Privacy-preserving data splitting: a combinatorial approach

Oriol Farràs\textsuperscript{1} · Jordi Ribes-González\textsuperscript{1} · Sara Ricci\textsuperscript{2}

Received: 20 May 2020 / Revised: 20 January 2021 / Accepted: 23 April 2021 / Published online: 22 May 2021
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract
Privacy-preserving data splitting is a technique that aims to protect data privacy by storing different fragments of data in different locations. In this work we give a new combinatorial formulation to the data splitting problem. We see the data splitting problem as a purely combinatorial problem, in which we have to split data attributes into different fragments in a way that satisfies certain combinatorial properties derived from processing and privacy constraints. Using this formulation, we develop new combinatorial and algebraic techniques to obtain solutions to the data splitting problem. We present an algebraic method which builds an optimal data splitting solution by using Gröbner bases. Since this method is not efficient in general, we also develop a greedy algorithm for finding solutions that are not necessarily minimally sized.

Keywords Data splitting · Data privacy · Graph colorings

Mathematics Subject Classification 05C15 · 68R10 · 05C25

1 Introduction
The size of the data sets collected by companies and organizations has increased over time, and it is nowadays unfeasible for some data owners to locally store and process data because of the associated costs (such as hardware, energy and maintenance costs). The cloud offers a suitable alternative for data storage, as it provides large and highly scalable storage and computational resources at a low cost and with ubiquitous access. However, many data owners are reluctant to embrace the cloud computing technology because of security and privacy

Communicated by C. J. Colbourn.
concerns, which are mainly centered around the cloud service provider (CSP). The problem is not only that the CSPs may read, use or even sell the data outsourced by their customers, but also that they may suffer attacks or data breaches that can compromise data confidentiality.

In the past few years, several techniques have been proposed to enforce data privacy in cloud computing, with different levels of security (for surveys in this topic, see [17,33,36]). Cryptographic techniques include for instance searchable encryption, homomorphic encryption or secure multi-party computation. In this work we study data splitting, which is a non-cryptographic privacy-preserving technique.

Privacy-preserving data splitting aims at protecting data privacy by leveraging architectures with multiple CSPs. It minimizes the leakage of information by distributing the data among several CSPs, assuming that they do not communicate with each other. Similar problems have been studied in other areas such as data mining, data sanitization, file splitting and data merging.

In general, in data splitting data sets are structured in a tabular format, according to a set of attributes. Data are then composed by records, and each record holds up to one value per attribute. We can consider the attributes ‘Name’, ‘Age’, ‘Occupation’, and a record ‘John’, ‘21’, ‘Student’ that holds values for all attributes, for instance.

In this work, we deal with vertical data splitting, where fragments consist of data on all records but only on part of the attributes [2,22]. If some combinations of attributes constitute sensitive information, one can vertically split the data set and distribute it among cloud servers, so that no CSP holds any sensitive attribute combination. Assuming CSPs do not communicate with each other, this measure enforces privacy. An example of a sensitive pair of attributes in a medical data setting is passport number and disease, whereas blood pressure and disease constitute a generally safe pair.

After using data splitting, one may wish to securely compute over distributed data (see [40]). Some computations may require the servers to exchange information, but the stored data must still be kept private for privacy reasons. This is studied in the context of parallel processing for statistical computations [18,27,31] and of privacy-preserving data mining [6,12,23].

The results presented in this work focus on data splitting, but they can be applied to other related areas such as file splitting [1,16], data sanitization [32], and data merging [5].

1.1 Our results

In this work, we give a new combinatorial formulation to data splitting. In the considered problem, we force certain subsets of attributes to be stored in different CSPs, because their combination may reveal sensitive information. Moreover, we impose some subsets of attributes to be jointly stored in some CSP, for instance because we want to query on them efficiently or to compute statistics on them. Regarding privacy and security, the CSPs are not trusted and hence they are not given access to the entire data set. We thus assume that their access to the data set is restricted to a single chosen fragment.

Our treatment of privacy-preserving data splitting assumes the honest-but-curious security model, where the CSPs honestly fill their role in the protocol and do not share information with each other, but they try to infer information on the available data. In particular, each CSP may analyze its stored data and the received message flows in order to acquire additional information. Therefore, in our model, the leaked information is that which can be extracted from single data fragments. This model is common in the literature (see e.g. [7]).
In this setting, we aim at minimizing the number of CSPs needed to store a data set using data splitting. The rationale behind this goal is that all costs associated to cloud data storage (economic, computational, environmental, etc.) are directly proportional to the number of CSPs, as also is the attack surface. It could also be desirable to minimize the maximum number of CSPs an attribute is stored in (i.e., the maximum redundancy) or, even better, the total amount of outsourced information.

Our first step is to present a formulation of the problem and an approach to solve it. The data splitting problem is presented combinatorially, by specifying two families of sets of attributes $A$ and $B$. The first family $A$ represents privacy constraints, and specifies sets of attributes that must not be stored together for privacy reasons. The second family $B$ represents processing constraints, and specifies sets of attributes that must be stored together in the same fragment in order to speed up processing. We introduce the notion of $(A, B)$-covering, which directly translates to solutions of the data splitting problem.

Once the combinatorial problem of finding $(A, B)$-coverings is stated, we show that it can be solved by using algebraic techniques by seeing it as a hypergraph-coloring problem. We exhibit an algebraic reformulation of the data splitting problem by translating privacy and processing constraints to a system of simultaneous equations. Through the use of Gröbner bases, this reformulation allows the computation of optimally-sized data decompositions.

Since finding an optimal covering is NP-hard, obtaining optimal solutions is often unfeasible. We hence present a greedy algorithm that sacrifices optimality for efficiency, achieving a polynomial running time in the size of the problem. We also present an heuristic improvement that provides smaller decompositions when the family of constraints is sparse enough.

A performance analysis is carried out to evaluate the presented solutions. First, the execution time of our algebraic approach is analyzed in a medical data set context. Next, we report execution times of our greedy and heuristic algorithms over random graphs, and we estimate the incurred size overhead with respect to the optimal size for a small number of attributes. In our experiments, the execution time of our greedy algorithm was several orders of magnitude smaller than our algorithm that computes an optimal solution.

### 1.2 Related work

In this section, we review the literature on privacy-preserving techniques related to data splitting. For a survey on other techniques to preserve data privacy in cloud computing, see [17, 33, 36].

Previous works in data splitting describe privacy constraints as sensitive pairs of attributes. In [2], Aggarwal et al. find a decomposition of a given data set into two privacy-preserving vertical fragments. By using graph-coloring techniques, their decomposition optimizes the cost of querying the data set. If sensitive attribute pairs require more than two fragments to preserve data privacy, the use of encryption is contemplated. The decomposition problem in this work is NP-hard. Due to Guruswami et al. [24], if all 4-tuples of attributes are sensitive, it is NP-hard to find a privacy-preserving partition of attributes into two sets. Three different heuristics are presented as a workaround.

Ganapathy et al. [22] study the same scenario as [2], considering just two fragments and allowing for data encryption. They introduce three additional heuristics to reduce query costs, based on the greedy hill climbing approach. They also study the time complexity of the data splitting problem. If there are few constraints, the problem becomes polynomial time by solving the minimum cut problem. If the problem is equivalent to the hitting set problem, it is $O(\log(n))$-time. If each constraint set has size 2, time cost is an approximation.
factor of $O(\sqrt{n})$ by solving the minimum edge deletion bipartition problem. It becomes intractable for sets of sensitive attributes of size 3, through a reduction to the not-all-equal 3-SAT NP-complete problem.

Ciriani et al. propose in [9–11] different data splitting solutions that explore the use of encryption and communication between CSPs.

The data splitting problem studied in this work is related to other well known combinatorial optimization problems. We emphasize the connection with the job shop scheduling problem, which consists in assigning jobs to resources at particular times. Welsh and Powell [39] describe it as follows: let $J = \{J_i\}_{i=1}^N$ be a set of $N$ jobs, each of which takes an entire day to complete, and suppose that resources are unbounded. Let $M = \{m_{ij}\}_{i,j=1}^N$ be an incompatibility matrix, where $m_{ij}$ is zero or one depending on whether or not $J_i$ and $J_j$ can be carried out concurrently. The problem consists in scheduling the $n$ jobs using the minimum number of days according to the restrictions imposed by $M$. An efficient algorithm to solve this problem is presented in [39], and subsequent improvements can be found in [4,28]. See [8] for a survey on scheduling problems.

By interpreting jobs as attributes, days as data locations and the incompatibility matrix as a set of privacy constraints, we observe the equivalence between the problem posed in [39] and the data splitting problem. Through this same analogy, our setting extends to the following job scheduling problem: let $J = \{J_i\}_{i=1}^N$ be a set of $N$ jobs, and suppose that it takes an entire day to complete each job, and that resources are unbounded. Let $A \subseteq \mathcal{P}(J)$ be a family of sets of jobs that cannot be carried out all on the same day. Similarly, let $B \subseteq \mathcal{P}(J)$ be a family of sets of jobs that must be carried out concurrently. The problem consists in scheduling the $n$ jobs using the least number of days according to the restrictions imposed by $A$ and $B$.

1.3 Outline of the work

Section 2 presents privacy-preserving data splitting as a purely combinatorial problem. Section 3 states an algebraic formulation of this problem. Here, Gröbner bases are used to find the optimal (i.e., minimally-sized) solutions. Section 4 proposes a polynomial-time method that sacrifices optimality in favor of efficiency. In addition, an heuristic improvement is proposed. Section 5 presents the experimental results obtained by implementing these methods. First, we depict a comparison between the methods in a practical situation. Then, a performance analysis of the polynomial-time methods is carried out over random graphs.

2 A combinatorial approach

In this section we state the problem of privacy-preserving data splitting as a purely combinatorial problem. This problem consists in splitting a given data set in which some attributes are sensitive. As discussed above, this situation also covers problems of file splitting, data sanitization and data merging.

In the considered data splitting setting we start with a set $P$ of attributes. To avoid leaking sensitive information, some combinations of the attributes must not be stored by any individual server. We assume that individual attributes, when considered in isolation, are not sensitive (otherwise, encryption can be used). Moreover, we want some other sets of attributes to be stored in the same location, for example to perform statistical analysis computations such as contingency tables, correlations or principal component analysis of the attributes. We thus describe a data splitting problem using two families of attributes: $A \subseteq \mathcal{P}(P)$ is the family
of subsets of attributes that cannot be stored jointly in any single server, and $B \subseteq \mathcal{P}(P)$ is the family of subsets of attributes that must be stored together in some server.

Before proceeding, we introduce some notation. Let $P$ be the attribute set, and let $C \subseteq \mathcal{P}(P)$ any collection of attribute sets. For any attribute $v \in P$, we define $\deg_C(v)$ as the number of sets of $C$ containing attribute $v$, and we define the degree $\deg(C)$ of $C$ as the maximum of $\deg_C(v)$ for every $v \in P$. For any $B \subseteq P$, we also denote by $\deg_C(B)$ the number of sets $A \in C$ such that $A \cap B \neq \emptyset$. Given a set $A \subseteq P$, we define its closure $\text{cl}(A) = \{B \subseteq P : A \subseteq B\}$. We define $\min C$ and $\max C$ as follows. A subset $A \subseteq P$ is in $\min C$ if and only if $A \in C$ and there does not exist $B \in C$ with $B \subseteq A$. Analogously, a subset $A \subseteq P$ is in $\max C$ if and only if $A \in C$ and there does not exist $B \in C$ with $A \subseteq B$. That is, $\min C$ is the family of minimal subsets in $C$, and $\max C$ is the family of maximal subsets in $C$.

We say that $C$ is an antichain if $A \nsubseteq B$ for every $A, B \in C$. In this case, $C = \min C = \max C$.

Now, we can state the data splitting problem in terms of $(A, B)$-coverings, a notion first introduced in [21].

**Definition 2.1 (Data splitting problem)** Let $A, B \subseteq \mathcal{P}(P)$. An $(A, B)$-covering $C$ is a family of subsets of $P$ satisfying that

1. for every $A \in A$ and for every $C \in C$, $A \nsubseteq C$, and
2. for every $B \in B$ there exists $C \in C$ with $B \subseteq C$.

Let $A, B$ be the families of subsets defined by the data splitting restrictions described above, and let $C$ be an $(A, B)$-covering. Then $C$ defines a solution for data splitting by associating each fragment $i$ with a set $C_i \in C$. That is, we solve the data splitting problem by storing the data corresponding to attributes in $C_i$ at the $i$th location. Hence, we distribute data in $|C|$ different cloud servers or fragments, and any attribute $v \in P$ is stored at $\deg_C(v)$ different locations. Observe that, according to the definition above, for each $B \in B$ there is at least one fragment containing all attributes in $B$, and none of the fragments contain all attributes in $A$ for any $A \in A$. These are exactly the restrictions we have for data splitting.

Since $B \subseteq \mathcal{P}(P)$ is the family of subsets of attributes that must be stored together in some location, singletons $\{v\}$ can be appended to $B$ to ensure that all attributes $v \in P$ appear in some fragment, without altering the data splitting problem. In the rest of the article, we always assume that $\cup_{B \in B} B = P$.

Our work is focused on minimizing the size of the coverings, in order to reduce the attack surface and the costs associated to cloud data storage. The size of coverings corresponds to the number of fragments in data splitting. Therefore, we say that $C$ is an optimal $(A, B)$-covering if $|C|$ is minimal among all $(A, B)$-coverings.

Next, in Examples 2.2 and 2.3, we show two instances of the data splitting problem. The first states a general non-trivial example for any attribute set, and the second examines the previously defined objects and concepts in the light of a medical data use case.

**Example 2.2** Let $P$ be a nonempty attribute set, let $B \subseteq \mathcal{P}(P)$ be a nonempty antichain, and let $A \in B$. Then $C = \{P \setminus \{i\} : i \in A\}$ is an $((A), B \setminus \{A\})$-covering.

**Example 2.3** In Table 1 we depict a set of six attributes from a medical data use case, and we attach numerical identifiers from 0 to 5 for brevity. Since the attributes patient ID and address completely identify patients, they must be stored in an encrypted form and excluded from the data splitting problem.

A sensitive combination of attributes is $\{0, 2, 3\}$ as shown in [35], where a 1990 federal census reports that in Dekalb, Illinois there were only two resident black women. We can also

 Springer
regard \{0, 1, 3\}, \{0, 1, 4\} and \{1, 2, 3\} as sensitive. Moreover, some attributes may need to be stored in the same fragment to perform statistical analysis, for instance \{1, 2, 5\}, \{1, 3, 5\} and \{0, 2, 5\}. Hence, we can consider the following attribute set \(P\) and the privacy and processing constraints \(A\) and \(B\):

\[
P = \{0, 1, 2, 3, 4, 5\},
\]

\[
A = \{\{0, 2, 3\}, \{0, 1, 2\}, \{0, 1, 4\}, \{1, 2, 3\}\},
\]

\[
B = \{\{1, 2, 5\}, \{1, 3, 5\}, \{0, 2, 5\}, \{4\}\}.
\]

Note that a singleton \{4\} appears in \(B\), so that attribute 4 is guaranteed to be outsourced. A solution to the associated data splitting problem would be the \((A, B)\)-covering \(C\) of size 3 defined by:

\[
C = \{\{0, 2, 4, 5\}, \{1, 2, 5\}, \{1, 3, 5\}\}.
\]

Hence, data should be split into three fragments, with attributes \{0, 2, 4, 5\}, \{1, 2, 5\}, and \{1, 3, 5\}, respectively.

Next we present some technical results about coverings. The main results of this section are Proposition 2.6, which characterizes the existence of coverings, and Proposition 2.8, which justifies the search for \((A, B)\)-coverings in the case that \(A\) and \(B\) are antichains. In addition, we present a theoretical lower bound on the size of \((A, B)\)-coverings in Proposition 2.11.

**Lemma 2.4** Let \(A, B, C \subseteq \mathcal{P}(P)\). Then \(C\) is an \((A, B)\)-covering if and only if

1. \(\text{cl}(A) \cap C = \emptyset\) for every \(A \in A\), and
2. \(\text{cl}(B) \cap C \neq \emptyset\) for every \(B \in B\).

The following result stands as a direct consequence of this lemma, which was presented in [21].

**Lemma 2.5** (Lemma 6.4 in [21]) Let \(A, A', B, B' \subseteq \mathcal{P}(P)\) with \(A' \subseteq A\) and \(B' \subseteq B\). Every \((A, B)\)-covering is also an \((A', B')\)-covering.

The next proposition characterizes the pairs of subsets \((A, B)\) that admit \((A, B)\)-coverings, and was also presented in [21].

**Proposition 2.6** (Lemma 6.3 in [21]) Let \(A, B \subseteq \mathcal{P}(P)\). There exists an \((A, B)\)-covering if and only if

\[
A \nsubseteq B \text{ for every } A \in A \text{ and } B \in B.
\]  

\[1\]

**Lemma 2.7** Let \(A, A', B, B' \subseteq \mathcal{P}(P)\). If

1. for every \(A' \in A'\) there exists \(A \in A\) with \(A \subseteq A'\), and
– for every \(B' \in \mathcal{B}'\) there exists \(B \in \mathcal{B}\) with \(B' \subseteq B\),

then any \((A, B)\)-covering is also a \((A', B')\)-covering.

**Proposition 2.8** Let \(A, B, C \subseteq \mathcal{P}(P)\). Then \(C\) is an \((A, B)\)-covering if and only if it is a (min \(A\), max \(B\))-covering.

According to the previous proposition, we can always restrict the search for \((A, B)\)-coverings to the case where \(A\) and \(B\) are antichains. Further, as a consequence of Lemma 2.7 we can define a partial hierarchy among the pairs of antichains \((A, B)\). For example, every \(((\{1, 2\}, \{3, 4, 5\}))\)-covering is also an \(((\{1, 2, 3\}, \{3, 4\}))\)-covering.

We next give two general methods to build \((A, B)\)-coverings.

**Lemma 2.9** Let \(A, B \subseteq \mathcal{P}(P)\) be families of subsets satisfying condition (1), and denote \(A = \{A_1, \ldots, A_r\}\). Then

1. \(B\) is an \((A, B)\)-covering.
2. \(\{P \setminus \{a_1, \ldots, a_r\} : a_i \in A_i \text{ for } i = 1, \ldots, r\}\) is an \((A, B)\)-covering.

**Proof** The first item is trivial. For the second one, let

\[C = \{P \setminus \{a_1, \ldots, a_r\} : a_i \in A_i \text{ for } i = 1, \ldots, r\}.\]

We see \(C\) is an \((A, B)\)-covering by verifying the two conditions of Definition 2.1:

1. For every \(A_i \in A\) and \(X = P \setminus \{a_1, \ldots, a_r\} \in C\), we have \(a_i \in A_i\) and \(a_i \notin X\). Hence, \(A \not\subseteq C\).
2. Due to condition (1), we have that \(A \not\subseteq B\) for every \(A \in A\) and \(B \in B\). Therefore, given \(B \in B\), for every \(A_i \in A\) there exists an \(a_i \in A_i\) with \(a_i \notin B\). Hence, \(B\) is contained in \(P \setminus \{a_1, \ldots, a_r\} \in C\).

\(\Box\)

A direct consequence of the last proposition is the following upper bound on the size of \((A, B)\)-coverings. Further upper bounds are shown later in Lemma 2.13, Theorem 4.1, Proposition 4.4 and Remark 4.5.

**Proposition 2.10** Let \(A, B \subseteq \mathcal{P}(P)\) be families of subsets satisfying condition (1). Then there exists an \((A, B)\)-covering \(C\) of size

\[|C| \leq \min \left\{ |B|, \prod_{A \in A} |A| \right\}.\]

To conclude this section, we describe a theoretical lower bound on the size of \((A, B)\)-coverings. Note that, in the case \(B = \binom{P}{1} = \{\{i\} : i \in P\}\) and \(A \subseteq \binom{P}{2}\), the problem of finding an \((A, B)\)-covering is equivalent to the graph coloring problem on the graph \(G = (P, A)\). In this case, the size of an optimal \((A, B)\)-covering is just the chromatic number \(\chi(G)\).

Existing general lower bounds on the chromatic number include the clique number, the minimum degree bound, Hoffman’s bound, the vector chromatic number, Lovász number and the fractional chromatic number. Our proposed bound generalizes the minimum degree bound \(\chi(G) \geq \frac{n}{n - \delta(G)}\), where \(n\) is the number of vertices and \(\delta(G)\) is the minimum degree of \(G\), to the case of \((A, B)\)-coverings.
Proposition 2.11 Let $A, B \subseteq \mathcal{P}(P)$ be families of subsets satisfying condition (1), and let $C$ be an $(A, B)$-covering. Then

$$|C| \geq \frac{|B|}{|B| - \max_{A \in A} \min_{a \in A} \deg_B(a)}.$$  

Proof Let $C$ be an $(A, B)$-covering. Given $C \in C$, denote $B_C = B \cap \mathcal{P}(C)$. By the properties of $(A, B)$-coverings, we have that $B \subseteq \bigcup_{C \in C} \mathcal{P}(C)$, and this implies that $B = \bigcup_{C \in C} B_C$. Hence $|B| \leq \sum_{C \in C} |B_C| \leq |C| \cdot \max_{C \in C} |B_C|$, and so $|C| \geq |B|/\max_{C \in C} |B_C|$. We now proceed to upper bound $\max_{C \in C} |B_C|$. Since for every $B \in B_C$ we have $B \subseteq C$, we see that $\bigcup_{B \in B_C} B \subseteq C$. Therefore, by the definition of $(A, B)$-coverings we have that $A \not\subseteq \bigcup_{B \in B_C} B$ for every $A \in A$. Denote by $\alpha(A, B)$ the size of the largest subfamily of $B$ with this property, i.e.

$$\alpha(A, B) = \max\{|B'| : B' \subseteq B \text{ and } A \not\subseteq \bigcup_{B \in B_C} B \text{ for every } A \in A\}.$$  

By the preceding observation, we get that $\max_{C \in C} |B_C| \leq \alpha(A, B)$. By definition of $\alpha(A, B)$, given any set $A \in A$ we have $\alpha(A, B) \leq \alpha(|A|, B)$, and so $\alpha(A, B) \leq \min_{A \in A} \alpha(|A|, B)$. Now, given a set $A \in A$, a family $B' \subseteq B$ satisfies $A \not\subseteq \bigcup_{B \in B_C} B$ if and only if there exists an element $a \in A$ such that $a \notin \bigcup_{B \in B_C} B$. Therefore $\alpha(|A|, B) = \max_{a \in A} \alpha(|a|, B)$. Finally, by definition we see that $\alpha(|a|, B) = |B| - \deg_B(a)$. By composing the obtained results, we see that $\max_{C \in C} |B_C| \leq \min_{A \in A} \max_{a \in A} (|B| - \deg_B(a)) = |B| - \max_{A \in A} \min_{a \in A} \deg_B(a)$. The proposition follows by applying the first obtained inequality. \qed

2.1 Multi-colorings of hypergraphs

In this section we introduce multi-colorings of hypergraphs, and we reveal their relation to $(A, B)$-coverings. The notion of multi-colorings ultimately allows us to reformulate the considered combinatorial problem in an algebraic way.

Let $\mathcal{H} = (P, E)$ be a hypergraph. A coloring of $\mathcal{H}$ with $k$ colors is a mapping $\mu : P \rightarrow \{1, \ldots, k\}$ such that, for every $A \in E$, there exists $u, v \in A$ with $\mu(u) \neq \mu(v)$.

Next, we describe the connection between colorings and coverings. Let $\mu$ be a coloring of the hypergraph $\mathcal{H} = (P, A)$ with $k$ colors. Consider the family of subsets of elements in $P$ that have the same color under $\mu$. That is, consider a family of subsets $C = \{C_1, \ldots, C_k\}$ that is a partition of $P$ satisfying that $\mu(j) = i$ for every $j \in C_i$ and for every color $i$.

Now consider the pair $(A, B)$ with $B = \binom{P}{k}$. Observe that $C$ satisfies condition 1 in Definition 2.1 because, if a subset $A$ is in $A$, then it cannot be monochromatic. Since each element in $P$ has a color, condition 2 is also satisfied. In order to construct $(A, B)$-coverings for other families of subsets $B \subseteq \mathcal{P}(P)$, we can use sequences of colorings. To define appropriately these constructions, we use hypergraph multi-colorings.

For any integer $k > 0$, we define a multi-coloring of the hypergraph $\mathcal{H} = (P, E)$ of $k$ colors as a mapping $\mu : P \rightarrow \{0, 1\}^k$ with the following property: for every $A \in E$ and for every $1 \leq j \leq k$, there exists $i \in A$ for which the $j$th coordinate of $\mu(i)$ is 0, namely $\mu(i)_j = 0$. If we associate each $1 \leq j \leq k$ with a different color, a multi-coloring of $\mathcal{H}$ is a mapping that maps each element in $P$ to a set of at most $k$ colors. This mapping must satisfy that for every subset in $A \in E$ and for each color, at least one element in $A$ does not have this color. A sequence of colorings of a hypergraph defines a multi-coloring. A multi-coloring defines a family of subsets in a natural way, and vice-versa. More concretely, a multi-coloring $\mu$ induces the family of subsets $C = \{C_i\}_{1 \leq i \leq k} \subseteq \mathcal{P}(P)$, where $C_i$ are the subsets of elements of $P$ mapped to the same color $i$ by $\mu$. O. Farràs et al.
Lemma 2.12 Let \( A, B, C \subseteq \mathcal{P}(P) \), with \(|C| = k\). Then \( C \) is an \((A, B)\)-covering if and only if \( C \) defines a multi-coloring \( \mu \) of \( \mathcal{H} = (P, A) \) of \( k \) colors with the property that, for every \( B \in B \), there exists \( 1 \leq j \leq k \) for which the \( j \)-coordinate of \( \mu(i) \) is 1 for every \( i \in B \).

**Proof** Let \( C = \{C_1, \ldots, C_k\} \) be an \((A, B)\)-covering. We define a multi-coloring \( \mu \) of \( k \) colors as follows. For every \( i \in P \) and \( 1 \leq j \leq k \), set \( \mu(i)_j = 1 \) if and only if \( i \) is in \( C_j \). Let \( B \in B \), and let \( C_j \) be a subset in \( C \) with \( B \subseteq C_j \). Then \( \mu(i)_j = 1 \) for every \( i \in B \).

Taking into account the comments detailed above, it is straightforward to prove that the converse implication also holds. \( \square \)

We use the connection between coverings and multi-colorings to find general constructions of coverings, and upper bounds on their size. Beimel, Farràs, and Mintz built efficient secret sharing schemes for very dense graphs [3]. One of the techniques developed in that work is connected to our work. In [21], this result was described in terms of \( (A, B) \)-coverings and upper bounds on their size. Beimel, Farràs, and Mintz built efficient secret sharing schemes for very dense graphs [3]. One of the techniques developed in that work is connected to our work. In [21], this result was described in terms of \( (A, B) \)-coverings and upper bounds on their size.

Lemma 2.13 Let \( A, B \subseteq \mathcal{P}(P) \) be families of subsets satisfying condition (1). Let \( d \) denote the degree of \( A \), and suppose that sets in \( A \) and \( B \) have size at most \( k \). Then there exists an \((A, B)\)-covering of degree \( 2(2kd)^{k-1} \ln n \) and size \( 2(2kd)^k \ln n \).

2.2 Optimal covers

Both the optimization problem of determining the size of an optimal \((A, B)\)-covering and the search problem of finding an optimal \((A, B)\)-covering are NP-hard. This is so because taking \( B = \binom{P}{k} \) and \( A \subseteq \binom{P}{d} \) transforms these problems to the corresponding graph coloring problems, and so there is a trivial reduction from the known NP-hard graph coloring problems to the \((A, B)\)-covering problems. Next, we see NP-completeness of the decisional problem.

**Proposition 2.14** The problem of deciding whether an \((A, B)\)-covering of size \( t \) exists is NP-complete.

**Proof** Let \( A, B, t \) define an instance of the problem where the answer is affirmative. Given an \((A, B)\)-covering \( C \) of size \( t \), a checking algorithm first verifies that \( C \) has size \( t \), that every \( B \in B \) is contained in some \( X \in C \), and that no \( A \in A \) is contained in any \( X \in C \). The running time of this checking algorithm is at most quadratic in the size of the problem input, and thus the given problem is in NP.

Now, note that the case \( B = \binom{P}{t} \) and \( A \subseteq \binom{P}{2} \) is equivalent to the graph coloring problem. Therefore the given problem is NP-complete. \( \square \)

3 Algebraic formulation of the problem

In this section we present an algebraic formulation of the combinatorial problem presented in the previous section. The purpose of this formulation is to exploit algebraic techniques to find solutions to the data splitting problem for a fixed number of fragments.

It is not unusual that graph-coloring problems are encoded into polynomial ideals [14, 15, 26, 29]. In this case, the existence of a coloring is reduced to the solvability of a related system of polynomial equations over the algebraic closure of the field. Furthermore, the weak
Hilbert’s Nullstellensatz theorem allows to obtain a certificate that a system of polynomial has no solutions [13], and, consequently, that a graph is not colorable. The focus of this section is the use of polynomial ideals and Gröbner bases to provide an optimal multi-coloring with the property described in Lemma 2.12. Recall that obtaining a multi-coloring $\mu$ is equivalent to finding an $(A, B)$-covering.

Let $H = (P, A)$ be a hypergraph and let $\mu$ be a multi-coloring of $H = (P, A)$ of $k$ colors with the property that, for every $B \in B$, there exists $1 \leq j \leq k$ for which the $j$-coordinate of $\mu(i)$ is 1 for every $i \in B$. The multi-coloring $\mu$ can be seen as assignment of $[0, 1]$ values to a set of $kn$ variables $x_{i,j}$, where $n = |P|$, $1 \leq j \leq k$, $1 \leq i \leq n$ and $x_{i,j} = 1$ if and only if $\mu(i) = j$. In other words, we assign $k$ variables $x_{i,j}$ to each vertex $i$ in $P$ in such a way that,

$$x_{i,j} = \begin{cases} 1 & \text{if vertex } i \text{ takes color } j \text{ by applying } \mu \\ 0 & \text{otherwise.} \end{cases}$$

Encoding $\mu$ to a polynomial ring allows an algebraic formulation of the multi-coloring problem. Since we focus on optimal multi-colorings, the number of colors is fixed to a designated minimal $k$. Furthermore, each variable $x_{i,j}$ takes values in $\{0, 1\}$, which allows working over $\mathbb{F}_2$.

Therefore, given $X = (x_{i,j})_{1 \leq j \leq k, 1 \leq i \leq n}$, we define the $k$-coloring ideal $I_k(H, B) \subset \mathbb{F}_2[X]$ as the ideal generated by $G_1 \cup G_2$, where

- $G_1 = \{\prod_{i \in A} x_{i,j} : 1 \leq j \leq k, A \in \mathcal{A}\}$
  - all vertices belonging to an edge set $A \in \mathcal{A}$ cannot have the same color,
- $G_2 = \{\prod_{i=1}^n (\prod_{j \in B} x_{i,j}) - 1 : B \in B\}$
  - there exists a color $j$ such that all the vertices in $B$ are colored with $j$.

Theorem 3.1 proves that finding a solution of $I_k(H, B)$ is equivalent to obtaining a suitable multi-coloring $\mu$.

**Theorem 3.1** Let $\mu : P \rightarrow \{0, 1\}^k$ be a multi-coloring of $H = (P, A)$, and assume that $\cup_{B \in B} B = P$. Then $\mu$ defines an $(A, B)$-covering (in the sense of Lemma 2.12) if and only if $I_k(H, B)$ has a common root in $\mathbb{F}_2[X]$. In other words, the multi-coloring $\mu$ of $H$ does not define an $(A, B)$-covering if and only if $I_k(H, B) = (1)$.

**Proof** According to the definition above, the map $\mu$ is a multi-coloring of $H$ defining an $(A, B)$-covering if and only if the following two conditions hold:

- for every $A \in \mathcal{A}$ and for every $1 \leq j \leq k$, there exists $i \in A$ for which $\mu(i) = j$ is 0,
- for every $B \in B$, there exists $1 \leq j \leq k$ for which the $j$-coordinate of $\mu(i)$ is 1 for every $i \in B$.

We next verify that the sets of polynomials $G_1$ and $G_2$ defined above respectively encode these two conditions. In other words, we see that the first (resp. second) condition holds for $\mu$ if and only if every polynomial in $G_1$ (resp. $G_2$) vanishes, under the assignment $x_{i,j} = \mu(i)$ for all $i \in P$ and $1 \leq j \leq k$.

- $G_1$: Suppose that, for all $A \in \mathcal{A}$ and for every color $j$, we have $\prod_{i \in A} x_{i,j} = 0$. This means that there exists $i \in A$ such that $x_{i,j} = 0$, which is equivalent to saying that there exists $i \in A$ such that $\mu(i)_j = 0$.
- $G_2$: Suppose that, for all $B \in B$, we have $\prod_{j=1}^k (\prod_{i \in B} x_{i,j} - 1) = 0$. This means that there exists $j$ such that $\prod_{i \in B} x_{i,j} = 1$, which amounts to say that there exists $j$ such that $x_{i,j} = 1$ for all $i \in B$. This is equivalent to saying that there exists $j$ such that $\mu(i)_j = 1$ for all $i \in B$. 

© Springer
Now, multi-colorings \( \mu \) satisfying these two conditions correspond to simultaneous roots of polynomials of \( G_1 \) and \( G_2 \). Since we defined \( I_k(\mathcal{H}, \mathcal{B}) = (G_1 \cup G_2) \), these correspond to common roots of \( I_k(\mathcal{H}, \mathcal{B}) \), as claimed.

Now that the data splitting problem is stated as an algebraic problem, a technique based on Gröbner bases can be used to solve it. A Gröbner basis is a generating set of an ideal \( I \) in a polynomial ring, which allows to determine if any polynomial belongs to \( I \) or not [13]. In other words, it allows to determine the variety associated to \( I \), i.e. the solutions of \( I \). It is proven that it is possible to associate a Gröbner basis to every polynomial ideal [13]. Informally, Gröbner basis computation can be viewed as a generalization of Gaussian elimination for non-linear equations.

In our case, Gröbner bases can be used to find the solutions of \( I_k(\mathcal{H}, \mathcal{B}) \). Once the Gröbner basis of the \( k \)-coloring ideal is obtained, the associated variety can be easily computed. For systems of polynomial equations of degree \( d \) in \( n \) variables, the complexity of computing Gröbner bases has been proven to be \( dO(n^2) \) when the number of solutions is finite [20]. In general, its complexity is \( 2^{O(n)} \). Since \( I_k(\mathcal{H}, \mathcal{B}) \) belongs to \( \mathbb{F}_2[X] \), then it has a finite number of solutions, and so a bound for the worst-case complexity of computing Gröbner basis in this case is \( 2^{O(k^2n^2)} \). We must note that this method can be slower than an exhaustive search, since the number of multi-colorings is \( 2^{kn} \), and checking if multi-coloring defines an \((\mathcal{A}, \mathcal{B})\)-covering can be done in time \( 2^{O(n)} \).

As stated before, it is possible to derive a certificate for the unsolvability of a system of polynomials from the weak Hilbert’s Nullstellensatz. In our case, this allows to prove that it is not possible to find a multi-coloring \( \mu \) with a designated number of colors \( k \).

**Theorem 3.2** (Weak Hilbert’s Nullstellensatz [13]) Let the polynomials \( f_1, \ldots, f_m \in \mathbb{K}[x_1, \ldots, x_n] \). Then there are no solutions to the system \( \{f_i = 0\}_{i=1}^m \) in the algebraic closure of \( \mathbb{K} \) if and only if there exist \( \alpha_1, \ldots, \alpha_m \in \mathbb{K}[x_1, \ldots, x_n] \) such that
\[
\alpha_1 f_1 + \cdots + \alpha_m f_m = 1
\]  
(2)

The set \( \{\alpha_i\} \) is called a Nullstellensatz certificate. The complexity of computing such a certificate depends on the degree of \( \{\alpha_i\}_{i=1}^m \), which is defined as the maximum degree of any \( \alpha_i \) as polynomials. Fast results have been achieved when the Nullstellensatz certificate has small constant degree [30].

According to Theorem 3.2, by using methods to compute a Nullstellensatz certificate it is possible to find out whether or not \( I_k(\mathcal{H}, \mathcal{B}) \) has any solutions. If we start by fixing a tentative number of colors \( k \), the data splitting problem can be solved by applying a Nullstellensatz certificate method. When there exists a Nullstellensatz certificate, \( I_k(\mathcal{H}, \mathcal{B}) \) does not have common root. The complexity of the Nullstellensatz certificate and the Gröbner basis methods grow with the number of variables which, in our case, grows with the number of colors. Therefore, it is convenient to start with as few colors as possible and increase them sequentially until a certificate of feasibility is found or until a Gröbner basis is computed. Notice that finding the optimal \( k \) is a NP-complete problem, because it has complexity equivalent to solving the system of equations.

**Example 3.3** Given \( \mathcal{A} = \{\{1, 2, 3\}\} \) and \( \mathcal{B} = \{\{1, 4\}, \{2, 4\}, \{3\}\} \), we want to compute an \((\mathcal{A}, \mathcal{B})\)-covering. As explained above, the problem can be encoded to polynomial ideals. We assign \( k \) variables to each attribute, where \( k \) is the number of colors needed to obtain an optimal covering. For example, we can encode vertex 1 to \( x_{1,1} \) and \( x_{1,2} \) when 2 colors are
considered. The variable \(x_{i,j}\) equals 1 if and only if vertex \(i\) takes color \(j\), and it equals 0 otherwise.

The ideal \(I_2(\mathcal{H}, \mathcal{B})\) is generated by the polynomials in \(G_1\) and \(G_2\), where

\[
\begin{align*}
G_1 &= \{x_{1,1}x_{2,1}x_{3,1}, \ x_{1,2}x_{2,2}x_{3,2}\}, \\
G_2 &= \{x_{2,1}x_{4,1}x_{2,2}x_{4,2} + x_{2,1}x_{4,2} + x_{2,2}x_{4,1} + 1, \ x_{1,1}x_{4,1}x_{1,2}x_{4,2} + x_{1,1}x_{4,1} + x_{1,2}x_{4,2} + 1, \ x_{3,1}x_{3,2} + x_{3,1} + x_{3,2} + 1\}.
\end{align*}
\]

Note that there does not exist an \((\mathcal{A}, \mathcal{B})\)-covering of size one, because we have \(\{1, 2, 3\} \subseteq \bigcup_{B \in \mathcal{B}} B\). Since we can compute the Gröbner basis of \(I_2(\mathcal{H}, \mathcal{B})\), there exist \((\mathcal{A}, \mathcal{B})\)-coverings of size two, and we obtain the optimal \((\mathcal{A}, \mathcal{B})\)-coverings:

\[
\begin{align*}
&\{x_{1,1}, x_{2,1} + 1, x_{3,1}, x_{4,1} + 1, x_{1,1} + 1, x_{2,2}, x_{3,2} + 1, x_{4,2} + 1\}, \\
&\{x_{1,1}, x_{2,1} + 1, x_{3,1} + 1, x_{4,1} + 1, x_{1,2} + 1, x_{2,2}, x_{3,2}, x_{4,2} + 1\}, \\
&\{x_{1,1}, x_{3,1} + 1, x_{1,2} + 1, x_{2,2} + 1, x_{3,2}, x_{4,2} + 1\}.
\end{align*}
\]

In the last solution, the variables \(x_{3,1}\) and \(x_{4,2}\) are missing, which means that they can take both 0 and 1 values. Therefore, the solutions can be re-written as the following coverings:

\[
\begin{align*}
&\{2, 4\}, \{1, 3, 4\}, \\
&\{2, 3, 4\}, \{1, 4\}, \\
&\{3\}, \{1, 2, 4\}, \{2, 3\}, \{1, 2, 4\}, \{3, 4\}, \{1, 2, 4\}, \{2, 3, 4\}, \{1, 2, 4\}.
\end{align*}
\]

In order to obtain the number of colors \(k\) which allows to compute an optimal covering, a small number of colors \(k = 2\) is fixed. If the Gröbner basis method applied to \(I_2(\mathcal{H}, \mathcal{B})\) outputs the ideal (1), then \(k\) is incremented. This process is repeated until an ideal different from (1) is obtained. The resulting number of colors \(k\) is the smallest one for which we have an \((\mathcal{A}, \mathcal{B})\)-covering, and thus it is optimal.

4 A greedy algorithm

In this section we aim for an efficient method to build \((\mathcal{A}, \mathcal{B})\)-coverings and for upper bounds on the size of an optimal \((\mathcal{A}, \mathcal{B})\)-covering.

As seen above, the problem of finding an optimal \((\mathcal{A}, \mathcal{B})\)-covering is NP-hard. Hence, as expected, the labour involved in finding an optimal \((\mathcal{A}, \mathcal{B})\)-covering can render methods inefficient when solving practical data splitting instances. Our strategy to circumvent this consists in sacrificing optimality to achieve a polynomial-time algorithm.

The problem of finding upper bounds on the size of an optimal \((\mathcal{A}, \mathcal{B})\)-covering \(\mathcal{C}\) has been studied in the literature for the following particular cases:

- In the case \(\mathcal{B} = (\binom{P}{l})\) and \(\mathcal{A} \subseteq \binom{P}{2}\), the problem of finding an \((\mathcal{A}, \mathcal{B})\)-covering is easily seen to be equivalent to the graph coloring problem. Then \(|\mathcal{C}|\) is the chromatic number of the graph \(G = (P, A)\). For instance, the greedy coloring bound gives \(|\mathcal{C}| \leq \deg(\mathcal{A}) + 1\).
- The case \(\mathcal{B} \subseteq \binom{P}{2} \cup (\binom{P}{3})\) and \(\mathcal{A} = (\binom{P}{k}) \setminus \mathcal{B}\) has been studied as the clique covering and the clique partition numbers. Hall [25] and Erdős et al. [19] showed that \(|\mathcal{C}| \leq \lfloor |P|/2/4 \rfloor\).
- In the case \(\mathcal{B} = (\binom{P}{l})\) and \(\mathcal{A} \subseteq \bigcup_{i : k \leq n} (\binom{P}{i})\) for \(1 \leq l \leq k < n\), the problem of finding a \(k\)-uniform \((\mathcal{A}, \mathcal{B})\)-covering is equivalent to finding an \((n, k, l)\)-covering design. In this case, by Spencer [34], \(|\mathcal{C}| \leq \binom{n}{l}/(l)! \left(1 + \ln \left(\binom{n}{l}\right)\right)\).
In the following, we first describe a general upper bound on the size of an optimal \((A, B)\)-covering. We then deduce from this bound an algorithm to build \((A, B)\)-coverings, and we analyze its worst-time complexity. Finally, we introduce an heuristic improvement and a theoretical bound that improve the prior results for sparse enough \(B\).

4.1 Our construction

The next result generalizes the greedy coloring bound to \((A, B)\)-coverings. It gives a general bound of the size of an optimal \((A, B)\)-covering in terms of the degrees of \(A, B\).

**Theorem 4.1** Let \(A, B \subseteq \mathcal{P}(P)\) be families of subsets satisfying condition (1), and suppose that sets in \(B\) have size at most \(k\). Then there exists an \((A, B)\)-covering \(C\) of size

\[
|C| \leq k \deg(A) \deg(B) + 1
\]

such that \(\deg_C(v) \leq \deg_B(v)\) for every \(v \in P\).

**Proof** We prove this by induction on \(|B|\). If \(|B| = 1\), then \(C = B\) satisfies the theorem. Now let \(s > 1\) be an integer and assume that the proposition holds for every pair \(A, B \subseteq \mathcal{P}(P)\) of families of subsets satisfying \(|B| < s\) and the proposition hypotheses. Let \(A, B' \subseteq \mathcal{P}(P)\) be a pair of families of subsets satisfying \(|B'| = s\) and the proposition hypotheses, and express \(B' = B \cup \{B\}\) for some fixed \(B \in B'\) and \(|B| = s - 1\). Then, by induction hypothesis, there exists an \((A, B'\)\)-covering \(C\) with \(|C| \leq k \deg(A) \deg(B') + 1\) and such that every \(v \in P\) is contained in at most \(\deg_B(v)\) elements of \(C\). We now build an \((A, B')\)-covering \(C'\) from \(C\), in such a way that \(|C'| \leq k \deg(A) \deg(B') + 1\) and that every \(v \in P\) is contained in at most \(\deg_B(v)\) elements of \(C\).

If \(B\) is contained in some \(X \in C\), then \(C' = C\) satisfies the theorem. Otherwise, let

\[
\mathcal{F}_B = \{X \in C : \text{there exists } A \in A \text{ with } A \subseteq X \cup B\}.
\]

Note that the condition \(A \subseteq X \cup B\) is equivalent to \(A \cap B \neq \emptyset\) and \(A \setminus B \subseteq X\). Since there are at most \(k \deg(A)\) elements \(A \in A\) with \(A \cap B \neq \emptyset\), and since every set of the form \(A \setminus B\) can be contained in at most \(\deg(B)\) elements of \(C\) (because \(\deg_C(v) \leq \deg_B(v)\) for every \(v \in P\) by hypothesis), we have that \(|\mathcal{F}_B| \leq k \deg(A) \deg(B)\).

Therefore, either there exists an element \(X \in C \setminus \mathcal{F}_B\), in which case we take

\[
C' = (X \cup B) \cup (C \setminus \{X\})
\]

or \(C = \mathcal{F}_B\), in which case \(|C| \leq k \deg(A) \deg(B)\) and we let \(C' = C \cup \{B\}\). \(\square\)

Algorithm 1 is a greedy algorithm to compute an \((A, B)\)-covering that follows directly from the constructive proof of the previous theorem. This algorithm simply builds an ordered \((A, B)\)-covering \(C\) by iterating through \(B\). Every set in \(B\) is merged with the first available element of \(C\), i.e., with the first element \(X \in C\) such that no \(A \in A\) is contained in \(X \cup B\). If no such \(X\) exists, then \(B\) is added as a singleton in \(C\). Note that this algorithm is a generalization of the usual greedy coloring algorithm.

To see the worst-time complexity of Algorithm 1, note that the first loop (line 2) is repeated \(|B|\) times. At step \(i\), the first \(i \in \mathcal{E}\) statement (line 3) requires checking at most \(i - 1\) inclusions, and the second \(i \in \mathcal{E}\) statement (line 4) requires checking at most \((i - 1)|A|\) inclusions. Therefore, Algorithm 1 runs in time \(O(|A| \cdot |B|^2)\).
Algorithm 1: Construction

Input: \( A = \{A_1, \ldots, A_r\}, B = \{B_1, \ldots, B_s\} \)
1. Initialize \( C \leftarrow \emptyset \)
2. for \( i = 1, \ldots, s \) do
3. \( \) if \( B_i \) is not contained in any \( X \in C \) then
4. \( \) if there exists \( X \in C \) such that \( A \not\subseteq X \cup B_i \) for every \( A \in A \) then
5. \( \) \( C \leftarrow \{X \cup B_i\} \cup (C \setminus \{X\}) \)
6. else
7. \( \) \( C \leftarrow C \cup \{B_i\} \)
8. end
9. end
10. end

Output: The \((A, B)\)-covering \( C \)

Remark 4.2 The total amount of outsourced information \( \sum_{X \in C} |X| \) of the \((A, B)\)-covering \( C \) output by Algorithm 1 is, by construction, at most \( \sum_{B \in B} |B| \). By Lemma 2.9, it is also at most \((n - |A|) \prod_{A \in A} |A| \). To reduce the total amount of outsourced information, line 4 of Algorithm 1 can be modified to find a set \( X \in C \) that, on top of satisfying the condition therein, minimizes \( |X \cup B_i| \). This change does not significantly impact worst-time complexity.

4.2 An heuristic improvement

In order to motivate the heuristic procedure proposed later, we must first note that the output of Algorithm 1 depends strongly on the particular order in which elements of \( B \) are taken in the first loop. In particular, we see in the following proposition that there always exists an optimal ordering of the elements of \( B \). Of course, since the problem of finding an optimal \((A, B)\)-covering is NP-hard and an optimal ordering can be verified in polynomial time, finding an optimal ordering in our case is NP-complete.

Proposition 4.3 Let \( A, B \subseteq P(P) \) be families of subsets satisfying condition (1). Then there exists an ordering of \( B \) such that Algorithm 1 outputs an optimal \((A, B)\)-covering.

Proof Let \( C = \{X_1, \ldots, X_t\} \) be an optimal \((A, B)\)-covering. For every \( j \in \{1, \ldots, t\} \), define \( S_j \) as the family of elements of \( B \) that are contained in \( X_j \) and that are not contained in any \( X_i \) for \( l < j \),
\[
S_j = \{B \in B : B \subseteq X_j \text{ and } B \not\subseteq X_l \text{ for every } l < j\}.
\]
We first prove that \( \{S_j\}_{j=1}^{t} \) defines a partition of \( B \).

Indeed, \( S_j \neq \emptyset \), because otherwise \( C \setminus \{X_j\} \) would be an \((A, B)\)-covering smaller than \( C \). Also, \( S_i \cap S_j = \emptyset \) for every \( i, j \). Otherwise, if \( B \in S_i \cap S_j \) with \( i < j \), then \( B \in S_i \) implies \( B \subseteq X_i \), and \( B \in S_j \) implies \( B \not\subseteq X_i \), a contradiction.

Finally, since every \( B \in B \) is contained in some element of \( C \) by the definition of \((A, B)\)-covering, we can take \( X_j \in C \) with minimal index \( j \) among those that contain \( B \). Then \( B \in S_j \) by definition, and therefore \( B = \bigcup_{j=1}^{t} S_i \).

Now, define a new ordering of \( B \) by taking the sets in \( S_j \) sequentially. That is, if \( S_j = \{B_{j,1}, \ldots, B_{j,k_i}\} \), define
\[
B' = \{B_{1,1}, \ldots, B_{1,k_1}, \ldots, B_{t,1}, \ldots, B_{t,k_t}\}.
\]

Consider the behavior of Algorithm 1 on input \( A, B' \). It is easy to see that, when the algorithm finishes processing the sets in \( S_j \), the local variable \( C \) holds at most \( j \) elements.
Therefore, since the covering $C$ is optimal, Algorithm 1 outputs an optimal $(A, B)$-covering. □

Following this result, we propose an heuristic procedure to build an ordering of $B$, inspired in the Welsh-Powell algorithm [39]. This procedure can be deduced from the proof of the following proposition, which effectively reduces the upper bound given in Theorem 4.1 for sparse enough $B$.

**Proposition 4.4** Assume the hypotheses of Theorem 4.1. Then there exists an $(A, B)$-covering $C$ of size

$$|C| \leq \max_i \min \{ \deg_A(B_i) \deg(B) + 1, i \}$$

such that $\deg_C(v) \leq \deg_B(v)$ for every $v \in P$.

**Proof** First reorder $B$ so that $B = \{B_1, \ldots, B_s\}$ satisfies

$$\deg_A(B_1) \geq \deg_A(B_2) \geq \cdots \geq \deg_A(B_s).$$

Now, consider the behavior of Algorithm 1 on input $A$ and the reordered $B$. At step $i$, Algorithm 1 processes $B_i$. In this step, there can be at most $\deg_A(B_i) \deg(B)$ sets $X \in C$ such that $B_i$ does not satisfy the condition in line 4 (that is, such that there exists $A \in A$ with $A \subseteq X \cup B_i$). To see this, note that by definition at most $\deg_A(B_i)$ elements $A \in A$ intersect $B_i$, and that each set of the form $A \setminus B_i$ can be contained in at most $\min \{ \deg(B), |C| \} \leq \deg(B)$ elements of $C$.

Now, at step $i$ the number of elements $X \in C$ checked in the condition of line 4 is at most $\min \{ \deg_A(B_i) \deg(B), |C| \}$. Since at step $i$ the family $C$ has at most $i - 1$ sets, at most $\min \{ \deg_A(B_i) \deg(B), i - 1 \}$ elements of $C$ are checked until either line 5 or 7 is executed, and line 7 can add an additional element to $C$. Hence, by iterating through all elements of $B$, the size of the final output can be at most $\max_i \min \{ \deg_A(B_i) \deg(B), i - 1 \} + 1$. □

We now state our heuristic improvement of Algorithm 1, which follows directly from the previous proof.

| Algorithm 2: Heuristic Improvement of Algorithm 1 |
|-----------------------------------------------|
| **Input:** $A = \{A_1, \ldots, A_r\}, B = \{B_1, \ldots, B_s\}$ |
| 1 for $B \in B$ do |
| 2 \quad Compute $\deg_A(B) = |\{A \in A : A \cap B \neq \emptyset\}|$ |
| 3 end |
| Sort $B$ so that $B = \{B'_1, \ldots, B'_s\}$ satisfies $\deg_A(B'_1) \geq \deg_A(B'_2) \geq \cdots \geq \deg_A(B'_s)$ |
| **Output:** The output of Algorithm 1 on input $A, B$ |

To see the worst-time complexity of Algorithm 2, note that the computation of each quantity $\deg_A(B)$ requires $O(|A|)$ time. Adding in the sorting time, we see that our heuristic takes $O(|B| \cdot (|A| + \log(|B|)))$ time. In turn, adding this to the cost of Algorithm 1 does not alter the total $O(|A| \cdot |B|^2)$ worst-time complexity.

**Remark 4.5** In fact, the previous proof indicates a slightly better bound. For an integer $\alpha$ define the function $f_\alpha$ by

$$f_\alpha(\beta) = \begin{cases} 
\beta & \text{if } \alpha < \beta \\
\beta + 1 & \text{otherwise.}
\end{cases}$$
Then the bound on the previous proposition can be replaced by

\[ |C| \leq f_{\deg_A(B_1)}(\deg(B)) \cdot f_{\deg_A(B_2)}(\deg(B)) \cdot \cdots \cdot (f_{\deg_A(B)}(\deg(B))(1) \cdots) + 1. \]

To wrap up this section, we comment on a possible alternative algorithm to compute an optimal \((A, B)\)-covering, based on the work by Tsukiyama et al. [38]. Denote \( B = \{B_1, \ldots, B_d\} \). We can consider a search tree with root node \( \{B_1\} \), where the children of each node at level \( d \) are \((A, \{B_1, \ldots, B_{d+1}\})\)-coverings computed by adding in the set \( B_{d+1} \) in all possible ways allowed by the privacy restrictions \( A \) (roughly as done in Algorithm 1). While this allows to find an optimal \((A, B)\)-covering \( C \), the worst-case running time can be seen to be \( O(|A| \cdot |C|^{2|B|-3}) \).

5 Experimental results

This section details the experimental results obtained by implementing the proposed methods in the Sage Mathematical Software System [37], version 7.4. First, a comparison between Algorithm 2 and the Gröbner basis method on a practical setting is shown. Then, a performance analysis of Algorithms 1 and 2 is carried out over random graphs. The reported experiments have been conducted on an AMD Ryzen 7 1700X Eight-core 3.4 GHz processor, with 32 GB of RAM, in Sage [37] and under Ubuntu 4.10.0-37. All experiments have been carried out without parallelization. All CPU running times have been collected using the function \( \text{cputime}(\text{subprocesses}=\text{True}) \) in Sage.

5.1 Medical data example

Medical data applications tend to be extremely storage and functionality-demanding, and thus it is often unfeasible for the data holder to locally store and manage the data. Therefore, medical data provides a good candidate for a privacy-preserving data splitting use case. We take up Example 2.3, where attributes from a medical data set are presented (see Table 1), and a data splitting problem is determined by

\[
\begin{align*}
P &= \{0, 1, 2, 3, 4, 5\}, \\
A &= \{\{0, 2, 3\}, \{0, 1, 2\}, \{0, 1, 4\}, \{1, 2, 3\}\}, \\
B &= \{\{1, 2, 5\}, \{1, 3, 5\}, \{0, 2, 5\}, \{4\}\}.
\end{align*}
\]

Both the Gröbner basis method (implemented as \( \text{buchberger2}() \) in Sage) and Algorithm 2 can be used to find an \((A, B)\)-covering \( C \). In Table 2 we document their executions, and we also compute the bounds on \(|C|\) given in Proposition 2.11, Remark 4.5, Proposition 4.4, and Theorem 4.1. We do this in the following cases, the first of which is the medical data splitting example above:

- **Case 1**: \( A = \{\{0, 2, 3\}, \{0, 1, 2\}, \{0, 1, 4\}, \{1, 2, 3\}\}, \)
  \( B = \{\{1, 2, 5\}, \{1, 3, 5\}, \{0, 2, 5\}, \{4\}\} \).
- **Case 2**: \( A = \{\{0, 2, 3\}, \{0, 1, 2\}, \{0, 1, 4\}\}, \)
  \( B = \{\{1, 2, 5\}, \{1, 3, 5\}, \{0, 2, 5\}, \{4\}\} \).
- **Case 3**: \( A = \{\{0, 4, 5\}, \{1, 2, 3\}, \{8, 9\}\}, \)
  \( B = \{\{1, 2, 4\}, \{4, 5, 8\}, \{0, 9\}, \{2, 3, 8\}\} \).
Table 2: The first two sets of columns give a comparison between the Gröbner basis method and Algorithm 2 on several hypergraphs.

| Case | Gröbner basis | Algorithm 2 | Bounds on | | | |
|------|---------------|-------------|------------|-------------|-------------|-------------|---------------|-------------|
|      | Sol. | | Opt. | | | | Proposition 2.11 | Remark 4.5 | Proposition 4.4 | Theorem 4.1 |
|      | | | | | | | | | | |
| 1    | 15   | 3   | 6.68 s | Yes | 3 | 3.84 ms | 1.3 | 4 | 4 | 28 |
| 2    | 3    | 2   | 1.12 ms | Yes | 2 | 1.19 ms | 1.3 | 4 | 4 | 28 |
| 3    | 1    | 316 ms | No | 3 | 1.41 ms | 1.3 | 4 | 4 | 7 |
| 4    | 204  | 3   | 16 h 45 m | Yes | 3 | 4.88 ms | 1.4 | 6 | 6 | 28 |
| 5    | 2    | 2   | 1.87 s | Yes | 2 | 2.19 ms | 1.3 | 7 | 7 | 13 |

The third set of columns states the lower bound given in Proposition 2.11 and the upper bounds of Remark 4.5, Proposition 4.4 and Theorem 4.1, for the same hypergraphs.

- "v.": number of vertices of the selected hypergraph
- "sol.": number of optimal coverings
- |C|: size of the (A, B)-coverings computed by the respective algorithm, which is optimal in the case of the Gröbner basis method
- "time": the time needed by the related method to compute the solution
- "opt.": whether or not the solution of Algorithm 2 is optimal
Case 4: \( A = \{\{1, 3\}, \{1, 6, 8\}, \{3, 4\}, \{7, 9\}, \{0, 3, 6\}\} \),
\( B = \{\{0, 2, 3\}, \{0, 1, 2\}, \{3, 6\}, \{4, 6\}, \{7, 8\}, \{0, 7\}, \{9\}\} \).

Case 5: \( A = \{\{0, 2\}, \{1, 6, 8\}, \{3, 4\}, \{7, 9\}, \{0, 3\}\} \),
\( B = \{\{0, 1\}, \{1, 2, 8\}, \{3, 5\}, \{4, 6\}, \{7, 8\}, \{0, 4\}, \{2, 3\}, \{9\}\} \).

The first column of Table 2 shows the results of applying the Gröbner basis method and Algorithm 2 to the medical data set. Algorithm 2 has been chosen for the tests above instead of Algorithm 1 due to efficiency reasons. Both the Gröbner basis method and Algorithm 2 provide solutions in the considered case, but with a considerable time difference. Observe that Algorithm 2 does not always compute an optimal solution. Two of the optimal solutions computed by the Gröbner basis method are depicted in Fig. 1.

The time needed to obtain a solution is strictly related to the degree of \( A \) and \( B \). Other parameters which affect the running time are the number of vertices and the size of the optimal covering (see Sect. 3 for more details). However, while having the same number of vertices, the needed time may vary greatly.

5.2 Performance analysis over random graphs

We now give some performance measures to analyze the presented results. In this performance analysis we restrict to the graph case, and thus we take \( A \subseteq \binom{\mathbb{P}_2}{2} \) and \( B \subseteq \binom{\mathbb{P}_2}{3} \cup \binom{\mathbb{P}_1}{1} \) to be disjoint families of subsets. We classify the test cases according to two parameters: the number \( n \) of vertices and the sum of densities \( \rho = \rho_A + \rho_B \) of \( A \) and \( B \). For each test case, we randomly generate graphs by choosing every single edge of the complete graph \( K_n \) with probability \( \rho \), and we then throw a uniform random coin for each chosen edge to determine if it belongs either to \( A \) or to \( B \). Next, we add the necessary singletons to \( B \) so that \( \cup_{B \in B} B = \mathbb{P} \). Finally, we randomly shuffle \( A \) and \( B \) and we apply the algorithm to test. Hence, in the considered experiments both \( A \) and \( B \) are generated with density \( \rho_A = \rho_B = \rho/2 \).

We divide the experiments into two subsections. In the first, we analyze the algorithms given in Sect. 4, examining their running time and the size of the computed \((A, B)\)-coverings. In the second, we study the bounds given in Proposition 2.11, Theorem 4.1, Proposition 4.4, and Remark 4.5.
### 5.2.1 Analysis of Algorithms 1 and 2

In Table 3 we analyze the time performance of Algorithms 1 and 2. For each of the considered $n$ and $\rho$, the reported CPU running times have been averaged over 1000 independent random experiments.

Observe that average running times increase both in the number of attributes and the density, and range between milliseconds and 20 minutes for the considered parameters.

Next, in Table 4 we compile evidence on the size of the result output by Algorithm 2 over the size of the result output by Algorithm 1. For every considered $n$ and $\rho$, we randomly instantiate $10^5$ different $A$ and $B$ as stated above, and for each of them we compute the sizes $s_{alg}$ and $s_{heur}$ of the $(A, B)$-coverings given by Algorithm 1 and by Algorithm 2, respectively. Then, we compute the decrease as the percentage $(100(s_{alg} - s_{heur})/s_{alg})\%$ in size offered by the heuristic. The reported percentage decreases have been averaged over at least $10^5$ independent random experiments.

Following the same procedure, in Table 5 we compile evidence on the size increase of the covering given by Algorithm 2 in relation to the size of an optimal covering. The reported percentage decreases have been averaged over at least $10^4$ independent random experiments.

In Table 4, we observe that our heuristic Algorithm 2 generally improves the greedy Algorithm 1 for a small number of attributes, and that this improvement grows in the number of attributes and is larger for medium densities. In addition, in Table 5 we confirm that our heuristic algorithm generally provides near-to-optimal sized decompositions for a small number of attributes, and that much better results are achieved for small densities. In the case $n = 5$ and $\rho = 1$, our algorithms always provide optimal coverings.

---

**Table 3** Time performance analysis (in seconds)

| $\rho$ | Algorithm 1 | Algorithm 2 |
|--------|-------------|-------------|
| $n$    | 0.1 | 0.4 | 0.7 | 1.0 | 0.1 | 0.4 | 0.7 | 1.0 |
| 10     | 0.003 | 0.01 | 0.029 | 0.066 | 0.005 | 0.02 | 0.057 | 0.13 |
| 20     | 0.018 | 0.18 | 0.74 | 2 | 0.036 | 0.35 | 1.26 | 3 |
| 30     | 0.065 | 1.15 | 5.1 | 14 | 0.14 | 2 | 7.8 | 20 |
| 40     | 0.19 | 4.4 | 21 | 62 | 0.38 | 7.1 | 29 | 78 |
| 50     | 0.44 | 13 | 63 | 190 | 0.89 | 19 | 82 | 228 |
| 60     | 0.94 | 30 | 156 | 479 | 1.9 | 44 | 193 | 553 |
| 70     | 1.8 | 64 | 339 | 1054 | 3.5 | 89 | 404 | 1177 |

**Table 4** Average percent size reduction given by Algorithm 2 from the size given by Algorithm 1

| $n$ | $\rho = 0.1$ | $\rho = 0.3$ | $\rho = 0.5$ | $\rho = 0.7$ | $\rho = 0.9$ | $\rho = 1.0$ |
|-----|--------------|--------------|--------------|--------------|--------------|--------------|
| 5   | $1.2 \times 10^{-2}$ | 0.26 | 0.51 | 0.36 | $9.5 \times 10^{-2}$ | 0 |
| 6   | $7.3 \times 10^{-2}$ | 1.0 | 1.2 | 0.75 | 0.32 | $8.7 \times 10^{-2}$ |
| 7   | 0.22 | 2.0 | 1.9 | 1.2 | 0.64 | 0.34 |
| 8   | 0.51 | 3.0 | 2.5 | 1.7 | 1.1 | 0.73 |
| 9   | 0.92 | 3.7 | 3.0 | 2.3 | 1.6 | 1.2 |
| 10  | 1.6 | 4.3 | 3.6 | 2.8 | 2.1 | 1.6 |
Table 5  Average percent size increase given by Algorithm 2 with respect to the optimal size

| n   | \( \rho = 0.1 \) | \( \rho = 0.3 \) | \( \rho = 0.5 \) | \( \rho = 0.7 \) | \( \rho = 0.9 \) | \( \rho = 1.0 \) |
|-----|----------------|----------------|----------------|----------------|----------------|----------------|
| 5   | \( 3.3 \times 10^{-5} \) | \( 2.6 \times 10^{-4} \) | \( 3.6 \times 10^{-3} \) | \( 1.3 \times 10^{-3} \) | \( 1.3 \times 10^{-5} \) | 0              |
| 6   | \( 1.3 \times 10^{-3} \) | \( 2.5 \times 10^{-2} \) | 0.10           | \( 7.5 \times 10^{-2} \) | \( 3.2 \times 10^{-2} \) | \( 2.7 \times 10^{-2} \) |
| 7   | \( 9.3 \times 10^{-3} \) | 0.33           | 0.33           | 0.17           | 0.16           | 0.15           |
| 8   | \( 6.7 \times 10^{-2} \) | 0.52           | 0.41           | 0.48           | 0.31           | 0.19           |
| 9   | 0.13           | 0.84           | 1.0            | 0.89           | 0.84           | 0.37           |
| 10  | 0.33           | 1.5            | 1.6            | 1.4            | 1.2            | 0.90           |

Table 6  Average percent size difference given by the lower bound of Proposition 2.11 and of the upper bound of Proposition 4.4 with respect to the optimal size

| \( \rho \) | Proposition 2.11 | Proposition 4.4 |
|-----------|-----------------|-----------------|
| \( n = 5 \) | \( 0.1 \) | 8.8 | 22 | 32 | 41 | 50 | 54 | 11 | 30 | 49 | 58 | 62 | 62 |
| \( n = 6 \) | 15 | 32 | 44 | 54 | 62 | 65 | 15 | 51 | 78 | 86 | 86 | 85 |
| \( n = 7 \) | 22 | 40 | 53 | 62 | 70 | 73 | 21 | 80 | 108 | 112 | 107 | 102 |
| \( n = 8 \) | 28 | 47 | 59 | 68 | 76 | 79 | 29 | 112 | 135 | 133 | 125 | 120 |
| \( n = 9 \) | 33 | 52 | 64 | 73 | 80 | 83 | 41 | 143 | 160 | 156 | 143 | 134 |
| \( n = 10 \) | 37 | 56 | 68 | 77 | 83 | 86 | 56 | 171 | 186 | 176 | 161 | 150 |

5.2.2 Analysis on the bounds on the size of \((\mathcal{A}, \mathcal{B})\)-coverings

In Table 6, we analyze the sizes of the lower bound presented in Proposition 2.11 and the upper bound of Proposition 4.4. As in the previous cases, for every considered \( n \) and \( \rho \) we randomly instantiate \( 10^5 \) different \( \mathcal{A} \) and \( \mathcal{B} \), and for each of them we compute the value \( b \) of the bound and the optimal size \( c \) of an \((\mathcal{A}, \mathcal{B})\)-covering. Then, we compute their difference as the percentage \( \left( 100 \cdot \frac{|b - c|}{c} \right) \% \). These percentages have been averaged over at least \( 10^5 \) independent random experiments.

The average-case analysis hints that the bounds are best for smaller number of attributes \( n \) and lower density \( \rho \). The bound of Theorem 4.1 is, on average, between 1.3 \((n = 5, \rho = 0.1)\) and 9 \((n = 10, \rho = 1.0)\) times larger than the optimal size in the studied cases. Our lower bound of Proposition 2.11 improves all our upper bounds in terms of the computed average difference with the optimal size. The upper bound of Remark 4.5 is slightly better than that of Proposition 4.4, with the values of Table 6 being between 0.3 \((n = 10, \rho = 1)\) and 33 \((n = 10, \rho = 0.3)\) units smaller.

In the analyzed cases, all the upper bounds give values that are substantially larger than the optimal size, which comes into contrast with the near-to-optimal sizes of the output of Algorithm 2 shown in Table 5.

Acknowledgements  This article is supported by the Ministry of the Interior of the Czech Republic (grant VJ01030002), by the Government of Catalonia (grant 2017 SGR 705), by the European Commission (project H2020-871042 “SoBigData++”), by the Spanish Government (project RTI2018-095094-B-C21, “CONSENT”), and by the DRAC project, which is co-financed by the European Union Regional Development
Fund within the framework of the ERDF Operational Program of Catalonia 2014-2020 with a grant of 50% of total cost eligible.

References

1. Abu-Libdeh H., Princehouse L., Weatherspoon H.: RACS: a case for cloud storage diversity. In: Proceedings of the 1st ACM symposium on Cloud computing. ACM, New York (2010).
2. Aggarwal G., Bawa M., Ganesan P., Garcia-Molina H., Kenthalapadi K., Motwani R., Srivastava U., Thomas D., Xu Y.: Two can keep a secret: a distributed architecture for secure database services. In: Conference on Innovative Data Systems Research, vol. 2005, pp. 186–199 (2005).
3. Beimel A., Farràs O., Mintz Y.: Secret sharing schemes for very dense graphs. J. Cryptol. 29(2), 336–362 (2016).
4. Brélaz D.: New methods to color the vertices of a graph. Commun. ACM 22(4), 251–256 (1979).
5. Brinkman R., Maubach S., Jonker W.: A lucky dip as a secure data store. In: Proceedings of Workshop on Information and System Security (2006).
6. Calviño A., Ricci S., Domingo-Ferrer J.: Privacy-preserving distributed statistical computation to a semi-honest multi-cloud. In: 2015 IEEE Conference on Communications and Network Security (CNS), pp. 506–514 (2015). https://doi.org/10.1109/CNS.2015.7346863.
7. Cao N., Wang C., Li M., Ren K., Lou W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Trans. Parallel Distrib. Syst. 25(1), 222–233 (2014).
8. Carter M.W.: A rvey of practical applications of examination timetabling algorithms. Oper. Res. 34(2), 193–202 (1986).
9. Ciriani V., De Capitani di Vimercati S., Foresti S., Jajodia S., Paraboschi S., Samarati P.: Fragmentation and encryption to enforce privacy in data storage. In Computer Security – ESORICS 2007. Lecture Notes in Computer Science, vol. 4734, pp. 171–186. Springer, Heidelberg (2007).
10. Ciriani V., De Capitani di Vimercati S., Foresti S., Jajodia S., Paraboschi S., Samarati P.: Combining fragmentation and encryption to protect privacy in data storage. ACM Trans. Inf. Syst. Secur. 13(3), 221–233 (2010).
11. Ciriani V., Capitani De, di Vimercati S., Foresti S., Jajodia S., Paraboschi S., Samarati P.: Selective data outsourcing for enforcing privacy. J. Comput. Secur. 19(3), 531–566 (2011).
12. Clifton C., Kantarcioğlu M., Vaidya J., Lin X., Zhu M.Y.: Tools for privacy preserving distributed data mining. ACM SIGKDD Explor. Newsl. 4(2), 28–34 (2002).
13. Cox D., Little J., O’shea D.: Ideals, Varieties, and Algorithms. Springer, New York (1992).
14. De Loera J.A.: Gröbner bases and graph colorings. Beitr. Algebra Geom. 36(1), 89–96 (1995).
15. De Loera J.A., Margulies S., Permeptnner M., Riedl E., Rolnick D., Spencer G., Stasi D., Swenson J.: Graph-coloring ideals: Nullstellensatz certificates, Gröbner bases for chordal graphs, and hardness of Gröbner bases. In: Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation, pp. 133–140 (2015).
16. Dev H., Sen T., Basak B., Ali M.E.: An approach to protect the privacy of cloud data from data mining based attacks. In: High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion. IEEE (2012).
17. Domingo-Ferrer J., Farràs O., Ribes-González J., Sánchez D.: Privacy-preserving cloud computing on sensitive data: a survey of methods, products and challenges. Comput. Commun. 140–141, 38–60 (2019).
18. Du W., Yungshiang S.H., Shigang C.: Privacy-preserving multivariate statistical analysis: linear regression and classification. In: Proceedings of the 2004 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics (2004).
19. Erdős P., Goodman A.W., Pósa L.: The representation of a graph by set intersections. Can. J. Math. 18, 106–112 (1966).
20. Faugere J.C., Gianni P., Lazard D., Mora T.: Efficient computation of zero-dimensional Gröbner bases by change of ordering. J. Symb. Comput. 16(4), 329–44 (1993).
21. Farràs O., Ribes-González J., Ricci S.: Local bounds for the optimal information ratio of secret sharing schemes. Des. Codes Cryptogr. 87(6), 1323–1344 (2019).
22. Ganapathy V., Thomas D., Feder T., Garcia-Molina H., Motwani R.: Distributing data for secure database services. Trans. Data Privacy 5(1), 253–272 (2012).
23. Goethals B., Laur S., Lipmaa H., Mielikäinen T.: On private scalar product computation for privacy-preserving data mining. In: International Conference on Information Security and Cryptology. Springer, Berlin (2004).
24. Guruswami V., Hastad J., Sudan M.: Hardness of approximate hypergraph coloring. SIAM J. Comput. 31(6), 1663–1686 (2002).
25. Hall Jr. M.: A problem in partitions. Bull. Am. Math. Soc. 47, 801–807 (1941).
26. Hillar C.J., Windfeldt T.: Algebraic characterization of uniquely vertex colorable graphs. J. Combin. Theory Ser. B 98(2), 400–414 (2008).
27. Kantarcioglu M.: A survey of privacy-preserving methods across horizontally partitioned data. In: Privacy-Preserving Data Mining, pp. 313–335. Springer, Boston (2008).
28. Leighton F.T.: A graph coloring algorithm for large scheduling problems. J. Res. Natl Bur. Stand. 84, 489–506 (1979).
29. Levy-dit-Vehel, F., Marinari, M.G., Perret, L., Traverso, C.: A survey on Polly Cracker systems. In: Gröbner Bases, Coding, and Cryptography, pp. 285–305. Springer, Berlin (2009).
30. Loera J.A., Lee J., Margulies S., Onn S.: Expressing combinatorial problems by systems of polynomial equations and Hilbert’s Nullstellensatz. Combin. Probab. Comput. 18(4), 551–82 (2009).
31. Ricci S., Domingo-Ferrer J., Sánchez D.: Privacy-preserving cloud-based statistical analyses on sensitive categorical data. In: Modeling Decisions for Artificial Intelligence. Springer, Cham (2016).
32. Sánchez D., Batet M.: Privacy-preserving data outsourcing in the cloud via semantic data splitting. Comput. Commun. 110, 187–201 (2017).
33. Shan Z., Ren K., Blanton M., Wang C.: Practical secure computation outsourcing: a survey. ACM Comput. Surv. 51(2), Article No. 31 (2018).
34. Spencer J.: Ten lectures on the probabilistic method. SIAM Regional Conference Series in Applied Mathematics, vol. 52. SIAM, Philadelphia (1987).
35. Sweeney L.: Simple demographics often identify people uniquely. Health (San Francisco) 671, 1–34 (2000).
36. Tang J., Cui Y., Li Q., Ren K., Liu J., Buyya R.: Ensuring security and privacy preservation for cloud data services. ACM Comput. Surv. 49(1), Article No. 13 (2016).
37. The Sage Mathematical Software System. http://www.sagemath.org/. Accessed 10 Jan 2021.
38. Tsukiyama S., Ide M., Ariyoshi H., Shirakawa I.: A new algorithm for generating all the maximal independent sets. SIAM J. Comput. 6(3), 505–517 (1977).
39. Welsh D.J.A., Powell M.B.: An upper bound for the chromatic number of a graph and its application to timetabling problems. Comput. J. 10(1), 85–86 (1967).
40. Yang Q., Wu X.: 10 challenging problems in data mining research. Int. J. Inf. Technol. Decis. Mak. 5(04), 597–604 (2006).

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.