{F}_i$ *dominates* $\mathcal{F}_j$ if $\mathcal{F}_i$ can be obtained by merging two (or more) fragments in $\mathcal{F}_j$. We note that fragmentations defined on different subsets of attributes in relation R cannot be compared with respect to the dominance relationship. As an example, consider relation PATIENTS in Figure 1(a), and fragmentation $\mathcal{F}_1 = \{\{$`Birth`,`ZIP`,`Disease`$\}$, $\{$`Job`,`InsRate`$\}\}$ in Figure 2. $\mathcal{F}_1$ dominates fragmentation $\mathcal{F}_2 = \{\{$`Birth`,`ZIP`$\}$, $\{$`Disease`$\}$, $\{$`Job`,`InsRate`$\}\}$ since $\mathcal{F}_1$ can be obtained by merging fragments $\{$`Birth`,`ZIP`$\}$ and $\{$`Disease`$\}$ in $\mathcal{F}_2$.

A locally minimal fragmentation is defined as a correct fragmentation whose fragments cannot be merged without violating any confidentiality constraint (i.e., a locally minimal fragmentation cannot be dominated by a correct fragmentation). Note that all the visibility constraints satisfied by a fragmentation $\mathcal{F}$ are also satisfied by any fragmentation $\mathcal{F}'$ dominating it. The problem of computing a locally minimal fragmentation is formally defined as follows.

Problem 2.7 (Locally minimal fragmentation) *Given a relation schema $R(a_1, \ldots, a_n)$, a set $\mathcal{C}$ of confidentiality constraints over R, and a set $\mathcal{V}$ of visibility constraints over R, determine (if it exists) a fragmentation $\mathcal{F} = \{F_1, \ldots, F_l\}$ of R such that:*

1. *$\mathcal{F}$ is a correct fragmentation of R with respect to $\mathcal{C}$ and $\mathcal{V}$ (Definition 2.4);*
2. *$\nexists \mathcal{F}'$ such that:* i) *$\mathcal{F}'$ is a correct fragmentation of R with respect to $\mathcal{C}$ and $\mathcal{V}$, and* ii) *$\mathcal{F}' \succ \mathcal{F}$.*

For instance, the fragmentation in Figure 2 is locally minimal since merging F_1 with F_2 would violate confidentiality constraint c_5.

It is important to note that a locally minimal fragmentation may not be a minimal fragmentation, while a minimal fragmentation is also a locally minimal fragmentation. For instance, consider relation PATIENTS in Figure 1(a) and the confidentiality and visibility constraints over it in Figure 1(b) and in Figure 1(c), respectively. Fragmentation $\mathcal{F} = \{\{$`Name`$\}$, $\{$`Race`,`Disease`$\}$,$\{$`Job`,`InsRate`$\}\}$ represents a locally minimal, but not a minimal, fragmentation for relation PATIENTS. Fragmentation $\mathcal{F}' = \{\{$`Birth`,`ZIP`,`Disease`$\}$, $\{$`Job`,`InsRate`$\}\}$ in Figure 2

is both locally minimal and minimal since there does not exist a correct fragmentation of relation PATIENTS composed of one fragment only.

3 OBDD-based Modeling of the Fragmentation Problem

We model the fragmentation problem as the problem of managing a set of Boolean formulas that are conveniently represented through *reduced and Ordered Binary Decision Diagrams* (OBDDs) [7]. OBDDs allow us to efficiently manipulate confidentiality and visibility constraints, and to easily compute a minimal (Section 5) or locally minimal (Section 6) fragmentation.

3.1 OBDD Representation of Constraints

In our modeling, attributes in R are interpreted as Boolean variables. Visibility constraints have already been defined as Boolean formulas (Definition 2.2). Each confidentiality constraint in $\mathcal{C}$ can be represented as the conjunction of the variables corresponding to the attributes in the constraint. For instance, Figure 3 represents the Boolean interpretation of the relation schema (i.e., the set $\mathcal{B}$ of Boolean variables), and of the constraints over it in Figure 1.

We use OBDDs as an effective and efficient approach for representing and manipulating Boolean formulas. An OBDD represents a Boolean formula as a rooted directed acyclic graph with two leaf nodes labeled 1 (true) and 0 (false), respectively, corresponding to the truth values of the formula. Each internal node in the graph represents a Boolean variable in the formula and has two outgoing edges, labeled 1 and 0, representing the assignment of values 1 and 0, respectively, to the variable. The variables occur in the same order on all the paths of the graph. Also, to guarantee a compact representation of the Boolean formula, the subgraphs rooted at the two direct descendants of each internal node in the graph are disjoint, and pairs of subgraphs rooted at two different nodes are not isomorphic. Figure 4 and Figure 5 illustrate the OBDDs of the Boolean formulas in Figure 3 that model the confidentiality and visibility constraints in Figure 1(b) and in Figure 1(c), respectively. For simplicity, in these figures and in the following, attributes are denoted with their initials, edges labeled 1 are represented by solid lines, and edges labeled 0 are represented by dashed lines. A truth assignment to the Boolean variables in a formula corresponds to a path from the root to one of the two leaf nodes of the OBDD of the formula. The outgoing edge of a node in the path is the value assigned to the variable represented by the node. For instance, in the OBDD of v_1 in Figure 5, the path traversing nodes `N`, `B`, `Z`, and 1 represents truth assignment [`N`=0, `B`=1, `Z`=1] since the edge in the path outgoing from node `N` is labeled 0, and the edges in the path outgoing from nodes `B` and `Z` are labeled 1. We call *one-paths* (*zero-paths*, respectively) all the paths of an OBDD that reach leaf node 1 (0, respectively), which correspond to the assignments that satisfy (do not

satisfy, respectively) the formula. For instance, path N, B, Z, and 1 is a one-path of the OBDD of v_1 in Figure 5. Variables in the formula that do not occur in a path from the root to a leaf node are called *don't care* variables, since their values do not influence the truth value of the formula. For instance, with respect to one-path N and 1 of the OBDD of v_1 in Figure 5, B and Z are don't care variables. In the remainder of the paper, we use '-' as value for the don't care variables. If there is at least a don't care variable in a truth assignment, this assignment is *partial* (in contrast to *complete*), since not all the variables in the formula have a value associated with them. We note that a partial truth assignment with k don't care variables is a compact representation of a set of 2^k complete truth assignments obtained by assigning to the don't care variables value 1 or 0. A complete truth assignment is *implicitly represented* by a partial truth assignment if, for each Boolean variable a in the formula, either a is a don't care variable for the partial truth assignment or the two truth assignments set a to the same value. For instance, the OBDD of v_1 in Figure 5 has two one-paths, corresponding to truth assignments [N=1, B=-, Z=-] and [N=0, B=1, Z=1]. Partial truth assignment [N=1, B=-, Z=-] is a compact representation for [N=1, B=0, Z=0], [N=1, B=0, Z=1], [N=1, B=1, Z=0], and [N=1, B=1, Z=1].

3.2 Truth Assignments

In the Boolean modeling of the fragmentation problem, a fragment $F \in \mathcal{F}$ can be interpreted as a *complete* truth assignment, denoted I_F, over the set $\mathcal{B}$ of Boolean variables. Function I_F assigns value 1 to each variable corresponding to an attribute in F, and value 0 to all the other variables in $\mathcal{B}$. A fragmentation is then represented by a set of complete truth assignments, which is formally defined as follows.

Definition 3.1 (Set of truth assignments) *Given a set $\mathcal{B}$ of Boolean variables, a* set $\mathcal{I}$ of truth assignments *is a set $\{I_1, \ldots, I_l\}$ of functions such that I_i:$\mathcal{B} \rightarrow \{0,1\}$, $i = 1, \ldots, l$.*

With a slight abuse of notation, we use I to denote also the list of truth values assigned by I to variables in $\mathcal{B}$. For instance, fragmentation $\mathcal{F}$ in Figure 2 corresponds to the set $\mathcal{I}=\{I_{F_1}, I_{F_2}\}$ of truth assignments, with I_{F_1} = [S=0, N=0, B=1, R=0, Z=1, J=0, I=0, D=1] and I_{F_2} = [S=0, N=0, B=0, R=0, Z=0, J=1, I=1, D=0] . Given a Boolean formula f, defined over Boolean variables $\mathcal{B}$, and a truth assignment I, $I(f)$ denotes the result of the evaluation of f with respect to truth assignment I. A set $\mathcal{I}$ of truth assignments corresponds to a correct fragmentation (Definition 2.4) if it satisfies all the confidentiality and visibility constraints and each Boolean variable is set to 1 by at most one truth assignment in $\mathcal{I}$, as formally defined in the following.

Definition 3.2 (Correct set of truth assignments) *Given a set $\mathcal{B}$ of Boolean variables, a set $\mathcal{C}$ of confidentiality constraints over $\mathcal{B}$, and a set $\mathcal{V}$ of visibility constraints over $\mathcal{B}$, a set $\mathcal{I}$ of truth assignments is* correct *with respect to $\mathcal{C}$ and $\mathcal{V}$ iff:*

1. $\forall c \in \mathcal{C}$, $\forall I \in \mathcal{I}$: $I(c) = 0$ (*confidentiality*);

2. $\forall v \in \mathcal{V}$, $\exists I \in \mathcal{I}$: $I(v) = 1$ (*visibility*);

3. $\forall I_i, I_j \in \mathcal{I}, i \neq j$, $\forall a \in \mathcal{B}$ *with* $I_i(a) = 1$: $I_j(a) = 0$ (*unlinkability*).

Condition 1 ensures that the evaluation of any confidentiality constraint with respect to any truth assignment is false (i.e., all fragments satisfy confidentiality constraints). Condition 2 ensures that, for each visibility constraint, there is at least one truth assignment that makes the visibility constraint true (i.e., every visibility constraint is satisfied by at least one fragment). Condition 3 ensures that there is at most one truth assignment that sets a variable to true (i.e., fragments do not have common attributes). It is immediate to see that a set of truth assignments is correct with respect to $\mathcal{C}$ and $\mathcal{V}$ iff the corresponding fragmentation is correct with respect to $\mathcal{C}$ and $\mathcal{V}$ (i.e., Definition 3.2 is equivalent to Definition 2.4). OBDDs representing confidentiality and visibility constraints can be used to efficiently verify if a set $\mathcal{I}$ of truth assignments satisfies Condition 1 and Condition 2 in Definition 3.2: *i)* each assignment $I \in \mathcal{I}$ must correspond to a zero-path in all the OBDDs of the confidentiality constraints; and *ii)* for each visibility constraint, at least one assignment $I \in \mathcal{I}$ must correspond to a one-path in the OBDD of the constraint. We also note that Condition 3 in Definition 3.2 can be efficiently verified by simply comparing the truth value assigned to each variable by the truth assignments in $\mathcal{I}$. For instance, consider the OBDDs of confidentiality and visibility constraints in Figures 4 and 5, respectively, and the set $\mathcal{I} = \{I_{F_1}, I_{F_2}\}$, with I_{F_1} = [S=0, N=0, B=1, R=0, Z=1, J=0, I=0, D=1] and I_{F_2} = [S=0, N=0, B=0, R=0, Z=0, J=1, I=1, D=0], representing the fragmentation in Figure 2. $\mathcal{I}$ is correct, since: *1)* I_{F_1} and I_{F_2} correspond to zero-paths of the OBDDs of the confidentiality constraints (confidentiality); *2)* I_{F_1} corresponds to a one-path of the OBDDs of v_1 and v_3, and I_{F_2} corresponds to a one-path of the OBDD of v_2 (visibility); and *3)* each variable in $\mathcal{B}$ is set to 1 by at most one assignment between I_{F_1} and I_{F_2} (unlinkability).

Problem 2.5 (minimal fragmentation) can be reformulated as the problem of computing a correct set of truth assignments composed of the minimum number of truth assignments, which is formally defined as follows.

Problem 3.3 (Minimal set of truth assignments) *Given a set $\mathcal{B}$ of Boolean variables, a set $\mathcal{C}$ of confidentiality constraints over $\mathcal{B}$, and a set $\mathcal{V}$ of visibility constraints over $\mathcal{B}$, determine (if it exists) a set $\mathcal{I}$ of truth assignments such that:*

1. *$\mathcal{I}$ is a correct set of truth assignments* (*Definition 3.2*);

2. *$\nexists \mathcal{I}'$ such that:* i) *$\mathcal{I}'$ is a correct set of truth assignments, and* ii) *$\mathcal{I}'$ is composed of fewer truth assignments than $\mathcal{I}$.*

Analogously, the problem of computing a locally minimal fragmentation (Problem 2.7) can be reformulated as the problem of computing a correct set $\mathcal{I}$ of truth assignments such that no pair of truth assignments I_i

and I_j in $\mathcal{I}$ can be combined producing a new assignment I_{ij} such that $\forall a \in \mathcal{B}, I_{ij}(a) = I_i(a) \vee I_j(a)$, and all the confidentiality constraints are satisfied. This condition can be formally formulated by first translating the dominance relationship between fragmentations into an equivalent dominance relationship between sets of truth assignments as follow.

Definition 3.4 (Dominance relationship) *Given a set $\mathcal{B}$ of Boolean variables and two sets of truth assignments $\mathcal{I}_i$ and $\mathcal{I}_j$ over $\mathcal{B}$, $\mathcal{I}_i$* dominates *$\mathcal{I}_j$, denoted $\mathcal{I}_i \succ \mathcal{I}_j$, iff $\mathcal{I}_i \neq \mathcal{I}_j$ and $\forall I_j \in \mathcal{I}_j$, $\exists I_i \in \mathcal{I}_i$ such that $\forall a \in \mathcal{B}$, with $I_j(a) = 1$, $I_i(a) = 1$ and $\forall I_i \in \mathcal{I}_i$, $\exists \{I_{j_h}, \ldots, I_{j_l}\} \in \mathcal{I}_j$ such that $\forall a \in \mathcal{B}, I_i(a) = I_{j_h}(a) \vee \ldots \vee I_{j_l}(a)$.*

The problem of computing a locally minimal fragmentation (Problem 2.7) can now be formally defined as the problem of computing a locally minimal set of truth assignments.

Problem 3.5 (Locally minimal set of truth assignments) *Given a set $\mathcal{B}$ of Boolean variables, a set $\mathcal{C}$ of confidentiality constraints over $\mathcal{B}$, and a set $\mathcal{V}$ of visibility constraints over $\mathcal{B}$, determine (if it exists) a set $\mathcal{I}$ of truth assignments such that:*

1. *$\mathcal{I}$ is a correct set of truth assignments (Definition 3.2);*
2. *$\nexists \mathcal{I}'$ such that:* i) *$\mathcal{I}'$ is a correct set of truth assignments, and* ii) *$\mathcal{I}' \succ \mathcal{I}$.*

Our approach to solve the minimal and locally minimal set of truth assignments problems uses properties of the OBDDs to efficiently check if a set of truth assignments is correct. In principle, a set of truth assignments should be checked for correctness against each confidentiality constraint and each visibility constraint. We can cut down on such controls by noting that if a truth assignment I does not make true any confidentiality constraint, Boolean formula $c_1 \vee \ldots \vee c_m$ evaluates to false with respect to I. Also, if truth assignment I makes true at least one of the confidentiality constraints in $\mathcal{C}$, Boolean formula $c_1 \vee \ldots \vee c_m$ evaluates to true with respect to I. In other words, we can check all the confidentiality constraints together in a single step. Formally, this observation is expressed as follows.

Observation 1 *Given a set $\mathcal{B} = \{a_1, \ldots, a_n\}$ of Boolean variables, a set $\mathcal{C} = \{c_1, \ldots, c_m\}$ of confidentiality constraints over $\mathcal{B}$, and a truth assignment I:*

$$\forall c_i \in \mathcal{C},\ I(c_i) = 0 \iff I(c_1 \vee \ldots \vee c_m) = 0.$$

To verify whether a truth assignment I satisfies the given confidentiality constraints, we can then simply check if I corresponds to a zero-path of the OBDD representing the disjunction of confidentiality constraints. For instance, consider the confidentiality constraints in Figure 3, the OBDD representing their disjunction in Figure 6, and truth assignment I_{F_1} = [S=0, N=0, B=1, R=0, Z=1, J=0, I=0, D=1], representing fragment F_1

in Figure 2. I_{F_1} corresponds to a zero-path of the OBDD in Figure 6, implying that I_{F_1} does not violate any confidentiality constraint.

For each visibility constraint v, a correct set of truth assignments must include at least a truth assignment I satisfying v, while not violating confidentiality constraints (i.e., $I(v)$=1 and $I(c_1 \vee \ldots \vee c_m)$=0). This is equivalent to say that Boolean formula $v \wedge \neg(c_1 \vee \ldots \vee c_m)$ with respect to truth assignment I evaluates to true, as formally observed in the following.

Observation 2 *Given a set $\mathcal{B} = \{a_1, \ldots, a_n\}$ of Boolean variables, a set $\mathcal{C} = \{c_1, \ldots, c_m\}$ of confidentiality constraints over $\mathcal{B}$, a visibility constraint v over $\mathcal{B}$, and a truth assignment I:*

$$I(v) = 1 \text{ and } I(c_1 \vee \ldots \vee c_m) = 0 \iff I(v \wedge \neg(c_1 \vee \ldots \vee c_m)) = 1.$$

In other words, the set of one-paths of the OBDD of Boolean formula $v_i \wedge \neg(c_1 \vee \ldots \vee c_m)$ represents in a compact way all and only the truth assignments that satisfy v_i and that do not violate any confidentiality constraint. In the following, we will use O_i to denote the OBDD of Boolean formula $v_i \wedge \neg(c_1 \vee \ldots \vee c_m)$, and $\mathcal{P}_{v_i}$ to denote the set of one-paths in O_i, which can represent both complete and partial truth assignments. For instance, consider the confidentiality and visibility constraints in Figures 4 and 5, respectively. Figure 7 illustrates the OBDDs of formulas $v_i \wedge \neg(c_1 \vee \ldots \vee c_5)$, $i = 1, \ldots, 3$, along with their one-paths. Note that all the variables in $\mathcal{B}$ not appearing in formula $v_i \wedge \neg(c_1 \vee \ldots \vee c_m)$ are considered as don't care variables for the one-paths in O_i, $i = 1, \ldots, k$.

To satisfy Condition 1 (confidentiality) and Condition 2 (visibility) in Definition 3.2, a set of truth assignments must include, for each $v \in \mathcal{V}$, at least a complete truth assignment implicitly represented by a (partial) truth assignment corresponding to a one-path in $\mathcal{P}_v$. However, not all the sets of truth assignments that include at least one complete truth assignment implicitly represented by a (partial) truth assignment in $\mathcal{P}_v$, for each $v \in \mathcal{V}$, are correct, since they may violate Condition 3 in Definition 3.2 (unlinkability). In the following, we discuss how to combine truth assignments in $\mathcal{P}_{v_1}, \ldots, \mathcal{P}_{v_k}$ to compute a correct set of truth assignments.

3.3 Comparison of Assignments

Goal of our approach is to compute a correct set of truth assignments that solves either the minimal or the locally minimal fragmentation problem. To this purpose, we first introduce the concepts of *linkable* and *mergeable* truth assignments.

Definition 3.6 (Linkable truth assignments) *Given two assignments I_i and I_j over Boolean variables $\mathcal{B}$, we say that I_i and I_j are* linkable*, denoted $I_i \leftrightarrow I_j$, iff $\exists a \in \mathcal{B} : I_i(a) = I_j(a) = 1$.*

According to Definition 3.6, two assignments are linkable iff there is a Boolean variable in $\mathcal{B}$ such that the truth value of the variable is set to 1 by the two given assignments, that is, the fragments corresponding to them have an attribute in common. For instance, assignments [S=0, N=0, B=1, R=0, Z=1, J=0, I=-, D=-] and [S=0, N=0, B=1, R=0, Z=-, J=0, I=-, D=1] are linkable since they both assign 1 to variable Birth. In the following, we will use the term *disjoint*, and notation $I_i \not\leftrightarrow I_j$, to refer to two truth assignments I_i and I_j that are not linkable. For instance, assignments [S=0, N=0, B=0, R=-, Z=-, J=1, I=1, D=0] and [S=0, N=1, B=0, R=-, Z=-, J=-, I=0, D=0] are disjoint. Note that variables with value 0 and - do not have any impact on the linkability of two truth assignments.

Definition 3.7 (Mergeable truth assignments) *Given two assignments I_i and I_j over Boolean variables $\mathcal{B}$, we say that I_i and I_j are* mergeable*, denoted $I_i \rightleftharpoons I_j$, iff $\nexists a$ s.t. $I_i(a) = 1$ and $I_j(a) = 0$, or viceversa.*

According to Definition 3.7, two truth assignments are mergeable iff the truth value of each variable a in $\mathcal{B}$ in the two assignments is not in contrast, where being in contrast for variable a means that a is assigned 1 by one assignment and is assigned 0 by the other one. For instance, the two assignments [S=0, N=0, B=1, R=0, Z=1, J=0, I=-, D=-] and [S=0, N=0, B=1, R=0, Z=-, J=0, I=-, D=1] are mergeable. While these two assignments are also linkable, linkability and mergeability are two independent properties and none of them implies the other. For instance, assignments [S=0, N=0, B=1, R=0, Z=1, J=0, I=-, D=-] and [S=0, N=0, B=1, R=0, Z=-, J=1, I=1, D=0] are linkable (Birth is set to 1 by both assignments) but not mergeable (there is a conflict on variable Job), while [S=0, N=0, B=1, R=0, Z=1, J=-, I=-, D=-] and [S=0, N=0, B=-, R=0, Z=-, J=1, I=1, D=0] are mergeable but not linkable.

It is interesting to note that the sets of complete truth assignments implicitly represented by mergeable partial truth assignments are overlapping (i.e., they have assignments in common), and that a complete truth assignment cannot be represented by two different partial truth assignments with variables in contrast. This is equivalent to say that two partial truth assignments are mergeable only if they represent at least a common complete truth assignment, as formally observed in the following.

Observation 3 *Given a set $\mathcal{B} = \{a_1, \ldots, a_n\}$ of Boolean variables and two truth assignments I_i and I_j over $\mathcal{B}$:*

$$I_i \rightleftharpoons I_j \iff \exists I_k \text{ s.t. } \forall a \in \mathcal{B},\ I_k(a){=}I_i(a) \text{ or } I_i(a){=}\text{-}, \text{ and } I_k(a){=}I_j(a) \text{ or } I_j(a){=}\text{-}.$$

For instance, consider mergeable truth assignments [S=0, N=0, B=1, R=0, Z=1, J=0, I=-, D=-] and [S=0, N=0, B=1, R=0, Z=-, J=0, I=-, D=1]. They both implicitly represent the following two complete truth assignments: [S=0, N=0, B=1, R=0, Z=1, J=0, I=0, D=1] and [S=0, N=0, B=1, R=0, Z=1, J=0, I=1, D=1].

Mergeable (partial) assignments can be composed (merged) according to operator $\odot$ in Figure 8. Merging truth assignments I_i and I_j results in a new truth assignment I_{ij}, where the truth value of a variable coincides with its truth value in the assignment in which it does not appear as a don't care variable. If a variable appears as a don't care variable in both I_i and I_j, then its value in the new assignment remains don't care. The result of the composition of I_i with I_j represents in a compact form all the complete truth assignments implicitly represented by both I_i and I_j. Note that if I_i and I_j are two (partial) truth assignments in the set $\mathcal{P}_{v_i}$ and $\mathcal{P}_{v_j}$, respectively, then I_{ij}=$I_i \odot I_j$ represents a set of complete truth assignments that satisfies all the confidentiality constraints and both v_i and v_j. For instance, with reference to the example in Figure 7, [S=0, N=0, B=1, R=0, Z=1, J=0, I=-, D=-] is a one-path in $\mathcal{P}_{v_1}$ and [S=0, N=0, B=1, R=0, Z=-, J=0, I=-, D=1] is a one-path in $\mathcal{P}_{v_3}$. These two assignments are mergeable and the result of their merging computed through operator $\odot$ is [S=0, N=0, B=1, R=0, Z=1, J=0, I=-, D=1], which implicitly represents two complete truth assignments (differing for the value of I) that satisfy both v_1 and v_3 and that do not violate any confidentiality constraint. Also, we note that no pair of one-paths in $\mathcal{P}_v$ is mergeable since they are two distinct one-paths of the same OBDD, and therefore differ by at least one edge, meaning that they are in conflict on at least one variable.

4 Graph Modeling of the Minimal Fragmentation Problem

To compute a correct set $\mathcal{I}$ of truth assignments (i.e., $\forall v_i \in \mathcal{V}$, $\mathcal{I}$ includes at least one complete truth assignment implicitly represented by a one-path in $\mathcal{P}_{v_i}$, and each pair of truth assignments in $\mathcal{I}$ is disjoint), we propose to model the one-paths of $\mathcal{P}_{v_i}$, for each $v_i \in \mathcal{V}$, and their relationships described in Section 3.3 through a *fragmentation graph*. We then translate the problem of computing a minimal set of truth assignments into the equivalent problem of computing a *maximum weighted clique* of the fragmentation graph.

A fragmentation graph is an undirected graph that implicitly represents all the truth assignments that may belong to a correct set of truth assignments as they satisfy all the confidentiality constraints and an arbitrary subset of visibility constraints. Edges in a fragmentation graph connect truth assignments that could appear together in a correct set of truth assignments. The fragmentation graph has therefore a node for each partial truth assignment in the set $\mathcal{P}^{\odot}$ obtained from the closure of $\mathcal{P}$ under operator $\odot$, where $\mathcal{P} = \mathcal{P}_{v_1} \cup \ldots \cup \mathcal{P}_{v_k}$ is the set of one-paths extracted from the OBDDs representing $v_i \wedge \neg(c_1 \vee \ldots \vee c_m)$, $i = 1, \ldots, k$ (see Section 3). Note that each truth assignment in $\mathcal{P}_{v_i}$ is explicitly associated with visibility constraint v_i. The rationale is that the truth assignments in $\mathcal{P}_{v_i}$ satisfy at least v_i, while not violating the confidentiality constraints. Set $\mathcal{P}^{\odot}$ includes both the truth assignments in $\mathcal{P}$ and the truth assignments resulting from the merging of any subset of mergeable one-paths in $\mathcal{P}$. The merging of two (partial) truth assignments I_i and I_j generates a (partial) truth assignment I_{ij} that is associated with a set of visibility constraints computed as the set-theoretic union

of those associated with I_i and I_j. We have therefore the guarantee that $\mathcal{P}^{\odot}$ contains all the (partial) truth assignments that represent fragments satisfying all the confidentiality constraints and a subset of the visibility constraints. Each node in the fragmentation graph is modeled as a pair $\langle I,V\rangle$, where I is a truth assignment in $\mathcal{P}^{\odot}$ and V is the set of the visibility constraints associated with I. Note that a complete truth assignment that satisfies a set $\{v_i,\ldots,v_j\}\subseteq\mathcal{V}$ of visibility constraints is represented by 2^n-1 nodes in the fragmentation graph, with $n=|\{v_i,\ldots,v_j\}|$, one for each subset of $\{v_i,\ldots,v_j\}$. Clearly, the set of nodes in the graph implicitly representing I may also represent other (different) truth assignments. The edges of the fragmentation graph connect nodes that represent disjoint truth assignments associated with non-overlapping sets of visibility constraints. Note that we add these edges because if there exist two nodes n_i and n_j representing two disjoint partial truth assignments with overlapping sets of visibility constraints, by construction, $\mathcal{P}^{\odot}$ must include also a node n_k that represents a partial truth assignment that is mergeable with the truth assignment represented by n_i (n_j, respectively) and is associated with a set of visibility constraints non-overlapping with the set of visibility constraints associated with n_j (n_i, respectively). The fragmentation graph therefore has an edge connecting node n_k with node n_j, thus making the edge between nodes n_i and n_j redundant. A fragmentation graph is formally defined as follows.

Definition 4.1 (Fragmentation graph) *Given a set $\mathcal{B}=\{a_1,\ldots,a_n\}$ of Boolean variables, a set $\mathcal{C}=\{c_1,\ldots,c_m\}$ of confidentiality constraints over $\mathcal{B}$, a set $\mathcal{V}=\{v_1,\ldots,v_k\}$ of visibility constraints over $\mathcal{B}$, and a set $\mathcal{P}=\mathcal{P}_{v_1}\cup\ldots\cup\mathcal{P}_{v_k}$ of one-paths in $O_1,\ldots,O_k$, a* fragmentation graph *is an undirected graph $G_F(N_F,E_F)$ where:*

- $N_F=\{\langle I,V\rangle$*: $I\in\mathcal{P}^{\odot}\wedge V\subseteq\mathcal{V}\wedge\forall v\in V$, $I(v)=1\}$, with $\mathcal{P}^{\odot}$ the closure of $\mathcal{P}$ under $\odot$;*
- $E_F=\{(n_i,n_j)$*: $n_i,n_j\in N_F\wedge n_i.I\not\leftrightarrow n_j.I\wedge n_i.V\cap n_j.V=\emptyset\}$, with $n_i.I$ the truth assignment represented by node n_i, and $n_i.V$ the set of visibility constraints associated with $n_i.I$.*

Note that nodes with sets of visibility constraints that have at least one visibility constraint in common are not connected by an edge in G_F since, in a correct set of truth assignments, it is sufficient that each visibility constraint is satisfied by one assignment. In fact, the release of multiple assignments satisfying the same visibility constraint may imply the release of unnecessary fragments, that is, of a fragmentation which is not minimal. Figure 9 illustrates the fragmentation graph resulting from the $\odot$-closure on $\mathcal{P}_{v_1}$, $\mathcal{P}_{v_2}$, $\mathcal{P}_{v_3}$ in Figure 7. In this figure and in the following, for readability purposes, we denote truth assignments by reporting attribute initials with a different notation, depending on the truth value assigned to the corresponding variable. More precisely, variables set to 1 are represented in uppercase and boldface (e.g., **A**), variables set to 0 are represented in lowercase (e.g., a), and variables set to - are represented in uppercase (e.g., A).

We note that a clique in G_F that includes, for each $v \in \mathcal{V}$, at least a node n such that $v \in n.V$ (i.e., $n.I$ is associated with v and satisfies it), represents a correct set $\mathcal{I}$ of truth assignments (Definition 3.2). In fact, by definition of fragmentation graph, the nodes in the clique represent a set of disjoint (and possibly partial) truth assignments (Condition 3) such that each of them satisfies all the confidentiality constraints (Condition 1). Also, each visibility constraint $v \in \mathcal{V}$ is satisfied by at least one of the truth assignments in the clique, the one represented by node n with $v \in n.V$ (Condition 2). Analogously, each correct set $\mathcal{I}$ of truth assignments is implicitly represented by a clique in G_F, and the same clique may represent more than one correct set of truth assignments. A correct set $\mathcal{I}$ of truth assignments is composed of complete truth assignments only, while the nodes in the fragmentation graph may represent partial truth assignments. Given a clique in the fragmentation graph, don't care variables in the truth assignments represented by the nodes in the clique must be set to either 0 or 1 to obtain one of the correct sets of truth assignments represented by the clique, with the restriction that no variable can assume value 1 in more than one fragment. Hence, we conveniently set all the don't care variables to 0. For instance, nodes $\langle$snBrZjlD,$\{v_1,v_3\}\rangle$ and $\langle$snbRZJId,$\{v_2\}\rangle$ form a clique for the fragmentation graph in Figure 9 such that the corresponding assignments satisfy all the confidentiality and visibility constraints in Figure 3. This clique corresponds to the set of truth assignments $\mathcal{I}$ = {[S=0, N=0, B=1, R=0, Z=1, J=0, I=0, D=1], [S=0, N=0, B=0, R=0, Z=0, J=1, I=1, D=0]}. We note that the clique also implicitly represents $\mathcal{I}$ = {[S=0, N=0, B=1, R=0, Z=1, J=0, I=0, D=1], [S=0, N=0, B=0, R=1, Z=0, J=1, I=1, D=0]}.

The problem of computing a correct set $\mathcal{I}$ of truth assignments can now be reformulated as the problem of computing a clique C of the fragmentation graph G_F such that $\bigcup_{n \in C} n.V = \mathcal{V}$. We are interested in computing a *minimal set of truth assignments* (Problem 3.3), which corresponds to a clique of the fragmentation graph that satisfies all the confidentiality and visibility constraints while *minimizing the number of nodes* composing it. The problem of computing a minimal set of truth assignments (Problem 3.3), or equivalently the problem of computing a minimal fragmentation (Problem 2.5), is then translated into the problem of computing a maximum weighted clique for the fragmentation graph, where a weight function w assigns a weight to the nodes of the graph so to model our minimization requirement. The maximum weighted clique problem has been widely studied in the literature and is formulated as follows [23, 24].

Problem 4.2 (Maximum weighted clique) *Given a weighted undirected graph $G(N, E, w)$, with $w : N \to \mathbb{R}^+$, determine a subset $C \subseteq N$ of nodes in N such that:*

1. $\forall n_i, n_j \in C$, $(n_i, n_j) \in E$ (C *is a* clique)*;*
2. $\nexists C' \subseteq N$ *such that:* i) C' *is a clique, and* ii) $\sum_{n \in C'} w(n) > \sum_{n \in C} w(n)$ (C *has* maximum weight)*.*

To reformulate the minimal set of truth assignments problem into the maximum weighted clique problem,

we define the weight function w in a way that satisfies the following three properties, which guarantee the equivalence between a maximum weighted clique in G_F (if it exists) and a minimal set of truth assignments.

1. *Monotonicity of w with respect to the number of visibility constraints*: given two cliques, the one associated with a higher number of visibility constraints has higher weight.

2. *Anti-monotonicity of w with respect to the number of nodes*: given two cliques associated with the same number of visibility constraints, the one composed of fewer nodes has higher weight.

3. *Equivalence of solutions*: cliques associated with the same number of visibility constraints and composed of the same number of nodes have the same weight.

A weight function that satisfies all the properties above is $w : N_F \to \mathbb{N}^+$ with $w(n)=(|\mathcal{V}| \cdot |n.V|) - 1$, where $|\mathcal{V}|$ is the number of visibility constraints, n is a node in N_F, and $|n.V|$ is the number of visibility constraints associated with n. The weight of a set $N'_F \subseteq N_F$ of nodes is the sum of the weights of the nodes composing it, that is, $w(N'_F)=\sum_{n \in N'_F} w(n)$. We first prove that our weight function satisfies the properties above, and then we show that such properties guarantee the equivalence between the minimum set of truth assignments problem and the maximum weighted clique problem.

Property 4.3 (Weight function) *Given a fragmentation graph $G_F(N_F, E_F)$, a weight function $w : N_F \to \mathbb{N}^+$ with $w(n)=(|\mathcal{V}|\cdot|n.V|)-1$, and two cliques of G_F, $C_i = \{n_{i_1}, \ldots, n_{i_x}\}$ and $C_j = \{n_{j_1}, \ldots, n_{j_y}\}$, the following conditions hold:*

1. $\sum_{k=1}^{x} |n_{i_k}.V| > \sum_{k=1}^{y} |n_{j_k}.V| \implies w(C_i)>w(C_j)$ (*monotonicity of w with respect to the number of visibility constraints*);

2. $\sum_{k=1}^{x} |n_{i_k}.V| = \sum_{k=1}^{y} |n_{j_k}.V|$ *and* $x < y \implies w(C_i)>w(C_j)$ (*anti-monotonicity of w with respect to the number of nodes*);

3. $\sum_{k=1}^{x} |n_{i_k}.V| = \sum_{k=1}^{y} |n_{j_k}.V|$ *and* $x = y \implies w(C_i)=w(C_j)$ (*equivalence of solutions*).

Proof:

1. Let us assume, by contradiction, that $w(C_i)\leq w(C_j)$, that is $\sum_{k=1}^{x} w(n_{i_k}) \leq \sum_{k=1}^{y} w(n_{j_k})$. Since $w(n) = (|\mathcal{V}|\cdot|n.V|) - 1$, the above equation can be rewritten as $\sum_{k=1}^{x}(|\mathcal{V}| \cdot |n_{i_k}.V| - 1) \leq \sum_{k=1}^{y}(|\mathcal{V}| \cdot |n_{j_k}.V| - 1)$, which is equivalent to $|\mathcal{V}| \cdot \sum_{k=1}^{x} |n_{i_k}.V| - x \leq |\mathcal{V}| \cdot \sum_{k=1}^{y} |n_{j_k}.V| - y$. This equation can be rewritten as $|\mathcal{V}| \cdot (\sum_{k=1}^{x} |n_{i_k}.V| - \sum_{k=1}^{y} |n_{j_k}.V|) - x + y \leq 0$. Since, by assumption, $\sum_{k=1}^{x} |n_{i_k}.V| > \sum_{k=1}^{y} |n_{j_k}.V|$, we have that $|\mathcal{V}| \cdot (\sum_{k=1}^{x} |n_{i_k}.V| - \sum_{k=1}^{y} |n_{j_k}.V|)$ is greater than $|\mathcal{V}|$. Also, $1 \leq x \leq |\mathcal{V}|$ and $1 \leq y \leq |\mathcal{V}|$. As a consequence, considering the worst case scenario, $\sum_{k=1}^{x} |n_{i_k}.V| - \sum_{k=1}^{y} |n_{j_k}.V| = 1$, $x = |\mathcal{V}|$, and

$y = 1$, the equation becomes $|\mathcal{V}| - |\mathcal{V}| + 1 \leq 0$, which is a contradiction proving the monotonicity of w with respect to the number of visibility constraints.

2. Let us now assume, by contradiction, that $w(C_i) \leq w(C_j)$, that is $|\mathcal{V}| \cdot \sum_{k=1}^{x} |n_{i_k}.V| - x \leq |\mathcal{V}| \cdot \sum_{k=1}^{y} |n_{j_k}.V| - y$. Since by assumption $\sum_{k=1}^{x} |n_{i_k}.V| = \sum_{k=1}^{y} |n_{j_k}.V|$, the above inequality holds only if $x > y$, which contradicts our hypothesis and proves the anti-monotonicity of w with respect to the number of nodes.

3. Let us now assume, by contradiction, that $w(C_i) \neq w(C_j)$, that is $|\mathcal{V}| \cdot \sum_{k=1}^{x} |n_{i_k}.V| - x \neq |\mathcal{V}| \cdot \sum_{k=1}^{y} |n_{j_k}.V| - y$. Since by assumption $\sum_{k=1}^{x} |n_{i_k}.V| = \sum_{k=1}^{y} |n_{j_k}.V|$, the above inequality holds only if $x \neq y$, which contradicts our hypothesis and proves the equivalence of solutions. □

To illustrate Property 4.3, consider the fragmentation graph in Figure 9, and cliques C_1={⟨snB**r**Z**j**I**D**,{v_1,v_3}⟩, ⟨snbRZ**JI**d,{v_2}⟩}, C_2={⟨s**N**bRZJid,{v_1}⟩, ⟨snbRZ**JI**d,{v_2}⟩}, C_3={⟨s**N**bRZJid,{v_1}⟩, ⟨snbRZ**JI**d,{v_2}⟩, ⟨snb**R**ZjI**D**,{v_3}⟩}, and C_4={⟨snB**r**Z**JI**d,{v_1,v_2}⟩, ⟨snb**R**ZjI**D**,{v_3}⟩}, with weight $w(C_1)$=7, $w(C_2)$=4, $w(C_3)$=6, and $w(C_4)$=7, respectively. According to the monotonicity of the weight function with respect to the number of visibility constraints, $w(C_1)$=7 > $w(C_2)$=4 since the nodes in C_1 are associated with three visibility constraints, while the nodes in C_2 are associated with two constraints only. According to the anti-monotonicity of the weight function with respect to the number of nodes, $w(C_1)$=7 > $w(C_3)$=6 although C_1 (composed of two nodes) and C_3 (composed of three nodes) are associated with all the visibility constraints in Figure 3. According to the equivalence of solutions, $w(C_1)$=$w(C_4)$=7 since the nodes in C_1 and in C_4 are associated with all the visibility constraints, and C_1 and C_4 are composed of two nodes.

Given a fragmentation graph G_F, a clique C of G_F represents a correct set of truth assignments iff C is associated with all the visibility constraints in $\mathcal{V}$. It is interesting to note that, according to the definition of weight function w of the fragmentation graph G_F as $w(n)$=($|\mathcal{V}| \cdot |n.V|$) − 1, C is associated with (and then satisfy) all the visibility constraints only if the weight of C is higher than or equal to $|\mathcal{V}| \cdot (|\mathcal{V}| - 1)$. Formally, this property can be formulated as follow.

Property 4.4 *Given a fragmentation graph $G_F(N_F, E_F)$, a weight function $w : N_F \rightarrow \mathbb{N}^+$, with $w(n) = (|\mathcal{V}| \cdot |n.V|) - 1$, and a clique C of G_F:*

$$\forall v \in \mathcal{V},\ \exists n \in C \text{ s.t. } v \in n.V \iff w(C) \geq |\mathcal{V}| \cdot (|\mathcal{V}| - 1).$$

PROOF: The weight of a clique C={n_1,...,n_i} is computed as: $\sum_{j=1}^{i}(|\mathcal{V}| \cdot |n_j.V| - 1) = |\mathcal{V}| \cdot \sum_{j=1}^{i} |n_j.V| - i$. Since, by hypothesis, C includes a node associated with v for each visibility constraint $v \in \mathcal{V}$ then $\sum_{j=1}^{i} |n_j.V| = |\mathcal{V}|$ and therefore $w(C) = |\mathcal{V}| \cdot |\mathcal{V}| - i$. In the worst case, each node in the clique is associated with one visibility

constraint and the clique is then composed of $|\mathcal{V}|$ nodes. The weight of the clique is then $w(C)=|\mathcal{V}|\cdot|\mathcal{V}|-|\mathcal{V}|=|\mathcal{V}|\cdot(|\mathcal{V}|-1)$. Therefore, by Property 4.3, all cliques of G_F associated with less than $|\mathcal{V}|$ visibility constraints have weight lower than $|\mathcal{V}|\cdot(|\mathcal{V}|-1)$. □

Property 4.4 guarantees that it is sufficient to check if the weight of the maximum weighted clique of G_F is higher than or equal to $|\mathcal{V}|\cdot(|\mathcal{V}|-1)$ to determine whether a correct set of truth assignments exists for the considered instance of the problem. To illustrate, consider the fragmentation graph in Figure 9. Clique $C_1 = \{\langle$sNbRZJid,$\{v_1\}\rangle$, $\langle$snbRZ**JI**d,$\{v_2\}\rangle\}$ is associated with two out of the three visibility constraints in $\mathcal{V}$, and has weight 2+2=4, which is lower than 3·(3−1)=6. Clique $C_2 = \{\langle$sNbRZJid,$\{v_1\}\rangle$, $\langle$snbRZ**JI**d,$\{v_2\}\rangle$, $\langle$snb**R**ZjI**D**,$\{v_3\}\rangle\}$ is associated with all the visibility constraints, and has weight 2+2+2=6.

We now formally prove that Properties 4.3 and 4.4 discussed above guarantee that the problem of computing a minimal set of truth assignments (Problem 3.3) is equivalent to the problem of computing a maximum weighted clique of a fragmentation graph with weight at least $|\mathcal{V}|\cdot(|\mathcal{V}|-1)$.

Theorem 4.5 (Problem equivalence) *The minimal set of truth assignments problem (Problem 3.3) is equivalent to the problem of determining a maximum weighted clique of weight at least* $|\mathcal{V}|\cdot(|\mathcal{V}|-1)$ *of the fragmentation graph* $G_F(N_F,E_F)$ *(Definition 4.1), with weight function* $w : N_F \to \mathbb{N}^+$ *s.t.* $w(n)=(|\mathcal{V}|\cdot|n.V|)-1$.

Proof: The proof of this theorem immediately follows from Properties 4.3 and 4.4. Indeed, the maximum weighted clique $C = \{n_1,\ldots,n_i\}$ of the fragmentation graph G_F satisfies the maximum number of visibility constraints, according to the monotonicity of w with respect to the number of visibility constraints associated with the nodes in C. If there are different cliques in G_F associated with the same number of visibility constraints, C is the one composed of the minimum number of nodes, according to the anti-monotonicity of w with respect to the number of nodes. Let us now suppose that the set of nodes composing the clique C having maximum weight is associated with all the visibility constraints. Property 4.4 guarantees that $w(C)$ is, in the worst case, equal to $|\mathcal{V}|\cdot(|\mathcal{V}|-1)$. □

Since the minimal set of truth assignments problem and the minimal fragmentation problem are equivalent, the minimal fragmentation problem is also equivalent to the maximum weighted clique problem on the fragmentation graph with the weight function defined above.

In the following section, we will present an algorithm for computing a minimal set of truth assignments, exploiting the equivalence proved by Theorem 4.5. In Section 6, we will introduce a heuristic algorithm for computing a locally minimal set of truth assignments.

5 Computing a Minimal Set of Truth Assignments

The algorithm we propose for computing a minimal set of truth assignments (see Figure 10) takes as input a set $\mathcal{B} = \{a_1, \ldots, a_n\}$ of Boolean variables (representing the attributes in R), a set $\mathcal{C} = \{c_1, \ldots, c_m\}$ of confidentiality constraints, a set $\mathcal{V} = \{v_1, \ldots, v_k\}$ of visibility constraints, and executes the following three steps: *1)* it computes the set of one-paths of the OBDDs representing Boolean formulas $v_i \wedge \neg(c_1 \vee \ldots \vee c_m)$, $i = 1, \ldots, k$; *2)* it builds the fragmentation graph; *3)* it determines a maximum weighted clique of the fragmentation graph, and checks if the clique represents a correct set of truth assignments. In the following, we describe these steps more in details.

Step 1: Compute one-paths. For each visibility constraint $v_i \in \mathcal{V}$, the algorithm defines the OBDD O_i representing Boolean formula $v_i \wedge \neg(c_1 \vee \ldots \vee c_m)$. Then, it extracts from O_i the set $\mathcal{P}_{v_i}$ of one-paths (lines 1–4), $i = 1, \ldots, k$. If, for a given O_i, the set $\mathcal{P}_{v_i}$ is empty, v_i cannot be satisfied without violating the confidentiality constraints and therefore the algorithm terminates, returning an empty set of truth assignments (line 5).

Step 2: Build the fragmentation graph. The algorithm first builds an undirected weighted graph $G(N, M \cup D, w)$ such that for each truth assignment I in $\mathcal{P}_{v_i}$, $i = 1, \ldots, k$, there is a node $n \in N$, with $n.I = I$, $n.V = \{v_i\}$, and $n.weight = (|\mathcal{V}| \cdot |n.V|) - 1$ (lines 10–14). Then, for each pair of nodes n_i and n_j in N, the algorithm inserts edge (n_i, n_j) in M if n_i and n_j represent a pair of mergeable truth assignments that are associated with non-overlapping sets of visibility constraints. This edge indicates that the one-paths represented by n_i and n_j can be merged, thus obtaining a truth assignment associated with both the visibility constraints in $n_i.V$ and in $n_j.V$ (lines 20–22). Edge (n_i, n_j) is inserted in D if n_i and n_j represent two disjoint truth assignments associated with non-overlapping sets of visibility constraints. In this case, edge (n_i, n_j) indicates that the one-paths represented by n_i and n_j can belong to the same correct set of truth assignments, and that these one-paths guarantee the satisfaction of different subsets of visibility constraints (lines 23–25). Note however that, as already discussed in Section 4.1, the truth assignments associated with n_i and n_j may also satisfy additional (possibly overlapping) visibility constraints that are not explicitly associated with them. If n_i and the nodes to which it is connected do not satisfy all the visibility constraints in $\mathcal{V}$, n_i cannot belong to any maximum weighted clique representing a correct set of truth assignments. In fact, a clique is a solution of the fragmentation problem only when, for each visibility constraint v in $\mathcal{V}$, there is a node in the clique such that v is associated with such a node. For this reason, the algorithm removes n_i from G (lines 26–29).

The algorithm then transforms the graph G representing the one-paths in $\mathcal{P} = \mathcal{P}_{v_1} \cup \ldots \cup \mathcal{P}_{v_k}$ into a fragmentation graph by computing the closure of $\mathcal{P}$ (i.e., the nodes in N) under merging operator $\odot$. To this end, the algorithm creates a copy M' of the set M of edges and initializes M to the empty set (lines 31–32).

Then, the algorithm iteratively extracts an edge (n_i,n_j) from M', and determines a new node n_{ij} such that $n_{ij}.I$=$n_i.I \odot n_j.I$ and $n_{ij}.V$=$n_i.V \cup n_j.V$(lines 34–36). The weight $n_{ij}.weight$ is set to $|\mathcal{V}|\cdot|n_{ij}.V| - 1$ (see Section 4), thus reflecting the number of visibility constraints associated with the node (line 37). Before inserting n_{ij} in G, the algorithm checks if n_{ij} satisfies all the visibility constraints (line 38). If this is the case, n_{ij} represents a maximum weighted clique for G. $\mathcal{I}_{sol}$ is then set to $n_{ij}.I$, don't care variables are set to 0, and the algorithm terminates returning $\mathcal{I}_{sol}$ (lines 39–41). Otherwise, node n_{ij} is inserted in G, and the algorithm checks if nodes adjacent either to n_i or to n_j are also adjacent (with a mergeable or disjoint edge) to n_{ij}, thus possibly inserting in M or in D the corresponding edges (lines 42–46). Note that the algorithm needs only to check n_{ij} against the nodes in N that are mergeable/disjoint with $n_i.I$ or $n_j.I$, since satisfying any of these conditions is a precondition for being mergeable/disjoint with $n_{ij}.I$. When the set M' of edges is empty, the algorithm checks whether there are nodes in N that can be removed from G since they cannot belong to any maximum weighted clique (lines 47–51). The algorithm then iteratively repeats the process of removing edges from M (i.e., it creates a copy M' of M and inserts new nodes and edges in N, M, and D, respectively) until the set M of edges is empty, that is, no edge is inserted in M during the process of merging nodes connected through the edges in M' (i.e., until G is a fragmentation graph).

Step 3: Compute a maximum weighted clique. The algorithm exploits a known algorithm [23] to compute a maximum weighted clique of the fragmentation graph (line 53). Function **FindMaxWeightedClique** takes the fragmentation graph as input and returns a maximum weighted clique. If the weight of the clique is lower than $|\mathcal{V}|\cdot(|\mathcal{V}|-1)$, the considered instance of the problem does not admit a correct set of truth assignments (line 54). Otherwise, if $w(C)$ is at least $|\mathcal{V}| \cdot (|\mathcal{V}| - 1)$, the one-paths represented by the nodes in C are inserted in $\mathcal{I}_{sol}$, don't care variables are set to 0 (lines 56–59), and $\mathcal{I}_{sol}$ is returned (line 60).

Example 5.1 *Consider relation* PATIENTS *and the confidentiality and visibility constraints over it in Figure 1. The execution of the algorithm in Figure 10 proceeds as follows.*

1) Compute one-paths. *The algorithm builds O_1, O_2, and O_3 in Figure 7, representing formula $v_i \wedge \neg(c_1 \vee \ldots \vee c_5)$, $i = 1, 2, 3$, and extracts their one-paths, which are listed in Figure 7.*

2) Build the fragmentation graph. *The algorithm inserts in G a node for every one-path in $\mathcal{P}_{v_1}$, $\mathcal{P}_{v_2}$, and $\mathcal{P}_{v_3}$ (see Figure 11(a)). Figure 11(b) shows the graph obtained connecting the nodes in Figure 11(a) that represent mergeable truth assignments (dotted edges) and disjoint truth assignments (continuous edges). Nodes* $\langle$sN**BR**zJid,$\{v_1\}\rangle$ *and* $\langle$sn**BR**z**JI**d,$\{v_2\}\rangle$ *in Figure 11(a) do not appear in the graph in Figure 11(b) since they neither are associated with v_3 nor could be connected with a node that is associated with v_3, and therefore cannot be part of a clique. The algorithm then computes the closure of*

the nodes in G. It first merges nodes $\langle$sn**B**r**ZJI**d,$\{v_1\}\rangle$ *and* $\langle$sn**B**r**Z**j**I**d,$\{v_2\}\rangle$*, inserts the resulting node* $\langle$sn**B**r**ZJI**d,$\{v_1,v_2\}\rangle$ *in N, and checks if it can be connected by an edge in D with node* $\langle$snb**R**Zj**I**d,$\{v_3\}\rangle$ *and/or with node* $\langle$s**N**bRZJid,$\{v_1\}\rangle$ *(the nodes adjacent to the merged nodes). Since* $\langle$sn**B**r**ZJI**d,$\{v_1,v_2\}\rangle$ *and* $\langle$snb**R**Zj**I**d,$\{v_3\}\rangle$ *represent disjoint assignments and are associated with non-overlapping sets of visibility constraints, the algorithm inserts edge* ($\langle$sn**B**r**ZJI**d,$\{v_1,v_2\}\rangle$,$\langle$snb**R**Zj**I**d,$\{v_3\}\rangle$) *in D, while it does not insert the edge connecting* $\langle$sn**B**r**ZJI**d,$\{v_1,v_2\}\rangle$ *with* $\langle$s**N**bRZJid,$\{v_1\}\rangle$ *since v_1 is associated with both nodes. The resulting graph is illustrated in Figure 11(c), where the new node is doubly circled. The algorithm then merges nodes* $\langle$sn**B**r**Z**jID,$\{v_1\}\rangle$ *and* $\langle$sn**B**r**Z**j**I**d,$\{v_3\}\rangle$*, inserts the resulting node* $\langle$sn**B**r**Z**j**I**d,$\{v_1,v_3\}\rangle$ *in N, and inserts edge* ($\langle$sn**B**r**Z**j**I**d,$\{v_1,v_3\}\rangle$,$\langle$snbRZ**JI**d,$\{v_2\}\rangle$) *in D. Figure 11(d) illustrates the resulting graph, where the new node is doubly circled. Since there are no more mergeable edges in M, the algorithm checks whether there are nodes in N that can possibly be removed from G. Node* $\langle$sn**B**r**ZJI**d,$\{v_1\}\rangle$ *is only connected with a node associated with v_3 and has no connections with nodes associated with v_2, and therefore it is removed from G. Figure 11(d) illustrates the resulting fragmentation graph.*

3) Compute a maximum weighted clique. *The algorithm calls function* **FindMaxWeightClique** *that returns one of the two maximum weighted cliques in G, C=*{$\langle$sn**B**r**Z**j**I**d,$\{v_1,v_3\}\rangle$,$\langle$snbRZ**JI**d,$\{v_2\}\rangle$}*. The weight of this clique is $w(C)=7$ and is higher than threshold $|\mathcal{V}|\cdot(|\mathcal{V}|-1)=6$. Therefore, C represents a solution to the minimal set of truth assignments problem. The algorithm extracts from C the corresponding set of truth assignments, and the don't care variables are set to 0, thus obtaining $\mathcal{I}_{sol}$ =* {[`S`*=0,* `N`*=0,* `B`*=1,* `R`*=0,* `Z`*=1,* `J`*=0,* `I`*=0,* `D`*=1*], [`S`*=0,* `N`*=0,* `B`*=0,* `R`*=0,* `Z`*=0,* `J`*=1,* `I`*=1,* `D`*=0*]} *that is finally returned. We note that this set of truth assignments corresponds to the minimal fragmentation $\mathcal{F}$=*{{`Birth`,`ZIP`,`Disease`}, {`Job`,`InsRate`}} *in Figure 2.*

The correctness and complexity of the algorithm in Figure 10 are stated by the following theorems.

Theorem 5.2 (Correctness of the exact algorithm) *Given a set $\mathcal{B}$ of Boolean variables, a set $\mathcal{C}$ of confidentiality constraints over $\mathcal{B}$, and a set $\mathcal{V}$ of visibility constraints over $\mathcal{B}$, the algorithm in Figure 10 terminates and computes (if it exists) a minimal set of truth assignments.*

PROOF: See Appendix A. □

Theorem 5.3 (Complexity of the exact algorithm) *Given a set $\mathcal{B}$ of Boolean variables, a set $\mathcal{C}$ of confidentiality constraints over $\mathcal{B}$, and a set $\mathcal{V}$ of visibility constraints over $\mathcal{B}$, the complexity of the algorithm in Figure 10 is $O(2^{\prod_{v\in\mathcal{V}}|\mathcal{P}_v|\cdot|\mathcal{B}|}+(|\mathcal{V}|+|\mathcal{C}|)2^{|\mathcal{B}|})$ in time, where $\mathcal{P}_v$ is the set of one-paths of the OBDD representing $v\wedge\neg(c_1\vee\ldots\vee c_m)$.*

PROOF: See Appendix A. □

The computational cost of the algorithm is obtained as the sum of the cost of building the OBDDs, which is $O((|\mathcal{V}|+|\mathcal{C}|)2^{|\mathcal{B}|})$, and of the cost of determining $\mathcal{I}_{sol}$ by building the fragmentation graph and searching for its maximum weighted clique, which is $O(2^{\prod_{v\in\mathcal{V}}|\mathcal{P}_v|\cdot|\mathcal{B}|})$. We note that the computational cost of the construction of the OBDDs is exponential in the worst case, but in the majority of real-world applications OBDD-based approaches are computationally efficient [7, 18].

6 Computing a Locally Minimal Set of Truth Assignments

Since the problem of computing a minimal set of truth assignments is NP-hard, the computational complexity of any algorithm that finds a solution to the problem is exponential in the size of the input. In this section, we therefore propose a heuristic algorithm that computes a locally minimal set of truth assignments (Problem 3.5) with a limited computational effort. The algorithm exploits Theorem 4.5 to take advantage of the graph modeling of the problem but does not explicitly create the fragmentation graph. The idea consists in using the relationships between the one-paths extracted from the OBDDs representing confidentiality and visibility constraints to iteratively build a clique. The algorithm does not compute the closure of the one-paths in $\mathcal{P}_{v_1}\cup\ldots\cup\mathcal{P}_{v_k}$ under operator $\odot$, but composes them when necessary. It then starts from an empty clique C and, at each iteration, tries to insert in C a node n (possibly composing it with nodes in C) that is associated with a visibility constraint that is not associated with any node already in C. The algorithm terminates when it finds a clique C of weight at least $|\mathcal{V}|\cdot(|\mathcal{V}|-1)$.

Figure 12 illustrates the pseudocode of the algorithm that takes as input a set $\mathcal{B}=\{a_1,\ldots,a_n\}$ of Boolean variables (representing the attributes in R), a set $\mathcal{C}=\{c_1,\ldots,c_m\}$ of confidentiality constraints, and a set $\mathcal{V}=\{v_1,\ldots,v_k\}$ of visibility constraints and computes, if it exists, a locally minimal set of truth assignments. The algorithm executes four steps: *1)* it extracts the set of one-paths from the OBDDs representing Boolean formulas $v_i\wedge\neg(c_1\vee\ldots\vee c_m)$, $i=1,\ldots,k$; *2)* it creates a node for each of these one-paths; *3)* it iteratively builds a clique of the fragmentation graph; *4)* it combines the one-paths represented by the nodes in the clique, if this combination does not violate confidentiality constraints, to minimize the number of assignments in the computed set. In the following, we describe these steps more in details.

Step 1: Compute one-paths. Like for the exact algorithm (Section 5, step 1), for each $v_i\in\mathcal{V}$ the algorithm extracts from the OBDD O_i representing Boolean formula $v_i\wedge\neg(c_1\vee\ldots\vee c_m)$ the set $\mathcal{P}_{v_i}$ of one-paths (lines 2–4). If, for a given O_i, the set $\mathcal{P}_{v_i}$ is empty, the algorithm terminates and returns an empty solution (line 5).

Step 2: Generate nodes representing one-paths. The algorithm inserts a node $n{=}\langle I,\{v\}\rangle$ in graph G for each one-path $I{\in}\mathcal{P}_{v_1}{\cup}\ldots{\cup}\mathcal{P}_{v_k}$. Unlike the exact algorithm, it does not explicitly insert the edges in G connecting pairs of nodes that represent mergeable and disjoint truth assignments. In contrast, it implicitly considers these relationships in the building process of a clique. The algorithm then partitions nodes in G according to the visibility constraint associated with them, and orders the obtained sets of nodes N_i, $i = 1,\ldots,|\mathcal{V}|$, by increasing cardinality (lines 10–13). The reason for this ordering is to consider first the visibility constraints that can be satisfied by a smaller set of truth assignments (represented by a smaller set of nodes in the graph). Nodes in N_i, $i = 1,\ldots,|\mathcal{V}|$, are ordered by decreasing number of don't care variables in the truth assignments they represent (lines 14–15). The intuition is that nodes representing truth assignments with a higher number of don't care variables implicitly represent a larger set of complete truth assignments (where don't care variables can be set to either 0 or 1) and therefore they impose less constraints on subsequent choices of the nodes that can be inserted in a clique with them. Indeed, as already noted in Section 3.3, don't care variables do not affect the linkability or the mergeability of truth assignments.

Step 3: Build a clique for the fragmentation graph. The algorithm iteratively builds a clique by calling recursive function **DefineClique** (line 17). Function **DefineClique** receives as input a clique C of the fragmentation graph and an integer number i, $1 \leq i \leq k$, indicating that C either includes a node in N_j, $j = 1,\ldots,(i-1)$, or a node resulting from the combination of a node in N_j with another node in N_l, with $l < j$ (i.e., C includes a node n such that $v{\in}n.V$, for each visibility constraint v associated with the nodes in $N_1,\ldots,N_j$). For each node n_j in N_i, function **DefineClique** verifies whether n_j can be inserted in C, that is, if: *i)* for each node n in C, $n.I$ and $n_j.I$ are disjoint; or *ii)* $n_j.I$ is mergeable with a subset of the truth assignments represented by the nodes in C and the resulting truth assignment is disjoint from all the other truth assignments represented by nodes in C. To efficiently check if n_j satisfies one of the conditions above, the function first identifies the set *LinkableNodes* of nodes in C representing truth assignments linkable with $n_j.I$ (line 33). For each n_l in *LinkableNodes*, if $n_l.I$ is mergeable with $n.I$ (with n initialized to n_j), $n.I$ is set to $n.I{\odot}n_l.I$, and $n.V$ is set to $n.V{\cup}n_l.V$ (lines 37–39). If $n_l.I$ is linkable but not mergeable with $n.I$, n cannot be part of clique C since $n_l.I$ and $n.I$ are not disjoint (line 39). We note that nodes in C representing truth assignments that are mergeable and disjoint to $n_j.I$ are not combined in a unique node. In fact, by composing a pair of disjoint truth assignments, the algorithm would discard, without evaluation, all the correct solutions where the two truth assignments are represented by distinct nodes. If all the nodes in *LinkableNodes* can be combined with n_j (i.e., the one-paths they represent are all mergeable), the algorithm then determines a new clique C' obtained by removing *LinkableNodes* from C and inserting n in C' (lines 40–41). If $i{=}|\mathcal{V}|$, C' satisfies all the visibility constraints, has a weight at least equal to $|\mathcal{V}| \cdot (|\mathcal{V}| - 1)$, and is returned (line 42). Otherwise,

function **DefineClique** is recursively called with C' and $i+1$ as input (line 43). If the clique resulting from this recursive call is not empty, it represents a correct set of truth assignments and is therefore returned (line 44). If no node in N_i can be inserted in C, an empty clique is returned and the algorithm looks for a different clique of the fragmentation graph.

Step 4: Minimize the number of assignments. The clique C computed by function **DefineClique** may represent a correct set of truth assignments that is not locally minimal. In fact, it may include one-paths that can be combined without violating confidentiality constraints. Every pair of nodes in C is then checked and their truth assignments are ORed whenever they can be combined without violating confidentiality constraints (i.e., the algorithm performs the union of the corresponding fragments). To this purpose, the algorithm first assigns value 0 to don't care variables in the truth assignments represented by the nodes in C (line 20). Then, it iteratively extracts a node n_i from C, assigns $n_i.I$ to I_i and, for each n_j in C, it checks if I_i can be composed with one-path I_j, with $I_j = n_j.I$ without violating confidentiality constraints (lines 21–28). If this is the case, I_i is set to $I_i \vee I_j$, and n_j is removed from C. When the algorithm has checked if I_i can be combined with all the one-paths represented by nodes in C, it inserts I_i in $\mathcal{I}_{sol}$ (line 29). It is important to note that the algorithm does not check if I_i can be combined with the truth assignments already in $\mathcal{I}_{sol}$. In fact, all the truth assignments in $\mathcal{I}_{sol}$ have already been checked against all the assignments in C, and therefore also against I_i. Finally, the algorithm returns $\mathcal{I}_{sol}$ (line 30).

Example 6.1 *Consider relation* PATIENTS *and the confidentiality and visibility constraints over it in Figure 1. The execution of the algorithm in Figure 12 proceeds as follows.*

1) Compute one-paths. *The algorithm first builds O_1, O_2, and O_3 in Figure 7, representing $v_i \wedge \neg(c_1 \vee \ldots \vee c_5)$, $i = 1, 2, 3$, and extracts their one-paths, which are listed in Figure 7.*

2) Generate nodes representing one-paths. *The algorithm creates a node for each one-path in $\mathcal{P}_{v_1}$, $\mathcal{P}_{v_2}$, and $\mathcal{P}_{v_3}$, obtaining the set N of nodes illustrated in Figure 13(a). The algorithm partitions N in three sets N_1, N_2, and N_3 depending on the visibility constraint associated with each node in N, and orders these sets by increasing cardinality (i.e., $|N_1| \leq |N_2| \leq |N_3|$). N_1 includes the nodes associated with v_2, N_2 includes the nodes associated with v_3, and N_3 includes the nodes associated with v_1. The nodes in N_1, N_2, and N_3 are then ordered by decreasing number of don't care variables, as illustrated in Figure 13(a).*

3–4) Build a clique for the fragmentation graph and minimize the number of assignments. *Figure 13(b) illustrates the recursive calls to function* **DefineClique** *showing for each execution: the value of input parameters C and i; the candidate node n_j in N_i to insert in C; its relationships with nodes already in C; and the computed clique C'. The clique finally returned by the function includes three nodes: $C =$*

$\{\langle$sNbRZJid,$\{v_1\}\rangle$, $\langle$snbRZ**JI**d,$\{v_2\}\rangle$, $\langle$snb**R**ZjId,$\{v_3\}\rangle\}$, *which cannot be further combined without violating confidentiality constraints. The corresponding set of truth assignments is* $\mathcal{I}_{sol}$ = {[S=*0*, N=*1*, B=*0*, R=*0*, Z=*0*, J=*0*, I=*0*, D=*0*], [S=*0*, N=*0*, B=*0*, R=*0*, Z=*0*, J=*1*, I=*1*, D=*0*], [S=*0*, N=*0*, B=*0*, R=*1*, Z=*0*, J=*0*, I=*0*, D=*1*]}, *which corresponds to locally minimal fragmentation* $\mathcal{F}$ = {{Name}, {Job,InsRate}, {Race,Disease}}. *We note that this fragmentation is not minimal, since there exists at least a correct fragmentation composed of two fragments* (*Example 5.1*).

The correctness and complexity of the algorithm in Figure 12 are stated by the following theorems.

Theorem 6.2 (Correctness of the heuristic algorithm) *Given a set* $\mathcal{B}$ *of Boolean variables, a set* $\mathcal{C}$ *of confidentiality constraints over* $\mathcal{B}$*, and a set* $\mathcal{V}$ *of visibility constraints over* $\mathcal{B}$*, the algorithm in Figure 12 terminates and computes (if it exists) a locally minimal set of truth assignments.*

Proof: See Appendix A. □

Theorem 6.3 (Complexity of the heuristic algorithm) *Given a set* $\mathcal{B}$ *of Boolean variables, a set* $\mathcal{C}$ *of confidentiality constraints over* $\mathcal{B}$*, and a set* $\mathcal{V}$ *of visibility constraints over* $\mathcal{B}$*, the complexity of the algorithm in Figure 12 is* $O(\prod_{v\in\mathcal{V}}|\mathcal{P}_v|\cdot|\mathcal{B}| + (|\mathcal{V}|+|\mathcal{C}|)2^{|\mathcal{B}|})$ *in time, where* $\mathcal{P}_v$ *is the set of one-paths of the OBDD representing* $v\wedge\neg(c_1\vee\ldots\vee c_m)$.

Proof: See Appendix A. □

We note that the computational cost of the algorithm includes, like the exact algorithm, the (exponential) cost of building the OBDDs. Indeed, both the algorithms first transform the input of the fragmentation problem into a set of one-paths, which represents the input to the problem of computing a (maximum weighted) clique of the fragmentation graph. The advantage of our heuristic over the exact algorithm illustrated in Section 5 is related to the search of the clique, which is exponential in the number of one-paths in the exact approach, and polynomial in the number of one-paths in the heuristic approach.

7 Experimental Results

The exact and heuristic algorithms presented in Sections 5 and 6, respectively, have been implemented as C programs to experimentally assess their behavior in terms of execution time and quality of the solution. To efficiently manage OBDDs we used the CUDD libraries [27], and to compute the maximum weighted clique of our fragmentation graph we used the implementation of the algorithm described in [23]. The experiments have been carried out on a laptop equipped with Intel 2 Duo 2GHz processor, 4 GB RAM, running Windows 7, 32 bit version.

As formally proved by Theorems 5.3 and 6.3, the computational complexity of both the exact and heuristic algorithms depends on the number of one-paths extracted from the OBDDs representing the constraints. As a consequence, we compared the execution time and the quality of the solution computed by the two algorithms varying the number of one-paths in the range between 10 and 30,000. The configurations considered in our experiments have been obtained starting from a relation schema composed of a number of attributes varying from 10 to 40. For each configuration, we randomly generated sets of confidentiality and visibility constraints. The number of confidentiality and visibility constraints varies from 5 to 25 and from 2 to 10, respectively. Each confidentiality and visibility constraint includes a number of attributes that varies from 2 to 8 and from 2 to 4, respectively. In line with real world scenarios, constraints include a limited number of attributes. Also, the number of visibility constraints is lower than the number of confidentiality constraints, since this choice reflects most real world scenarios, where the need for privacy imposes more constraints than the need for data release.

Our experimental results evaluate three aspects: *i)* the number of one-paths extracted from the OBDDs representing the constraints; *ii)* the execution time of the exact and heuristic algorithms; and *iii)* the quality of the solution, in terms of number of fragments, computed by the exact and the heuristic algorithms.

Number of one-paths. One of the main advantages of OBDDs is that the number of their one-paths is not related to the complexity of the Boolean formulas they represent. Complex Boolean formulas expressed on a high number of variables may therefore be characterized by an extremely low number of one-paths. We then experimentally measure the number of one-paths of configurations with a growing number of attributes and of confidentiality and visibility constraints. Figure 14 illustrates the number of one-paths characterizing configurations with a number of attributes varying from 10 to 40 (the scale of the y-axis is logarithmic). The results illustrated in the graph have been computed as the average of the number of one-paths obtained with 30 simulations for each configuration, where the number of constraints in each configuration varies as explained above. Note that the overall number of simulations is more than 30 since we discarded the best and worst cases, and those configurations characterized by visibility constraints that are in contrast with confidentiality constraints (i.e., configurations that do not admit a solution). As expected, the average number of one-paths grows more than linearly with the number of attributes. It is however interesting to note that the number of one-paths remains considerably lower than $2^{|\mathcal{B}|}$ in all the considered configurations. This is consistent with real world scenarios, where confidentiality constraints (visibility constraints, respectively) involve a limited number of attributes.

Execution time. As expected from the analysis of the computational complexity of our algorithms (see Theorems 5.3 and 6.3), the heuristic algorithm outperforms the exact algorithm. Indeed, consistently with

the fact that the minimal fragmentation problem is NP-hard, the exact approach requires exponential time in the number of one-paths, that is, of nodes in the fragmentation graph, which is even higher than the number of one-paths extracted from OBDDs. We run the exact algorithm only for configurations with at most 1,000 one-paths, since this configuration is characterized by a fragmentation graph including more than 6,000 nodes and more than 190,000 edges. To further confirm the exponential growth of the computational time required by the exact algorithm, we compute the fragmentation graph also for larger configurations (up to 5,000 one-paths, for which the computation of the fragmentation graph takes more than 806.99 seconds). Figure 15 compares the execution time of our heuristic and exact algorithms (for configurations with up to 1,000 one-paths), varying the number of one-paths represented by the OBDDs (the scale of both the x-axis and the y-axis is logarithmic). The figure also reports the time required for computing the fragmentation graph for configurations including between 1,000 and 5,000 one-paths, and the execution time of the heuristic algorithm for configurations including between 5,000 and 30,000 one-paths, highlighting the benefit of the heuristic approach that does not explicitly build the fragmentation graph.

To better understand the impact of building the OBDDs modeling the constraints and extracting their one-paths, we measure the execution time required by this step, which is common to the exact and heuristic algorithms. It is interesting to note that the impact of this step on the overall execution time of both our algorithms is negligible. In fact, it remains under 454 milliseconds in all the considered configurations.

Quality of the solution. Figure 16 reports the comparison between the number of fragments obtained by the execution of the exact and the heuristic algorithms (the scale of the x-axis is logarithmic). The comparison shows that, in the majority of the configurations where the comparison was possible (i.e., for configurations with less than 1,000 one-paths), our heuristic algorithm computes a locally minimal fragmentation that is also minimal since the fragmentations computed by the two algorithms have the same number of fragments. Figure 16 reports the number of fragments in the locally minimal fragmentations computed by the heuristic algorithm also for configurations with a number of one-paths between 1,000 and 5,000. It is interesting to note that also for these configurations, characterized by a considerable number of attributes and of confidentiality and visibility constraints, the number of fragments in the locally minimal fragmentation remains limited (in our experiments, it varies between 1 and 3 fragments). We can then conclude that our heuristic algorithm is efficient, computes a solution close to optimum, and can therefore be conveniently adopted in many scenarios.

8 Related Work

Data fragmentation has been studied as a solution to enforce confidentiality constraints in outsourcing scenarios, where data are stored and managed at external honest-but-curious servers [13, 16, 26]. In fact, most of the proposals in this scenario are based on the assumption that data are all equally sensitive and therefore must be entirely encrypted to protect their confidentiality.

The proposals based on fragmentation can be classified as solutions that: *1)* combine fragmentation and encryption [1, 9, 11]; and *2)* depart from encryption and satisfy confidentiality constraints by splitting the data over two fragments, one of which is stored at the data holder site [8, 10, 28, 29]. Among the first class of solutions, the approach introduced in [1] relies on the presence of two non-communicating servers. Confidentiality constraints are satisfied by splitting data between two fragments and resorting to encryption to protect sensitive attribute values and sensitive associations whenever two fragments are not sufficient to break all of them. The approach in [9, 11] enforces association constraints by splitting the data among multiple unlinkable fragments, which can also be stored on a single server, and encrypting only the attributes in singleton constraints. The fragmentation approach in [9, 11] is aimed at minimizing query execution costs for the client. The solutions that exploit storage space resources at the data holder side are aimed at completely departing from encryption, to avoid key management overheads. The approach in [8, 10] adopts vertical fragmentation to split the data between two fragments, in such a way to minimize the storage and/or computational burden of the data holder caused by the direct management of one of these fragments. The technique in [29] adopts instead horizontal fragmentation to protect sensitive associations possibly specified at the instance level (i.e., that depend on attribute values). Although the approach presented in this paper shares with these proposals the use of fragmentation for properly protecting sensitive data and/or associations, we take into consideration a different scenario and address a different problem. In fact, our solution considers a data publishing scenario, in contrast to data outsourcing, and aims at satisfying, besides confidentiality, also visibility constraints, which have been introduced in [14] where the authors adopt SAT solvers to compute a correct fragmentation. All these approaches are based on the assumption that an attacker does not have any additional knowledge on the original data collection. Therefore, she cannot infer sensitive information by combining her a-priori knowledge with the published fragments. Note that in [4] the authors prove that, for solutions that exploit storage at the data holder side, the impossibility of inferring sensitive information also holds when the knowledge of a possible attacker is limited to Equality and Tuple Generating Dependencies (which include both functional and join dependencies).

The work presented in this paper has some affinities with the proposals that introduce a policy-based classification of the data to protect their confidentiality (e.g., [2, 3, 5, 6]). The solution in [3] guarantees

confidentiality of sensitive data through a policy-based classification of databases. The confidentiality policy is described through a classification instance representing the combinations of values that need to be protected. The result of the evaluation of a query is then released to the requesting user only if it does not contain values that are considered sensitive according to the classification instance. Since a user's prior knowledge could be exploited to infer sensitive information, in [5, 6] the authors propose a model for the modification of the database instance visible to the user. This solution guarantees both data confidentiality (as specified by the data holder's policy) and consistency with a user's prior knowledge while ensuring maximum visibility of correct database answers. The approach in [2] aims instead at defining privacy constraints by reducing the problem of protecting the data from inference to the problem of enforcing access control in relational databases. These approaches differ from the proposal in this paper since they do not use fragmentation to protect confidentiality, but are concerned with returning to users query results that do not contain combinations of values that are sensitive or that can be exploited to infer sensitive information.

The problem of fragmenting relational databases while maximizing query efficiency has already been studied and some approaches have been proposed (e.g., [21, 22]). However, these techniques are not applicable to our problem, since they are only aimed at performance optimization and do not take into consideration confidentiality and visibility constraints.

Other related work is represented by proposals that introduce OBDD-based approaches for solving constraint satisfaction problems (e.g. [15, 17, 19]). These approaches compute a truth assignment for a set of variables that satisfies a set of constraints among the Boolean variable values. The solution described in this paper differs from the techniques proposed for general constraint satisfaction problems, since our approach takes advantage of the monotonicity of confidentiality and visibility constrains and therefore fully exploits the implicit representation of sets of truth assignments provided by OBDDs. These peculiarities of the fragmentation problem permit to limit the computational effort required to compute an optimal solution.

9 Conclusions

We addressed the problem of fulfilling both the needs of properly protecting sensitive data and of guaranteeing visibility requirements in data publishing scenarios. The proposed solution relies on a graph-based modeling of the fragmentation problem that takes advantage of a novel OBDD-based approach compactly representing confidentiality and visibility constraints. The fragmentation problem is then reformulated in terms of the problem of computing a maximum weighted clique over a graph modeling the fragments that satisfy confidentiality and (a subset of) visibility constraints. The set of fragments in the graph are efficiently computed adopting OBDDs that represent the Boolean formulas corresponding to confidentiality and visibility constraints. We presented

both an exact and a heuristic algorithm to solve the fragmentation problem and experimentally compared their efficiency and the quality of the fragmentations computed by the heuristics. Our work leaves space for further investigations, including: dynamic datasets, where data in the original relation change over time; multiple input relations; and possible dependencies among attributes in the original relation, which observers can exploit for reconstructing the association between data in different fragments.

Acknowledgements

This work was partially supported by the Italian Ministry of Research within the PRIN 2008 project "PEPPER" (2008SY2PH4).

References

[1] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R. Motwani, U. Srivastava, D. Thomas, and Y. Xu. Two can keep a secret: A distributed architecture for secure database services. In *Proc. of CIDR 2005*, Asilomar, CA, USA, January 2005.

[2] J. Biskup, D.W. Embley, and J. Lochner. Reducing inference control to access control for normalized database schemas. *Information Processing Letters*, 106(1):8–12, March 2008.

[3] J. Biskup and J. Lochner. Enforcing confidentiality in relational databases by reducing inference control to access control. In *Proc. of ISC 2007*, Valparaíso, Chile, October 2007.

[4] J. Biskup, M. Preuß, and L. Wiese. On the inference-proofness of database fragmentation satisfying confidentiality constraints. In *Proc. of ISC 2011*, Xi'an, China, October 2011.

[5] J. Biskup and L. Wiese. Combining consistency and confidentiality requirements in first-order databases. In *Proc. of ISC 2009*, Pisa, Italy, September 2009.

[6] J. Biskup and L. Wiese. A sound and complete model-generation procedure for consistent and confidentiality-preserving databases. *Theoretical Computer Science*, 412(31):4044 – 4072, June 2011.

[7] R.E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE TC*, 35(8):677–691, August 1986.

[8] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Enforcing confidentiality constraints on sensitive databases with lightweight trusted clients. In *Proc. of DBSec 2009*, Montreal, Quebec, Canada, July 2009.

[9] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Fragmentation design for efficient query execution over sensitive distributed databases. In *Proc. of ICDCS 2009*, Montreal, Canada, June 2009.

[10] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Keep a few: Outsourcing data while maintaining confidentiality. In *Proc. of ESORICS 2009*, Saint Malo, France, September 2009.

[11] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Combining fragmentation and encryption to protect privacy in data storage. *ACM TISSEC*, 13(3):22:1–22:33, July 2010.

[12] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, G. Livraga, and P. Samarati. Enforcing confidentiality and data visibility constraints: An OBDD approach. In *Proc. of DBSec 2011*, Richmond, VA, USA, July 2011.

[13] E. Damiani, S. De Capitani di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Balancing confidentiality and efficiency in untrusted relational DBMSs. In *Proc. of CCS 2003*, Washington, DC, USA, October 2003.

[14] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Fragments and loose associations: Respecting privacy in data publishing. *Proc. of the VLDB Endowment*, 3(1):1370–1381, September 2010.

[15] G. Gange, P.J. Stuckey, and V. Lagoon. Fast set bounds propagation using a BDD-SAT hybrid. *International Journal of Artificial Intelligence*, 38(1):307–338, May 2010.

[16] H. Hacigümüs, B. Iyer, and S. Mehrotra. Providing database as a service. In *Proc. of ICDE 2002*, San Jose, CA, USA, February 2002.

[17] T. Hadzic, E.R. Hansen, and B. O'Sullivan. On automata, MDDs and BDDs in constraint satisfaction. In *Proc. of ECAI 2008*, Patras, Greece, July 2008.

[18] D.E. Knuth. *The Art of Computer Programming, Volume 4, Fascicle 1: Bitwise Tricks & Techniques; Binary Decision Diagrams*. Addison-Wesley Professional, 2009.

[19] M. Kurihara and H. Kondo. Efficient BDD encodings for partial order constraints with application to expert systems in software verification. In B. Orchard, C. Yang, and M. Ali, editors, *Innovations in Applied Artificial Intelligence*. Springer, 2004.

[20] C. Meinel and T. Theobald. *Algorithms and Data Structures in VLSI Design*. Springer-Verlag, 1998.

[21] S. Navathe, S. Ceri, G. Wiederhold, and J. Dou. Vertical partitioning algorithms for database design. *ACM TODS*, 9(4):680–710, December 1984.

[22] S. Navathe and M. Ra. Vertical partitioning for database design: A graphical algorithm. In *Proc. of SIGMOD 1989*, Portland, OR, USA, June 1989.

[23] P.R.J. Östergård. A new algorithm for the maximum-weight clique problem. *Nordic Journal of Computing*, 8(4):424–436, December 2001.

[24] P.R.J. Östergård. A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, 120(1–3):197–207, August 2002.

[25] P. Samarati. Protecting respondents' identities in microdata release. *IEEE TKDE*, 13(6):1010–1027, November/December 2001.

[26] P. Samarati and S. De Capitani di Vimercati. Data protection in outsourcing scenarios: Issues and directions. In *Proc. of the ASIACCS 2010*, Beijing, China, April 2010. Invited paper.

[27] F. Somenzi. CUDD: CU Decision Diagram package – release 2.4.2. Department of Electrical and Computer Engineering – University of Colorado at Boulder, 2009.

[28] A. Steele and K.B. Frikken. An index structure for private data outsourcing. In *Proc. of DBSec 2011*, Richmond, VA, USA, July 2011.

[29] L. Wiese. Horizontal fragmentation for data outsourcing with formula-based confidentiality constraints. In *Proc. of IWSEC 2010*, Kobe, Japan, November 2010.

A Proof of Theorems

Theorem 5.2 (Correctness of the exact algorithm) *Given a set $\mathcal{B}$ of Boolean variables, a set $\mathcal{C}$ of confidentiality constraints over $\mathcal{B}$, and a set $\mathcal{V}$ of visibility constraints over $\mathcal{B}$, the algorithm in Figure 10 terminates and computes (if it exists) a minimal set of truth assignments.*

PROOF: To prove the correctness of the algorithm in Figure 10, we have to show that *i)* it terminates; *ii)* it computes a correct set of truth assignments; *iii)* if there exists a correct set $\mathcal{I}$ of truth assignments with respect to $\mathcal{C}$ and $\mathcal{V}$, the algorithm finds it; and *iv)* it computes a minimal set of truth assignments.

Termination. Since the number of confidentiality and visibility constraints is finite, the **for each** loop in Step 1 terminates. The first two **for each** loops in Step 2 (line 10 and line 16, respectively) terminate since for each $v \in \mathcal{V}$ the set $\mathcal{P}_v$ of one-paths is finite, and N includes at most one node for each one-path in $\mathcal{P}_{v_1} \cup \ldots \cup \mathcal{P}_{v_k}$. The **while** loop in Step 2 (line 30) terminates since the number of nodes that it inserts in N is finite. In fact, let n_{ij} be a node obtained through the combination of n_i and n_j, which are nodes connected by a mergeable edge: *i)* n_{ij} is inserted in N only if N does not include a node n associated with a truth assignment and a set of visibility constraints equal to those of node n_{ij} (N is a set); *ii)* n_{ij} is adjacent to neither n_i nor n_j; and *iii)* edge $(n_i,n_j) \in M$ is removed from M when node n_{ij} is inserted in N. Therefore, at each iteration of the **while** loop, at most one node is inserted in N, an edge is removed from M, and a limited number of edges is inserted in M. Since the number of Boolean variables in $\mathcal{B}$ and of visibility constraints in $\mathcal{V}$ is finite, the number of possible nodes generated by merging nodes in N is finite. As a consequence, the **while** loop terminates. Function **FindMaxWeightClique** in Step 3 (line 53) terminates since it exploits a classical algorithm for finding a maximum weight clique. The last **for each** loop in Step 3 (line 56) terminates since C is a subset of N, which is a finite set of nodes.

Correctness of the set of truth assignments. $\mathcal{I}_{sol}$ is correct iff it satisfies the conditions in Definition 3.2.

1. $\forall c \in \mathcal{C}, \forall I \in \mathcal{I}_{sol}$: $I(c) = 0$ (confidentiality). In the last **for each** loop of Step 3 (line 56), the algorithm extracts $\mathcal{I}_{sol}$ from the clique C computed by function **FindMaxWeightClique** (the don't care variables in the truth assignments represented by the nodes in C are set to 0). Since function **FindMaxWeightClique** does not modify the truth assignments represented by the nodes in the graph received as input, $\mathcal{I}_{sol}$ does not violate confidentiality constraints iff the nodes in the fragmentation graph resulting from Step 2 represent truth assignments that do not violate confidentiality constraints. Each node in G either represents a truth assignment $I \in \mathcal{P}_{v_i}$ or a truth assignment resulting from the composition of a subset of one-paths in $\mathcal{P}_{v_1} \cup \ldots \cup \mathcal{P}_{v_k}$ under operator $\odot$. In the first case, I represents a one-path in the OBDD modeling Boolean formula $v \wedge \neg(c_1 \vee \ldots \vee c_m)$, $v \in \mathcal{V}$, and therefore it satisfies all the confidentiality constraints. In the second case, I represents in a compact way the complete truth assignments implicitly represented by

the composed assignments. Since the assignments composed to generate I satisfy all the confidentiality constraints, also I satisfies all of them.

2. $\forall v{\in}\mathcal{V}$, $\exists I \in \mathcal{I}_{sol}$: $I(v) = 1$ (visibility). In Step 3, the algorithm checks if the clique C computed by function **FindMaxWeightClique** has weight at least $|\mathcal{V}| \cdot (|\mathcal{V}| - 1)$, which is equivalent to check if $\forall v{\in}\mathcal{V}$, $\exists n{\in}C$ such that $v{\in}n.V$ as proved by Property 4.4. If the weight of the clique C is greater than or equal to $|\mathcal{V}| \cdot (|\mathcal{V}| - 1)$, the algorithm computes a solution $\mathcal{I}_{sol}$ obtained by setting to 0 all the don't care variables in the truth assignments represented by the nodes in C. Therefore $\mathcal{I}_{sol}$ satisfies all the visibility constraints.

3. $\forall I_i,I_j{\in}\mathcal{I}_{sol}, i \neq j$, $\forall a \in \mathcal{B}$ s.t. $I_i(a) = 1$: $I_j(a) = 0$ (unlinkability). In the last **for each** loop of Step 3 (line 56), the algorithm extracts $\mathcal{I}_{sol}$ from the clique C computed by function **FindMaxWeightClique** by setting to 0 all the don't care variables in the truth assignments represented by the nodes in the clique. Since function **FindMaxWeightClique** does not modify the graph received as input, $\mathcal{I}_{sol}$ satisfies unlinkability iff C is a clique and G includes only edges connecting nodes representing unlinkable truth assignments. Since the **while** loop in Step 2 of the algorithm (line 30) removes all the edges in M connecting nodes representing mergeable truth assignments, the graph G given as input to the **FindMaxWeightClique** function includes only edges in D, which connect nodes representing disjoint truth assignments. Any pair of nodes in C is then connected by an edge in D (i.e., nodes in C represent unlinkable assignments).

Completeness. Suppose by contradiction that there exists a correct set $\mathcal{I}$ of truth assignments and that our algorithm returns an empty solution. The algorithm returns an empty solution only if **FindMaxWeightClique** returns a clique for which at least one visibility constraint in $\mathcal{V}$ remains unsatisfied (line 54). Since function **FindMaxWeightClique** implements a known algorithm for the maximum weighted clique problem and according to Theorem 4.5, the function returns a clique that does not satisfy all the visibility constraints only if a clique representing a correct set of truth assignments does not exists in G. However, since we assume that $\mathcal{I}$ is a correct set of truth assignments, $\mathcal{I}$ has to be represented by a clique in G.

Suppose now by contradiction that $\mathcal{I}$ is not represented by a clique in G. Each $I{\in}\mathcal{I}$ satisfies all the confidentiality constraints and at least one visibility constraint (otherwise I could be removed from $\mathcal{I}$ preserving its correctness). Let $v_{i_1},\ldots,v_{i_k},{\in}\mathcal{V}$ be the visibility constraints satisfied by I. By construction of the OBDDs $O_{i_1},\ldots,O_{i_k}$, $\mathcal{P}_{i_1},\ldots,\mathcal{P}_{i_k}$ must contain at least one one-path $I_{i_1},\ldots,I_{i_k}$ that implicitly represents I. Since, by Observation 3, truth assignments are mergeable only if they represent at least a common complete truth assignment, $I_{i_1},\ldots,I_{i_k}$ are mergeable (they all implicitly represent I). For each $I{\in}\mathcal{I}$ and for each subset $V \subseteq \{v_{i_1},\ldots,v_{i_k}\}$ of visibility constraints satisfied by I, the fragmentation graph computed by our algorithm will then include a node n implicitly representing I, with $n.V{=}V$. Since $\mathcal{I}$ is a correct set of truth assignments,

there is at least a complete truth assignment $I \in \mathcal{I}$ that satisfies v for each $v \in \mathcal{V}$. Therefore, $\mathcal{V}$ can be partitioned in non-empty and non-overlapping subsets V_i, $i = 1, \ldots, |\mathcal{I}|$, such that each $I_i \in \mathcal{I}$ is associated with V_i and V_i includes only visibility constraints satisfied by I_i. As a consequence, there exists a set N of nodes in G with a node n_i for each $I_i \in \mathcal{I}$ such that n_i implicitly represents I_i and n_i is associated with the set V_i of visibility constraints. For each pair $I_i, I_j \in \mathcal{I}$, I_i and I_j are neither mergeable nor in conflict. As a consequence, nodes n_i and n_j in N implicitly representing I_i and I_j, respectively, are not in conflict (i.e., they do not associate conflicting values to the same variable) and are then connected by an edge in D. Therefore, N is a clique for G that implicitly represents $\mathcal{I}$, thus contradicting the initial hypothesis.

Minimality. Suppose by contradiction that the algorithm computes a correct set $\mathcal{I}$ of k truth assignments and that there exists a correct set $\mathcal{I}'$ of $k' < k$ truth assignments. We prove that, if $\mathcal{I}'$ exists, the set $\mathcal{I}$ of truth assignments computed by our algorithm includes k' truth assignments. In the last **for each** loop of Step 3 (line 56), the algorithm extracts $\mathcal{I}$ from the clique C of maximum weight in G computed by function **FindMaxWeightClique** by setting to 0 all the don't care variables in the truth assignments represented by the nodes in the clique. According to the anti-monotonicity of the weight function w with respect to the number of nodes (Property 4.3), C is the clique satisfying all the visibility constraints composed of the minimum number k of nodes. Since function **FindMaxWeightClique** implements a known algorithm for the maximum weighted clique problem, there cannot exist a clique C' of G satisfying all the visibility constraints and composed of $k' < k$ nodes. In other words, there cannot exist a correct set $\mathcal{I}'$ of $k' < k$ truth assignments, where for each $I \in \mathcal{I}'$ there exists a node n in G such that $n.I$ implicitly represents I. However, since we assume that $\mathcal{I}'$ is a correct set of truth assignments, $\mathcal{I}'$ has to be represented by a clique in G.

Suppose now by contradiction that $\mathcal{I}'$ is not represented by a clique in G. Let I be a complete truth assignment in $\mathcal{I}'$. Since $\mathcal{I}'$ is correct, I satisfies all the confidentiality constraints. Moreover, I is associated with (and then satisfies) at least one visibility constraint, otherwise I could be removed from $\mathcal{I}'$ preserving its correctness, thus contradicting the hypothesis of minimality of $\mathcal{I}'$. Let $v_{i_1}, \ldots, v_{i_k}, \in \mathcal{V}$ be the visibility constraints associated with I. By construction of the OBDDs $O_{i_1}, \ldots, O_{i_k}$, $\mathcal{P}_{i_1}, \ldots, \mathcal{P}_{i_k}$ must contain at least one one-path $I_{i_1}, \ldots, I_{i_k}$ that implicitly represents I. Since, by Observation 3, truth assignments are mergeable only if they represent at least a common complete truth assignment, $I_{i_1}, \ldots, I_{i_k}$ are mergeable (they all implicitly represent I). The truth assignment I' obtained as $I_{i_1} \odot \ldots \odot I_{i_k}$ implicitly represents I. As a consequence, the fragmentation graph computed by our algorithm will include a node n with $n.I = I'$ and $n.V = \{v_{i_1}, \ldots, v_{i_k}\}$ that implicitly represents I. This is true for each complete truth assignment $I \in \mathcal{I}'$. Therefore, G includes a clique that implicitly represents $\mathcal{I}'$ and composed of k' nodes. As a consequence, the clique computed by our algorithm implicitly represents $\mathcal{I}'$. This contradicts the original hypothesis that our algorithm computes a set of truth assignments that is not minimal.

Theorem 5.3 (Complexity of the exact algorithm) *Given a set $\mathcal{B}$ of Boolean variables, a set $\mathcal{C}$ of confidentiality constraints over $\mathcal{B}$, and a set $\mathcal{V}$ of visibility constraints over $\mathcal{B}$, the complexity of the algorithm in Figure 10 is $O(2^{\prod_{v\in\mathcal{V}}|\mathcal{P}_v|\cdot|\mathcal{B}|}+(|\mathcal{V}|+|\mathcal{C}|)2^{|\mathcal{B}|})$ in time, where $\mathcal{P}_v$ is the set of one-paths of the OBDD representing $v\wedge\neg(c_1\vee\ldots\vee c_m)$.*

PROOF: The construction of the OBDDs in Step 1 can be, in the worst case, exponential in the number of variables in the formula they represent, that is, $O(2^{|\mathcal{B}|})$. Therefore, the construction of the OBDDs representing the confidentiality constraints in $\mathcal{C}$, the visibility constraints in $\mathcal{V}$, and their combination has computational complexity $O((|\mathcal{V}|+|\mathcal{C}|)2^{|\mathcal{B}|})$. The construction of the fragmentation graph in Step 2 requires to compute the closure of the set $\mathcal{P}_{v_1}\cup\ldots\cup\mathcal{P}_{v_k}$ under operator $\odot$ since nodes in G represent, in the worst case, all the truth assignments in $\mathcal{P}^{\odot}$. To this purpose, the algorithm inserts edges in M and in D, connecting pairs of nodes representing mergeable or disjoint truth assignments, respectively, that are associated with non-overlapping sets of visibility constraints. The cost of inserting edges in G is then $O(\prod_{v\in\mathcal{V}}|\mathcal{P}_v|\cdot|\mathcal{B}|)$ since the cost of evaluating if two truth assignments are mergeable or disjoint is linear in the number of Boolean variables composing the truth assignments. Since for each edge in M the algorithm inserts a node in G, which can only be connected to the nodes adjacent to the incident nodes of the removed edge, the overall cost of building G is $O(\prod_{v\in\mathcal{V}}|\mathcal{P}_v|\cdot|\mathcal{B}|)$. Also, G includes at most $O(\prod_{v\in\mathcal{V}}|\mathcal{P}_v|)$ nodes. Function **FindMaxWeightClique** has exponential cost in the number of nodes of the input graph (i.e., $O(2^{\prod_{v\in\mathcal{V}}|\mathcal{P}_v|\cdot|\mathcal{B}|})$) since the maximum weighted clique problem is NP-hard. The last **for each** loop in Step 3 (line 56) has computational complexity $O(|C|\cdot|\mathcal{B}|)$, since it scans all the nodes in C to set to 0 the don't care variables in the truth assignments they represent. The cost of this loop is however dominated by the cost of the previous steps of the algorithm. The computational complexity of the algorithm is therefore $O(2^{\prod_{v\in\mathcal{V}}|\mathcal{P}_v|\cdot|\mathcal{B}|}+(|\mathcal{V}|+|\mathcal{C}|)2^{|\mathcal{B}|})$.

Theorem 6.2 (Correctness of the heuristic algorithm) *Given a set $\mathcal{B}$ of Boolean variables, a set $\mathcal{C}$ of confidentiality constraints over $\mathcal{B}$, and a set $\mathcal{V}$ of visibility constraints over $\mathcal{B}$, the algorithm in Figure 12 terminates and computes (if it exists) a locally minimal set of truth assignments.*

PROOF: To prove the correctness of the algorithm in Figure 12, we need to prove that: *i)* it terminates; *ii)* it computes a correct set of truth assignments; *iii)* if there exists a correct set $\mathcal{I}$ of truth assignments with respect to $\mathcal{C}$ and $\mathcal{V}$, the algorithm finds it; and *iv)* it computes a locally minimal set of truth assignments.

Termination. Since the number of confidentiality constraints, the number of visibility constraints, and the number of one-paths in $\mathcal{P}_v$, with $v\in\mathcal{V}$, is finite, the three **for each** loops in Step 1 (line 2) and in Step 2 (line 8 and line 14, respectively) terminate. The recursive call to function **DefineClique** in Step 3 (line17) terminates when variable i is greater than or equal to $|\mathcal{V}|$. We note that at each recursive call of function **DefineClique**, at most one node is inserted in C. Therefore, if function **DefineClique** terminates, C is a finite set. Function

DefineClique terminates because: *i)* the sets N_i, $i = 1, \ldots, |\mathcal{V}|$, of nodes are finite ($\forall v_i \in \mathcal{V}$, $\mathcal{P}_{v_i}$ is a finite set of one-paths); *ii) LinkableNodes* is a subset of $\mathcal{I}_{sol}$; and *iii)* variable i increases by one at each recursive call. The **for** loop and the **while** loop in function **DefineClique** (line 31 and line 36, respectively) terminate since the clique C received as input includes at most i nodes and is therefore finite.

Correctness of the set of truth assignments. $\mathcal{I}_{sol}$ is correct iff it satisfies the conditions in Definition 3.2.

1. $\forall c \in \mathcal{C}$, $\forall I \in \mathcal{I}_{sol}$: $I(c) = 0$ (confidentiality). The **while** loop in Step 4 (line 21) computes $\mathcal{I}_{sol}$ (starting from the one-paths represented by nodes in C) by trying to compose sets of truth assignments represented by nodes in C through $\vee$ operator and explicitly checking if the result of the composition violates confidentiality constraints. Since truth assignments are composed only if their composition does not violate the confidentiality constraints, $\mathcal{I}_{sol}$ satisfies all the confidentiality constraints iff the truth assignments represented by nodes in the clique C computed by function **DefineClique** do not violate the confidentiality constraints. Function **DefineClique** inserts a node n in C if n either belongs to N_i (i.e., it represents a one-path in $\mathcal{P}_{v_k}$) or has been obtained by composing a set of nodes in $N_1, \ldots, N_k$ (i.e., it represents the composition of a subset of one-paths in $\mathcal{P}_{v_x}, \ldots, \mathcal{P}_{v_y}$ under operator $\odot$). Since $\mathcal{P}_v$ represents the one-paths in the OBDD modeling Boolean formula $v \wedge \neg(c_1 \vee \ldots \vee c_m)$, $v \in \mathcal{V}$, all the truth assignments in $\mathcal{P}_v$ satisfy confidentiality constraints. Also, since the truth assignment resulting from the composition of I_i and I_j under $\odot$ represents, in a compact way, the set of complete truth assignments implicitly represented by both I_i and I_j, also $I_i \odot I_j$ does not violate confidentiality constraints. As a consequence, each node $n \in C$ represents a truth assignment that satisfies all the confidentiality constraints.

2. $\forall v \in \mathcal{V}$, $\exists I \in \mathcal{I}_{sol}$: $I(v) = 1$ (visibility). Recalling that in Step 4 all the don't care variables in truth assignments represented by nodes in C are set to 0 (line 20), and that the algorithm computes $\mathcal{I}_{sol}$ by trying to compose the truth assignments represented by nodes in C without violating confidentiality constraints (**for each** loop in line 24), $\mathcal{I}_{sol}$ satisfies the visibility constraints if, at the end of function **DefineClique**, $\forall v \in \mathcal{V}$, $\exists\ n \in C$ such that $v \in n.V$. Function **DefineClique** is recursively called for $i = 1, \ldots, |\mathcal{V}|$ and, at each recursive call, it inserts in C a node $n_j \in N_i$ that represents a truth assignment I in $\mathcal{P}_v$. In fact, node n_j is either inserted as a new node in C, or composed with a node n already in C. As already noted, the node resulting from the composition of n_j with n is associated with (and then satisfies) both the visibility constraints in $n.V$ and in $n_j.V$. Therefore, at the end of the i-th recursive call, C is associated with (and then satisfies) all the visibility constraints v such that $v \in n.V$ and n is a node in N_j, with $j \leq i$. We can conclude that, at the end of the $|\mathcal{V}|$-th recursive call, C includes, for each $v \in \mathcal{V}$, at least a node n with $v \in n.V$.

3. $\forall I_i, I_j \in \mathcal{I}_{sol}, i \neq j$, $\forall a \in \mathcal{B}$ s.t. $I_i(a) = 1$: $I_j(a) = 0$ (unlinkability). Since Step 4 sets to 0 all the don't

care variables in the truth assignments represented by nodes in C, $\mathcal{I}_{sol}$ satisfies unlinkability iff the truth assignments represented by the nodes in C computed by function **DefineClique** are disjoint. Function **DefineClique** tries to insert, at each iteration of the **for** loop (line 31), a node n_j in C. The function does not insert $n_j \in N_i$ in C if there exists at least a node in C that represents a truth assignment *linkable but not mergeable* with $n_j.I$, and composes $n_j.I$ with all the linkable and mergeable truth assignments represented by a node already in C. All nodes in C therefore represent disjoint truth assignments.

Completeness. Completeness is guaranteed if recursive function **DefineClique** computes, if it exists, a clique C of the fragmentation graph that satisfies all the visibility constraints in $\mathcal{V}$. Function **DefineClique** is recursively called for $i = 1, \ldots, |\mathcal{V}|$ and, at each recursive call, it inserts in C a node n_j in N_i, which represents a truth assignment in $\mathcal{P}_v$ (i.e., a truth assignment associated with and then satisfying visibility constraint v). If there is no clique in G including C together with a node in N_i, the function uses a back-track strategy and tries to insert in C a different node from N_{i-1}. Note that two nodes are combined (operator $\odot$) by function **DefineClique** iff they represent linkable truth assignments (i.e., they represent fragments with a common attribute). Indeed, a correct set of truth assignments cannot contain two linkable truth assignments (Condition 3 in Definition 3.2). Therefore, the composition performed in this phase is mandatory for finding a correct set of truth assignments. Nodes representing non linkable but mergeable truth assignments are not combined in this phase (they will be combined by Step 4 to guarantee local minimality). Recursive function **DefineClique** tries all the possible subsets of nodes in G including a node for each set N_i, $i = 1, \ldots, k$, using the back-track strategy. Thus, if there exist a clique for G, it will be found by the recursive call.

Local minimality. $\mathcal{I}_{sol}$ is locally minimal iff no pair of truth assignments in $\mathcal{I}_{sol}$ can be composed through the $\vee$ operator without violating confidentiality constraints. $\mathcal{I}_{sol}$ is computed by the **while** loop in Step 4 (line 21), where the algorithm tries to iteratively combine ($\vee$) the truth assignments represented by nodes in C, after all don't care variables have been set to 0. Such a combination is performed only if the disjunction of the confidentiality constraints is not violated. As a consequence, no pair of truth assignments in $\mathcal{I}_{sol}$ are combined through the $\vee$ operator without violating the confidentiality constraints.

Theorem 6.3 (Complexity of the heuristic algorithm) *Given a set $\mathcal{B}$ of Boolean variables, a set $\mathcal{C}$ of confidentiality constraints over $\mathcal{B}$, and a set $\mathcal{V}$ of visibility constraints over $\mathcal{B}$, the complexity of the algorithm in Figure 12 is $O(\prod_{v \in \mathcal{V}} |\mathcal{P}_v| \cdot |\mathcal{B}| + (|\mathcal{V}| + |\mathcal{C}|)2^{|\mathcal{B}|})$ in time, where $\mathcal{P}_v$ is the set of one-paths of the OBDD representing $v \wedge \neg(c_1 \vee \ldots \vee c_m)$.*

PROOF: The construction of the OBDDs in Step 1 can be, in the worst case, exponential in the number of variables in the formula they represent, that is, $O(2^{|\mathcal{B}|})$. Therefore, the construction of the OBDDs representing the confidentiality constraints in $\mathcal{C}$ and the visibility constraints in $\mathcal{V}$, and their combination have computational

complexity $O((|\mathcal{V}|+|\mathcal{C}|)2^{|\mathcal{B}|})$. The cost of building a node n for each one-path in $\mathcal{P}_{v_1}\cup\ldots\cup\mathcal{P}_{v_k}$ is linear in the number of one-paths. The cost of partitioning the resulting set of nodes in sets of nodes that are associated with the same visibility constraint and of ordering these sets by their cardinality is $O(\sum_{v\in\mathcal{V}}|\mathcal{P}_v|+|\mathcal{V}|\log|\mathcal{V}|)$. The cost of further ordering the nodes in each set N_i by decreasing number of don't care variables in the one-path they represent is $O(\sum_{v\in\mathcal{V}}(|\mathcal{B}||\mathcal{P}_v|+|\mathcal{P}_v|\log|\mathcal{P}_v|))$, since each set N_i includes $|\mathcal{P}_v|$ nodes (one for each one-path) and each one-path in $\mathcal{P}_v$ has $|\mathcal{B}|$ variables. The overall cost of Step 2 is then $O(|\mathcal{V}|\log|\mathcal{V}|+|\mathcal{P}_v|\sum_{v\in\mathcal{V}}(1+|\mathcal{B}|+\log|\mathcal{P}_v|))$. Recursive function **DefineClique** is invoked by the algorithm in Figure 12 at most $\prod_{v\in\mathcal{V}}|\mathcal{P}_v|$ times, since it needs to evaluate any possible combination of nodes (truth assignments), including a node from each N_i (a truth assignment from each $\mathcal{P}_v$, $v\in\mathcal{V}$). The comparison between two truth assignments $n_i.I$ and $n_j.I$ represented by two nodes in G has computational complexity $O(|\mathcal{B}|)$, since each Boolean variable in $\mathcal{B}$ must be checked. In the worst case, each node in N_i (truth assignment in $\mathcal{P}_v$) is compared with all the nodes in N_j (truth assignments in all the other sets of one-paths), $i\neq j$. The first **for each** loop in Step 4 has computational complexity $O(|C|\cdot|\mathcal{B}|)$, since it scans all the truth assignments represented by nodes in C to set to 0 all the don't care variables. The **while** loop in Step 4 has instead computational complexity $O(|C|^2\cdot|\mathcal{B}|)$, since it compares each pair of truth assignments represented by nodes in C. The computational complexity of the algorithm is therefore $O(\prod_{v\in\mathcal{V}}|\mathcal{P}_v|\cdot|\mathcal{B}|+(|\mathcal{V}|+|\mathcal{C}|)2^{|\mathcal{B}|})$, since the costs of Step 2 and of Step 4 are dominated by the costs of Step 1 and of Step 3.

Figure 1: Example of relation (a) and of confidentiality (b) and visibility constraints (c) over it

Figure 2: Example of fragmentation of relation PATIENTS in Figure 1(a) satisfying the constraints in Figures 1(b) and 1(c)

Figure 3: Boolean interpretation of the relation schema and of the confidentiality and visibility constraints in Figure 1

Figure 4: OBDDs representing the confidentiality constraints in Figure 3

Figure 5: OBDDs representing the visibility constraints in Figure 3

Figure 6: OBDD representing the disjunction of the confidentiality constraints in Figure 3

Figure 7: OBDDs representing the composition of each visibility constraint in Figure 5 with the negated disjunction of the confidentiality constraints in Figure 4, and their one-paths

Figure 8: Assignment merging operator

Figure 9: Fragmentation graph representing the one-paths extracted from the OBDDs in Figure 7 and their closure under operator $\odot$

Figure 10: Algorithm that computes a minimal set of truth assignments

Figure 11: Example of the execution of the algorithm in Figure 10 with the inputs in Figure 3

Figure 12: Algorithm that computes a locally minimal set of truth assignments

Figure 13: Example of the execution of the algorithm in Figure 12 with the inputs in Figure 3

Figure 14: Average number of one-paths varying the number of attributes

Figure 15: Execution time of the exact and heuristic algorithms

Figure 16: Number of fragments of the solution computed by the exact and heuristic algorithms

PATIENTS

SSN	Name	Birth	Race	ZIP	Job	InsRate	Disease
123-45-6789	Alice	74/01/17	white	24201	nurse	5K	diabetes
234-56-7654	Barbara	49/02/21	white	24223	clerk	9K	stomach ulcer
345-67-8123	Carol	55/10/01	asian	25273	manager	7K	hearth attack
456-78-9876	Donna	68/12/29	white	26134	lawyer	8K	gastritis
567-89-0534	Emma	81/10/02	black	24343	chef	6K	asthma

(a)

$\mathcal{C}$

$c_1 = \{\texttt{SSN}\}$
$c_2 = \{\texttt{Name}, \texttt{InsRate}\}$
$c_3 = \{\texttt{Name}, \texttt{Disease}\}$
$c_4 = \{\texttt{Birth}, \texttt{Race}, \texttt{ZIP}\}$
$c_5 = \{\texttt{Job}, \texttt{Disease}\}$

(b)

$\mathcal{V}$

$v_1 = \texttt{Name} \vee (\texttt{Birth} \wedge \texttt{ZIP})$
$v_2 = \texttt{Job} \wedge \texttt{InsRate}$
$v_3 = \texttt{Disease} \wedge (\texttt{Birth} \vee \texttt{Race})$

(c)

Figure 1:

F_1

Birth	ZIP	Disease
74/01/17	24201	diabetes
49/02/21	24223	stomach ulcer
55/10/01	25273	hearth attack
68/12/29	26134	gastritis
81/10/02	24343	asthma

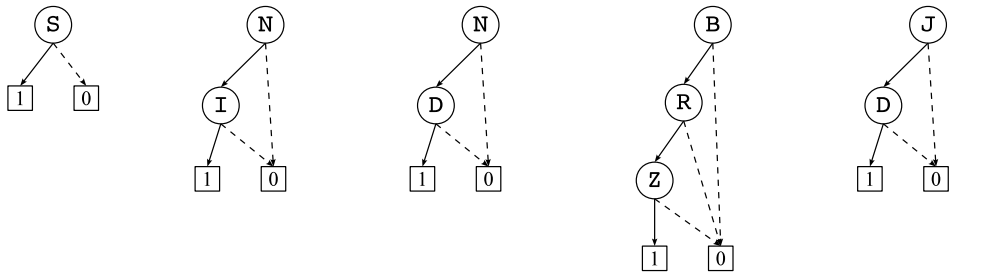
F_2

Job	InsRate
nurse	5K
clerk	9K
manager	7K
lawyer	8K
chef	6K

Figure 2:

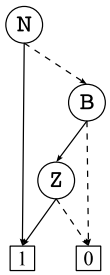
$\mathcal{B}$	$\mathcal{C}$	$\mathcal{V}$
SSN	c_1 = SSN	v_1 = Name∨(Birth∧ZIP)
Name	c_2 = Name∧InsRate	v_2 = Job∧InsRate
Birth	c_3 = Name∧Disease	v_3 = Disease∧(Birth∨Race)
Race	c_4 = Birth∧Race∧ZIP	
ZIP	c_5 = Job∧Disease	
Job		
InsRate		
Disease		

Figure 3:

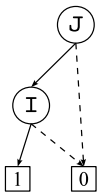


c_1=SSN c_2=Name∧InsRate c_3= Name∧Disease c_4=Birth∧Race∧ZIP c_5=Job∧Disease

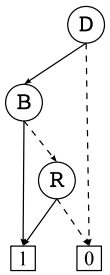
Figure 4:



v_1=Name∨(Birth∧ZIP)



v_2=Job∧InsRate



v_3=Disease∧(Birth∨Race)

Figure 5:

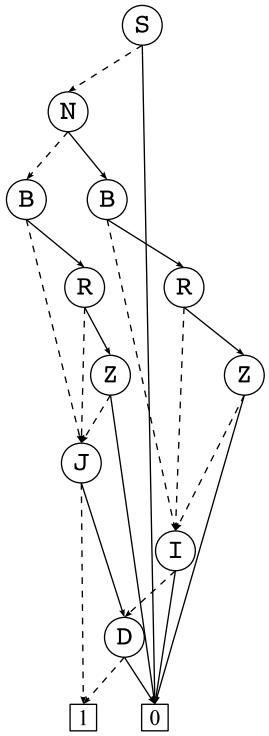


Figure 6:

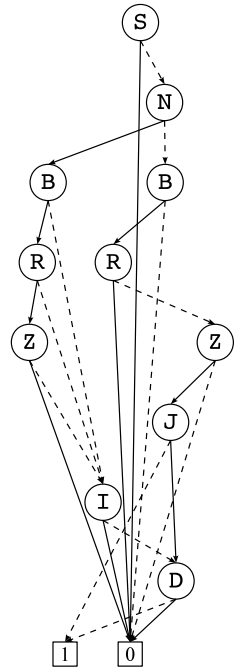
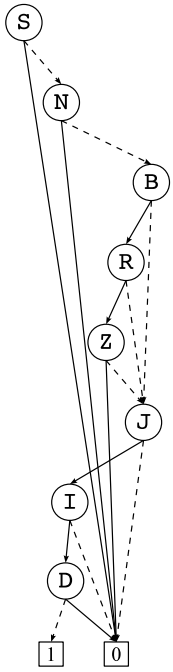
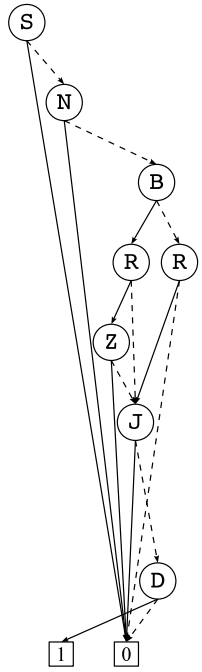
$v_1 \wedge \neg(c_1 \vee c_2 \vee c_3 \vee c_4 \vee c_5)$	$v_2 \wedge \neg(c_1 \vee c_2 \vee c_3 \vee c_4 \vee c_5)$	$v_3 \wedge \neg(c_1 \vee c_2 \vee c_3 \vee c_4 \vee c_5)$
		
S N B R Z J I D 0 1 0 - - - 0 0 0 0 1 0 1 0 - - 0 1 1 0 - - 0 0 0 0 1 0 1 1 - 0 0 1 1 1 0 - 0 0	S N B R Z J I D 0 0 0 - - 1 1 0 0 0 1 0 - 1 1 0 0 0 1 1 0 1 1 0	S N B R Z J I D 0 0 0 1 - 0 - 1 0 0 1 0 - 0 - 1 0 0 1 1 0 0 - 1

Figure 7:

$\odot$	0	1	-
0	0	n.a.	0
1	n.a.	1	1
-	0	1	-

Figure 8:

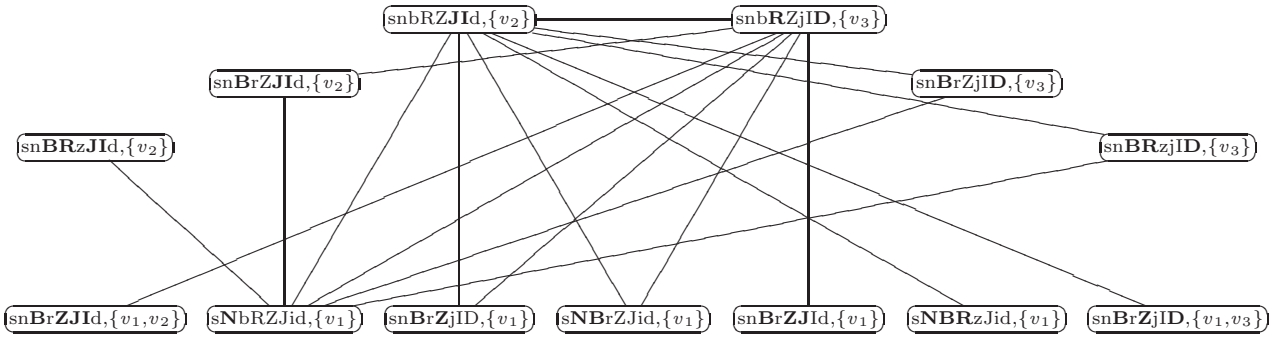

snbRZJId,{v2}
snbRZjID,{v3}
snBrZJId,{v2}
snBrZjID,{v3}
snBRzJId,{v2}
snBRzjID,{v3}
snBrZJId,{v1,v2}
sNbRZJid,{v1}
snBrZjID,{v1}
sNBrZJid,{v1}
snBrZJId,{v1}
sNBRzJid,{v1}
snBrZjID,{v1,v3}

Figure 9:

INPUT: $\mathcal{B} = \{a_1,\ldots,a_n\}$, $\mathcal{C} = \{c_1,\ldots,c_m\}$, $\mathcal{V} = \{v_1,\ldots,v_k\}$ /* Boolean variables, confidentiality and visibility constraints */
OUTPUT: $\mathcal{I}_{sol} = \{I_1,\ldots,I_l\}$ /* minimal set of truth assignments */
MAIN
/* **Step 1**: extract the one-paths from the OBDDs representing constraints */
for each $v_i \in \mathcal{V}$ **do**
  let O_i be the OBDD representing $v_i \wedge \neg(c_1 \vee \ldots \vee c_m)$
  let $\mathcal{P}_{v_i}$ be the set of one-paths of O_i
  if $\mathcal{P}_{v_i} = \emptyset$ **then return**($\emptyset$) /* no solution exists */
/* **Step 2**: build the fragmentation graph G */
$N := \emptyset$ /* set of nodes in G */
$M := \emptyset$ /* set of edges connecting nodes representing mergeable truth assignments */
$D := \emptyset$ /* set of edges connecting nodes representing disjoint truth assignments */
for each $v \in \mathcal{V}$ **do** /* insert nodes in G */
  for each $I \in \mathcal{P}_v$ **do**
    $n := \langle I, \{v\}\rangle$
    $N := N \cup \{n\}$
    $n.weight := (|\mathcal{V}| \cdot |n.V|) - 1$
$N' := N$
for each $n_i \in N$ **do** /* insert edges in G */
  $N' := N' - \{n_i\}$
  satisfied $:= n_i.V$
  for each $n_j \in N'$ **do**
    if $n_i.I \rightleftharpoons n_j.I \wedge n_i.V \cap n_j.V = \emptyset$ **then** /* the nodes represent mergeable truth assignments */
      $M := M \cup \{(n_i,n_j)\}$
      satisfied $:=$ *satisfied* $\cup\, n_j.V$
    if $n_i.I \not\rightleftharpoons n_j.I \wedge n_i.V \cap n_j.V = \emptyset$ **then** /* the nodes represent disjoint truth assignments */
      $D := D \cup \{(n_i,n_j)\}$
      satisfied $:=$ *satisfied* $\cup\, n_j.V$
  if *satisfied* $\neq \mathcal{V}$ **then** /* remove n_i from G, since it cannot be part of any solution */
    $M := M - \{(n_i,n_j)\colon n_j \in N\}$
    $D := D - \{(n_i,n_j)\colon n_j \in N\}$
    $N := N - \{n_i\}$
while $M \neq \emptyset$ **do** /* close the one-paths in N w.r.t. $\odot$ operator */
  $M' := M$
  $M := \emptyset$
  while $M' \neq \emptyset$ **do**
    let (n_i,n_j) be an edge in M' /* choose a mergeable edge */
    $M' := M' - \{(n_i,n_j)\}$
    $n_{ij} := \langle n_i.I \odot n_j.I,\ n_i.V \cup n_j.V\rangle$ /* compute the merged node */
    $n_{ij}.weight := |\mathcal{V}| \cdot |v_{ij}.V| - 1$ /* weight of the new node */
    if $n_{ij}.V = \mathcal{V}$ **then** /* n_{ij} is a clique of size 1 */
      $\mathcal{I}_{sol} := \{n_{ij}.I\}$
      assign 0 to don't care variables in $n_{ij}.I$
      return($\mathcal{I}_{sol}$)
    $N := N \cup \{n_{ij}\}$ /* insert the node in the fragmentation graph */
    for each $n_k \in \{n \in N\colon (n,n_i) \in M \vee (n,n_j) \in M\}$ **do**
      if $n_{ij}.I \rightleftharpoons n_k.I \wedge n_{ij}.V \cap n_k.V = \emptyset$ **then** $M := M \cup \{(n_{ij},n_k)\}$
    for each $n_k \in \{n \in N\colon (n,n_i) \in D \vee (n,n_j) \in D\}$ **do**
      if $n_{ij}.I \not\rightleftharpoons n_k.I \wedge n_{ij}.V \cap n_k.V = \emptyset$ **then** $D := D \cup \{(n_{ij},n_k)\}$
  for each $n_i \in N$ **do**
    if $\bigcup n.V\colon\ n \in \{n \in N\colon (n,n_i) \in M \cup D\} \neq \mathcal{V}$ **then** /* remove n_i from G if it cannot be part of a solution */
      $M := M - \{(n_i,n_j)\colon n_j \in N\}$
      $D := D - \{(n_i,n_j)\colon n_j \in N\}$
      $N := N - \{n_i\}$
/* **Step 3**: find the maximum weighted clique */
$C :=$ **FindMaxWeightClique**(G)
if $\sum_{n \in C} w(n) < |\mathcal{V}| \cdot (|\mathcal{V}| - 1)$ **then return**($\emptyset$) /* no solution exists */
$\mathcal{I}_{sol} := \emptyset$
for each $n \in C$ **do**
  $I := n.I$
  assign 0 to don't care variables in I
  $\mathcal{I}_{sol} := \mathcal{I}_{sol} \cup \{I\}$
return($\mathcal{I}_{sol}$)

Figure 10:

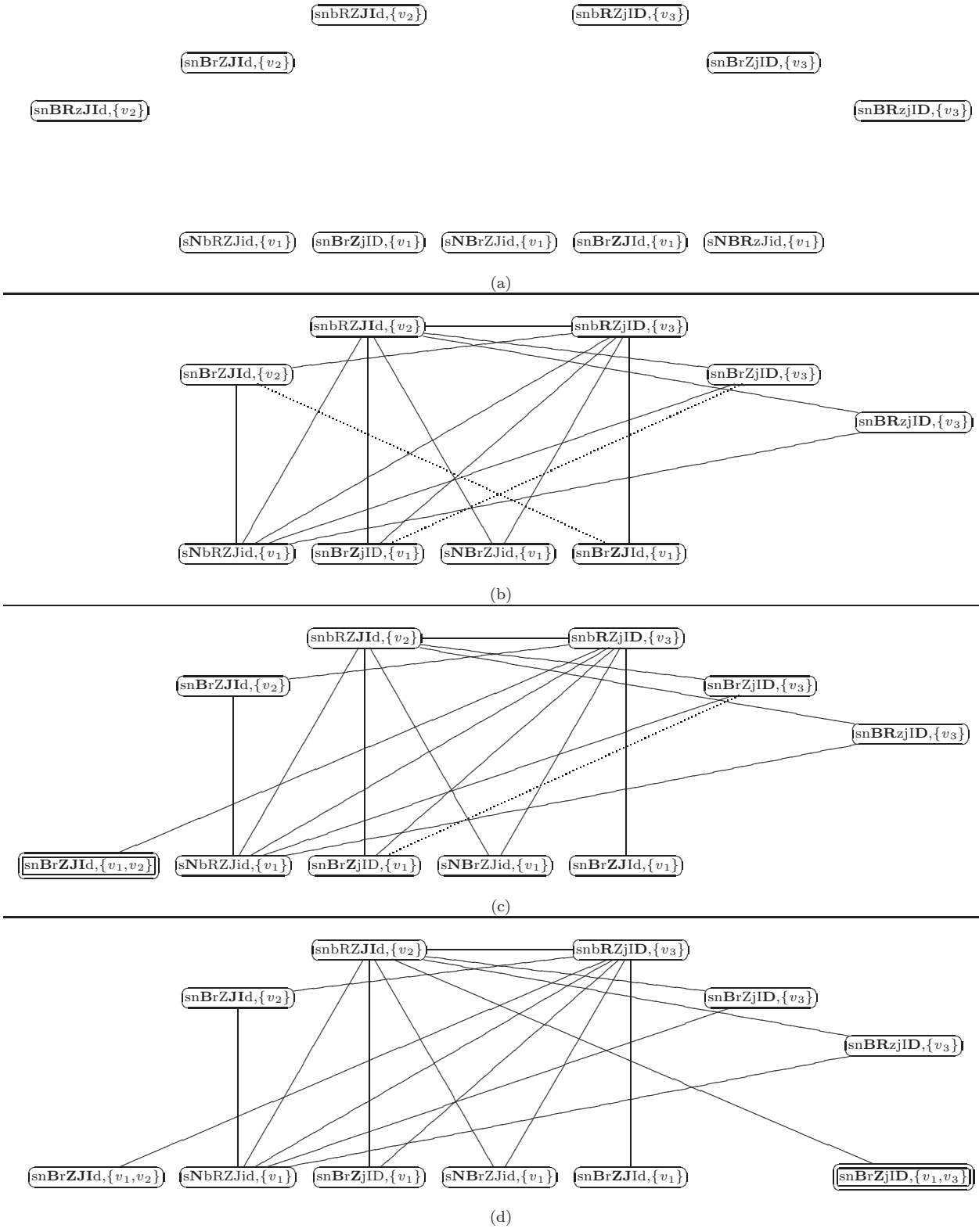

snbRZJId,{v2}
snbRZjID,{v3}
snBrZJId,{v2}
snBrZjID,{v3}
snBRzJId,{v2}
snBRzjID,{v3}
sNbRZJid,{v1}
snBrZjID,{v1}
sNBrZJid,{v1}
snBrZJId,{v1}
sNBRzJid,{v1}
(a)
snbRZJId,{v2}
snbRZjID,{v3}
snBrZJId,{v2}
snBrZjID,{v3}
snBRzjID,{v3}
sNbRZJid,{v1}
snBrZjID,{v1}
sNBrZJid,{v1}
snBrZJId,{v1}
(b)
snbRZJId,{v2}
snbRZjID,{v3}
snBrZJId,{v2}
snBrZjID,{v3}
snBRzjID,{v3}
snBrZJId,{v1,v2}
sNbRZJid,{v1}
snBrZjID,{v1}
sNBrZJid,{v1}
snBrZJId,{v1}
(c)
snbRZJId,{v2}
snbRZjID,{v3}
snBrZJId,{v2}
snBrZjID,{v3}
snBRzjID,{v3}
snBrZJId,{v1,v2}
sNbRZJid,{v1}
snBrZjID,{v1}
sNBrZJid,{v1}
snBrZJId,{v1}
snBrZjID,{v1,v3}
(d)

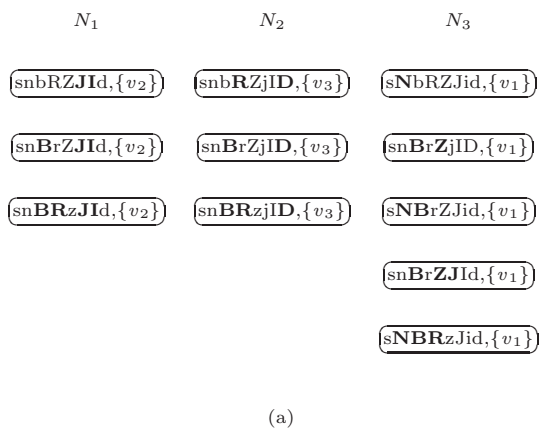
Figure 11:

INPUT: $\mathcal{B} = \{a_1,\ldots,a_n\}$, $\mathcal{C} = \{c_1,\ldots,c_m\}$, $\mathcal{V} = \{v_1,\ldots,v_k\}$ /* Boolean variables, confidentiality and visibility constraints */
OUTPUT: $\mathcal{I}_{sol} = \{I_1,\ldots,I_l\}$ /* locally minimal set of truth assignments */

MAIN
/* **Step 1**: extract the one-paths from the OBDDs representing constraints */
for each $v_i \in \mathcal{V}$ **do**
  let O_i be the OBDD representing $v_i \wedge \neg(c_1 \vee \ldots \vee c_m)$
  let $\mathcal{P}_{v_i}$ be the set of one-paths of O_i
  if $\mathcal{P}_{v_i} = \emptyset$ **then return**($\emptyset$) /* no solution exists */
/* **Step 2**: generate nodes representing the one-paths in O_i, $i = 1,\ldots,k$ */
$N := \emptyset$
for each $v \in \mathcal{V}$ **do** /* define a node for each one-path */
  for each $I \in \mathcal{P}_v$ **do**
    $n := \langle I, \{v\}\rangle$
    $N := N \cup \{n\}$
/* partition N depending on the visibility constraint that each node satisfies */
let $N_i = \{n \in N\colon n.V = v\}$, for all $v \in \mathcal{V}$, with $|N_i| > |N_j|$ iff $i > j$
for each $v \in \mathcal{V}$ **do**
  order nodes in N_i by decreasing number of don't care variables in $n.I$
/* **Step 3**: build a clique for the fragmentation graph */
$C :=$ **DefineClique**($\emptyset$,1)
/* **Step 4**: minimize the number of truth assignments in $\mathcal{I}_{sol}$ */
$\mathcal{I}_{sol} := \emptyset$
for each $n \in C$ **do** assign 0 to don't care variables in $n.I$
while $C \neq \emptyset$ **do**
  $n_i :=$ **ExtractNode**(C)
  $I_i := n_i.I$
  for each $n_j \in C$ **do**
    $I_j := n_j.I$
    if $I_i \vee I_j$ satisfies $\neg(c_1 \vee \ldots \vee c_m)$ **then**
      $I_i := I_i \vee I_j$
      $C := C - \{n_j\}$
  $\mathcal{I}_{sol} := \mathcal{I}_{sol} \cup \{I_i\}$
return($\mathcal{I}_{sol}$)

DEFINE_CLIQUE(C,i)
for j:=1,...,$|N_i|$ **do**
  satisfied := TRUE /* true if C includes a node that belongs to N_i */
  LinkableNodes := $\{n \in C\colon n.I \leftrightarrow n_j.I\}$ /* nodes in C representing truth assignments linkable to $n_j.I$ */
  $C' := C \backslash$ *LinkableNodes* /* remove from C the nodes that represent truth assignments linkable to $n_j.I$ */
  $n := n_j$
  while *satisfied* AND *LinkableNodes* $\neq \emptyset$ **do**
    $n_l :=$ **ExtractNode**(*LinkableNodes*) /* extract a node representing a truth assignment linkable to $n_j.I$ */
    if $n_l.I \rightleftharpoons n.I$ **then** $n := \langle n_l.I \odot n.I,\ n_l.V \cup n.V\rangle$ /* merge the two nodes */
    else *satisfied* := FALSE /* $n.I$ is linkable but not mergeable with $n_l.I$, then $C' \cup \{n\}$ cannot be a clique */
  if *satisfied* **then**
    $C' := C' \cup \{n\}$
    if $i = |\mathcal{V}|$ **then return**(C') /* C' represents a clique with weight at least $|\mathcal{V}| \cdot (|\mathcal{V}| - 1)$ */
    $C' :=$ **DefineClique**(C',$i+1$) /* recursive call */
    if $C' \neq \emptyset$ **then return**(C') /* C' represents a clique with weight at least $|\mathcal{V}| \cdot (|\mathcal{V}| - 1)$ */
return($\emptyset$)

Figure 12:



(a)

DefineClique(∅,1)
$n_1 \in N_1$:= ⟨snbRZ**JI**d,{v_2}⟩
LinkableNodes := ∅
n := ⟨snbRZ**JI**d,{v_2}⟩
C' := {⟨snbRZ**JI**d,{v_2}⟩}

DefineClique({⟨snbRZ**JI**d,{v_2}⟩},2)
$n_1 \in N_2$:= ⟨snb**R**ZjId, {v_3}⟩
LinkableNodes := ∅
n := ⟨snb**R**ZjId, {v_3}⟩
C' := {⟨snbRZ**JI**d,{v_2}⟩, ⟨snb**R**ZjId,{v_3}⟩}

DefineClique({⟨snbRZ**JI**d,{v_2}⟩, ⟨snb**R**ZjId,{v_3}⟩},3)
$n_1 \in N_3$:= ⟨s**N**bRZJid,{v_1}⟩
LinkableNodes := ∅
n := ⟨s**N**bRZJid,{v_1}⟩
C' := {⟨s**N**bRZJid,{v_1}⟩, ⟨snbRZ**JI**d,{v_2}⟩, ⟨snb**R**ZjId,{v_3}⟩}

(b)

Figure 13:

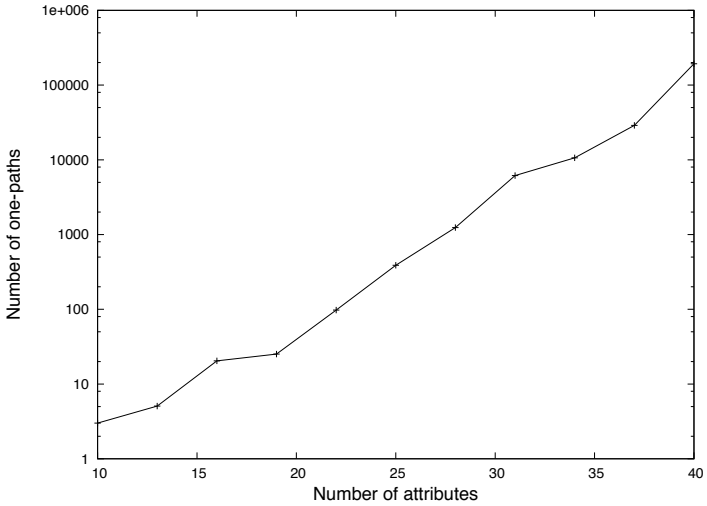

1e+006
100000
10000
1000
100
10
1
Number of one-paths
10
15
20
25
30
35
40
Number of attributes

Figure 14:

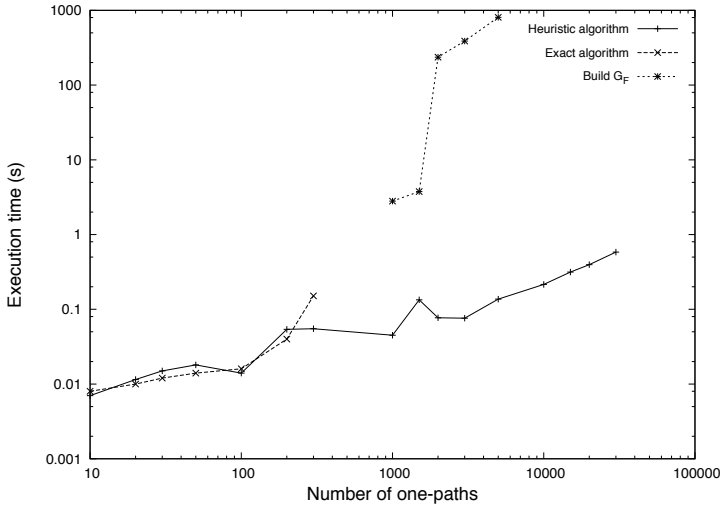

Execution time (s)
Number of one-paths
Heuristic algorithm
Exact algorithm
Build G_F
1000
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000

Figure 15:

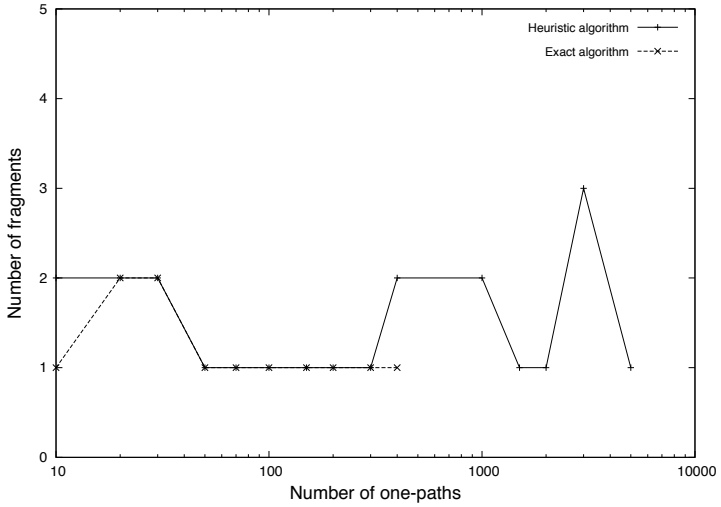

Heuristic algorithm
Exact algorithm
Number of fragments
Number of one-paths
0
1
2
3
4
5
10
100
1000
10000

Figure 16: