Finding Non-Redundant Simpson's Paradox from Multidimensional Data

Yi Yang
Duke University
Durham, NC, USA
owen.yang@duke.edu

Jian Pei
Duke University
Durham, NC, USA
j.pei@duke.edu

Jun Yang
Duke University
Durham, NC, USA
junyang@cs.duke.edu

Jichun Xie
Duke University
Durham, NC, USA
jichun.xie@duke.edu

ABSTRACT

Simpson's paradox, a long-standing statistical phenomenon, describes the reversal of an observed association when data are disaggregated into sub-populations. It has critical implications across statistics, epidemiology, economics, and causal inference. Existing methods for detecting Simpson's paradox overlook a key issue: many paradoxes are *redundant*, arising from equivalent selections of data subsets, identical partitioning of sub-populations, and correlated outcome variables, which obscure essential patterns and inflate computational cost. In this paper, we present the first framework for discovering *non-redundant* Simpson's paradoxes. We formalize three types of redundancy – sibling child, separator, and statistic equivalence – and show that redundancy forms an equivalence relation. Leveraging this insight, we propose a concise representation framework for systematically organizing redundant paradoxes and design efficient algorithms that integrate depth-first materialization of the base table with redundancy-aware paradox discovery. Experiments on real-world datasets and synthetic benchmarks show that redundant paradoxes are widespread, on some real datasets constituting over 40% of all paradoxes, while our algorithms scale to millions of records, reduce run time by up to 60%, and discover paradoxes that are structurally robust under data perturbation. These results demonstrate that Simpson's paradoxes can be efficiently identified, concisely summarized, and meaningfully interpreted in large multidimensional datasets.

PVLDB Reference Format:
Yi Yang, Jian Pei, Jun Yang, and Jichun Xie. Finding Non-Redundant Simpson's Paradox from Multidimensional Data. PVLDB, XX(X): XXX-XXX, XXXX.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Owen-Yang-18/non-redundant-simpson-paradox.

1 INTRODUCTION

Simpson's paradox [33, 42] is a classic and widely studied phenomenon in statistics, probability, and data science [2, 7, 14, 21, 25, 28, 32, 35, 37, 40, 41, 44, 50, 51, 54, 60]. This paradox refers to the reversal of an observed association between two variables when data are disaggregated into sub-populations. It has been recognized for more than a century and continues to play a central role in fields such as epidemiology, economics, machine learning, and causal inference [2, 12, 18, 25, 29, 31, 32, 36, 46, 47, 57], where decisions depend critically on understanding relationships in multidimensional data.

Table 1: Data table $T(A, B, C, Y_1)$ containing 7 records.

	A	B	C	Y_1
t_1	a_1	b_1	c_1	0
t_2	a_1	b_1	c_1	0
t_3	a_1	b_2	c_1	0
t_4	a_2	b_1	c_2	1
t_5	a_2	b_1	c_2	1
t_6	a_2	b_2	c_2	1
t_7	a_2	b_2	c_2	1

Example 1.1 (Simpson's paradox). Consider the dataset in Table 1. Overall, the probability of $Y_1 = 1$ is lower for records with $B = b_2$ than for those with $B = b_1$:

$$P(Y_1 = 1 \mid B = b_2) = \tfrac{2}{4} = 0.50 \ < \ P(Y_1 = 1 \mid B = b_1) = \tfrac{2}{3} \approx 0.67.$$

However, when the data are partitioned by attribute A, the trend reverses. For $A = a_1$, both $B = b_1$ and $B = b_2$ yield $P(Y_1 = 1) = 0$. For $A = a_2$, both $B = b_1$ and $B = b_2$ yield $P(Y_1 = 1) = 1$. In each subgroup defined by A, the conditional probabilities satisfy

$$P(Y_1 = 1 \mid A = a_i, B = b_2) \geq P(Y_1 = 1 \mid A = a_i, B = b_1),\ i \in \{1, 2\}.$$

Thus, although the aggregated data suggest $B = b_1$ is more favorable, conditioning on A eliminates the apparent disadvantage of $B = b_2$. This reversal of association between B and Y_1 after conditioning on A is an instance of Simpson's paradox. □

Simpson's paradox has been observed in diverse real-world domains, including medicine and social science [6, 8, 24]. In a well-known study of treatment effectiveness for kidney stones [9], the overall recovery rate appears higher for one treatment, but when patients are stratified by stone size, the trend reverses in both subgroups. A similar paradox was documented in graduate admissions at the University of California, Berkeley [5], where aggregate data suggested gender bias, yet department-level data showed the opposite pattern. These counterintuitive reversals—the essence of Simpson's paradox—demonstrate how aggregated data can obscure underlying relationships and highlight the importance of identifying such paradoxes for reliable analysis and decision-making.

Despite its importance, an overlooked issue in the literature is that instances of Simpson's paradox can be highly *redundant*. In high-dimensional data, many partitions share identical sets of

Proceedings of the VLDB Endowment, Vol. XX, No. X ISSN 2150-8097.
doi:XX.XX/XXX.XX

records in the base table, or different choices of separator or label attributes may yield equivalent partitions. As a result, multiple paradoxes can describe the same underlying phenomenon. For example, in Table 1, there is a one-to-one correspondence between attributes A and C: every Simpson's paradox involving A can also be expressed as one involving C. Although such paradoxes differ syntactically, they arise from the same overlapping population structure. Treating them as distinct not only inflates the number of reported paradoxes but also obscures the essential insights that analysts aim to extract.

One might question whether redundant paradoxes occur merely in theory or isolated cases. However, our empirical analysis using real-world datasets in different domains, reported in Section 5.2, shows that redundant Simpson's paradoxes account for 20.3–47.8% of all observed paradoxes.

Identifying *non-redundant* Simpson's paradoxes poses several technical challenges. First, the search space of potential paradoxes grows exponentially with the number of attributes, making brute-force enumeration computation-heavy. Second, redundancies can arise in multiple ways, as analyzed in Section 3.1, and distinguishing among them requires careful formalization. Third, even after redundancies are recognized, a principled method is needed to group redundant paradoxes and produce concise, non-overlapping representations without information loss.

To address these challenges, our key idea is to exploit the mathematical structure underlying how data subsets (populations) relate to one another. We discover that redundant paradoxes exhibit patterns that allow us to group them into well-defined equivalence classes. We propose a concise representation for these equivalence classes that eliminates redundancy while ensuring completeness (i.e., discovery of all Simpson's paradoxes). This approach enables us to compactly capture large numbers of redundant paradoxes.

We make four main contributions in this paper. First, we formally define three sources of redundancy – sibling child, separator, and statistic equivalence – and show that redundancy is an equivalence relation. Second, we propose a concise representation framework that groups redundant paradoxes into compact, systematic summaries. Third, we develop efficient algorithms that combine depth-first materialization of the input base table and redundancy-aware discovery of Simpson's paradoxes. Last, through experiments on both real-world and synthetic datasets, we demonstrate that redundant paradoxes are common in practice, our methods scale efficiently, and the identified paradoxes (and redundancies) are structurally robust.

The rest of the paper is organized as follows. Section 2 introduces preliminaries and definitions of Simpson's paradox. Section 3 formalizes redundancy and presents our concise representation framework. Section 4 describes algorithms for discovering non-redundant paradoxes. Section 5 reports experimental results on real and synthetic datasets. Section 6 reviews related work, and Section 7 concludes the paper.

2 PRELIMINARIES

We introduce the foundational concepts used throughout the paper. We then present the formal definition of Simpson's Paradox in this context, along with illustrative examples and related variants.

2.1 Basic Notations

Consider a base table T containing n *categorical attributes* $\{X_1, \dots, X_n\}$ and m *label attributes* $\{Y_1, \dots, Y_m\}$, where the domain $\mathsf{Dom}(X_i)$ of each categorical attribute X_i $(1 \le i \le n)$ is finite, and each label attribute Y_i $(1 \le i \le m)$ is binary. Our results can be generalized to cases where label attributes are categorical with more than two classes. For simplicity, we focus on binary labels in this paper. Each record $t \in T$ is an $(n+m)$-dimensional tuple $(t.X_1, \dots, t.X_n, t.Y_1, \dots, t.Y_m)$.

A **population** s is an n-dimensional tuple $(s[1], \dots, s[n])$ such that $s[i] \in \mathsf{Dom}(X_i) \cup \{*\}$, where $*$ is a wildcard character equivalent to ALL in data cube terminology [17, 20, 49, 59]. Populations serve as selection criteria to define subsets of records from the base table. In paritcular, the **coverage** of a population s with respect to T, denoted by $\mathrm{cov}_T(s) = \{t \in T \mid t.X_i = s[i] \ \vee \ s[i] = *, 1 \le i \le n\}$, is the set of records in T matching s. We omit the subscript T in $\mathrm{cov}_T(\cdot)$ when the context is clear.

Given populations s and s', s is a **parent** of s' (and s' a **child** of s), denoted by $s \dot{\succ} s'$, if (1) there exists an attribute X_j $(1 \le j \le n)$ such that $s[j] = *$ and $s'[j] \ne *$, and (2) for all attributes X_i $(1 \le i \ne j \le n)$, $s[i] = s'[i]$. Clearly, if $s \dot{\succ} s'$, then $\mathrm{cov}(s) \supseteq \mathrm{cov}(s')$. We call X_j the **differential attribute** between s and s' and $s'[j]$ the **differential value**. We write the child as $s' = s\langle X_j \mapsto s'[j] \rangle$, where $\langle X_j \mapsto \cdot \rangle$ denotes the substitution of the j-th component. Generally, a population may have multiple parents and children.

A population s is an **ancestor** of population s' (and s' a **descendant** of s), denoted by $s \succ s'$, if for all attributes X_i $(1 \le i \le n)$ either $s[i] = *$ or $s[i] = s'[i]$, and for at least one attribute $s[i] \ne s'[i]$. In this case, $\mathrm{cov}(s) \supseteq \mathrm{cov}(s')$. If s is a parent of s', then s is also an ancestor of s', but not vice versa. We write $s \succeq s'$ if $s \succ s'$ or $s = s'$.

Two populations s_1 and s_2 are **siblings** if both are children of a common parent s, under the differential attribute X_j, with different differential values. In this case, $s_1[i] = s_2[i]$ for all attributes X_i $(1 \le i \le n, i \ne j)$, and $s_1[j] \ne s_2[j]$ and neither equals $*$. Moreover, $\mathrm{cov}(s_1) \cap \mathrm{cov}(s_2) = \emptyset$.

We are interested in how often each label attribute Y_i $(1 \le i \le m)$ takes the value 1 within a given population s where $\mathrm{cov}_T(s) \ne \emptyset$. We define the **frequency statistics** of a non-empty population s (w.r.t. T) under label attribute Y_i as the conditional probability $P(Y_i = 1|s)$, or simply denoted by $P(Y_i|s)$, given by $\frac{|\mathrm{cov}_T(s) \cap \{t \in T | t.Y_i = 1\}|}{|\mathrm{cov}_T(s)|}$.

Example 2.1 (Notations). Consider the table $T(A, B, C, Y_1)$ in Table 1, which contains three categorical attributes A, B, and C and a label attribute Y_1. Population $(a_1, b_1, *)$ covers the set of records with $A = a_1$ and $B = b_1$, i.e., $\mathrm{cov}(a_1, b_1, *) = \{t_1, t_2\}$. Population $(a_1, b_2, *)$ is a sibling of $(a_1, b_1, *)$, sharing the common parent $(a_1, *, *)$ under the differential attribute B. We can write $(a_1, b_1, *) = (a_1, *, *)\langle B \mapsto b_1 \rangle$ and $(a_1, b_2, *) = (a_1, *, *)\langle B \mapsto b_2 \rangle$. It is easy to verify that $P(Y_1|(*, b_1, *)) = \frac{2}{4} = 0.50$. □

The set of all populations in a given table forms a lattice under the parent-child relation $\dot{\succ}$. Let $\mathcal{P}$ be the set of all populations in a given table. Consider a subset $\mathcal{E} \subseteq \mathcal{P}$. Two populations $s, s' \in \mathcal{E}$ are *directly connected*, denoted $s \dot{\sim} s'$, if either $s \dot{\prec} s'$ or $s \dot{\succ} s'$. They are *connected*, denoted $s \sim s'$, if either $s \dot{\sim} s'$ or there exists a sequence $s_1, \dots, s_k \in \mathcal{E}$ $(k > 2)$ such that $s = s_1$, $s' = s_k$, and $s_j \dot{\sim} s_{j+1}$ $(1 \le j < k)$. $\mathcal{E}$ is **convex** if all pairs of populations in $\mathcal{E}$

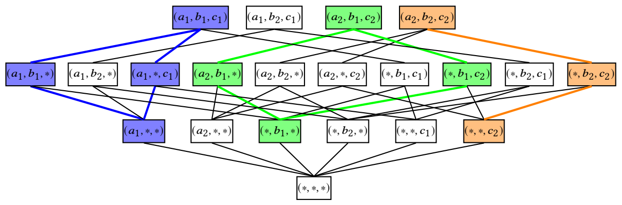

Figure 1: Hasse diagram of the lattice formed by all populations in Table 1 with respect to the parent-child relation $\dot{\succ}$. A parent is placed lower than its child. The blue and green subsets are convex, while the orange subset is non-convex.

are connected and, whenever a pair $s, s' \in \mathcal{E}$ satisfies $s \succ s'$, then all intermediate populations s'' with $s \succ s'' \succ s'$ also belong to $\mathcal{E}$. Figure 1 shows the population lattice of Table 1.

2.2 Simpson's Paradox

Simpson's Paradox [33, 42] describes the counterintuitive phenomenon where the type of relationship (e.g., positive, negative, or independent) between two variables reverses when the population is partitioned into sub-populations. In this subsection, we formalize Simpson's Paradox in the multidimensional data, provide an illustrative example, and briefly review its well-known variants.

Definition 2.2. Consider a population s and two sibling child populations $s_1 = s\langle X_j \mapsto u_1\rangle$ and $s_2 = s\langle X_j \mapsto u_2\rangle$ with differential attribute X_j $(1 \leq j \leq n)$, where $u_1, u_2 \in \mathrm{Dom}(X_j)$ $(u_1 \neq u_2)$ are the respective *differential values*. Let $X \in \{X_1, \ldots, X_n\} \setminus \{X_j\}$ be a **separator attribute** and $Y \in \{Y_1, \ldots, Y_m\}$ be a label attribute. The tuple (s_1, s_2, X, Y) is called an **association configuration (AC)**. An AC is a **Simpson's Paradox** if the following holds:

(1) $P(Y|s_1) \geq P(Y|s_2)$;
(2) For every separator attribute value $v \in \mathrm{Dom}(X)$ with $\mathrm{cov}(s_1\langle X \mapsto v\rangle) \neq \emptyset$ and $\mathrm{cov}(s_2\langle X \mapsto v\rangle) \neq \emptyset$:
$$P(Y|s_1\langle X \mapsto v\rangle) \leq P(Y|s_2\langle X \mapsto v\rangle);$$
(3) Either the inequality in (1) is strict or all inequalities in (2) are strict. □

The directions of the inequalities in (1) and (2) may be reversed simultaneously. In addition, partitioning can be generalized to a set of multiple separator attributes $\mathbf{X}$: for each value combination $\mathbf{v} \in \prod_{X_j \in \mathbf{X}} \mathrm{Dom}(X_j)$, we consider sub-populations $s_1\langle \mathbf{X} \mapsto \mathbf{v}\rangle$ and $s_2\langle \mathbf{X} \mapsto \mathbf{v}\rangle$.

Example 1.1 shows an example of Simpson's paradox, where $((*, b_1, *), (*, b_2, *), A, Y_1)$ is an associate configuration. For clarity, the remainder of this paper assumes a single separator attribute, though our results extend directly to the multi-attribute case.

Over the past century, several variants of Simpson's Paradox have been studied. The most widely used is the **Association Reversal (AR)** [40], as formalized in Definition 2.2. A special case, **Yule's Association Paradox (YAP)** [58], occurs when there is no association in the sub-populations, yet an association emerges in the aggregate. Example 1.1 is an example of YAP. Another form, the **Amalgamation Paradox (AMP)** [15], arises when the strength of association in the aggregate is greater (or smaller) than in each sub-population. A variant of AMP, the **Averaged Association Reversal (AAR)** [2, 51], occurs when the aggregate association differs from the average association across sub-populations. Both AMP and AAR are special cases of AR. For an in-depth review of Simpson's Paradox, we refer the reader to the survey by Sprenger and Weinberger [45]. Our framework naturally extends to these variants.

Table 2: Data table $T(A, B, C, D, Y_1, Y_2)$ containing 7 records.

	A	B	C	D	Y_1	Y_2
t_1	a_1	b_1	c_1	d_1	0	0
t_2	a_1	b_1	c_1	d_1	0	0
t_3	a_1	b_2	c_1	d_2	0	0
t_4	a_2	b_1	c_2	d_1	1	1
t_5	a_2	b_1	c_2	d_1	1	1
t_6	a_2	b_2	c_2	d_2	1	1
t_7	a_2	b_2	c_2	d_2	1	1

3 REDUNDANCY AMONG INSTANCES OF SIMPSON'S PARADOX

In practice, multiple populations in a table may have identical coverage, leading to different association configurations that capture essentially the same paradoxical behavior. For example, in Table 1, $\mathrm{cov}(a_1, *, *) = \mathrm{cov}(a_1, *, c_1) = \{t_1, t_2, t_3\}$. When a table has many attributes but relatively sparse records, such overlaps are common [4, 10, 19, 23, 26]. This incidental identicality can generate multiple Simpson's paradoxes that are redundant. In this section, we formalize this insight by defining **redundancy** through three types of equivalences that give rise to it, and then unifying them into a single definition.

3.1 Three Types of Redundancies

Redundancy may arise from three distinct sources. We first describe each case with formal statements and examples.

3.1.1 Sibling Child Equivalence. When sibling populations have identical coverage, their corresponding paradoxes are redundant.

Lemma 3.1 (Sibling child equivalence). *Consider two association configurations $p = (s_1, s_2, X, Y)$ and $p' = (s'_1, s'_2, X, Y)$ where $\mathrm{cov}(s_1) = \mathrm{cov}(s'_1)$ and $\mathrm{cov}(s_2) = \mathrm{cov}(s'_2)$. If p is a Simpson's paradox, then p' is also a Simpson's paradox.* □

Example 3.2 (Sibling child equivalence). We extend Table 1 to Table 2 by adding attribute D and label attribute Y_2. Similar to Example 1.1, $((*, b_1, *, *), (*, b_2, *, *), A, Y_1)$ is a Simpson's paradox. It can be verified that $((*, *, *, d_1), (*, *, *, d_2), A, Y_1)$ is also a Simpson's paradox due to sibling child equivalence. □

3.1.2 Separator Equivalence. When two separator attributes induce partitions that are aligned via a one-to-one mapping, the resulting paradoxes are redundant.

Lemma 3.3 (Separator equivalence). *Consider two association configurations $p = (s_1, s_2, X, Y)$ and $p' = (s_1, s_2, X', Y)$, where $X \neq X'$ and there exists a one-to-one mapping $f : \mathrm{Dom}(X) \mapsto \mathrm{Dom}(X')$ such that for every $v \in \mathrm{Dom}(X)$ and $s \in \{s_1, s_2\}$, $\mathrm{cov}(s\langle X \mapsto v\rangle) =$*

$\mathrm{cov}(s\langle X' \mapsto f(v)\rangle)$. *If p is a Simpson's paradox, then p' is also a Simpson's paradox.* □

Example 3.4 (Separator equivalence). In Table 2, $((*, b_1, *, *), (*, b_2, *, *), A, Y_1)$ is a Simpson's paradox. It can be verified that $((*, b_1, *, *), (*, b_2, *, *), C, Y_1)$ is also a Simpson's paradox due to separator equivalence. □

3.1.3 Statistic Equivalence.

When label attributes are dependent, their paradoxes may be redundant. We identify three sufficient conditions for such equivalence.

LEMMA 3.5 (STATISTIC EQUIVALENCE). *Consider two association configurations $p = (s_1, s_2, X, Y)$ and $p' = (s_1, s_2, X, Y')$ such that $Y \neq Y'$. If p is a Simpson's paradox and if any of the following (sufficient, and progressively less restrictive) conditions hold, then p' is also a Simpson's paradox:*

1. *For every $t \in \mathrm{cov}(s_1) \cup \mathrm{cov}(s_2)$, $t.Y = t.Y'$;*
2. *For every $s \in \{s_1, s_2\}$, $P(Y|s) = P(Y'|s)$ and for every $v \in \mathrm{Dom}(X)$, $P(Y|s\langle X \mapsto v\rangle) = P(Y'|s\langle X \mapsto v\rangle)$;*
3. *$\mathrm{sign}(P(Y|s_1) - P(Y|s_2)) = \mathrm{sign}(P(Y'|s_1) - P(Y'|s_2))$, and for every $v \in \mathrm{Dom}(X)$, $\mathrm{sign}(P(Y|s_1\langle X \mapsto v\rangle) - P(Y|s_2\langle X \mapsto v\rangle)) = \mathrm{sign}(P(Y'|s_1\langle X \mapsto v\rangle) - P(Y'|s_2\langle X \mapsto v\rangle))$.* □

Example 3.6 (Statistic equivalence). We extend Table 2 to Table 3. Consider $p = ((*, b_1, *, *), (*, b_2, *, *), A, Y_1)$. We observe $P(Y_1|(*, b_1, *, *)) = 0.33 < P(Y_1|(*, b_2, *, *)) = 0.40$. Partitioning by A yields $P(Y_1|(a_1, b_1, *, *)) = P(Y_1|(a_1, b_2, *, *)) = 0$ and $P(Y_1|(a_2, b_1, *, *)) = P(Y_1|(a_2, b_2, *, *)) = 0.50$, confirming that p is a Simpson's paradox. It follows that:

- $((*, b_1, *, *), (*, b_2, *, *), A, Y_2)$ is statistic equivalent to p (Case 1), since $t.Y_1 = t.Y_2$ for every record t.
- $((*, b_1, *, *), (*, b_2, *, *), A, Y_3)$ is statistic equivalent to p (Case 2), as the frequency statistics of Y_3 match those of Y_1 across all relevant populations, even though $Y_1 \neq Y_3$ for some records.
- $((*, b_1, *, *), (*, b_2, *, *), A, Y_4)$ is statistic equivalent to p (Case 3), since the signs of the frequency statistics differences coincide for Y_1 and Y_4 at both the aggregate and sub-populations levels.
- Note that Y_4 could be constructed such that the sign of frequency statistics differences at the aggregate level matches that of Y_1, while for one sub-population the difference is negative for Y_1 but zero for Y_4. Despite sign mismatch, Simpson's Paradox is still preserved with Y_4, showing that Case 3 is sufficient but not necessary. □

3.2 Equivalence Classes of Simpson's Paradoxes

Redundant Simpson's paradoxes may arise from more than one of the equivalence types, sometimes combining sibling child, separator, and statistic equivalence within the same instances.

Example 3.7 (Redundancy). In Table 2, $((*, b_1, *, *), (*, b_2, *, *), A, Y_1)$ is a Simpson's paradox. It can be verified that $((*, *, *, d_1), (*, *, *, d_2), C, Y_2)$ is also a Simpson's paradox, and in fact redundant due to sibling child, separator, and statistic equivalence simultaneously. □

Motivated by the above observation, we integrate the three equivalences into a unified notion.

Table 3: Data table $T(A, B, C, D, Y_1, Y_2, Y_3, Y_4)$ with 11 records.

	A	B	C	D	Y_1	Y_2	Y_3	Y_4
t_1	a_1	b_1	c_1	d_1	0	0	0	0
t_2	a_1	b_1	c_1	d_1	0	0	0	0
t_3	a_1	b_1	c_1	d_2	0	0	0	0
t_4	a_1	b_2	c_1	d_1	0	0	0	0
t_5	a_1	b_2	c_1	d_2	0	0	0	0
t_6	a_2	b_1	c_2	d_1	0	0	1	1
t_7	a_2	b_1	c_2	d_1	1	1	0	1
t_8	a_2	b_1	c_2	d_2	1	1	1	1
t_9	a_2	b_2	c_2	d_1	0	0	1	1
t_{10}	a_2	b_2	c_2	d_2	1	1	0	1
t_{11}	a_2	b_2	c_2	d_2	1	1	1	1

Definition 3.8 (Redundancy). Two distinct paradoxes are:

- **sibling child equivalent** if they satisfy Lemma 3.1;
- **separator equivalent** if they satisfy Lemma 3.3;
- **statistic equivalent** if they satisfy at least one of the conditions in Lemma 3.5.

They are **redundant** if any of the above holds. □

THEOREM 3.9 (EQUIVALENCE). *Redundancy of Simpson's paradoxes is an equivalence relation.* □

Because redundancy is an equivalence relation, the set of all Simpson's paradoxes can be partitioned into equivalence classes. Each class corresponds to a group of mutually redundant paradoxes. In some cases, an equivalence class of Simpson's paradox may contain only a single instance when no redundancy is observed.

Example 3.10 (Equivalence class). Consider Table 2. From Example 1.1, $p_1 = ((*, b_1, *, *), (*, b_2, *, *), A, Y_1)$ is a paradox. We also obtain the following paradoxes redundant to p_1:

- $p_2 = ((*, *, *, d_1), (*, *, *, d_2), A, Y_1)$ (sibling child equiv.);
- $p_3 = ((*, b_1, *, *), (*, b_2, *, *), C, Y_1)$ (separator equiv.);
- $p_4 = ((*, b_1, *, *), (*, b_2, *, *), A, Y_2)$ (statistic equiv.);
- $p_5 = ((*, *, *, d_1), (*, *, *, d_2), C, Y_1)$ (sibling and separator);
- $p_6 = ((*, *, *, d_1), (*, *, *, d_2), A, Y_2)$ (sibling and statistic);
- $p_7 = ((*, b_1, *, *), (*, b_2, *, *), C, Y_2)$ (separator and stat.);
- $p_8 = ((*, *, *, d_1), (*, *, *, d_2), C, Y_2)$ (all equivalences).

The set $\{p_1, p_2, \ldots, p_8\}$ forms an equivalence class. □

3.3 Representing Equivalence Classes Concisely

In real datasets with many attributes, the number of redundant paradoxes can be large. We therefore propose a concise representation for an equivalence class of redundant Simpson's paradoxes (or **redundant paradox group** for short). To begin, we make the following observation.

LEMMA 3.11 (PRODUCT SPACE). *Each redundant paradox group can be characterized by the product of: $\mathcal{E}_1 \times \mathcal{E}_2 \times \mathbf{X} \times \mathrm{Y}$, where $\mathbf{X}$ is a set of separator attributes, Y is a set of label attributes, and $\mathcal{E}_1, \mathcal{E}_2$ are sets of sibling populations, each containing populations with identical coverage. Any choice of $(s_1, s_2, X, Y) \in \mathcal{E}_1 \times \mathcal{E}_2 \times \mathbf{X} \times \mathrm{Y}$ where s_1, s_2 are siblings is a Simpson's paradox in the redundant paradox group.* □

Example 3.12. The redundant paradox group from Example 3.10 is characterized by the product space of the following:

- $\mathcal{E}_1 = \{(*, b_1, *, *), (*, *, *, d_1), (*, b_1, *, d_1)\}$,
- $\mathcal{E}_2 = \{(*, b_2, *, *), (*, *, *, d_2), (*, b_2, *, d_2)\}$,
- $\mathbf{X} = \{A, C\}$,
- $\mathbf{Y} = \{Y_1, Y_2\}$.

This product space encompasses multiple paradoxes. For instance, $((*, b_1, *, *), (*, b_2, *, *), A, Y_1)$ and $((*, *, *, d_1), (*, *, *, d_2), C, Y_2)$ are included. However, $((*, b_1, *, d_1), (*, b_2, *, d_2), A, Y_1)$ is not, since these two populations are not valid siblings. □

Next, we show that $\mathcal{E}_1$ and $\mathcal{E}_2$ can be represented more concisely. We partition the set of all populations $\mathcal{P}$ based on coverage, denoted $\mathcal{P}/\equiv_{\text{cov}}$, where each **coverage group** $\mathcal{E} \in \mathcal{P}/\equiv_{\text{cov}}$ contains populations with identical coverage. We show that each such coverage group exhibits a structural property in the population lattice: they form convex subsets. This means that if two populations belong to the same coverage group, then all populations that lie between them in the lattice hierarchy must also belong to that group. Given convexity of any $\mathcal{E} \in \mathcal{P}/\equiv_{\text{cov}}$, we call a population $s \in \mathcal{E}$ an **upper bound** of $\mathcal{E}$ if there is no $s' \in \mathcal{E}$ with $s \prec s'$ (i.e., least descendant); similarly, we call s a **lower bound** of $\mathcal{E}$ if there is no $s' \in \mathcal{E}$ with $s' \prec s$ (i.e., greatest ancestor). We denote the set of upper bounds of $\mathcal{E}$ by $\mathsf{up}(\mathcal{E})$ and the set of lower bounds of $\mathcal{E}$ by $\mathsf{low}(\mathcal{E})$. Convexity ensures that we can represent coverage groups $\mathcal{E}_1$ and $\mathcal{E}_2$ using only these bounds, thereby avoiding enumerating all members, which can be prohibitively large in high-dimensional datasets. Furthermore, we show that each coverage group has a unique upper bound (though it can have multiple lower bounds).

Property 1 (Convexity of coverage groups). *Let $\mathcal{P}$ be the set of all populations. For each coverage group $\mathcal{E} \in \mathcal{P}/\equiv_{\text{cov}}$, $\mathcal{E}$ is a convex subset of coverage-identical populations. Furthermore, $|\mathsf{up}(\mathcal{E})| = 1$ and the least descendant is the unique upper bound.* □

Property 2 (Reconstruction from bounds). *Let $\mathcal{E} \subseteq \mathcal{P}$ be a convex subset of populations. Then $s \in \mathcal{E}$ if and only if there exist $s_l \in \mathsf{low}(\mathcal{E})$ and $\{s_u\} = \mathsf{up}(\mathcal{E})$ such that $s_l \preceq s \preceq s_u$.* □

Using these properties and Lemma 3.11, we can concisely represent each redundant paradox group. We remark that while Property 1 establishes that upper bounds uniquely identify (convex) coverage groups, reconstruction requires lower bounds. Given only the upper bound s_u, enumerating all $s \in \mathcal{E}$ requires verifying equality of coverage for each ancestor $s \prec s_u$. With lower bounds $\mathsf{low}(\mathcal{E})$, reconstruction is straightforward and efficient: following Property 2, enumerate all s satisfying $s_l \preceq s \preceq s_u$ for every $s_l \in \mathsf{low}(\mathcal{E})$. This reconstruction is essential because our concise representation must generate all Simpson's paradoxes in the group by enumerating every valid sibling pair across populations in $\mathcal{E}_1$ and $\mathcal{E}_2$ (as shown in Examples 3.10, 3.12).

Definition 3.13 (Concise representation). A redundant paradox group characterized by $\mathcal{E}_1 \times \mathcal{E}_2 \times \mathbf{X} \times \mathbf{Y}$ can be represented as:

$$(\mathsf{up}(\mathcal{E}_1), \mathsf{low}(\mathcal{E}_1), \mathsf{up}(\mathcal{E}_2), \mathsf{low}(\mathcal{E}_2), \mathbf{X}, \mathbf{Y}),$$

where $\mathsf{up}(\mathcal{E}_1)$ and $\mathsf{up}(\mathcal{E}_2)$ are the (unique) upper bounds of $\mathcal{E}_1$ and $\mathcal{E}_2$, and $\mathsf{low}(\mathcal{E}_1)$ and $\mathsf{low}(\mathcal{E}_2)$ are their lower bounds. □

By Property 2, this representation is precise: all populations in $\mathcal{E}_1$ and $\mathcal{E}_2$ (and thus all redundant Simpson's paradoxes in the group) can be reconstructed from the bounds.

Example 3.14 (Concise representation). From Example 3.10, the redundant paradox group can be represented as:

- $\mathsf{up}(\mathcal{E}_1) = \{(*, a_1, *, b_1)\}$,
- $\mathsf{low}(\mathcal{E}_1) = \{(*, a_1, *, *), (*, *, *, b_1)\}$,
- $\mathsf{up}(\mathcal{E}_2) = \{(*, a_2, *, b_2)\}$,
- $\mathsf{low}(\mathcal{E}_2) = \{(*, a_2, *, *), (*, *, *, b_2)\}$,
- $\mathbf{X} = \{A, C\}$,
- $\mathbf{Y} = \{Y_1, Y_2\}$.

All eight redundant Simpson's paradoxes from Example 3.10 are captured. For instance, $((*, b_1, *, *), (*, b_2, *, *), A, Y_1)$ is valid because $(*, b_1, *, *) \in \mathcal{E}_1$ and $(*, b_2, *, *) \in \mathcal{E}_2$ are siblings. By contrast, $((*, b_1, *, d_1), (*, b_2, *, d_2), A, Y_1)$ is not valid, since these two populations are not siblings. □

4 FINDING NON-REDUNDANT SIMPSON'S PARADOXES

In this section, we first establish the #P-hardness of finding non-redundant Simpson's paradoxes. Then we present our algorithmic framework for fast identification all non-redundant Simpson's paradoxes (i.e., redundant paradox groups) in a given table.

4.1 Complexity

Theorem 4.1 (#P-Hardness). *Finding all redundant paradox groups in a multidimensional table is #P-hard.*

Proof sketch. We reduce from #SAT [48]. Given a Boolean formula ϕ, we construct a table where each satisfying assignment of ϕ corresponds to a distinct group of redundant Simpson's paradoxes. Specifically, each satisfying assignment maps to a subset of records that exhibit paradoxes under Definition 2.2 and redundancies under Lemmas 3.1, 3.3, and 3.5. The reduction preserves the number of satisfying assignments of ϕ as the number of redundant paradox groups in the constructed table, making the problem #P-hard. □

This hardness result shows that identifying non-redundant Simpson's paradoxes is computationally challenging. Nevertheless, given their importance and practical relevance, in the remainder of this section we present techniques to accelerate the computation.

4.2 General Algorithmic Framework

Our approach builds directly on the concise representation developed in Section 3, ensuring that redundant instances are grouped into equivalence classes and represented concisely. Our framework proceeds in two main steps:

(1) **Materialization.** We enumerate all non-empty populations, compute their coverage and frequency statistics, and organize populations into (convex) coverage groups.
(2) **Paradox discovery.** Using the materialized coverage and frequency statistics, we detect all instances of Simpson's paradox and simultaneously construct concise representations (Section 3.2) for redundant paradox groups.

Algorithm 1 Brute-force Materialization

```
Input: Data table $T = (\{X_i\}_{i=1}^n, \{Y_j\}_{j=1}^m)$
Output: Materialized populations
1: for each record $t \in T$ do
2:     for each ancestor $s$ of $t$ do
3:         Update $\mathrm{cov}(s)$ and $P(Y|s)$ for each $Y \in \{Y_1, \ldots, Y_m\}$.
```

Algorithm 2 DFS-based Materialization

```
Input: Data table $T = (\{X_i\}_{i=1}^n, \{Y_j\}_{j=1}^m)$, coverage threshold $\theta$
Output: Coverage-based partitioning $\mathcal{P}/\equiv_{\mathrm{cov}}$ of populations,
    where each group $\mathcal{E}$ is represented by $(\mathrm{up}(\mathcal{E}), \mathrm{low}(\mathcal{E}))$; fre-
    quency STATS for each $\mathcal{E} \in \mathcal{P}/\equiv_{\mathrm{cov}}$, indexed by $\mathrm{up}(\mathcal{E})$.
1: Initialize the set $G$ of candidate coverage groups and STATS;
2: DFS($(*, *, \ldots, *), T, 0$);                          ▷ updates G and STATS
3: for each unique upper bound $u$ such that $(u, \_) \in G$ do
4:     $L \leftarrow \{s \mid (u, s) \in G \wedge (\nexists s' : (u, s') \in G \wedge s \prec s')\}$;
5:     Add $(u, L)$ as a coverage group in $\mathcal{P}/\equiv_{\mathrm{cov}}$;
6: return $\mathcal{P}/\equiv_{\mathrm{cov}}$, STATS;
7: function DFS($s, T', k$):                                ▷ updates G and STATS
8:     $d \leftarrow s$;
9:     for each attribute $X_i$ $(1 \le i \le n)$ with $s[i] = *$ do
10:        if $\exists v \in \mathrm{Dom}(X_i) : \mathrm{cov}(s) = \mathrm{cov}(s\langle X_i \mapsto v\rangle)$ then
11:            $d[i] \leftarrow v$;
12:    Add $(d, s)$ to $G$; STATS$(d) \leftarrow \{P(Y_j|d)\}_{j=1}^m$;
13:    for each attribute $X_h$ $(k < h \le n)$ with $d[h] = *$ do
14:        for each $v \in \mathrm{Dom}(X_h)$ do
15:            $T'_v \leftarrow \{t \mid t \in T' \wedge t.X_h = v\}$;
16:            if $|T'_v| \ge \theta \cdot |T|$ then
17:                DFS($d\langle X_h \mapsto v\rangle, T'_v, h$);
```

4.3 Materialization

The first step of our algorithm materializes all non-empty populations, computing their coverage and frequency statistics. A brute-force approach, illustrated in Algorithm 1, is to iterate through each record t in the base table T and updates the coverage and frequency statistics for all ancestor populations of t. For each record, this requires enumerating all 2^n ancestors s where $s[i] \in \{t.X_i, *\}$ for each attribute X_i. This method suffers from two inefficiencies. First, it performs repetitive computation by separately processing all 2^n ancestors for every record, leading to $|T| \times 2^n$ population updates even when many have identical coverage. Second, it fails to organize materialized populations into convex coverage groups, thereby missing opportunities to avoid repetitive materialization of intermediate populations within each coverage group.

To address these limitations, we propose a depth-first search (DFS) approach adapted from [4, 19, 26]. Effectively, the output will be coverage groups (i.e., convex subsets) of populations with identical coverage, along with frequency statistics for each. Algorithm 2 summarizes the procedure.

4.3.1 DFS-based Population Materialization. The algorithm builds the population lattice (see Figure 1 as an example) in a bottom-up manner. It starts from the root population $s_{\mathrm{root}} = (*, *, \ldots, *)$ covering the entire dataset T, and progressively materializes populations that cover fewer records, thereby moving upward in the lattice.

At each recursive step, the DFS aims to identify a (candidate) convex coverage group populations. We begin with some lower-bound population s and attempt to find its corresponding upper bound s'. The upper bound s', intitially the same as s, is constructed by scanning the records in $\mathrm{cov}(s)$: for each attribute X_i where $s[i] = *$, if all records share the value $v \in \mathrm{Dom}(X_i)$, we set $s'[i] = v$; otherwise, $s'[i] = *$. This constructions ensures $\mathrm{cov}(s') = \mathrm{cov}(s)$.

Example 4.2. Consider Table 2 and population $s = (a_1, *, *, *)$ with $\mathrm{cov}(s) = \{t_1, t_2, t_3\}$. Scanning these records, we find that all share the value c_1 for attribute C. For B and D, the records do not share a common value. Thus, the upper bound s' is $(a_1, *, c_1, *)$. □

By Property 2, a convex coverage group can be reconstructed from its upper and lower bounds. Thus, any population s'' between s and s' (i.e., $s \succeq s'' \succeq s'$) must have the same coverage. These intermediate populations do not need explicit materialization; their coverage and statistics can be inferred, greatly improving efficiency.

After identifying s', the pair (s, s') serves as a candidate coverage group, and we recursively explore each child $\hat{s}$ of s', continuing DFS with $\hat{s}$ as the lower bound of the next candidate coverage group.

Example 4.3. Continuing from Example 4.2, the pair $(s = (a_1, *, *, *), s' = (a_1, *, c_1, *))$ defines a convex group. We then explore children of $(a_1, *, c_1, *)$. For instance, $(a_1, b_1, c_1, *)$ is a child since $\mathrm{cov}((a_1, b_1, c_1, *)) \subset \mathrm{cov}((a_1, *, c_1, *))$ and is non-empty. DFS proceeds with $(a_1, b_1, c_1, *)$ as the next lower bound. □

The recursion stops when (1) a population s covers fewer than a threshold $\theta \cdot |T|$ records (see Section 4.3.3, population pruning), or (2) the DFS reaches the top of the lattice.

4.3.2 Coverage Group Merging. DFS may discover the same coverage group via different lower bounds. Therefore, we merge candidate coverage groups that share the same upper bound. Each merged group has one upper bound and potentially multiple lower bounds. We then refine the lower bounds by removing invalid ones, i.e., those that are descendants of others in the same group.

Example 4.4. In Table 2, populations $(a_1, *, *, *)$, $(*, *, c_1, *)$, and $(a_1, *, c_1, *)$ all have coverage $\{t_1, t_2\}$. DFS may discover this convex coverage group via two paths: (1) $(a_1, *, *, *) \longrightarrow (a_1, *, c_1, *)$, or (2) $(*, *, c_1, *) \longrightarrow (a_1, *, c_1, *)$. This yields two candidate coverage groups. After merging, the consolidated coverage group has upper bound $\{(a_1, *, c_1, *)\}$ and lower bounds $\{(a_1, *, *, *), (*, *, c_1, *)\}$, both valid since neither is an ancestor of the other. □

4.3.3 Population Pruning. An important practical insight is that populations with very small coverage often have low analytical significance and are unlikely to be of interest to users. To address this, we introduce a simple pruning threshold $0 \le \theta \le 1$. Any population with coverage less than $\theta \cdot |T|$ is neither materialized nor considered in Simpson's paradoxes. This practical constraint also reduces computational cost: in the DFS-based materialization algorithm, if a population s covers fewer than $\theta \cdot |T|$ records, we skip materializing both s and all its descendants.

Example 4.5. In Table 2, suppose $\theta = 0.4$. Population $s = (*, *, c_1, *)$ covers $\{t_1, t_2, t_3\}$. Since $3/7 \approx 43\% > 40\%$, s is not pruned. If $\theta = 0.6$, then s and all its descendants, such as $(a_1, *, c_1, *)$ and (a_1, b_1, c_1, d_1), would be pruned. □

Algorithm 3 Brute-force finding of Simpson's paradoxes

```
Input: Materialized populations $S$ from $T = (\{X_i\}_{i=1}^n, \{Y_j\}_{j=1}^m)$
Output: All instances of Simpson's paradox
1: for each population $s \in S$ do
2:     for each $X_i$ with $s[i] = *$ do
3:         for each pair $v_1, v_2 \in \mathrm{Dom}(X_i)$ where $u_1 \neq u_2$ do
4:             $s_1 \leftarrow s\langle X_i \mapsto v_1 \rangle$;   $s_2 \leftarrow s\langle X_i \mapsto v_2 \rangle$;
5:             for each $i' \neq i$ with $s[i'] = *$ do
6:                 for each label attribute $Y$ do
7:                     Evaluate $(s_1, s_2, X_{i'}, Y)$ using Definition 2.2;
```

Theorem 4.6 (Completeness). *Algorithm 2 materializes all non-empty populations that satisfy the coverage threshold. Furthermore, after group merging, Algorithm 2 yields maximal convex coverage groups of coverage-identical populations; that is, no population outside a group shares the same coverage as any population within it.* □

4.4 Finding Redundant Paradox Groups

The discovery of Simpson's paradoxes can be viewed as a two-step process: (1) systematically enumerating all possible ACs, and (2) evaluating each AC against the definition of Simpson's paradox (Definition 2.2). Algorithm 3 illustrates a brute-force method: for each non-empty population s, we enumerate all sibling child pairs (s_1, s_2), and then combine each pair with every valid separator attribute and label attribute to form candidate ACs. Each AC is then checked against Definition 2.2.

This exhaustive search has two major drawbacks. First, it does not organize discovered paradoxes into redundant paradox groups. Second, it wastes computation by (i) repeatedly iterating over populations with identical coverage, and (ii) evaluating ACs that are redundant to already discovered paradoxes.

We therefore design optimizations to avoid repeated computation and concisely represent redundant paradox groups.

4.4.1 Iteration over Coverage Groups. Instead of iterating over every non-empty population in Algorithm 3, we exploit the convex coverage groups discovered from Algorithm 2. From each coverage group, it suffices to consider only one representative population – specifically, its unique upper bound (Property 1) – since all populations in the coverage group have identical coverage. This helps avoid repeated computation over such populations.

4.4.2 Constructing Redundant Paradox Groups. Even after restricting to iteration over coverage groups, many ACs remain redundant. This occurs due to three types of equivalence. First, sibling child equivalence: coverage groups contain multiple populations beyond their upper bounds, and these populations can form valid sibling pairs, generating sibling child equivalent Simpson's paradoxes. For example, if coverage groups $\mathcal{E}_1$ and $\mathcal{E}_2$ have upper bounds $(*, b_1, *, d_1)$ and $(*, b_2, *, d_1)$, the AC using the upper bounds is just one instance – other populations like $(*, b_1, *, *) \in \mathcal{E}_1$ and $(*, b_2, *, *) \in \mathcal{E}_2$ are also siblings, creating sibling child equivalent ACs. Second, separator and statistic equivalence: different separator attributes may induce identical sub-population partitions, and different label attributes may be perfectly correlated. For example, consider sibling populations $(*, b_1, *, *)$ and $(*, b_2, *, *)$ from Table 2. Observe that attributes A and C partition the data identically, and label values for Y_1 and Y_2 are perfectly correlated, then ACs $((*, b_1, *, *), (*, b_2, *, *), A, Y_1)$ and $((*, b_1, *, *), (*, b_2, *, *), C, Y_2)$ would be redundant.

Algorithm 4 Construction by sibling child equivalence

```
1: function ConstructBySib($p, \mathcal{P}/\equiv_{\mathrm{cov}}$)
2:     Input: Simpson's paradox $p = (s_1, s_2, X, Y)$; coverage-based
       partitioning $\mathcal{P}/\equiv_{\mathrm{cov}}$ of populations
3:     Output: Concise representation of a set of paradoxes
       sibling-child-equivalent to $p$
4:     Let $\mathcal{E}_1, \mathcal{E}_2 \in \mathcal{P}/\equiv_{\mathrm{cov}}$ be groups containing $s_1$ and $s_2$, resp.;
5:     return $(\mathrm{up}(\mathcal{E}_1), \mathrm{low}(\mathcal{E}_1), \mathrm{up}(\mathcal{E}_2), \mathrm{low}(\mathcal{E}_2), \{X\}, \{Y\})$;
```

We propose two strategies that exploit the redundancy conditions from Section 3.2 to avoid repeated evaluations when constructing redundant paradox groups. Importantly, we leverage the concise representation of such groups (Section 3.3), and ensure that we maintain representational conciseness when constructing/extending these groups.

Construction by sibling child equivalence. Once a Simpson's paradox $p = (s_1, s_2, X, Y)$ is identified, all paradoxes sibling child equivalent to p can be inferred without evaluation. Thanks to Section 3.3, we can also encode the entire set of such paradoxes concisely. This strategy is formalized below and implemented by Algorithm 4.

Proposition 4.7. *Let $p = (s_1, s_2, X, Y)$ be a Simpson's paradox, where s_1 and s_2 belong to coverage groups $\mathcal{E}_1$ and $\mathcal{E}_2$ in $\mathcal{P}/\equiv_{\mathrm{cov}}$, respectively. Then for any $(s'_1, s'_2) \in \mathcal{E}_1 \times \mathcal{E}_2$ such that s'_1 and s'_2 are siblings, the AC $p' = (s'_1, s'_2, X, Y)$ is also a Simpson's paradox and redundant with respect to p.* □

Example 4.8 (Construction by sibling child equivalence). In Table 2, consider the Simpson's paradox $p = ((*, b_1, *, *), (*, b_2, *, *), A, Y_1)$ from Example 3.2. With materialization by Algorithm 2, populations $(*, b_1, *, *)$ and $(*, b_2, *, *)$ belong to coverage groups $\{(*, b_1, *, *), (*, *, *, d_1), (*, b_1, *, d_1)\}$ and $\{(*, b_2, *, *), (*, *, *, d_2), (*, b_2, *, d_2)\}$, respectively. Valid sibling pairs across these groups yield additional paradoxes, such as $((*, *, *, d_1), (*, *, *, d_2), A, Y_1)$, which are redundant to p. These can be directly included in the same redundant paradox group without further evaluation. Furthermore, instead of enumerating these paradoxes, the group can be concisely represented by:

$$\mathrm{up}(\mathcal{E}_1) = \{(*, b_1, *, d_1)\}, \quad \mathrm{low}(\mathcal{E}_1) = \{(*, b_1, *, *), (*, *, *, d_1)\},$$
$$\mathrm{up}(\mathcal{E}_2) = \{(*, b_2, *, d_2)\}, \quad \mathrm{low}(\mathcal{E}_2) = \{(*, b_2, *, *), (*, *, *, d_2)\},$$
$$\mathbf{X} = \{A\}, \quad \mathbf{Y} = \{Y_1\}$$
□

Extension by separator and statistic equivalence. Many paradoxes differ only by separator or label attributes but are still redundant. Once we know a paradox p' is separator- or statistic-equivalent to some p in a group $\mathbf{P}$ of sibling-child-equivalent paradoxes, we can apply the separator and label attributes of p' to all members of $\mathbf{P}$ to obtain more redundant paradoxes, without evaluation. Again, thanks to Section 3.3, such an extension can be efficiently carried out by the concise represention of $\mathbf{P}$, without enumerating members of $\mathbf{P}$. Algorithm 5 implements this strategy.

Algorithm 5 Extension by separator/statistic equivalence

```
1: function ExtendBySepStat($\tilde{\mathbf{P}}, p'$)
2:     Input: Concise rep. $\tilde{\mathbf{P}}$ for a sibling-child-equivalent paradox group; new Simpson's paradox $p' = (s'_1, s'_2, X', Y')$ redundant with respect to some paradoxes in $\tilde{\mathbf{P}}$
3:     Output: Updated $\tilde{\mathbf{P}}$
4:     $(\mathrm{up}(\mathcal{E}_1), \mathrm{low}(\mathcal{E}_1), \mathrm{up}(\mathcal{E}_2), \mathrm{low}(\mathcal{E}_2), \mathbf{X}, \mathbf{Y}) \leftarrow \tilde{\mathbf{P}}$;
5:     return $(\mathrm{up}(\mathcal{E}_1), \mathrm{low}(\mathcal{E}_1), \mathrm{up}(\mathcal{E}_2), \mathrm{low}(\mathcal{E}_2),$
               $\mathbf{X} \cup \{X'\}, \mathbf{Y} \cup \{Y'\})$;
```

Proposition 4.9. *Let* **P** *be a set of sibling-child-equivalent Simpson's paradoxes with separator X and label Y. Suppose (s'_1, s'_2, X', Y'), where $X' \neq X$ or $Y' \neq Y$, is a Simpson's paradox redundant with respect to some paradox in* **P***. Then for every $p = (s_1, s_2, X, Y) \in \mathbf{P}$, the AC (s_1, s_2, X', Y') is also a redundant Simpson's paradox with respect to p.* □

Example 4.10 (Extension by separator and statistic equivalence). Continuing from Example 4.8, suppose we have now identified $p' = ((*, *, *, d_1), (*, *, *, d_2), C, Y_2)$ as redundant with respect to the sibling-child-equivalent redundant paradox group **P** characterized by $\mathcal{E}_1 \times \mathcal{E}_2 \times \{A\} \times \{Y_1\}$. Proposition 4.9 implies that all combinations from $\mathcal{E}_1$ and $\mathcal{E}_2$ with separator C and label Y_2 are also redundant paradoxes. For example, $((*, b_1, *, *), (*, b_2, *, *), C, Y_2)$ can be added directly. The characterization of the group simply becomes $\mathcal{E}_1 \times \mathcal{E}_2 \times \{A, C\} \times \{Y_1, Y_2\}$. □

4.4.3 Complete Algorithm. Finally, we integrate these optimizations into a comprehensive algorithm (Algorithm 6). We iterate only over coverage groups (using upper bounds), constructing a sibling-child-equivalence redundant paradox group as soon as one paradox is found, and extending groups with separator and statistic equivalence when applicable. We maintain a hashmap I where each key is $\textsc{Sig}(p)$ (see Definition 4.11 below) and the value is the concise representation of a redundant paradox group (though a group may contain only a single paradox if no redundancy is observed).

Definition 4.11 (Signature). Given populations s_1 and s_2, we define their *joint signature* with respect to label attribute Y as a triple:

$$\textsc{JSig}_Y(s_1, s_2) = \langle \mathrm{cov}(s_1), \mathrm{cov}(s_2), \mathrm{sign}(P(Y|s_1) - P(Y|s_2)) \rangle.$$

For an AC $p = (s_1, s_2, X, Y)$, its **signature** defined as:

$$\textsc{Sig}(p) = \langle \textsc{JSig}_Y(s_1, s_2), \{\textsc{JSig}_Y(s_1\langle X \mapsto v\rangle, s_2\langle X \mapsto v\rangle) \mid v \in \mathrm{Dom}(X)\} \rangle.$$ □

In implementation, coverage sets cov(·) are represented using integer hashes rather than storing full record sets. The signature $\textsc{Sig}(\cdot)$ becomes a vector containing integer hashes for coverage sets and sign values from $\{-1, 0, +1\}$ for frequency statistic differences. This enables efficient detection of redundant paradoxes: as established by Lemma 4.12, all paradoxes within the same redundant paradox group share identical signatures. Thus, we can efficiently determine if a paradox p belongs to an already discovered redundant paradox group by checking if $\textsc{Sig}(p)$ exists as a key in I.

Lemma 4.12. *Two Simpson's paradoxes p and p′ are redundant if and only if $\textsc{Sig}(p) = \textsc{Sig}(p')$.* □

Algorithm 6 Finding non-redundant Simpson's paradoxes

```
Input: $\mathcal{P}/\equiv_{\mathrm{cov}}$ and stats as produced by Algorithm 2
Output: Hashmap $I$ storing concise representations of redundant paradox groups, keyed by Sig
1: Initialize $I \leftarrow \emptyset$;
2: for each coverage group $\mathcal{E} \in \mathcal{P}/\equiv_{\mathrm{cov}}$ do
3:     Let $s \leftarrow \mathrm{up}(\mathcal{E})$        ▷ use upper bound as representative
4:     for each $X_i$ with $s[i] = *$ do
5:         for each pair $v_1, v_2 \in \mathrm{Dom}(X_{i_0})$ do
6:             $s_1 \leftarrow s\langle X_i \mapsto v_1\rangle$,   $s_2 \leftarrow \langle X_i \mapsto v_2\rangle$;
7:             for each $X_j \neq X_i$ with $s[j] = *$ do
8:                 for each label attribute $Y$ do
9:                     $p \leftarrow (s_1, s_2, X_j, Y)$; ▷ check first if p is included in existing redundant paradox groups
10:                    if $\exists(\mathcal{E}_1, \mathcal{E}_2, \mathbf{X}, \mathbf{Y}) \in I : s_1 \in \mathcal{E}_1 \wedge s_2 \in \mathcal{E}_2 \wedge X_j \in \mathbf{X} \wedge Y \in \mathbf{Y}$ then continue
11:                    Evaluate $p$ acc. Def. 2.2; compute Sig($p$);
12:                    if $p$ is a Simpson's paradox then
13:                        if $I(\textsc{Sig}(p)) = \emptyset$ then
14:                            $I(\textsc{Sig}(p)) \leftarrow$
                               ConstructBySib($p, \mathcal{E}/\equiv_{\mathrm{cov}}$);
15:                        else
16:                            $I(\textsc{Sig}(p)) \leftarrow$
                               ExtendBySepStat($I(\textsc{Sig}(p)), p$);
17: return $I$
```

Together, these optimizations transform the brute-force enumeration into an efficient method that avoids repeated work while producing concise representations of redundant paradox groups.

5 EXPERIMENTAL RESULTS

In this section, we evaluate instances of coverage redundant Simpson's paradoxes on both real-world and synthetic datasets, described in Section 5.1. Our study is guided by three research questions (RQs): **RQ1** investigates whether (coverage) redundant Simpson's paradoxes are rare in practice (Section 5.2); **RQ2** evaluates the scalability of our computational framework (Section 5.3); and **RQ3** examines the structural robustness of the discovered (coverage) redundant Simpson's paradoxes (Section 5.4).

For each question, we conduct quantitative experiments and provide detailed analyses. Overall, our results show that redundant Simpson's paradoxes occur frequently in real-world datasets, our method scales effectively in practice, and the identified paradoxes are structurally robust under data perturbation.

5.1 Experimental Setup

We conducted our experiments on the Duke Computer Science Department's computing cluster, using nodes equipped with Intel Xeon Gold 5317 processors (3.0 GHz, 12 cores) and 64 GB of RAM.

We evaluate our methods on both real-world categorical datasets and synthetic datasets generated with controlled parameters. The real-world datasets allow us to measure the prevalence of redundant paradoxes, while the synthetic datasets provide a way to systematically assess efficiency and scalability under controlled conditions.

5.1.1 Real-World Datasets. We use datasets from diverse domains:

Table 4: Simpson's paradoxes in real-world datasets.

Dataset	Adult	Mushroom	Loan	Diabetes
#paradoxes	3,880	6,878	18,330	1,464,250
#groups	3,460	4,931	16,293	1,065,189
#standalone	3,094	3,590	14,354	809,388
#sibling-child eq.	366	1,220	1,939	255,690
#separator eq.	0	146	0	340
#statistic eq.	0	0	0	0

- **Adult:** A census income dataset with 48,842 records, 8 attributes (e.g., education, occupation), and a binary label indicating whether annual income exceeds \$50K [3].
- **Mushroom:** A dataset of 8,124 records describing mushrooms using 22 categorical attributes (e.g., cap shape, habitat), with edibility as the binary label [1].
- **Loan:** A large financial dataset[1] containing about 3 million loan applications, with 12 categorical attributes (e.g., loan purpose, home ownership) and a label of loan approval.
- **CDC Diabetes Health Indicators:** A healthcare dataset[2] of 253,681 individuals, with 35 categorical attributes (covering demographics, laboratory results, and lifestyle factors) and a binary label indicating diabetes status.

5.1.2 Synthetic Datasets. To evaluate performance in a controlled setting, we employ a synthetic data generator that produces datasets with user-specified structural properties. The generator accepts the following key parameters:

- n: number of categorical attributes;
- m: number of label attributes;
- d: number of values per attribute.

The generation process proceeds in two steps. First, it constructs individual instances of Simpson's paradox. For a given AC (s_1, s_2, X, Y), the generator enforces a consistent trend across all sub-populations (e.g., $P(Y|s_1\langle X \mapsto v\rangle) > P(Y|s_2\langle X \mapsto v\rangle)$ for all $v \in \mathrm{Dom}(X)$). Then, it solves an optimization problem to distribute records across sub-populations such that the aggregate association reverses (e.g., $P(Y|s_1) \leq P(Y|s_2)$). Detailed procedures are provided in the artifact supplements.

Second, the framework introduces redundancy by modifying the generated records. Sibling child equivalence is induced by aligning sibling pairs (s_1, s_2) and (s'_1, s'_2) so that $\mathrm{cov}(s_1) = \mathrm{cov}(s'_1)$ and $\mathrm{cov}(s_2) = \mathrm{cov}(s'_2)$. Separator equivalence is enforced by mapping domains of two separator attributes X and X' and updating each record r with $r.X' = f(r.X)$. Finally, statistic equivalence is created by defining a new label Y' as a direct copy of an existing label Y.

This workflow is repeated until the dataset reaches a target size. Each iteration generates at least one unique non-redundant paradox, together with multiple redundant variants, while ensuring that all populations and sub-populations involved in a paradox contain at least a minimum number of records.

By varying n, d, and m, the generator naturally controls the richness of paradoxes and redundancies. Larger n and d values, for instance, expand the number of potential sibling values and separator attributes, increasing opportunities for Simpson's paradoxes and sibling child and separator equivalences. We study these effects in detail in Figure 3 in Section 5.2.

[1]https://www.kaggle.com/datasets/ikpeleambrose/irish-loan-data
[2]https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset/data

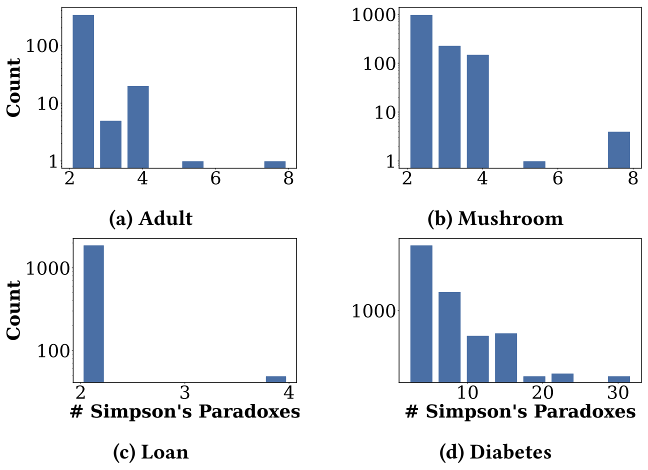

Figure 2: Distribution of the number of Simpson's paradoxes per redundant group in four real-world datasets.

5.2 Q1: Are Coverage-Redundant Simpson's Paradoxes Rare?

Our analysis shows that (coverage) redundant Simpson's paradoxes are common in real-world datasets. As summarized in Table 4, a substantial fraction of discovered paradoxes are redundant: 20.3% in Adult, 47.8% in Mushroom, 21.7% in Loan, and 44.7% in Diabetes. Among the three types of equivalence, *sibling child equivalence* is the most prevalent across all datasets. *Separator equivalence*, which requires a one-to-one correspondence between separator attributes, is less frequent and appears only in the higher-dimensional Mushroom (10.7%) and Diabetes (0.1%) datasets. No *statistic equivalence* is observed because each dataset contains only a single label attribute.

To further analyze redundancy, Figure 2 shows the distribution of group sizes. In all datasets, most redundant paradox groups consist of only 2 or 3 paradoxes. In higher-dimensional datasets such as Mushroom and Diabetes, however, groups can grow much larger, containing up to 30 paradoxes due to the combined effects of sibling child and separator equivalences.

We also examine how redundancy patterns emerge in synthetic datasets. Since each synthetic dataset is generated with a fixed number of records (Section 5.1.2), the number of unique, non-redundant paradoxes is limited, keeping the number of redundant paradox groups relatively stable across parameter settings. Our analysis therefore focuses on how generator parameters ($n = 8$, $m = 4$, $d = 8$ by default) affect the *total* number of paradoxes. With a stable number of paradox groups, growth in the total count reflects an increase in redundant instances. Figure 3 presents the results, from which we draw the following observations:

- **Number of categorical attributes (n):** The total number of paradoxes grows exponentially with n, as additional

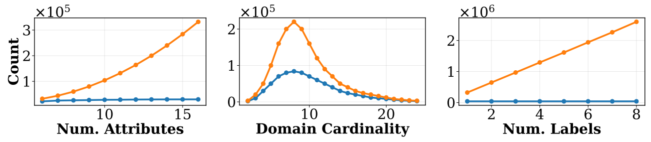

Figure 3: Effect of dataset parameters on the total number of Simpson's paradoxes (orange) and redundant paradox groups (blue) in synthetic data.

attributes create more opportunities for sibling child and separator equivalences.

- **Domain cardinality (d):** The total number of paradoxes first increases and then decreases with d. Larger domains cause each paradox to cover more records, reducing the number of unique, non-redundant paradoxes that can be generated under the fixed record budget.
- **Number of label attributes (m):** The total number of paradoxes increases linearly with m. Each additional label introduces a statistic-equivalent version of every existing paradox, effectively scaling the total count.

5.3 Q2: Scalability

We next evaluate the computational efficiency of our method on both real-world and synthetic datasets. Our experiments yield three main findings: (i) our base algorithm (cf. Alg. 2, 6), without population pruning (Sec. 4.3.3), achieves an average 6.72× speedup over brute-force baselines (cf. Alg. 1, 3) on real-world datasets, enabled by DFS-based materialization and redundancy-aware paradox discovery; (ii) population pruning, which excludes populations covering fewer than 0.1% of total records, yields an additional 50% run time reduction on top of our base algorithm; and (iii) run time on synthetic datasets scales predictably with the total number of Simpson's paradoxes, as characterized in Section 5.2 (Fig. 3).

5.3.1 Scalability on Real-World Datasets. Figure 4 (a), on the left, compares three methods: (1) the brute-force baseline (including brute-force materialization cf. Alg. 1 and paradox discovery cf. Alg. 3); (2) our base algorithm (including DFS-based materialization cf. Alg. 2 and redundancy-aware paradox discovery cf. Alg. 6) with no population pruning (Sec. 4.3.3); and (3) our base algorithm with 0.1% population pruning.

Across all datasets, the base algorithm achieves an average 6.72× speedup relative to brute-force. This gain is driven by two key optimizations: (1) DFS-based materialization (Alg. 2) yields an average 4.94× improvement by identifying upper and lower bounds of coverage groups, avoiding explicit enumeration of intermediate populations; (2) redundancy-aware paradox discovery (Alg. 4–6) achieves an average 20.24× improvement by eliminating repeated enumeration and evaluation of redundant paradoxes. Finally, pruning populations below the 0.1% threshold reduces run time by an additional 50%, confirming that small-coverage populations are abundant in practice. Together, these optimizations make the problem computationally tractable even for high-dimensional datasets such as Diabetes, with over 250,000 records and 35 attributes.

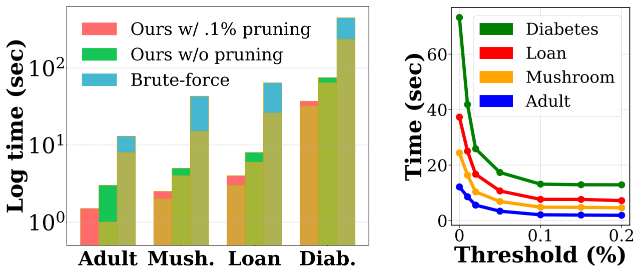

Figure 4: (a) Run time comparison on real-world datasets. Yellow shaded regions represent materialization time. (b) Run time vs. pruning threshold on real-world datasets.

Figure 4 (b), on the right, shows the effect of varying the pruning threshold θ from 0% to 0.2% of total records. We observe a reciprocal ($1/x$-like) trend: even minimal pruning (0.05%) reduces run time by an average of 41.2%, as many low-coverage populations are immediately eliminated. Improvements diminish beyond 0.1%, where most such populations have already been pruned and run time stabilizes.

5.3.2 Scalability on Synthetic Datasets. Figure 5 reports run time with respect to three parameters in synthetic datasets: number of attributes (n), domain cardinality (d), and number of labels (m), with default values $n = 8$, $d = 8$, and $m = 4$. To stress-test scalability, we generate datasets with up to 30 million records. Each plot shows both materialization time and total run time.

Run time trends align closely with the total number of Simpson's paradoxes observed in Figure 3. This correlation arises because the materialization phase dominates computation and directly depends on the number of populations:

- **Number of categorical attributes (n):** As n grows, the population lattice expands exponentially, since each population can branch into multiple children for the new attribute's values. This explains the exponential increase in materialization time.
- **Domain cardinality (d):** Larger domains reduce the number of paradoxes (Figure 3), which in turn reduces the number of populations requiring materialization. Run time therefore decreases after an initial peak.
- **Number of label attributes (m):** Each additional label requires computing frequency statistics for the same set of populations, yielding linear scaling in materialization time.

In contrast, paradox discovery time (the difference between total and materialization time) remains nearly constant across parameter settings. This stability follows from our synthetic generation procedure, where the number of redundant paradox groups is held constant regardless of the total number of paradoxes (Figure 3).

5.4 Q3: Are Coverage-Redundant Simpson's Paradoxes Robust?

Beyond demonstrating the prevalence of (coverage) redundant Simpson's paradoxes, we now investigate whether these paradoxical reversals and redundancies reflect *genuine structural properties* of the data or are merely random artifacts introduced by noise or

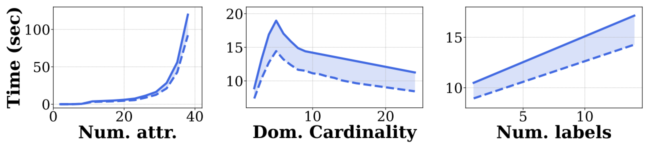

Figure 5: Run time scaling with synthetic dataset parameters. Solid lines denote total run time; dotted lines denote materialization time.

errors in data collection. To this end, we adopt a perturbation-based framework to test the *robustness* of paradoxes and redundancies.

At a high level, we measure *tolerance:* given an observed Simpson's paradox (or redundancy), we quantify how much the data can be perturbed before the paradoxical (or redundant) relationship disappears. Robust patterns should persist under small perturbations, whereas random artifacts should vanish quickly.

We evaluate two aspects: (1) the **robustness** of individual Simpson's paradoxes under label and record perturbations; and (2) the **persistence** of coverage redundancies under record perturbations.

5.4.1 *Robustness of Individual Simpson's Paradoxes.* We first examine whether Simpson's paradoxes persist under perturbations. Any frequency statistics $P(Y|s)$ can be decomposed as

$$P(Y|s) = \sum_{v \in \mathrm{Dom}(X)} \frac{|\operatorname{cov}(s\langle X \mapsto v\rangle)|}{|\operatorname{cov}(s)|} P(Y|s\langle X \mapsto v\rangle).$$

This weighted sum consists of two components: (i) weights representing record distribution across sub-populations, and (ii) frequency statistics within each sub-population. Simpson's paradoxes emerge from specific interactions between these components.

Accordingly, for each paradox $p = (s_1, s_2, X, Y)$, we apply two perturbation strategies:

- **Label perturbation:** Randomly flip labels of 5% of the records in $\operatorname{cov}(s_1) \cup \operatorname{cov}(s_2)$ to alter the frequency statistics $P(Y|s\langle X \mapsto v\rangle)$. This tests whether paradoxical reversals depend critically on exact label assignments.
- **Coverage perturbation:** Randomly modify the separator attribute X for 5% of records to change the weights $|\operatorname{cov}(s\langle X \mapsto v\rangle)|/|\operatorname{cov}(s)|$ across sub-populations. We then reassign labels in each sub-population according to their original frequency statistics, thereby isolating the effect of record distribution.

Each perturbation is repeated 10,000 times, and robustness is measured as the survival rate (percentage of trials where the paradox persists). Figure 6 shows robustness under label perturbations. The fraction of robust paradoxes – those surviving in at least 95% of trials – increases with higher pruning thresholds. This indicates that paradoxes supported by larger populations are more tolerant to perturbations.

5.4.2 *Robustness of Redundant Groups.* Sibling child and separator equivalence rely on populations with identical coverage, which can be organized into convex coverage groups. Redundancy is therefore robust if coverage identicality persist under perturbations. We test robustness with two strategies:

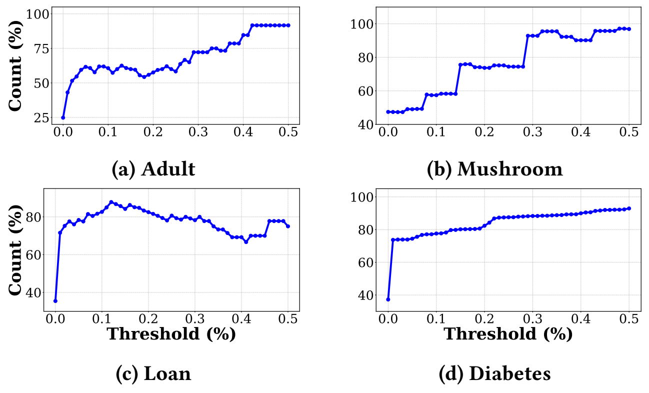

Figure 6: Fraction of structurally robust Simpson's paradoxes vs. pruning threshold across four real-world datasets.

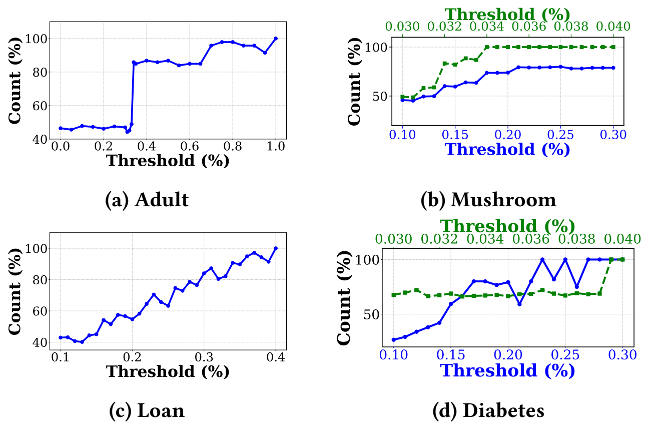

Figure 7: Fraction of robust redundant paradox groups vs. pruning threshold across four real-world datasets.

- **Sibling child equivalence** (Lemma 3.1): For sibling-child-equivalent paradoxes $p = (s_1, s_2, X, Y)$ and $p' = (s'_1, s'_2, X, Y)$, we randomly alter one attribute value in 5% of the records in $\operatorname{cov}(s_1) \cup \operatorname{cov}(s_2)$. This tests whether $\operatorname{cov}(s_1) = \operatorname{cov}(s'_1)$ and $\operatorname{cov}(s_2) = \operatorname{cov}(s'_2)$ remain intact.
- **Separator equivalence** (Lemma 3.3): For separator-equivalent paradoxes $p = (s_1, s_2, X, Y)$ and $p' = (s_1, s_2, X', Y)$, we perturb 5% of records in $\operatorname{cov}(s_1) \cup \operatorname{cov}(s_2)$ on attributes other than X and X'. The one-to-one mapping $f : X \mapsto X'$ is preserved, and we test whether equivalence $\operatorname{cov}(s\langle X \mapsto v\rangle) = \operatorname{cov}(s\langle X' \mapsto f(v)\rangle)$ persists despite changes in other attributes.

As before, each perturbation is repeated 10,000 times, and robustness is measured as the survival rate of redundant paradox groups. Figure 7 reports the results. The fraction of robust redundant groups rises with higher pruning thresholds, indicating that sibling- and division-equivalent paradoxes with larger coverage are more tolerant to perturbations.

Summary of Findings

Across RQ1-RQ3, our experiments highlight three main insights. First, coverage-redundant Simpson's paradoxes are common: they represent a substantial fraction of all paradoxes in real-world datasets, with sibling child equivalence being the most frequent redundancy type. Second, our computational framework scales efficiently, achieving practical run times even on moderate- to high-dimensional data, and further benefits from population pruning that eliminate low-coverage populations early. Third, both individual paradoxes and their coverage redundancies exhibit structural robustness under data perturbations, confirming that these patterns reflect genuine properties of the data rather than random noise or collection errors.

Overall, our findings show that coverage-redundant Simpson's paradoxes are prevalent, can be detected efficiently, and capture meaningful structural characteristics in real-world datasets.

6 RELATED WORK

To the best of our knowledge, this is the first work to study redundancy among Simpson's paradoxes. Our contribution connects to two lines of prior research: (1) methods for detecting Simpson's paradox in high-dimensional data; and (2) techniques for concisely summarizing data populations.

6.1 Detecting Simpson's Paradox

Simpson's paradox has been extensively studied since the introduction of Association Reversal (AR) and Amalgamation Paradox (AMP) [58]. Early detection methods relied on statistical modeling. Freitas et al. [14] constructed Bayesian networks to detect paradoxes by analyzing network structure, and Fabris and Freitas [13] extended this approach by ranking paradoxes according to their estimated surprisingness for knowledge discovery. Alipourfard et al. [2] introduced a statistical test comparing global trends with disaggregated sub-population patterns, while Xu et al. [54] used Pearson correlation to detect reversals between continuous variables. Sharma et al. [41] further extended this approach to categorical variables through binarization.

Beyond standalone methods, several works integrate paradox detection into broader analytical frameworks. Salimi et al. [37–39] developed a system that identifies biased OLAP queries susceptible to Simpson's paradox using independence tests and resolves them through automated query rewriting. Liu et al. [30] proposed a data-driven framework that discovers sub-populations for hypothesis testing and reveals confounding factors underlying paradoxes. More recently, automated methods have been introduced: Wang et al. [51] employed neural models to disaggregate data and evaluate associations across subgroups, and Jiang et al. [21] designed a federated learning framework that mitigates association reversals in distributed data via counterfactual learning. Domain-specific efforts include Portela et al. [35], who applied regression trees to identify conditional outliers affected by Simpson's paradox.

Despite these advances, existing methods emphasize global analyses and overlook paradoxes within *local populations* defined by subspaces of the data. Such local paradoxes can still reveal structural patterns and are important for causal analysis and decision-making. The closest related work is Xu et al. [55], who proposed a combinatorial search over all possible local subspaces. However, their algorithm (cf. Alg.1,3) does not detect redundancy and is computationally expensive, repeatedly enumerating populations with identical coverage and evaluating redundant paradoxes.

6.2 Summarizing Data Populations

A cornerstone of our approach is identifying and organizing subsets of populations with identical coverage. This connects to the extensive literature on data cubes [11, 16, 34, 53], which address computational challenges in OLAP. Kenneth and Srivastava [23] proposed efficient materialization algorithms for sparse multidimensional data using divide-and-conquer partitioning. Beyer and Ramakrishnan [4] introduced bottom-up materialization with coverage-based pruning, which is closely related to our method (cf. Alg. 2). Other work has focused on cube condensation to reduce storage and improve query performance [43, 52]. More recently, John and Koch [22] proposed partial materialization that reconstructs missing populations via linear programming, and You et al. [56] developed an adaptive caching system that selectively materializes convex equivalence classes under memory constraints.

Most directly relevant are quotient cubes [26, 27], which partition the population lattice into equivalence classes defined by monotone aggregate functions (e.g., coverage, count, min, max). The quotient cube preserves roll-up and drill-down semantics, improving the efficiency of analytical queries such as `GROUP BY` and `CUBE BY`. While our method exploits similar structural properties – specifically convex equivalence classes via coverage-based partitioning – our focus is different. Unlike prior cube condensation techniques [4, 23, 43, 52, 56], which aim to optimize storage and query time, we leverage convex partitions of populations to identify and eliminate redundancy among Simpson's paradoxes.

7 CONCLUSIONS

In this paper, we addressed the problem of redundancy in Simpson's paradox, a long-standing statistical phenomenon with broad applications in data analysis and causal inference. We showed that many paradoxes in multidimensional data are redundant, arising from populations with identical coverage or equivalent separator and label attributes. To resolve this issue, we formally defined three types of coverage redundancy, proved that redundancy forms an equivalence relation, and introduced a concise representation based on convexity properties of the population lattice. We further developed efficient algorithms that combine depth-first materialization, pruning, and redundancy-aware evaluation to discover all non-redundant Simpson's paradoxes. Experiments on both real-world and synthetic datasets demonstrated that redundant paradoxes are prevalent in practice, that our algorithms scale efficiently, and that the discovered paradoxes are structurally robust.

Future work includes extending our framework to richer data types and continuous attributes, incorporating causal semantics to further refine redundancy definitions, and applying our methods in practical domains such as healthcare, finance, and social science where Simpson's paradox continues to pose challenges for interpretation and decision-making.

REFERENCES

[1] 1987. Mushroom. UCI Machine Learning Repository.

[2] Nazanin Alipourfard, Peter G Fennell, and Kristina Lerman. 2018. Can you trust the trend? discovering simpson's paradoxes in social data. In *Proceedings of the eleventh ACM international conference on web search and data mining*.

[3] Barry Becker and Ronny Kohavi. 1996. Adult. UCI Machine Learning Repository.

[4] Kevin Beyer and Raghu Ramakrishnan. 1999. Bottom-up computation of sparse and iceberg cube. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*. 359–370.

[5] Peter J Bickel, Eugene A Hammel, and J William O'Connell. 1975. Sex bias in graduate admissions: Data from Berkeley. *Science* (1975).

[6] Colin R Blyth. 1972. On Simpson's paradox and the sure-thing principle. *J. Amer. Statist. Assoc.* 67, 338 (1972), 364–366.

[7] Francesco Bonchi, Francesco Gullo, Bud Mishra, and Daniele Ramazzotti. 2018. Probabilistic causal analysis of social influence. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 1003–1012.

[8] Christopher J Cates. 2002. Simpson's paradox and calculation of number needed to treat from meta-analysis. *BMC Medical research methodology* 2, 1 (2002), 1.

[9] Christopher R Charig, David R Webb, Stephen R Payne, and John EA Wickham. 1986. Comparison of treatment of renal calculi by open surgery, percutaneous nephrolithotomy, and extracorporeal shockwave lithotripsy. *British Medical Journal (Clinical research ed.)* (1986).

[10] Changqing Chen, Jianlin Feng, and Longgang Xiang. 2003. Computation of sparse data cubes with constraints. In *International Conference on Data Warehousing and Knowledge Discovery*. Springer, 14–23.

[11] Yu Chen, Jinguo You, Benyuan Zou, Guoyu Gan, Ting Zhang, and L Jia. 2020. Exploring Structural Characteristics of Lattices in Real World. *Complexity* (2020).

[12] Yuhao Deng, Yu Wang, Lei Cao, Lianpeng Qiao, Yuping Wang, Jingzhe Xu, Yizhou Yan, and Samuel Madden. 2024. Outlier summarization via human interpretable rules. *Proceedings of the VLDB Endowment* 17, 7 (2024), 1591–1604.

[13] Carem C Fabris and Alex A Freitas. 2006. Discovering surprising instances of Simpson's paradox in hierarchical multidimensional data. *International Journal of Data Warehousing and Mining (IJDWM)* 2, 1 (2006), 27–49.

[14] Alex Freitas, Kenneth McGarry, and Elon Correa. 2007. *Integrating Bayesian networks and Simpson's paradox in data mining*. College Publications.

[15] Irving John Good and Yashaswini Mittal. 1987. The amalgamation and geometry of two-by-two contingency tables. *The Annals of Statistics* (1987).

[16] Jim Gray, Adam Bosworth, Andrew Layman, and Hamid Pirahesh. 1996. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total. In *Proceedings of the Twelfth International Conference on Data Engineering (ICDE '96)*. IEEE Computer Society, USA, 152–159.

[17] J. Gray, A. Bosworth, A. Lyaman, and H. Pirahesh. 1996. Data cube: a relational aggregation operator generalizing GROUP-BY, CROSS-TAB, and SUB-TOTALS. In *Proceedings of the Twelfth International Conference on Data Engineering*.

[18] Yue Guo, Carsten Binnig, and Tim Kraska. 2017. What you see is not what you get! Detecting Simpson's Paradoxes during Data Exploration. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*.

[19] Jiawei Han, Jian Pei, Guozhu Dong, and Ke Wang. 2001. Efficient computation of iceberg cubes with complex measures. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*. 1–12.

[20] Venky Harinarayan, Anand Rajaraman, and Jeffrey D Ullman. 1996. Implementing data cubes efficiently. *Acm Sigmod Record* 25, 2 (1996), 205–216.

[21] Zhonghua Jiang, Jimin Xu, Shengyu Zhang, Tao Shen, Jiwei Li, Kun Kuang, Haibin Cai, and Fei Wu. 2025. Fedcfa: Alleviating simpson's paradox in model aggregation with counterfactual federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 17662–17670.

[22] Sachin Basil John and Christoph Koch. 2022. High-dimensional data cubes. *Proceedings of the VLDB Endowment* (2022).

[23] A Kenneth and D Srivastava. 1997. Fast computation of sparse datacubes. In *Proceedings of the 23rd International Conference on Very Large Data Bases, VLDB*.

[24] R. Kievit, W. Frankenhuis, L. Waldorp, and D. Borsboom. 2013. Simpson's paradox in psychological science: a practical guide. *Frontiers in psychology* (2013).

[25] Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J Franklin, and Ken Goldberg. 2016. Activeclean: Interactive data cleaning for statistical modeling. *Proceedings of the VLDB Endowment* 9, 12 (2016), 948–959.

[26] Laks VS Lakshmanan, Jian Pei, and Jiawei Han. 2002. Quotient cube: How to summarize the semantics of a data cube. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*. Elsevier, 778–789.

[27] Laks VS Lakshmanan, Jian Pei, and Yan Zhao. 2003. QC-Trees: An efficient summary structure for semantic OLAP. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. 64–75.

[28] H-J Lenz and Arie Shoshani. 1997. Summarizability in OLAP and statistical data bases. In *Proceedings. Ninth International Conference on Scientific and Statistical Database Management (Cat. No. 97TB100150)*. IEEE, 132–143.

[29] Y. Lin, B. Youngmann, Y. Moskovitch, HV. Jagadish, and T. Milo. 2021. On detecting cherry-picked generalizations. *Proc. of the VLDB Endowment* (2021).

[30] Guimei Liu, Mengling Feng, Yue Wang, Limsoon Wong, See-Kiong Ng, Tzia Liang Mah, and Edmund Jon Deoon Lee. 2011. Towards exploratory hypothesis testing and analysis. In *2011 IEEE 27th International Conference on Data Engineering*.

[31] Y Zee Ma. 2015. Simpson's paradox in GDP and per capita GDP growths. *Empirical Economics* (2015).

[32] Judea Pearl. 2014. Comment: Understanding Simpson's Paradox. *The American Statistician* 68, 1 (2014), 8–13.

[33] LA Pearson Karl and BM Leslie. 1899. Genetic (reproductive) selection: inheritance of fertility in man, and of fecundity in thoroughbred racehorses. *Philos. Trans. R. Soc. Lond. Ser. A* (1899).

[34] Viet Phan-Luong. 2016. A Data Cube Representation for Efficient Querying and Updating. In *2016 International Conference on Computational Science and Computational Intelligence*.

[35] Eduarda Portela, Rita P Ribeiro, and Joao Gama. 2019. The search of conditional outliers. *Intelligent Data Analysis* 23, 1 (2019), 23–39.

[36] Ricardo Salazar, Felix Neutatz, and Ziawasch Abedjan. 2021. Automated feature engineering for algorithmic fairness. *Proceedings of the VLDB Endowment* (2021).

[37] Babak Salimi, Corey Cole, Peter Li, Johannes Gehrke, and Dan Suciu. 2018. HypDB: a demonstration of detecting, explaining and resolving bias in OLAP queries. *Proceedings of the VLDB Endowment* 11, 12 (2018), 2062–2065.

[38] Babak Salimi, Johannes Gehrke, and Dan Suciu. 2018. Bias in OLAP queries: Detection, explanation, and removal. In *Proceedings of the 2018 International Conference on Management of Data*. 1021–1035.

[39] Babak Salimi, Bill Howe, and Dan Suciu. 2020. Database repair meets algorithmic fairness. *ACM SIGMOD Record* 49, 1 (2020), 34–41.

[40] Myra L Samuels. 1993. Simpson's paradox and related phenomena. *J. Amer. Statist. Assoc.* (1993).

[41] Rahul Sharma, Huseyn Garayev, Minakshi Kaushik, Sijo Arakkal Peious, Prayag Tiwari, and Dirk Draheim. 2022. Detecting Simpson's Paradox: A Machine Learning Perspective. In *International Conference on Database and Expert Systems Applications*. Springer, 323–335.

[42] E. H. Simpson. 1951. The Interpretation of Interaction in Contingency Tables. *Journal of the Royal Statistical Society: Series B (Methodological)* 13, 2 (1951).

[43] Yannis Sismanis, Antonios Deligiannakis, Nick Roussopoulos, and Yannis Kotidis. 2002. Dwarf: Shrinking the petacube. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. 464–475.

[44] Peter Spirtes, Clark Glymour, and Richard Scheines. 2000. *Causation, Prediction, and Search* (2nd ed.). MIT Press, Cambridge, MA.

[45] Jan Sprenger and Naftali Weinberger. 2021. Simpson's Paradox. In *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University.

[46] Guanting Tang, James Bailey, Jian Pei, and G. Dong. 2013. Mining multidimensional contextual outliers from categorical relational data. In *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*.

[47] Yu-Kang Tu, David Gunnell, and Mark S Gilthorpe. 2008. Simpson's Paradox, Lord's Paradox, and Suppression Effects are the same phenomenon–the reversal paradox. *Emerging themes in epidemiology* 5, 1 (2008), 2.

[48] Leslie G Valiant. 1979. The complexity of enumeration and reliability problems. *siam Journal on Computing* 8, 3 (1979), 410–421.

[49] Jeffrey Scott Vitter, Min Wang, and Bala Iyer. 1998. Data cube approximation and histograms via wavelets. In *Proceedings of the seventh international conference on Information and knowledge management*. 96–104.

[50] C. Wagner. 1982. Simpson's paradox in real life. *The American Statistician* (1982).

[51] Jingwei Wang, Jianshan He, Weidi Xu, Ruopeng Li, and Wei Chu. 2023. Learning to Discover Various Simpson's Paradoxes. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5092–5103.

[52] Wei Wang, Jianlin Feng, Hongjun Lu, and Jeffrey Xu Yu. 2002. Condensed cube: An effective approach to reducing data cube size. In *Proceedings 18th International Conference on Data Engineering*. IEEE, 155–165.

[53] Xike Xie, Xingjun Hao, Torben Bach Pedersen, Peiquan Jin, and Jinchuan Chen. 2016. OLAP over probabilistic data cubes I: Aggregating, materializing, and querying. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*.

[54] Chenguang Xu, Sarah M Brown, and Christan Grant. 2018. Detecting Simpson's paradox. In *The Thirty-First International Flairs Conference*.

[55] Jay Xu, Jian Pei, and Zicun Cong. 2022. Finding Multidimensional Simpson's Paradox. *ACM SIGKDD Explorations Newsletter* 24, 2 (2022), 48–60.

[56] Jinguo You, Yuxuan Wang, Xingrui Huang, Zhenrui Yi, Wanting Fu, Kaiqi Liu, Pengchen Zhang, and Bin Yao. 2025. SOC: A Succinct Adaptive Semantic OLAP Caching: J. You et al. *Data Science and Engineering* (2025), 1–18.

[57] Brit Youngmann, Michael Cafarella, Amir Gilad, and Sudeepa Roy. 2024. Summarized causal explanations for aggregate views. *Proceedings of the ACM on Management of Data* 2, 1 (2024), 1–27.

[58] G Udny Yule. 1903. Notes on the theory of association of attributes in statistics. *Biometrika* (1903).

[59] Yihong Zhao, Prasad M Deshpande, and Jeffrey F Naughton. 1997. An array-based algorithm for simultaneous multidimensional aggregates. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*. 159–170.

[60] Pierre Zuyderhoff, M Denuit, and J Trufin. 2025. Simpson's Paradox for Kendall's Rank Coefficient. *Methodology and Computing in Applied Probability* (2025).