Computational Complexity And Algorithms For Dirty Data Evaluation And Repairing

Dongjing Miao

Follow this and additional works at: https://scholarworks.gsu.edu/cs_diss

Recommended Citation
https://scholarworks.gsu.edu/cs_diss/145

This Dissertation is brought to you for free and open access by the Department of Computer Science at ScholarWorks @ Georgia State University. It has been accepted for inclusion in Computer Science Dissertations by an authorized administrator of ScholarWorks @ Georgia State University. For more information, please contact scholarworks@gsu.edu.
COMPUTATIONAL COMPLEXITY AND ALGORITHMS FOR DIRTY DATA EVALUATION AND REPAIRING

by

DONGJING MIAO

Under the Direction of Dr. Yingshu Li and Dr. Zhipeng Cai

ABSTRACT

In this dissertation, we study the dirty data evaluation and repairing problem in relational database. Dirty data is usually inconsistent, inaccurate, incomplete and stale. Existing methods and theories of consistency describe using integrity constraints, such as data dependencies. However, integrity constraints are good at detection but not at evaluating the degree of data inconsistency and cannot guide the data repairing. This dissertation first studies the computational complexity of and algorithms for the database inconsistency evaluation. We define and use the minimum tuple deletion to evaluate the database inconsistency. For such minimum tuple deletion problem, we study the relationship between the size of rule set and its computational complexity. We show that the minimum tuple deletion problem is NP-hard to approximate the minimum tuple deletion within \(\frac{17}{16}\) if given three functional dependencies and four attributes involved. A near optimal approximated algorithm for computing the minimum tuple deletion is proposed with a ratio of \(2 - \frac{1}{r}\), where \(r\) is the number of given functional dependencies. To guide the data repairing, this dissertation also investigates the data repairing method by using query feedbacks, formally studies two decision problems, functional dependency restricted deletion and insertion propagation problem, corresponding to the feedbacks of deletion and insertion. A comprehensive analysis on both combined and data complexity of the cases is provided by considering different relational operators and feedback types. We have
identified the intractable and tractable cases to picture the complexity hierarchy of these problems, and provided the efficient algorithm on these tractable cases. Two improvements are proposed, one focuses on figuring out the minimum vertex cover in conflict graph to improve the upper bound of tuple deletion problem, and the other one is a better dichotomy for deletion and insertion propagation problems at the absence of functional dependencies from the point of respectively considering data, combined and parameterized complexities.

INDEX WORDS: Consistency Evaluation, Inconsistent Data Repairing, Functional Dependency, Query Feedback, Deletion Propagation, Insertion Propagation, Inconsistent Data, Algorithm, Computational Complexity, Databases
COMPUTATIONAL COMPLEXITY AND ALGORITHMS FOR DIRTY DATA EVALUATION AND REPAIRING

by

Dongjing Miao

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy
in the College of Arts and Sciences
Georgia State University

2018
COMPUTATIONAL COMPLEXITY AND ALGORITHMS FOR DIRTY DATA EVALUATION
AND REPAIRING

by

DONGJING MIAO

Committee Chair: Dr. Yingshu Li
Committee: Dr. Kai Zhao
         Dr. Wei Li
         Dr. Zhipeng Cai

Electronic Version Approved:

Office of Graduate Studies
College of Arts and Sciences
Georgia State University
March 2018
DEDICATION

This dissertation is dedicated to my friends.
# TABLE OF CONTENTS

**LIST OF TABLES** ......................................................... vii

**LIST OF FIGURES** ....................................................... viii

**Chapter 1  INTRODUCTION** ............................................. 1

1.1 Background ........................................................... 1
1.2 Survey of Current Research .......................................... 3
1.3 Research context and contributions .................................... 10

**Chapter 2  DATA INCONSISTENCY EVALUATION** ................... 12

2.1 Introduction .......................................................... 12
2.2 Conflict graph and its vertex cover .................................. 15
2.3 literature review of the vertex cover problem ...................... 16
2.4 Complexity and Inapproximation ..................................... 18
2.5 A \( (2 - \frac{1}{\ln r}) \)-approximation .................................. 24
   2.5.1 A basic approximation algorithm ............................. 25
   2.5.2 Improve the approximation by triangle eliminating .......... 27
   2.5.3 Near optimality .................................................. 28
2.6 A modified linear program .......................................... 31
2.7 Conclusion ............................................................ 36

**Chapter 3  VIEW PROPAGATION FOR DATA REPAIR VALIDATION** 37

3.1 Introduction .......................................................... 37
3.2 Preparation ........................................................... 40
3.3 Complexity results ................................................... 41
   3.3.1 Deletion Propagation ............................................. 42
   3.3.2 Insertion Propagation ............................................. 47
LIST OF TABLES

Table 1.1  An instance of employee information ................................................. 2
Table 2.1  Results summary in this paper ............................................................ 24
Table 3.1  Notations for this chapter ................................................................. 41
Table 3.2  Results in this paper .......................................................................... 47
Table 5.1  Polynomial tractable cases of $S$-set ................................................. 79
Table 5.2  Hard cases of $S$-set .......................................................................... 79
Table 5.3  Polynomial tractable cases of $V$-set ................................................. 79
Table 5.4  Hard cases of $V$-set .......................................................................... 80
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Relations of data dependencies</td>
<td>4</td>
</tr>
<tr>
<td>2.1</td>
<td>Example for a conflict graph built by two forests</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>Reduction for the instance $(x_1 + x_2 + x_3)(x_1 + \overline{x}_2 + \overline{x}_3)(\overline{x}_1 + x_2 + \overline{x}_3)$</td>
<td>22</td>
</tr>
<tr>
<td>2.3</td>
<td>Example for a 3-local biclique coloring</td>
<td>29</td>
</tr>
<tr>
<td>2.4</td>
<td>Conflict graph constructed in the proof of Proposition 1.3</td>
<td>30</td>
</tr>
<tr>
<td>2.5</td>
<td>Limitation of linear inequality</td>
<td>31</td>
</tr>
<tr>
<td>2.6</td>
<td>Example of a wheel complete multipartite</td>
<td>32</td>
</tr>
<tr>
<td>3.1</td>
<td>Example for the reduction of Theorem 10</td>
<td>45</td>
</tr>
<tr>
<td>3.2</td>
<td>Example for the reduction of theorem 15</td>
<td>52</td>
</tr>
<tr>
<td>3.3</td>
<td>Example reduction of Theorem 16</td>
<td>55</td>
</tr>
<tr>
<td>4.1</td>
<td>Example data and the result of query $Q := \pi_{f_1.uid,f_3.fid}(\sigma_{f_1.uid=u_0 \land f_1.fid=f_2.uid \land f_2.fid=f_3.uid}(f_1 \times f_2 \times f_3))$ which is finding all persons that user $u_0$ is able to get in touch with by at most two men-in-middle, figure at right side shows the graph represented by this database. $(u_0,u_3)$ is the view deletion from the result of query $Q$.</td>
<td>59</td>
</tr>
<tr>
<td>4.2</td>
<td>The data complexity of bounded, general, $fd$-restricted deletion propagation</td>
<td>61</td>
</tr>
<tr>
<td>4.3</td>
<td>Example for the reduction of step i, ii and iii 17, input instance of SAT-UNSAT $\phi = (x_1 + x_2 + x_3)(x_2 + \overline{x}_3 + x_4)(\overline{x}_4 + \overline{x}_5 + x_6)$ and $\phi' = (x'_1 + x'_2 + x'_3)(x'_3 + x'_4 + \overline{x}'_5)(x'_4 + \overline{x}'_5 + \overline{x}'_6 + \overline{x}'_7)$. Gray shaded tuples is permitted source deletion. Underlined tuple is the view deletion.</td>
<td>67</td>
</tr>
<tr>
<td>5.1</td>
<td>Example database instance</td>
<td>76</td>
</tr>
<tr>
<td>5.2</td>
<td>The witness graph for Example 4</td>
<td>92</td>
</tr>
<tr>
<td>5.3</td>
<td>The first possible gadget for the instance of $g(u)$</td>
<td>93</td>
</tr>
<tr>
<td>5.4</td>
<td>The second possible gadget for the instance of $g(u)$</td>
<td>93</td>
</tr>
<tr>
<td>5.5</td>
<td>The third possible gadget for the instance of $g(u)$</td>
<td>93</td>
</tr>
<tr>
<td>5.6</td>
<td>The fourth possible gadget for the instance of $g(u)$</td>
<td>94</td>
</tr>
<tr>
<td>5.7</td>
<td>Four possible visualizations of gadget $g(u)$</td>
<td>94</td>
</tr>
</tbody>
</table>
Figure 5.8  Connecting the gadgets of two adjacent vertices ........................ 94
Figure 5.9  Local view of connecting two gadgets ....................................... 96
Figure 5.10  Conflict resulting from odd circle of connecting gadgets .............. 96
Figure 5.11  At most 4 vertices could be added ............................................ 98
Chapter 1

INTRODUCTION

Data is always not available in real database applications. This kind of data is usually called low quality or dirty data, and shows up in general as inconsistent, inaccuracy, incomplete and stale. These problems greatly decrease the usability of the data in practical application, damage the real value of the data, and even lead to serious consequences in many fields. According to statistics, in the US banking industry, credit card fraud caused $4.8 billion in losses in 2006 because the inconsistent data was undetected. What’s worse, dirty data is not rare, but widespread. According to the statistics of authoritative institutions, there are various errors and inaccuracies in 1% - 30% of the data in American enterprise information system, and 13.6% - 81% of the key data in American medical information system is incomplete or stale. A survey by Gartner, a leading international technology consultancy, found that data from more than 25% of corporate information systems in global fortune 1000 companies were inaccurate or incorrect. There are still data from dirty source such as wireless sensor networks for data collecting always compress data [1–4]. In conclusion, the low quality of data is an important and urgent problem.

1.1 Background

Inconsistent data is the most typical dirty data and has been long time studied. Recall the development of traditional database management systems, it is always considered that how to design the language to express queries and how to do the query evaluation and optimization. The quality of the data is not considered, therefore whenever a user submits a query, the answer of the query returned by database management system is always assumed correct. However, if the data is dirty, no matter how carefully the query is designed and how good the query is optimized, the correctness of the query result cannot be guaranteed.

In database community, data dependency rules are widely used in the study of data consistency, and as the main way to improve the ability of describing data consistency. At first, functional dependencies and other forms of data dependency rules were introduced into the database system
to normalize the relational schema \[5\]. While at the presence of large amount of dirty data, especially the inconsistent data, it is more important to consider how to express and normalize data semantic other than the relational schema normalization. Extending these classic data dependence rules in form, especially functional dependencies, is still a good way to capture data semantic errors, such as conditional function dependencies and so on. Based on these extensions, a number of methods has been developed to improve the quality of data. In particular, in the case of relational data, that is, a set of declarative semantic constraints (such as functional or conditional functional dependencies) are first specified for the database, so that once inconsistent data occurs, some corresponding constraints are violated. Compared with other methods, the method based on the data dependence rules has several obvious advantages, such as it is essentially a declarative method, and can conduct automatic reasoning. More important, this kind of rules depicts the semantics of data in essential, so that it is capable to capture semantic errors in data. It is also expected in this way to provide a systematic method to deduce the semantics of data, derive and discover data quality rules \[6, 7\].

For example, each employee is represented as a relation tuple including employee’s name (FN: first name, LN: last name), work phone (CC: country code, AC: area code, phn: phone number), work address (street, city).

<table>
<thead>
<tr>
<th>Table 1.1. An instance of employee information</th>
</tr>
</thead>
<tbody>
<tr>
<td>FN</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>t1</td>
</tr>
<tr>
<td>t2</td>
</tr>
</tbody>
</table>

Consider an aggregation query \( Q \) asking the number of employees named “Mary”, note that the answer of \( Q \) is 2. However, it is possible that both the tuples are recording the information of a same person, so that the truth ground should be 1.

In this case, if set a functional dependency “fd : CC, AC, phn→ FN, LN” saying that each phone number could be owned by only one employee, then the answer of query \( Q \) is known to be 1.

Such a rule-based method expresses the inconsistency data and improves the usability of large-scale data. Unfortunately, this processing method based on the data dependency rules to describe data consistency development is still not perfect enough, and the method itself has a lot of inherent defects, mainly has the following points: (1) consistency rules can only be used to detect
inconsistency errors, but cannot provide an intuitive review on the degree of the inconsistency of
given data; (2) consistency rules cannot guide the data repairing, automatic repairing method based
on rules cannot ensure their repair derived is reliable, such as in the example above, even if the
function dependency rules is a strong constraint, still not sure how to fix the LN fields, namely
choose "Smith" or "Luth"; (3) it is difficult to establish sufficient consistency rules to fully detect
the consistency errors of data. As a result, more effective ways must be proposed to compensate for
these shortcomings, to develop and complement the data consistency evaluation and repair methods
based on dependency rules, make sure they are able to return the correct or at least with a certain
quality assurance.

1.2 Survey of Current Research

It has been a long history of research on data consistency, especially for the fundamental
theory, key techniques and systems on inconsistency detection, and repairing based on the semantic
constraints.

**Semantic constraints.** The first problem to improve data consistency is to define inconsis-
tency. Unified frameworks based on declarative semantic constraints (a.k.a, data quality rules) has
been introduced to deal with inconsistency problems by database and artificial intelligence commu-
nity. Roughly speaking, the widely applied constraint hierarchy can be shown as the figure 1.1 of
secondary relationship.

Given a relational schema $R = \{R_1, \ldots, R_c\}$, all the existing languages to define constraints,
excepting domain independent first order logic declarative sentences, can be written as a unified
form,

$$\forall \bar{x} (\varphi(\bar{x}) \land \beta(x) \rightarrow \bigvee_{i=1}^{n} \exists \bar{y}_i \Phi_i(\bar{x}, \bar{y}_i))$$

where $\varphi$ and $\Phi_i$ is relational atoms like $R_i(x_1, \ldots, x_h)$, and the equality atoms like $x = x'$. 
To define it strictly, $x, x', x_1, \ldots, x_h$ should occur in the declarative sentences, $\beta$ is a quantifier-free
formula embedded with predicates only. If a constraint or dependency has no existential quantifier in
it, then it is called full or universal constraint (UC). If the $\beta$ is empty in a constraint or dependency
has no existential quantifier in it, then it is called disjunctive tuple generating dependency (-tgd),
and if there is no equality atoms, *e.g.* $n = 1$, then it is called tuple generating dependency (tgd). Besides this, if the right part of a dependency is a single equality atom, then it is called an equality generating dependency (egd). If the $\varphi$ contains only two atoms with the same symbol in an equality generating dependency, then it is the classical dependency, “functional dependency (fd)” [8].

On the other hand, if the $\varphi$ in a tuple generating dependency is a single atom, then such type of dependency is a Local-As-View tuple generating dependency (LAV tgd), correspondingly, there are still other dual forms. If $\Phi_1$ is a single atom, the dependency is the classical dependency, “inclusion dependency (ind)” [9]. In addition, when the right part of the unified form is empty, *i.e.* the sentence is now $\forall \bar{x}(\varphi(\bar{x}) \land \beta(x))$, then the dependency is a denial constraint (denial). More other types of constraints could be found in literatures as [8].

Moreover, this dissertation is aware of the canonicity and arity of a dependency. A canonical dependency is a dependency such that there is a column specifying schema, where each variable in relation atoms only locates their position specified, and each equality atom contains only one pair of variables at the same position. And a $k$-ary dependency contains $k$ relation atoms in $\varphi$. 

---

Figure 1.1. Relations of data dependencies
(Conditional) functional dependency. It is easy to observe that data dependencies above is defined on relation atoms, variables and equalities, but not on constant. To capture errors in real world data, a novel constraints, conditional functional dependency, proposed in literature [6]. It modifies functional dependencies by adding a pattern tabular, advanced such well-developed constraints, and has been widely used in database community. However, due to the high complexity of related reasoning problems (e.g., it is EXPTIME-complete for full dependencies [5]), not all types of dependencies can be modified in this way. The key point is the balance between expressive abilities and computational complexities, conditional functional dependencies reach this point in some degree, thus being widely applied. Formally, a conditional functional dependency defined on schema $R$ is a pair $\varphi(x \rightarrow Y, T_p)$, where $X \rightarrow Y$ is a canonical functional dependency as stated in [8]. $T_p$ is table defined on attribute sets $X$ and $Y$, called pattern tabular, where for each attribute $A \in X \cup Y$ and each pattern tuple $t_p \in T_p$, $t_p[A]$ is either a constant taken from the domain of $A$, or a nameless variable “\_” matching any possible value.

In order to guarantee the practical use of this kind of rule, the fundamental problem involved here is that some classical decision problems related to conditional function dependencies need to be solved.

For example, in the finite database instance, the increase of expressive abilities raises the complexity of the reasoning problem, namely the finite inference problem [5,8]. The second is the satisfiable decision problem. When detecting inconsistency in real data using conditional function depends, the first question is whether a given set of conditional functional dependencies is conflicting or inconsistent with itself, if the set of dependencies is unsatisfiable or inconsistent, then it is necessary to correct them in the data, and find out where the rules went wrong or why. The intractability of satisfiability decision and reasoning problem has been shown in [10]. Finite axiomatization of conditional functional dependencies has been proven in [11], and the completeness and correctness of inference system are also proposed. Meanwhile, the tractable cases and approximation algorithms are also provided, that is, for a set of arbitrary conditional functional dependencies defined on a given schema $R$, in the case of infinite domain, it can be done in $O(|\Sigma|^2)$ to determine whether the set $\Sigma$ is satisfiable. Therefore, when we consider the data inconsistency related problems and methods, key based constraints or (conditional) functional dependencies are always used as the criterions to study the lower bound of the computational complexity of these decision problems.
The problems studied in this dissertation are defined by (conditional) functional dependencies, and so does the theory developed in this dissertation.

**Data inconsistency detection.** There is few work on the data inconsistency detection, the detection method in the real world distributed system is provided in [12], and the incremental detection method is studied in [13], aiming to find the minimum necessary data transmission to detect the change of conflicting sets, in order to avoid the redundant computation. However, no matter horizontal or vertical partitioning, this problem is proven to be NP-complete. Nevertheless, the necessary data transmission has been bounded in [13], computation and transmit cost is linear to the incremental of change and independent from database instance, and an optimized partition method is provided.

**Inconsistent data repairing** Optimal data repairing and consistent query answer are the most popular approaches to deal with violations of FDs and other integrity constraints. The former aim to find a repair with a minimum modifications on the given database, including minimally differs from the original one (e.g., [14], [15], [16], [17]), minimize the description length (e.g., [18]) and so on. The limitation of them is that there may be many different optimal repairs. The latter aims to find answer of a query that are true in every possible repair. It usually employ techniques of condensed representation of possible repairs(e.g., [19], [20]) or query rewriting (e.g., [21], [22]) to obtain consistent answer. Unfortunately, there are lots of classes of queries have to be answered approximately. Sampling repairs is an alternative approach proposed to overcome several drawbacks of optimal repairing and consistent query answering. It is to generate a sample of possible repairs of the input database under some repair semantic, moreover, it will return empty query results when no consistent answers are found, such as [23] proposed three classes of repairs and the corresponding sampling algorithm.

Different from the existing work, this dissertation consider using user query feedback to guide how to resolve conflicts so that a more reasonable sampling space could be obtained. Comparing with the preferred repair in [24], query feedback restricted repair defined in this dissertation has a stronger expressive ability. Another benefit is that source data could not be read by users directly in this way, so that it avoids the difficulties by privacy protection mechanisms [25–32]. Moreover, we focus on sampling repairs, not the consistent query answering. We study the complexity of repair
existence problem not the repair checking problem that whether a given instance is a repair of the input instance \[33\].

Another related problems is view update problem that given a view and an update against a view, the problem is to translate the update into a corresponding update against the base data, see \[34\]. There are several complexity bounds are known on relational view updates, \[35\], \[36\]–\[38\] give out the tractability and intractability results of finding a minimal view complement for relational views. Cong et al. \[39\] gave out the complexity of view update analysis under key preserving condition which can not be extended to ours. There are lots of works on the algorithms for translating view update to base table update, such as \[40\], \[42\] and so on. In modern scenarios, the update or query feedback comes from everywhere especially the crowd-sourcing \[43\], \[44\] and the truthfulness could be guaranteed by lots of methods presented \[28\], \[45\], \[46\]. Their goal was to define correctness properties of these translations and to characterize precisely the conditions for the existence of translations possessing these properties. There are still similar works trying to deal with more complicated queries, such as iceberg \[47\] and aggregation \[48\]–\[50\]. Different from our work, the database they considered is consistent, it is not inconsistent, so that the presence of FDs and other integrity constrains simplifies view update problem which it is in contrary to this dissertation.

There have been some complexity results on the view side effect problem for insertion \[51\], \[52\] and deletion propagation \[39\], \[52\], \[56\], moreover, on the data complexity of deletion propagation, B. Kimfield et al. \[54\] showed the dichotomy ‘head domination’ for every conjunctive query without self joins, deletion propagation is either APX-hard or solvable (in polynomial time) by the unidimensional algorithm, they also showed the dichotomy ‘functional head domination’ \[55\] using functional dependencies to simply the general deletion propagation problem, further more, they especially showed the trichotomy for group deletion a more general case \[56\]. On the combined complexity of deletion propagation, \[39\], \[52\] showed the variety results for different combination of relational algebraic operators and showed the tractable and intractable results on both data and combined complexity aspects. All the previous results showed that, for general cases, the deletion propagation is hard due to the huge searching space. Conclusions in these works are still not precise, and no results for bounded case, while we will show the precise complexity class where the general deletion and insertion propagation problem is, and where the bounded case is in this dissertation.
There are still related works including view update, data provenance, explanations in databases. As a classical problem, view update research has been studied several decades ago [35, 39, 40, 57–59]. It aims to find the set of operations that could modify the database in order to obtain a certain update on the view exactly. Deletion propagation problem we studies actually is a special cases of view updates. Nevertheless, the concern of previous research are mainly on find the exactly operation set. When it comes to 2000s, data provenance has been studied the ways to model relation between the input and output for database queries [60–63], mainly focusing on explaining ‘Why-provenance’ and ‘Where-provenance’. A variant of ‘Why-provenance’ is resilience [64], which defined as the minimum set of source tuples whose deletion making the given query false. Causality and responsibility are also based on similar idea [65–68].

In the applications dealing with data quality [67–75], they try to produce explanations for query results, in order to trace errors in the source data, such as inconsistencies, incompleteness or some other kind of errors. In fact, all the methods mentioned above can be used to do this, and there are several existing frameworks [66, 76]. Some work not only try to provide explanation, but also try to modify queries [77, 78], this kind of work actually deal with the view update problem on low quality data. To deal with the incompleteness of data, some work [67, 68, 74, 75, 79] try to explain the missing query results, it is also a variant of the insertion case studied in this dissertation. The other relevant quality

Dependency propagation is another related problem, it is to determine that given a view defined on data sources and a set of dependencies on the sources, whether another dependency is guaranteed to hold on the view, e.g., [80], [81] which are the first to investigate dependency propagation. Fan et al [82] extended [80], [81] by providing complexity bounds for FD propagation in the general setting, and for CFD propagation. However, this is a problem different from ours.

Reviews on vertex covers The related work is vast, and its application is very wide [83], however, we only mention those very related. For any $\varepsilon > 0$, minimum vertex cover might be hard to be approximated within $2-\varepsilon$ [84], this is its UGC-hardness. For any $\varepsilon > 0$, $7/6-\varepsilon$ inapproximability for minimum vertex cover has been proved by Hastad et al. [85], and this factor was improved by Dinur and Safra [86] to 1.36, this is its NP-hardness. A practical 2-approximation algorithm of minimum vertex cover was provided by Gavril et al. [87], it is to find a maximal matching of a graph and output all vertices in the matching, since the size of any maximal matching of a graph
is always a 2-approximation of its minimum vertex cover. For complete multipartite graphs, there are few works about computing its vertex cover or maximal matching. A formula of computing maximum matching size has been developed in \[88\].

On the other side, the vertex cover problem on general graph is approximable within \(2 - \frac{\log \log |V|}{2 \log |V|}\) \[89\] and \(2 - \frac{\ln \ln |V|}{2 \ln |V|} \left(1 - o\left(1\right)\right)\) \[90\]. Karakostas reduced the approximation factor to \(2 - \Theta\left(\frac{1}{\sqrt{\log |V|}}\right)\) \[91\] instead of the previous \(2 - \frac{\log \log |V|}{2 \log |V|}\). We mention that several simple 2-approximation algorithms are known. Half-integral solution \[92\] is a well-known approach to obtain a better approximation.

The algorithm uses a k-coloring of the subgraph induced by the half value vertices as input to reduce approximation ratio into \(2 - \frac{k}{2}\). For graphs with degree bounded by \(d\), based on Brook’s theorem, this directly leads to a \(\left(2 - \frac{2}{d}\right)\)-approximation. As referred above, the basic approach has been improved to \(2 - \Theta\left(\frac{1}{\sqrt{\log |V|}}\right)\) \[91\] by replacing the LP relaxation to SDP relaxation. We also mentioned the recent work on local biclique coloring given by F. Kuhn \[93\]. They generalize the result on bounded degree obtained by SDP relaxation to bounded local chromatic number. It can be guaranteed that any valid coloring with \(k\) colors also is a local \(k\)-coloring. Some other works on dominating sets \[94, 96\] are also related to this dissertation but solutions cannot be applied directly.

**Brief summary** The current work can be summarized as the following both sides. Firstly, there is no work that can be applied to conduct the consistency evaluation of dirty data directly. Modifying the existing method based on confidence of individual rule is a possible way to evaluate the data consistency, but it can be severely restricted by locality. Secondly, for the inconsistent data repairing, the existing methods which try to repair dirty data automatically are far more limited in scope, the consensus in the data quality research community is that artificial feedbacks are the keys to get high quality repair which should be introduced into repairing framework. However, some work assumes that (1) the artificial answer is correct and can be used directly, (2) the artificial feedbacks can be translated to the data without side effects, and (3) the given rules are correct. These assumptions are unreasonable in practice.

Therefore, this dissertation studies an alternative way to evaluate the degree of data inconsistency by ”minimum tuple deletion sets”, and then the computational complexity of the problem is considered, and several linear programming based algorithms with good approximation ratios are proposed to solve the data inconsistency evaluation problem. And then, this dissertation studies how to guarantee the correctness of artificial feedback when these assumptions are incorrect. To
solve this problem, two decision problems are formally formulated, namely the view deletion and insertion propagation problems restricted by integrity constraints. Furthermore, the complexities and corresponding decision algorithms of both problems are studied.

1.3 Research context and contributions

As stated above, improving inconsistent data is of great importance for increasing data availability and ensuring the use of large scale data. However, the existing processing method of data consistency based on data dependency rules is not well developed, and there are many inherent defects in this method as followings, (1) the integrity rules can be used to detect inconsistencies only, but it cannot provide an intuition of the degree of data inconsistency; (2) consistency rules cannot guide the repair completely, and the automatic repairing method based on the rules cannot guarantee a reliable and complete repair of inconsistent data; (3) it is difficult to declare sufficient rules enough to detect all the consistent errors in the dirty data.

To solve these problems, this dissertation studies the key problem of computational complexity and algorithm for data evaluation and repairing. This dissertation first studies the computational complexity of and algorithms for the database inconsistency evaluation. We define and use the minimum tuple deletion to evaluate the database inconsistency. For such minimum tuple deletion problem, we study the relationship between the size of rule set and its computational complexity. We show that the minimum tuple deletion problem is still NP-complete, if given two conditional functional dependencies and three attributes involved in them; And it is NP-hard to approximate the minimum tuple deletion problem within $\frac{17}{16}$ if given three conditional functional dependencies and four attributes involved in them. We design a near optimal approximated algorithm for computing the minimum tuple deletion, the ratio is $2 - \frac{1}{2^r}$, where $r$ is the number of conditional functional dependencies in the given rule set $\Sigma$. Under the unique gaming conjecture, this ratio is near optimal, its hard to improve it with a constant independent of $n$.

To guide the data repairing, this dissertation also investigates the data repairing method by using query feedbacks, formally studies two decision problems, functional dependency restricted deletion and insertion propagation problem, corresponding to the feedbacks of deletion and insertion. A comprehensive analysis on both combined and data complexity of the cases is provided by
considering different relational operators and feedback types. We have identified the intractable and tractable cases to picture the complexity hierarchy of these problems, and provided the efficient algorithm on these tractable cases. Last but not the least, we propose the potential future research topics in order to complete this dissertation. For the future work, we will study the precise complexity of the constant $b$-DP problem. We believe it is in $\Delta^P_2$-complete instead of $P^{NP[O(log n)]}$. Furthermore, we will design an efficient practical algorithm for $b$-DP to improve the naive one on the aspect of data complexity.
Chapter 2

DATA INCONSISTENCY EVALUATION

Managing inconsistent data is a core problem in the area of data quality management. Inconsistent data indicates that there is conflicted information in the data, which can be formalized as the violations of given semantic constraints. For relational data, usually, dependencies are utilized to capture the inconsistencies of data.

2.1 Introduction

There are many kinds of dependencies in database theory. Given a relational database schema $\mathcal{R} = \{R_1, \ldots, R_c\}$, a formal form of general dependency can be specified as a first-order logic sentence as follows.

$$\forall x_1, \ldots, \forall x_n [\varphi(x_1, \ldots, x_n)] \rightarrow \exists z_1, \ldots, \exists z_k \psi(y_1, \ldots, y_m)$$

Here, the variables in $\psi$ are taken from $\{x_i\}$ and $\{z_j\}$, both $\varphi$ and $\psi$ are conjunctions of relation atoms of the form $R_l(w_1, \ldots, w_h)$ and equality atoms of the form $w = w'$, where $R_l$ is a relation in schema $\mathcal{R}$ and each of the $w, w', w_1, \ldots, w_h$ is a variable appearing in the sentence. Furthermore, the dependencies can be classified from following aspects. (i) Full versus embedded. A full dependency is a dependency that has no existential quantifiers. (ii) Tuple generating versus equality generating. A tuple generating dependency (tg) is a dependency in which no equality atoms occur, while an equality generating dependency (eg) is a dependency for which the right-hand formula is a single equality atom. (iii) Typed versus untyped. A dependency is typed if there is an assignment of variables to column positions such that variables in relation atoms occur only in their assigned position, and each equality atom involves a pair of variables assigned to the same position. (iv) $k$-ary. A $k$-ary dependency contains $k$ relation atoms in $\varphi$.

Several kinds of dependencies have been successfully applied in the area of inconsistent data management, such as functional dependency (FD for short) [1], conditional functional dependency
(CFD for short) \[34\] and so on. Most of them can be classified into \textit{full}, \textit{equality generating}, \textit{typed} and \textit{2-ary} dependencies, and the reason is easy to understand after observing that the inconsistency is usually explained to be the conflict between specified tuples (usually 2 tuples) when some conditions are satisfied. CFD is a typical example of such dependencies utilized in capturing inconsistencies. For example, a CFD constraint may say, for two tuples in relation \( R \), if their values on attribute \( A \) are both 1 and they have equal values on attributes \( B \) and \( C \), they must have equal values on attribute \( D \) also. Given the above CFD constraint, it is expected that the constraint is valid on all possible data instances. Then, if two tuples \( t_1 = \{ A : 1, B : x, C : abc, D : m \} \) and \( t_2 = \{ A : 1, B : x, C : abc, D : n \} \) are found in the real instance, it is easy to verify that \( t_1 \) and \( t_2 \) violate the given constraint. That is, inconsistent data are found according to the given semantic constraints.

**Example.** An FD constraint \( AB \rightarrow C \) defined over relation \( R \) says, “for two tuples in relation \( R \), if their have equal values on attributes \( A \) and \( B \), they must have equal values on attribute \( C \) also”. Given the above FD constraint, it is expected that the constraint is valid on all possible data instances. Then, if two tuples \( t_1 = \{ A : x, B : abc, C : m \} \) and \( t_2 = \{ A : x, B : abc, C : n \} \) are found in the real instance, it is easy to verify that \( t_1 \) and \( t_2 \) violate the given constraint. That is, \textit{inconsistent data} are found according to the given semantic constraints.

Based on dependency theory, many research works focusing on managing inconsistent data have appeared. For example, \[11\], \[13\], \[97\] and \[10\] study the inconsistency detection problem, and \[14\], \[15\], \[16\], \[17\] and \[18\] study the data repairing problem. However, one important problem, called inconsistency evaluation, has not been well studied.

Rather than simply providing an consistency measure, the aim of evaluating inconsistency is to figure out the lower bound of the cost one should pay for data repairing based on only dependencies. In real life, given a (dirty) database instance and some consistency rules defined over its schema, it is possible that all the tuples should be repaired if without any useful, that is, the upper bound of data repairing is naive. However, if without enough knowledge on the dirty data, it is important to know the lower bound of repairing the data, so as to help user select data sources, determine whether or not to repair the data and so on. For this purpose, a natural lower bound is data inconsistency
defined over minimum number of tuple deletion, because any exact repair algorithm need to do such many changes on data at least, in order to eliminate all violations no matter what strategies applied. Essentially, it has a close relation with the conception tuple-deletion repair studied in some previous work. This kind of definition has been investigated in the area of repairing data and consistent query answering, and it is well known that it’s NP-hard to find minimum tuple deletion based repair generally or even given two FDs [17, 98, 99], and have a constant approximation [16]. However, we note that to evaluate the inconsistency of database, it is important to give an accuracy value of data inconsistency as much as possible, thus, we need to formally study the inconsistency evaluation problem, give a more careful analysis on the complexity and approximation ratio of it.

An important task focusing on managing inconsistent data is to evaluate how inconsistent the data is. A natural model is using the vertex cover size to show this value based on the input dependencies. Concretely, for an inconsistent database instance with respect to several given FDs, we model each tuple as a vertex in graph, and each tuple pair has an edge between them if they are conflict with each other. Then we can see that each FD will generate a set of disjoint complete multipartite graphs, because if a tuple has the same value on the left attributes with some other tuples, it must be conflict with other tuples with different right attributes’ values, i.e., it has edges with all of them. Finally, if we want to minimum tuple deletions to make it satisfying all the given FDs, we must compute the vertex cover of the sum graph of those generated by single FD, here the sum graph is just a conflict graph.

Therefore, in this scenario, vertex cover problem in conflict graph is the basis of evaluating data inconsistency and provide several theoretical analysis results. Since inconsistent data is defined under the help of dependencies, a specific kind of dependency, FD, is chosen in this dissertation when introducing our results in the following parts. In fact, all results in this paper can be extended to model the full, equality-generating, typed and 2-ary dependencies [34].

In this paper, we consider the problem of evaluating data inconsistency and provide several theoretical analysis results. Since inconsistent data is defined under the help of dependencies, a specific kind of dependency CFD is chosen when introducing our results in the following parts. In fact, all results in this paper can be extended to the full, equality-generating, typed and 2-ary dependencies.
Given a database instance $D$ including $n$ tuples and a dependency set $\Sigma$ including $r$ constraints, the inconsistency evaluation problem is formalized as computing the ratio of the minimum culprit size to $n$, where a culprit is defined to be a set of tuples leading to violations of constraints. First, it is proved that (i) if $\Sigma$ contains only one constraint, the inconsistency evaluation problem can be solved in polynomial time, and (ii) when $\Sigma$ contains two constraints, even if it is assumed that only 3 attributes are used and each tuple can be involved at most 6 violations, the inconsistency evaluation problem is still NP-complete.

Then, to investigate the hardness of designing approximation algorithms for this problem, only considering the simple scenario of $r = 3$ and 4 attributes, it is shown to be still NP-hard to find an algorithm with approximation ratio smaller than $\frac{17}{16}$. By directly using previous results on vertex cover problem, it is shown that finding 1.36-approximation algorithms is NP-hard even when the sizes of attributes and constraints are fixed, and there are no $(2 - \epsilon)$-approximation algorithms for any $\epsilon > 0$ unless the UGC(unique gaming conjecture) is false.

Finally, considering that in real applications, the number of constraints in $\Sigma$ are usually fixed, we design an approximation algorithm for that setting and show that the approximation ratio can be bounded by $2 - \frac{1}{r}$. Furthermore, the approximation algorithm is shown to be near optimal by proving that it is impossible to improve the ratio with any constant factor under the assumption of UGC. Actually, it is an approximation algorithm for minimum vertex cover on a special graph class, namely conflict graph. Comparing with the previous results, the results in this paper are obtained based on the characteristic of conflict graph.

### 2.2 Conflict graph and its vertex cover

In this paper, we study a restricted graph class, called conflict graph, which is defined as follows.

**Definition 2.2.1** (Conflict Graph). Given an vertex set $V$ and $r$ sets of graphs of disjoint complete multipartites $F_1, \ldots, F_r$ defined on $V$, such that each connected component is a multipartite graph in each $F_i$. The conflict graph $G(V,E)$ is an undirected graph where edge $(u,v) \in E$ if there is an $F_i$ such that edge $(u,v) \in F_i$. 
An example of conflict graph $G$ consisting of 2 sets of graphs of disjoint complete multipartites can be shown as fig. 2.5. In $G$, $F_1$ includes two components, both are complete bipartite, while $F_2$ includes there components, one is complete quadripartite and the other two are complete bipartite.

**Vertex Cover.** Vertex cover is one of the classical problems in graph theory: given a graph $G(V, E)$, find a vertex set of vertices in $V$, say $V'$, such that for each edge $(u, v)$ of $E$, at least one of $u$ and $v$ belongs to $V'$, and $V'$ has the minimize size. We want to study the vertex cover problem on conflict graph, including complexity and approximation bound.

### 2.3 literature review of the vertex cover problem

The related work is vast, and its application is very wide [83], however, we only mention those very related. For any $\varepsilon > 0$, minimum vertex cover might be hard to be approximated within $2 - \varepsilon$ [84], this is its UGC-hardness. For any $\varepsilon > 0$, $7/6 - \varepsilon$ inapproximability for minimum vertex cover has been proved by Hastad et al. [85], and this factor was improved by Dinur and Safra [86] to 1.36, this is its NP-hardness. A practical 2-approximation algorithm of minimum vertex cover was provided by Gavril et al. [87], it is to find a maximal matching of a graph and output all vertices in the matching, since the size of any maximal matching of a graph is always a 2-approximation of its minimum vertex cover. For complete multipartite graph, there are few works about computing its vertex cover or maximal matching. A formula of computing maximum matching size has been developed in [88].
On the other side, the vertex cover problem on general graph is approximable within $2 - \frac{\log \log |V|}{2 \log |V|}$ and $2 - \frac{\ln \ln |V|}{2 \ln |V|} \big(1 - o(1)\big)$ ~[89]~. Karakostas reduce the approximation factor to $2 - \Theta \left( \frac{1}{\sqrt{\log |V|}} \right)$ instead of the previous $2 - \frac{\log \log |V|}{2 \log |V|}$. We mention that several simple 2-approximation algorithms are known. Half-integral solution ~[92]~ is a well-known approach to obtain a better approximation. The algorithm uses a k-coloring of the subgraph induced by the half value vertices as input to reduce approximation ratio into $2 - \frac{k}{2}$. For graphs with degree bounded by $d$, based on Brook’s theorem, this directly leads to a $\left(2 - \frac{2}{d}\right)$-approximation. As referred above, the basic approach has been improved to $2 - \Theta \left( \frac{1}{\sqrt{\log |V|}} \right)$ ~[91]~ by replacing the LP relaxation to SDP relaxation. We also mentioned the recent work on local biclique coloring given by F. Kuhn ~[93]~. They generalize the result on bounded degree obtained by SDP relaxation to bounded local chromatic number. It can be guaranteed that any valid coloring with $k$ colors also is a local $k$-coloring. They proved the following theorem ~[93]~.

**Theorem 1.** Assuming the UGC, it is NP-hard to approximate the vertex cover problem in graphs for which a $(\Delta + 1)$-local coloring is given as input, within any constant factor better than $2 - \frac{2}{\Delta + 1}$. If the given coloring is also a biclique coloring, there will be a randomized polynomial-time algorithm with approximation ratio $2 - \Omega \left( \frac{\ln \ln \Delta}{\ln \Delta} \right)$.

In this paper, based on the above theorem, we show that it is hard to give a bounded local (biclique) coloring with size independent of $|V|$ to improve the ratio with use of the approach they gave and our approach seems good enough to solve vertex cover problem on conflict graph. We first prove that conflict graph is a non-trivial property when the number of sets of graphs of disjoint complete multipartites is fixed, i.e., if we fix the number $r$, it is always possible to find a graph can not be represented by sum of $r$ sets of graphs of disjoint complete multipartites. However, we proved that (i) If the number of sets of graphs of disjoint complete multipartites is fixed, finding 1.36-approximation is also NP-hard. (ii) Given 2 sets of graphs of disjoint complete multipartites and maximum degree less than 7, vertex cover problem of conflict graph is NP-complete. Without the degree restriction, it is shown to be still NP-hard to find an algorithm for vertex cover of conflict graph within $\frac{41}{40} - \varepsilon$, for any $\varepsilon > 0$. By directly using previous results on vertex cover problem, it is shown that it is NP-hard to obtain a $(2 - \varepsilon)$-approximation for any $\varepsilon > 0$ unless the UGC(unique gaming conjecture) does not hold.
Given conflict graph consisting of \( r \) sets of graphs of disjoint complete multipartites, we design an deterministic approximation algorithm and show that the approximation ratio can be bounded by \( 2 - \frac{1}{2^r} \) which is a bound does not depend on \(|V|\) but only \( r \) usually a small constant in real applications. Also, this bound is not a expected ratio of a random algorithm, but it cannot be improved by applying the related SDP approach [93]. Furthermore, the approximation algorithm is shown to be near optimal by proving that it is impossible to improve the ratio with any constant factor under the assumption of UGC. Actually, it is an approximation algorithm for vertex cover on a special graph class, namely conflict graph. Comparing with the previous results, the results in this paper are obtained based on the characteristic of conflict graph.

2.4 Complexity and Inapproximation

In the following, we study the complexity of vertex cover problem on conflict graph. First, a trivial case is the vertex cover on only one set of disjoint complete multipartite graphs.

**Property 1.** Minimum vertex cover in a set of disjoint complete multipartite graphs can be found polynomially.

**Proof.** Due to the definition of conflict graph, each two connected component are disjoint and there is no edge between them. Given a conflict graph \( G \), for each connected component (a complete multipartite graph), take all vertices of it into cover \( C \) excepting those of the largest part. At last, \( C \) is a minimum vertex cover of \( G \).

**Lemma 1.** Given any graph \( G \), it’s not always possible to find a collection of subgraphs such that

a) each subgraph is a complete \( k \)-partite graph for some \( k \),

b) each edge in occurs in at least one of the subgraphs, and

c) each vertex in occurs in at most a constant number of the subgraphs.

Moreover, the probability is close to 1.

**Proof.** Given \( n \), let’s take a random bipartite graph \( G = ([n], [n], E) \) where \( Pr [(v_i, v_j) \in E] = \frac{1}{2} \) for each pair \( (v_i, v_j) \). (The answer is no with probability close to 1.)
Since $G$ is bipartite, any complete $k$-partite subgraph has to be bipartite. First, we claim that with probability $1 - o(1)$, every complete bipartite subgraph in $G$ has at most $4n$ edges. For any pair of subsets $L \subseteq [n]$ and $R \subseteq [n]$ with $|L \times R| \geq 4n$, the probability that “the complete bipartite subgraph with edge set $L \times R$ is in $G$” is $2^{-|L \times R|}$, and it is no more than $2^{-4n}$. There are fewer than $2^n \times 2^n = 4^n$ such pairs $L$ and $R$, thus, by the naive union bound, the probability that any of the corresponding subgraphs is present in $G$ is at most $4n2^{-4n}$, and it is no more than $2^{-2n}$.

Also, with probability $1 - o(1)$, the graph $G$ has at least $n^2/4$ edges.

Thus, with probability $1 - o(1)$, every complete bipartite subgraph in $G$ has at most $4n$ edges, and $G$ has at least $n^2/4$ edges. Assume this happens.

Now suppose for contradiction that a collection of subgraphs with the desired properties exists. Each of the subgraphs has edge set $L \times R$ for some pair of subsets $L$ and $R$. In $G$, direct all the edges in $L \times R$ from the larger side to the smaller side (to the left if $|L| \leq |R|$, and to the right otherwise). Since $|L \times R| \leq 4n$, the smaller of $L$ or $R$ must have size at most $2\sqrt{n}$.

Since each vertex is in $O(1)$ of the subgraphs, each vertex now has $O(\sqrt{n})$ edges directed out of it. But all edges are directed one way or the other, so the number of edges in $G$ is at most the number of vertices times the maximum out-degree of any vertex, that is, at most $O(n\sqrt{n})$. This contradicts the graph having at least $n^2/4$ edges.

Lemma 1 shows that conflict graph is a non-trivial graph class under the constant number restriction. Inspired by this, we next have the following theorem on the complexity for a more restricted condition.

**Theorem 2.** Given 2 sets of graphs of disjoint complete multipartites and maximum degree no more than 6, decision version of vertex cover problem on conflict graph is NP-complete.

**Proof.** Thus the problem is in NP obviously. We give the proof of the lower bound as follow.

**NP-hardness.** The lower bound is established by a reduction from 3-SAT problem. An instance of 3-SAT problem includes a set $U$ of $n$ variables $x_1, ..., x_n$ and a collection $S$ of $m$ clauses $s_1, ..., s_m$, while in each clause $s_i = \alpha_{i1} + \alpha_{i2} + \alpha_{i3}$, each $\alpha_{ij}$ (1 $\leq$ $j$ $\leq$ 3) is the $j$-th literal of $s_i$. Given an instance of 3-SAT problem, it is to decide whether there is a satisfying truth assignment for $S$. The 3-SAT problem is NP-complete, and it remains NP-complete even if for each $x_i \in U$, there are at most 5 clauses in $S$ that contain either $x_i$ or $\overline{x_i}$. 
A polynomial reduction from 3-SAT can be constructed as follows.

Given an instance of 3-SAT with \( n \) variables and \( m \) clauses, let conflict graph \( G = F_1 + F_2 \) and \( k = n + 2m \). Then we introduce two sets of graphs of disjoint complete multipartites \( F_1 \) and \( F_2 \) share a vertex set with \( 2n + 3m \) vertices,

\( F_1 \): Set edge \((2i - 1, 2i)\) for each \( i \in [n] \), and edges \((3i - 2, 3i - 1), (3i - 2, 3i), (3i - 2, 3i - 1)\) (i.e., a triangle for each clause) for each \( i \in [m] \). That is, \( F_1 \) consists of \( n \) complete bipartite subgraphs and \( m \) complete tripartite subgraphs;

\( F_2 \): For each variable \( x_i \) of 3-SAT instance, build a complete bipartite subgraph \( L_i \times R_i \), let \( L_i \) includes vertex \( 2i - 1 \) and all vertices corresponding to the positive literals of \( x_i \), and \( R_i \) includes vertex \( 2i \) and all vertices corresponding to the negative literals of \( x_i \);

Note that, each vertex has a degree at most 7 assuming each variable occurs at most 5 clauses.

Suppose the 3-SAT instance is satisfiable, i.e., there is an satisfying truth assignment \( \rho : U \rightarrow \{0, 1\}^n \) for \( S \), then there is a vertex cover \( VC \) of \( G \) such that its size is at most \( n + 2m \). Concretely, it can be computed as follows, for each variable \( x_i \), (1) if \( \rho(x_i) = 1 \), delete vertex \( 2i \) from \( G \). And for each clause \( s_j \), if \( \alpha_{jq} \) is a positive literal of \( x_i \) and vertices \( 2n + 3j - 2, 2n + 3j - 1, 2n + 3j \) is currently in \( G \), delete vertex \( 2n + 3(j - 1) + q \) from \( G \); (2) if \( \rho(x_i) = 0 \), delete vertex \( 2i + 1 \) from \( G \). And for each clause \( s_j \), if \( \alpha_{jq} \) is a negative literal of \( x_i \) and vertices \( 2n + 3j - 2, 2n + 3j - 1, 2n + 3j \) is currently in \( G \), delete vertex \( 2n + 3(j - 1) + q \) from \( G \). We have that for each \( i \), either \( 2i \) or \( 2i + 1 \) is deleted from \( G \), and for each \( j \), either of \( \{2n + 3j - 2, 2n + 3j - 1, 2n + 3j\} \) is deleted from \( G \) for each \( j \). This is because in each clause, there is at least one literal that is made true by assignment \( \rho \). Therefore, there is a set \( VC \) of the rest tuples such that it is a cover and has a size no more than \( n + 2m = k \).

To see the converse, let \( VC \) is the cover such that \( |VC| \leq k = n + 2m \). To cover \( F_1 \), either \( 2i - 1 \) or \( 2i \) should be included in \( VC \) for each \( i \in [n] \), and at least two of \( 2n + 3j - 2, 2n + 3j - 1, 2n + 3j \) should be included in \( VC \). That is, the size of \( VC \) is at least \( n + 2m \). Thus, \( |VC| = n + 2m \). After covering \( F_2 \), at most one literal of each variable rests in \( G - VC \). Then, there is a satisfying truth
assignment $\tau$ for $S$ such that, for each $i \in [n],$

$$
\tau(x_i) = \begin{cases}
0, & \text{if } 2i - 1 \in V(G) - VC, \\
1, & \text{otherwise}
\end{cases}
$$

(2.1)

It is sure that $\tau$ will make all clauses true. Therefore, it is NP-complete, even if given only 2 set of disjoint complete multipartite graphs and vertex with at most 7 degree.

We next prove that vertex cover problem on conflict graph is Max-SNP-hard if without the degree restriction. To analyze the lower bound of approximation of vertex cover, we next use the same reduction in Theorem 2 above, but from MAX-E3SAT [100] to this problem if given 2 set of disjoint complete multipartite graphs. We know that unless P$\neq$NP, there is no polynomial-time algorithm approximates MAX-E3SAT with $\frac{7}{8} + \varepsilon$ [100] (we use the ratio notion less than 1 for maximizing problem) for any $\varepsilon > 0$. That is, there is no guarantee that ”more clause satisfied, more vertex preserved”. This is really because three free tuples are built for each clause in the reduction. Concretely, in that reduction, there may exist two an assignments $\tau_p$ and $\tau_q$ ($p > q$) where they makes $p$ and $q$ clauses true. However, in its corresponding instance, there may be $p'$ and $q'$ tuples can be preserved respectively where $p' < q'$.

Therefore, we give a linear reduction carefully designed by modifying the one used in Theorem 2.

**Theorem 3.** Vertex cover problem on conflict graph can not be approximated in $\frac{41}{40}$ if given 2 set of disjoint complete multipartite graphs.

**Proof.** This lower bound is established by a reduction from MAX-E3SAT whose instance includes a set $U$ of $n$ variables and a collection $S$ of $m$ disjunctive clauses of exactly 3 literals. Given an instance of MAX-E3SAT problem, it is to find a satisfying truth assignment maximizing the number of clauses satisfied by it.

We build the same reduction in the proof of Theorem 2 but only delete all the vertices $i \in [2n]$, shown in fig. 2.2. We also use the assignment function $\tau$ such that

$$
\tau(x_i) = \begin{cases}
0, & \text{if } 2i - 1 \in V(G) - VC, \\
1, & \text{otherwise}
\end{cases}
$$

(2.2)
Let $\#\tau$ is the number of clauses satisfied by any assignment $\tau$, let $\#\tau_{\text{max}}$ refer to the optimal assignment $\tau_{\text{max}}$ which maximizing the number of clauses satisfied in the MAX-E3SAT instance. Let $|V(G)|$ is the number of vertex in $G$.

\[ \begin{align*}
&C_{11} \quad C_{21} \quad C_{31} \\
&C_{12} \quad C_{13} \quad C_{22} \quad C_{23} \quad C_{32} \quad C_{33}
\end{align*} \]

Figure 2.2. Reduction for the instance $(x_1 + x_2 + x_3)(x_1 + \overline{x}_2 + \overline{x}_3)(\overline{x}_1 + x_2 + \overline{x}_3)$

We first consider $VC_{\text{min}}$ the minimum vertex cover of $G$. We claim that any pair of variable vertices cannot occurs in $VC_{\text{min}}$, because if not, there must be a smaller $VC'_{\text{min}}$ obtained by returning redundant variable vertices from $VC_{\text{min}}$ into $V(G) - VC_{\text{min}}$ without any edge left. Based on this, given the optimal assignment $\tau_{\text{max}}$ maximizing the number of clauses satisfied in the MAX-E3SAT instance, we have

\[ \#\tau_{\text{max}} = 3m - |VC_{\text{min}}| \quad (2.3) \]

And for any solution $VC$ of $V(G)$, we have

\[ \#\tau_{\text{vc}} = 3m - |VC| \quad (2.4) \]

Now, let $\hat{VC}$ is an $r$-approximation ($r > 1$) of minimum culprit $VC_{\text{min}}$ such that $|\hat{VC}| \leq r \cdot |VC_{\text{min}}|$. We have

\[ \frac{\#\tau_{\text{vc}}}{\#\tau_{\text{max}}} = \frac{3m - |\hat{VC}|}{3m - |VC_{\text{min}}|} > 1 + \frac{(1 - r) \cdot |VC_{\text{min}}|}{3m - |VC_{\text{min}}|} \quad (2.5) \]
Since each clause has exactly 3 literals and at least a half of clauses in any SAT instance can be satisfied by any variable assignment, we have

\[ |VC_{\text{min}}| \leq 3 \cdot \frac{m}{2} + 2 \cdot \frac{m}{2} \tag{2.6} \]

Apply this fact in the right hand of inequality 2.5 it is

\[ \frac{|VC_{\text{min}}|}{3m - |VC_{\text{min}}|} \leq 5 \tag{2.7} \]

Therefore we get

\[ \frac{\#_{\tau_{\text{vc}}}}{\#_{\tau_{\text{max}}}} > 1 + (1 - r) \cdot 5 \tag{2.8} \]

Then, apply this into inequality 2.5, then

\[ \frac{\#_{\tau_{\text{vc}}}}{\#_{\tau_{\text{max}}}} > 6 - 5r \tag{2.9} \]

That is, if minimum culprit can be approximated within ration \( r \), then MAX-E3SAT can be approximated within \( 6 - 5r \). However, if the former can be approximated within \( \frac{41}{3606} \), then the later will be approximated better than \( \frac{7}{8} \), and this contraries to the result of [100].

We next show that for the fixed number of subgraph, minimum vertex cover has a global inapproximability bound. The observation is that sum of fixed number of subgraphs is able to represent any bounded degree graph.

**Theorem 4.** For sufficiently large fixed number of sets of graphs of disjoint complete multipartites, it is NP-hard to approximate vertex cover on conflict graph within 1.3606.

**Proof.** Consider any input instance is a graph \( G(V, E) \) and each vertex has a degree constraint \( d \) \((d > 0)\), where each vertex in \( V \) has a degree no more than \( d \). It is easy to build \( d \) sets of graphs of disjoint complete multipartites to represent \( G \). For each vertex \( i \), we can easily distribute its each edge into \( d \) sets of graphs of disjoint complete multipartites one by one. For sufficiently large bounded degree \( d \), Dinur and Safra [86] has proved that it is NP-hard to approximate minimum vertex cover within any constant factor smaller than 1.3606. The number of sets of graphs of disjoint complete multipartites are \( O(d) \) in the reduction herein, this concludes the theorem. \( \square \)
The theorem gives an approximation lower bound for the case with fixed number of set of disjoint complete multipartite graphs. For the non-fixed case, it is able to encode arbitrary graphs with unbounded degree, just set $|E|$ subgraphs. Beyond the NP-hardness of 1.3606, Khot et. al [84] had proved that, for any $\varepsilon > 0$, there is no $(2 - \varepsilon)$-polynomial approximation, unless the unique games conjecture is false.

For the sake of clarity, we summary the hardness results of vertex cover problem in Table 2.1.

<table>
<thead>
<tr>
<th>Cases</th>
<th>Complexity/Inapproximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 subgraph</td>
<td>PTIME</td>
</tr>
<tr>
<td>2 subgraphs (with degree $\leq 7$)</td>
<td>$\frac{31}{40}$ (NP-complete)</td>
</tr>
<tr>
<td>Fixed number</td>
<td>1.3606</td>
</tr>
<tr>
<td>Non-fixed number</td>
<td>1.3606(NP-hardness), $2 - \varepsilon$ (UGC-hardness [84])</td>
</tr>
</tbody>
</table>

2.5 A $(2 - \frac{1}{2^r})$-approximation

In this section, we use the a linear programming based $(2 - \frac{1}{2^r})$-approximation for our input graph where $C$ is the number of connected components in the input graph, then improve the approximation.

We start by the following observations on the chromatic number of the input graph $G$, given $r$ graphs where each graph is a set of disjoint complete multipartite graphs, namely $F_1, ..., F_r$. We have the following useful observation of a set of disjoint complete multipartite graphs.

**Property 2.** If a graph $G$ is a sum of $C$ complete multipartite graphs, $k_i$-partite graph respectively, then it has a chromatic number $\prod k_i$. Moreover, this chromatic number is tight.

**Proof.** The graph $G$ is $\prod k_i$-colorable. We can do this in linear time by assigning a distinct color for each partite of each complete $k_i$-partite graph. A $k$-partite graph (assuming non-empty parts) is $k$-colorable (color all of the vertices in one part with the same color). A complete $k$-partite graph has chromatic number $k$ (clearly two vertices in two different parts require different colors). In fact, it has a maximal set of edges such that the graph has chromatic number (adding an edge within a part will force another color). If the graphs $g_1$ and $g_2$ are $k_1$ and $k_2$ colorable respectively, the sum
\( G = g_1 + g_2 \) is \( k_1k_2 \)-colorable by the product coloring. It follows that each \( g_i \) is \( k_i \)-colorable, and that the sum, \( G \) is \( \prod k_i \)-colorable.

This property is tight if we know nothing more about the graphs. For arbitrary \( n \), let \( p_1p_2 \cdots p_k \) be the prime factorization of \( n \) (where repeated primes are listed repeatedly). If \( G \) is the sum of \( k \) correctly-chosen complete \( p_i \)-partite graphs, \( G = K_n \). To choose \( G_1 \), let the parts be the residues modulo \( p_1 \). For \( G_2 \), let \( i \) be in a part according to the residue of \( \lfloor \frac{i}{p_1} \rfloor \) modulo \( p_2 \). Continuing this for each \( k \) makes the right subgraphs. The best way to imagine this is to express \( i \) has a number where the 1s place goes up to \( p_1 \), the \( p_1 \)s place (the next digit over) goes up to \( p_2 \), etc. Then \( G \) is complete because if \( i \neq j \) then The representations of \( i \) and \( j \) are different. 

\[ 2.5.1 \text{ A basic approximation algorithm} \]

We next combine this property and linear program relaxation to give a better approximation. Recall the classical linear program of vertex cover.

\[
\begin{align*}
\text{minimize} & \quad \sum_{i \in V} x_i \\
\text{subject to:} & \quad x_i + x_j \geq 1, \forall (i, j) \in E, \\
& \quad x_i \geq 0, i \in V
\end{align*}
\]

We can relax it with condition: \( x_i \in \{0, \frac{1}{2}, 1\} \), and we have that \( OPT_{relax} \leq OPT \). Then, we use a coloring on the input graph which can be found trivially based on Proposition. \[2\] to improve the approximation. We use a solution similar with \[92\] to improve the approximation as follow.
Algorithm 1

1: Solve the linear programming relaxation to obtain a solution $x'$ such that $x' \in \{0, \frac{1}{2}, 1\}$ for all $i \in V$.

2: Let $P_j$ is the set of vertices of color $j$

3: $j \leftarrow \arg \max_j |\{x_i' \mid i \in P_j \land x_i' = \frac{1}{2}\}|$

4: for each $i \in V$ do

5: if $x_i' = 1$ or ($x_i' = \frac{1}{2}$ and $i \notin P_j$) then

6: add $v_i$ into $\tilde{VC}$

7: return $\tilde{VC}$

First, $\text{OPT}_{\text{relax}}$ can be returned in polynomial time as shown in [101], and it is easy to see that $\tilde{VC}$ is a cover. Actually, if an $x_i' = \frac{1}{2}$ and it is not chosen into $\tilde{VC}$, then its neighbor must be added into $\tilde{VC}$ since its neighbor is not in $P_j$, and the sum of two adjacent variables is at least 1.

Second, we claim that the approximation ratio is $2 \left(1 - \frac{1}{\prod k_i}\right)$.

Lemma 2. Algorithm 1 returns a $2 \left(1 - \frac{1}{\prod k_i}\right)$-approximation.

Proof. Let $S_1$ be the set $\{x_i' \mid i \in V, x_i' = 1\}$, $S_\frac{1}{2}$ be the set $\{x_i' \mid i \in V, x_i' = \frac{1}{2}\}$, and $S_{P_j}$ be the set $\{x_i' \mid i \in P_j, x_i' = \frac{1}{2}\}$. Obviously, $\text{OPT}_{\text{relax}} = \sum_{i \in S_1 \cup S_\frac{1}{2}} x_i' \leq \text{OPT}$. We have,

$$|\tilde{VC}| \leq |S_1| + |S_\frac{1}{2}| - |S_{P_j}| \quad (2.13)$$

$$= \sum_{i \in S_1} x_i' + 2 \sum_{i \in S_\frac{1}{2}} x_i' - 2 \sum_{i \in S_{P_j}} x_i' \quad (2.14)$$

$$\leq \sum_{i \in S_1} x_i' + 2 \sum_{i \in S_\frac{1}{2}} x_i' - 2 \cdot \frac{1}{\prod k_i} \sum_{i \in S_\frac{1}{2}} x_i' \quad (2.15)$$

$$\leq \sum_{i \in S_1} x_i' + (2 - \frac{2}{\prod k_i}) \sum_{i \in S_\frac{1}{2}} x_i' \quad (2.16)$$

$$\leq (2 - \frac{2}{\prod k_i}) \sum_{i \in S_1 \cup S_\frac{1}{2}} x_i' \quad (2.17)$$

$$\leq (2 - \frac{2}{\prod k_i}) \text{OPT} \quad (2.18)$$
2.5.2 Improve the approximation by triangle eliminating

To improve the approximation, it is necessary to decrease the chromatic number of the input graph. We next use triangle eliminating technique to decrease the chromatic number.

However, due to the possible semantic in real applications, we give two eliminating methods and their corresponding bounds. In the first case, triangles formed by at least two forests cannot be eliminated. While, for in the second case, any triangle can be eliminated.

The first improved algorithm is shown as follow.

Algorithm 2

1: **for** each set of disjoint complete multipartite graphs $F_i$ **do**
2:     **while** there is a triangle **do**
3:         include the three vertices of a triangle into $\tilde{VC}_1$
4:         remove them and their adjacent edges in $G$
5:     **return** $\tilde{VC}_2$
6: **return** $\tilde{VC}_1 \cup \tilde{VC}_2$

We formally state that the algorithm above will return a $(2 - \frac{1}{2^C})$-approximation as the following theorem.

**Theorem 5.** Algorithm 2 returns a $(2 - \frac{1}{2^C})$-approximation.

**Proof.** First, Algorithm 2 does find a cover, because the edges connected triangles removed and the residual graph are covered by $\tilde{VC}_1$, meanwhile, Algorithm 1 guarantees $\tilde{VC}_2$ is a cover of the residual graph.

Second, as it known [102], in a graph $G$, if $v$, $u$ and $w$ form a triangle, then we can include all three vertices in a vertex cover for a $\frac{3}{2}$-approximation. This is because at least two vertices of a triangle is needed in order to cover it. Therefore, $\tilde{VC}_1$ is a $\frac{3}{2}$-approximation for the subgraph induced by all the triangles removed.

Third, for each connected component, if all the triangles are removed, it is at most a complete bipartite graph, therefore it has a chromatic number at most 2. Then the residual graph has a chromatic number at most $2^C$. Due to lemma 2, $\tilde{C}_2$ is an $(2 - \frac{1}{2^C})$-approximation on the residual graph. Then combine the two covers will obtain a max $\{\frac{3}{2}, 2 - \frac{1}{2^C}\}$-approximation. Without loss of generality, we have $C \geq 1$, then Algorithm 2 returns a $(2 - \frac{1}{2^C})$-approximation. \qed
Remark. In fact, after removing all the triangles, the residual graph has a chromatic number \(2^r\) which is far less than \(C\). Moreover, there is also a simple polynomial coloring algorithm as follow.

**Algorithm Coloring**

1. Removing all the triangles from each set of disjoint complete multipartite graphs
2. **for** each set of disjoint complete multipartite graphs \(F_i\) **do**
3. color each connected component in \(F_i\) with colors \(col_{2i-1}\) and \(clo_{2i}\), such that this subgraph is colored with only two colors.
4. **for** each vertex in \(G\) **do**
5. color it with \(col(col_{j_1}, col_{j_2}, \ldots, col_{j_r})\), where graph \(col_{j_i}\) is its color in \(F_i\)

Using the coloring algorithm after triangles elimination, we can reduce the approximation ratio to \(2 - \frac{1}{2^r}\). Theoretically, this ratio does not depend on size of input graph. Practically, when the number of set of disjoint complete multipartite graphs is a fixed small constant, it is a good approximation ratio.

**Corollary 1.** Algorithm 2 returns a \((2 - \frac{1}{2^r})\)-approximation.

2.5.3 Near optimality

In the following, we show that Algorithm 2 is a near optimal approximation based on the following two definitions in [93].

**Local Coloring.** Let \(\Delta\) be a positive integer. A \(\Delta\)-local coloring of a graph is a valid vertex coloring such that for every vertex \(u \in V\), all the neighbors of \(u\) are colored by at most \(\Delta - 1\) colors.

**Biclique Coloring.** A coloring of a graph is called a biclique coloring if for any two colors \(i\) and \(j\), the subgraph induced by “the vertices with color \(j\) that have a neighbor with color \(i\)” and “the vertices with color \(i\) that have a neighbor with color \(j\)” is either empty or a complete bipartite graph.

We show an example of 3-local biclique coloring in fig. 2.3

**Remark.** On one side, the best approximation for vertex cover of general graph is \(2 - \Theta\left(\frac{1}{\log |V|}\right)\) [91]. More strictly, it is a randomized algorithm and the ratio is parameterized by size of vertex set \(|V|\). While Algorithm 2 is deterministic and returns an approximation independent with \(|V|\) but only depends on \(|\Sigma|\) always small.
On the other side, as proven in [93] that it is UGC-hard to improve the approximation ratio of 

\[ 2 - \frac{2}{\Delta + 1} \]

by any constant factor if a \((\Delta + 1)\)-local coloring is given as input and it is not a biclique coloring. In general, computing a local biclique coloring is a hard problem for general graphs. Actually, due to the reduction in the proof of [4], the instance of our problem can simulate arbitrary graph, that is, finding a local biclique coloring for the input conflict graph is also hard. However, Algorithm Coloring provides a \(2^r\)-local coloring, this is to say that if assume UGC, then it is hard to improve any constant factor better than \((2 - \frac{1}{2^r})\).

Moreover, we also claim that Algorithm Coloring will return a \(2^r\)-local coloring and it is tight but not a biclique coloring. Actually, This property is tight if we know nothing more about the graphs. For arbitrary \(n\), let \(p_1p_2 \cdots p_k\) be the prime factorization of \(n\) (where repeated primes are listed repeatedly). If \(G'\) is the sum of \(k\) correctly-chosen complete \(p_i\)-partite graphs, \(G' = K_n\). To choose \(G_1\), let the parts be the residues modulo \(p_1\). For \(G_2\), let \(i\) be in a part according to the residue of \(\lfloor \frac{1}{p_1} \rfloor\) modulo \(p_2\). Continuing this for each \(k\) makes the right subgraphs. The best way to imagine this is to express \(i\) has a number where the 1st place goes up to \(p_1\), the \(p_1\)s place (the next digit over) goes up to \(p_2\), etc. Then \(G'\) is complete because if \(i \neq j\) then The representations of \(i\) and \(j\) are different.

**Property 3.** The biclique local coloring of conflict graph depends on the number of vertices, even if the number of set of disjoint complete multipartite graphs is fixed \((r \geq 2)\).

**Proof.** We can build a valid conflict graph with \(4n\) vertices and its biclique coloring of size \(\Omega(\sqrt{n})\). We show the conflict graph built by two forests.

![Figure 2.3. Example for a 3-local biclique coloring](image)
Construct two subgraphs, a complete bipartite graph $F_1 (V \cup U, E_1)$ where $V = \{v_1, \ldots, v_{2n}\}$, $U = \{u_1, \ldots, u_{2n}\}$ and $E_1 = V \times U$, and a bipartite graph $F_2 (V \cup U, E_2)$ where

$$E_2 = \{v_1v_2, v_3v_4, \ldots, v_{2n-1}v_{2n}, u_1u_2, u_3u_4, \ldots, u_{2n-1}u_{2n}\}$$

Let the conflict graph $G$ is $F_1 + F_2$ as follow example (dash edges from $E_2$, others from $E_1$),

For each odd $i$, we see that $v_i (u_i)$ is not adjacent with other vertices of $V (U)$ except $v_{i+1} (u_{i+1})$, therefore, in order to keep “biclique coloring” property, we have the following conditions,

(a) $v_i(u_i)$ should be colored different from $v_{i+1} (u_{i+1})$;

(b) the color pair of $v_i$ and $v_{i+1} (u_i$ and $u_{i+1})$ can not be the same as any other color pair of $v_j$ and $v_{j+1} (u_j$ and $u_{j+1})$ for any other odd $j$.

(c) all the colors of vertices in $V$ should be different from those in $U$.

Based on this, if a biclique local coloring has a size $f$, then it must satisfy that

$$\left( \frac{2}{f^2} \right) \geq \frac{n}{2} \quad (2.19)$$

Therefore, the biclique local coloring depends on the number of vertices.

**Corollary 2.** Algorithm 2 is near optimal.

Besides the algorithm above, we provide another for the input cases with a desired properties, in order to reduce the approximation ratio.
2.6 A modified linear program

The intuition of our idea is that for each complete $k$-partite graph, vertices in at least $k - 1$ partites should be add into the minimum vertex cover, if add constraints into classic linear program for vertex cover to limit the lower bound of variables taking value of 1, then we can get a better ratio on dense conflict graph. But the trouble is that if the variables of vertex cover linear program can only take 1, 0.5 and 0, then as the following figure shows, if we want to ensure that at least one of two given variables must take 1, then it is impossible to invent any linear inequality to guarantee the lower bound of variables taking value of 1.

![Figure 2.5. limitation of linear inequality](image)

Fortunately, we can at least invent some constraints to limit the lower bound to some degree if every variable can only take value of 1, 0.5 and 0. At the same time, the half-integral property should be kept if add such constraints. Base on this intuition, we show a modified linear program for vertex cover on $\beta$-conflict graph.

Given a complete multipartite component $\alpha$ with $m$ partites, i.e., independent set. Let partite $p$ has $b_p$ vertices and the size of $\alpha$ is $|\alpha|$, i.e., $|\alpha| = \sum_{p=1}^{m} b_p$. It is called wheel complete multipartite graph if $\alpha$ satisfy (a) $m \geq 3$ and (b) for any $p$, $|\alpha| - b_p \geq kb_p$ where $k > 1$. Obviously, any complete graph is a ideal wheel complete multipartite graph. Let $G = (\bigcup_{i=1}^{r} V_i, \bigcup_{i=1}^{r} E_i)$ be a $\beta$-conflict graph where $\beta \leq 1$ such that the ratio of vertices belong to some wheel complete multipartite graph in some $(V_i, E_i)$ is more than $(1 - \beta)$. For example, in fig. 2.6 $K_{2,2,2,3}$ is a wheel complete multipartite
graph, more concretely, $|\alpha| = 9$, therefore we can find a $k = 2(k > 1)$, such that for any partite, $|\alpha| - 3 = 6 \geq 3k$. But for the second one $K_{1,2,2,5}$, it is not a wheel complete multipartite graph, since $|\alpha| = 10$ and we can not find a $k > 1$ such that $|\alpha| - 5 = 5 \geq 5k$. Intuitively, if one partite of a complete multipartite is so large that even the sum of sizes of all the other partites is smaller than it. Also, it shows that for complete multipartite with more than 3 partites, in a wheel complete multipartite, size distribution of partite is roughly uniform.

Given a complete multipartite component $\alpha$ with $m$ partites, let partite $p$ has $b_p$ vertices, without loss of generality, let partite $m$ is of largest size $b_m$ and the size of $\alpha$ is $|\alpha|$, i.e., $|\alpha| = \sum_{p=1}^{m} b_p$.

We add constraints into the standard linear program as following,

\[
\begin{align*}
\text{minimizing} & \quad \sum_{v \in V} X_v \\
\text{subject to:} & \quad x_v + x_u \geq 1, \forall (v, u) \in E, \\
& \quad \sum_{v \in \alpha_i} x_v > |\alpha_i| - b_{m_i} - \varepsilon, \text{for every given complete multipartite graph } \alpha_i, \\
& \quad x_v \geq 0, v \in V
\end{align*}
\]

Figure 2.6. Example of a wheel complete multipartite
For a small enough $\varepsilon > 0$, we know that for any integral solution, the inequality will make sure that for each complete multipartite component, there are at least $|\alpha_i| - b_{m_i}$ variables $x$ take value of 1. Linear program solution is not able to guarantee the number of 1 up to $|\alpha_i| - b_{m_i}$, however,

**Lemma 1.** Every extreme point solution to the Conflict Graph Vertex Cover Linear Programme (1) is still half-integral, i.e., for all $v \in V$, $x_v \in \{0, \frac{1}{2}, 1\}$.

**Proof.** Contrary to the statement of this lemma, let the contradiction be that there is an extreme point solution which has a dimension $x_v$ not in $\{0, \frac{1}{2}, 1\}$. Let $V^{\frac{1}{2}} = \{v \in V | \frac{1}{2} < x_v < 1\}$ and $V < \frac{1}{2} = \{v \in V | 0 < x_v < \frac{1}{2}\}$. Note that $V^{\frac{1}{2}} \cup V < \frac{1}{2} = \emptyset$, and for any pair of vertices $u, v$, if there is $x_u + x_v = 1$, then $v \in V^{\frac{1}{2}}$ will result in $u \in V < \frac{1}{2}$.

Given a solution $x$ to the LP, consider the following solutions $y, z$ for every vertex $v \in V$:

\[
y_v = \begin{cases} 
x_v + \epsilon, & \text{if } v \in V^{\frac{1}{2}} \\
x_v - \epsilon, & \text{if } v \in V < \frac{1}{2} \\
x_v, & \text{otherwise}
\end{cases}
\]

\[
z_v = \begin{cases} 
x_v + \epsilon, & \text{if } v \in V < \frac{1}{2} \\
x_v - \epsilon, & \text{if } v \in V^{\frac{1}{2}} \\
x_v, & \text{otherwise}
\end{cases}
\]

Then, $x = \frac{y + z}{2}$, which means that $x$ is the convex combination of $y$ and $z$. All we need to prove in order to get a contradiction is that $y$ and $z$ are feasible. Consider the edges $e = \{v, u\}$, by selecting $\varepsilon$ to be small enough, so that:

- When $x_v > \frac{1}{2}$ and $x_u > \frac{1}{2}$, then we have $y_v + y_u = x_v + x_u + 2\epsilon > 1$ and $z_v + z_u = x_v + x_u - 2\epsilon > 1$

- When $\sum_{v \in \alpha_i} x_v > |\alpha_i| - b_{m_i} - \varepsilon$, then we have

\[
\begin{align*}
\sum_{v \in \alpha_i} y_v & \geq \sum_{v \in \alpha_i} x_v - (m_i \cdot \epsilon) > |\alpha_i| - b_{m_i} - \varepsilon \\
\sum_{v \in \alpha_i} z_v & \geq \sum_{v \in \alpha_i} x_v - (m_i \cdot \epsilon) > |\alpha_i| - b_{m_i} - \varepsilon
\end{align*}
\]
where

\[ 0 < \epsilon \leq \min \left\{ \frac{\epsilon}{m_i}, \min_{x_v, x_u \geq \frac{1}{2}} \frac{x_v + x_u - 1}{2} \right\} \]

Then we can still concern ourselves only with the edges \( e = \{v, u\} \) for which it holds \( x_v + x_u = 1 \).

Then, we can distinguish three cases:

- when \( x_v = \frac{1}{2} \), we have \( x_u = \frac{1}{2} \) and therefore \( y_v + y_u = x_v + x_u = 1 \) and \( z_v + z_u = x_v + x_u = 1 \)
- when \( x_v > \frac{1}{2} \), we have \( x_u < \frac{1}{2} \) and therefore \( y_v + y_u = x_v + \epsilon + x_u - \epsilon = 1 \) and \( z_v + z_u = x_v + \epsilon + x_u - \epsilon = 1 \)
- when \( x_v < \frac{1}{2} \), we have \( x_u > \frac{1}{2} \) and therefore \( y_v + y_u = x_v - \epsilon + x_u + \epsilon = 1 \) and \( z_v + z_u = x_v - \epsilon + x_u + \epsilon = 1 \)

That is, every non-half-integral point solution is not a extreme point solution.

Every extreme point solution is a half-integral, therefore we can claim that for each complete multipartite component, there are at most \( b_{m_i} \) will be round to 1 wrongly.

**Lemma 2.** In any complete multipartite graph \( \alpha \), let \( S_{\frac{1}{2}}^{\alpha} \) and \( S_{\frac{1}{2}}^{\alpha} \) be the set of variables taking value of \( \frac{1}{2} \) and \( \frac{1}{2} \), then they can be bounded by (a) \( |S_{\frac{1}{2}}^{\alpha}| \geq |\alpha| - 2b_m \) and (b) \( |S_{\frac{1}{2}}^{\alpha}| \leq 2b_m \).

**Proof.** For every given complete multipartite \( \alpha \), let \( S_{\frac{1}{2}}^{\alpha} \) be the set \( \{x'_v | v \in \alpha, x'_v = \frac{1}{2}\} \), \( S_{\frac{1}{2}}^{\alpha} \) be the set \( \{x'_v | v \in V, x'_v = \frac{1}{2}\} \).

Due the second LP constraint

\[ \sum_{v \in \alpha} x_v \geq |\alpha| - b_m - \epsilon, \]

so we have that,

\[ |S_{\frac{1}{2}}^{\alpha}| + \frac{1}{2}|S_{\frac{1}{2}}^{\alpha}| \geq |\alpha| - b_m - \epsilon \]

Then

\[ |S_{\frac{1}{2}}^{\alpha}| \geq |\alpha| - b_m - \epsilon - \frac{1}{2}|S_{\frac{1}{2}}^{\alpha}| \]

The worst case is that \( |S_{\frac{1}{2}}^{\alpha}| = |\alpha| - |S_{\frac{1}{2}}^{\alpha}| \), then

\[ |S_{\frac{1}{2}}^{\alpha}| \geq |\alpha| - b_m - \epsilon - \frac{1}{2}(|\alpha| - |S_{\frac{1}{2}}^{\alpha}|) \]
\[ \frac{|S^\alpha_1|}{2} \geq \frac{|\alpha|}{2} - b_m - \varepsilon \]
\[ |S^\alpha_1| \geq |\alpha| - 2b_m - 2\varepsilon \]

We choose a small enough \( \varepsilon \) such that \( 2\varepsilon < 1 \) then we get
\[ |S^\alpha_1| \geq |\alpha| - 2b_m, \text{ and } |S^\alpha_1| \leq 2b_m \]

Consider a \( \beta \)-conflict graph, that is, there are \((1-\beta)n\) vertices belongs to some wheel complete multipartite, and the other \( \beta n \) vertices are not. Then we can derive the ratio of modified linear program.

**Theorem 6.** Our LP will give a \((1 + \beta + \frac{1-\beta}{k})\)-approximation for \( \beta \)-conflict graph.

**Proof.** Let \( OPT \) is the optimal solution of modified linear program, i.e \( OPT = |S_1| + \frac{1}{2}|S^\gamma_2| \). For the subgraph \( \gamma \) induced by the vertices belong to some wheel complete multipartite graph in some \((V_i, E_i)\), let the solution of modified linear program involved in \( \gamma \) is \( OPT_\gamma = |S^\gamma_1| + \frac{1}{2}|S^\gamma_2| \). Obviously, let the approximated vertex cover \( \tilde{VC} \) in \( \gamma \), the number of vertices rounded into our approximated vertex cover which is
\[ |\tilde{VC} \cap V_\gamma| = OPT_\gamma + \frac{1}{2}|S^\gamma_2|, \]
moreover,
\[ |VC^* \cap V_\gamma| \geq OPT_\gamma, \]
then we can see that ratio of this part is
\[ \frac{|\tilde{VC} \cap V_\gamma|}{VC^* \cap V_\gamma} \leq \frac{OPT_\gamma + \frac{1}{2}|S^\gamma_2|}{OPT_\gamma} \leq 1 + \min_{\alpha} \left\{ \frac{1}{2} \frac{2b^\alpha_m}{|\alpha| - b^\alpha_m} \right\} = 1 + \min_{\alpha} \left\{ \frac{b^\alpha_m}{|\alpha| - b^\alpha_m} \right\} \leq 1 + \frac{1}{k} \]

Then for the other part, say
\[ \frac{\tilde{VC} \cap (V_G - V_\gamma)}{VC^* \cap (V_G - V_\gamma)} \leq 2 \]

Therefore we have
\[ \frac{\tilde{VC}}{VC^*} \leq (1 - \beta)(1 + \frac{1}{k}) + 2\beta = 1 + \beta + \frac{1 - \beta}{k} \]
Remark. For complete multipartite graph, complete graph is densest case while star is the sparsest case. Therefore if a conflict graph is sparse, the previous algorithm will provide a better approximation, but when the graph is dense, i.e., $\beta$ is small, then our algorithm will provide a very good approximation.

2.7 Conclusion

In this paper, we study conflict graph and show that this property is non-trivial if limiting the number of sets of graphs of disjoint complete multipartites, and show that vertex cover on this graph class is also NP-hard even if the condition is simple. At last, we show that the vertex cover on this graph class can be approximation within $2 - \frac{1}{2}$ which is near optimal. We also study approximation of vertex cover on $\alpha$-conflict graph, and this approximation is better for dense cases. If the ratio of vertex not belongs to any $k$-wheel complete multipartite graph is no more than $\beta < 1$, then our algorithm will provide a $(1 + \beta + \frac{1-\beta}{k})$-approximation. However, the gap between upper and lower bound of vertex cover problem on conflict graph remains open.
Chapter 3

VIEW PROPAGATION FOR DATA REPAIR VALIDATION

The view update problem has been extensively investigated for more than 35 years in the database community, which is stated as follows: given a desired update to a database view, what update should be performed towards the source tables to reflect this update to the view \[35, 37, 40, 41, 58\]. Generally, the previous works mainly focus on identifying the condition to make the update unique, and studying under the identified condition how to carry out the update.

3.1 Introduction

These works are only effective for very restricted circumstances where there is a unique update \(\Delta D\) to a source database \(D\) that will cause a specified update to the view \(Q(D)\). In practice, an update to \(D\) is not always unique. Therefore, an alternative is to find a minimum update to \(D\) resulting in the specified update to \(Q(D)\), which is the task of view propagation.

View propagation. With the emergence of research on data provenance and data quality in the recent score of years, view propagation analysis \[53\], as a new problem in classic view update analysis, has been formally proposed and widely studied. The goal of view propagation analysis is to figure out the \((minimum) side-effect\)' on either a view or a source database, which is caused by the asymmetry of update between the view and the source database. The study on computational complexity of the view side effect problem gives important insights into the computational issues involved in data provenance, which is the process by which data move among databases. It is fundamentally relative to why-provenance. The complexity of it can be used to show the complexity of some data quality tasks including the causality of an unanticipated certain query result. It is identified and defined in \[53\] as follows: given a source database \(D\), a monotonic relational query \(Q\), the materialized view \(V = Q(D)\), and the update on the view (a set of tuples) \(\Delta V\), the view
side-effect problem is to find a minimum $\Delta D$ such that $Q(f(D, \Delta D)) = f(V, \Delta V)$,

$$f(a, b) = \begin{cases} 
    a \setminus b, & \text{for deletion update,} \\
    a \cup b, & \text{for insertion update}
\end{cases} \quad (3.1)$$

i.e., side effect free whenever such $\Delta D$ exists.

Besides research on source side effect [64], there are some complexity results on the view side-effect problem for insertion [51, 52] and deletion propagation [39, 56, 103]. Moreover, for deletion propagation, Kimfield et. al. [54] identified the dichotomy ‘head domination’ for every conjunctive query without self joins. Deletion propagation is either APX-hard or solvable (in polynomial time) by the unidimensional algorithm. They also showed the dichotomy ‘functional head domination’ [55] for the functional-dependency-restricted version. For multiple deletion [56], they especially showed the trichotomy for group deletion a more general case. It is radically different from the case without functional dependencies (FD). Besides the previous researches, there are no results on insertion propagation considering FDs. Note that it is very common that FDs are defined in a database schema.

A conjunctive query usually makes a deletion update difficult to decide if there is a side-effect-free solution. The behind-the-scene intuition is the combination of Projection and Join which introduces many possible ways of removing an output tuple. Such a fact is called asymmetry. Specifically, the desired deletion in the view may have many witnesses and there may exist many possible ways of destroying each witness. Once any witness is removed from the source, the witness will be destroyed. The difficulty arises when one has to consider how a set of source tuple deletions affects the existence of other output tuples in order to minimize side-effects. Similarly, an insertion update should also consider this factor. To understand the computation difficulty, we next illustrate some examples.

**Side-effect.** Consider the following scenario of an insertion update. An archive management database of a company include two relations: Group(group, user) keeps track of the group to which each user belongs, and Access(group, file) records the files that each group has the authority to access. There is also a view defined as a conjunctive query (Selection-Projection-Join) “show the files to which a user has authority to access”.

Given a desired view insertion $\Delta V = (u_1, f_2)$, the task is to find a side-effect-free insertion. Here are some possible ways to update the source database:
(a) Insert (g2, u1) into Group;

(b) Insert (g1, f2) into Access;

(c) Insert (x, u1) into Group and (x, f2) into Access, where x is a value taken from the domain of attribute group and x is different from g1 and g2,

Except the desired \( \Delta V \), (u1, f3) which is the side-effect on the view must be produced in method (a). However, the other two methods are side-effect-free solutions.

**Finite domain of an attribute.** Besides the combination of Projection and Join, there are some other factors that need to be considered. For example, in practice, the domain of attribute group is often finite. Therefore, if no new value can be taken from the domain, then method (c) is never valid. Thus, the domain finity of the projection attributes plays a key role in the computation of this insertion propagation problem. The analysis in this paper considers it as a factor.

**Functional dependency restriction.** In practice, usually there are some functional dependencies defined in a database to guarantee semantic correctness. Following the above example database, an underlying functional dependency

\[
\text{user} \rightarrow \text{group}
\]

is defined on Group(group, user) in order to guarantee that ‘each user can take part in only one group’. In such a case, different from the above example, method (c) is not a valid solution any more due to the functional dependency. This is because, for any tuple \((x, u1)\) to be inserted into Group, the value of \(x\) is different from \(g1\), but if the insertion is carried out, functional dependency ‘user \(\rightarrow\) group’ will be violated by this tuple together with \((g1, u1)\).

In this paper, we investigate such FD-restricted insertion propagation with more effort on the complexity aspect of this problem.
3.2 Preparation

A *schema* is a finite sequence \( R = \langle R_1, \ldots, R_m \rangle \) of distinct relation symbols, where each \( R_i \) has an arity \( r_i > 0 \) and includes several attributes, denoted by \( R_i = \{ A_1, \ldots, A_{r_i} \} \). Each attribute \( A_j \) has a corresponding set \( \text{dom}(A_j) \) which is the domain of the valid values for \( A_j \). A database instance \( D \) (over \( R \)) is a sequence \( \langle R^D_1, \ldots, R^D_m \rangle \), such that each \( R^D_i \) is a finite set of tuples \( \{t_1, \ldots, t_N\} \), and each tuple \( t_k \) belongs to the set \( \text{dom}(A_1) \times \cdots \times \text{dom}(A_{r_i}) \). We use \( R.A_i \) to denote attribute \( A_i \) of relation \( R \), and we denote \( R^D \) as \( R \) without loss of clarity.

**Fact.** We extend the concept of *fact* in the database theory. Given a database instance \( D \) and an SPJ query \( Q \) in the form of \( \pi_A (\sigma_{\text{con}}(R_1 \times \cdots \times R_n)) \). A *fact* \( \mu \) of \( D \) is a tuple sequence \( (t_1, t_2, \ldots, t_n) \in R_1 \times \cdots \times R_n \), where \( t_i \in R_i \) for each \( 1 \leq i \leq n \). If \( (t_1, t_2, \ldots, t_n) \) satisfies the selection condition \( \text{con} \), then we denote it as \( Q(\mu) \in Q(D) \).

**Functional Dependency (FD)** \[34\]. \( \varphi \) over a relation \( R \) can be represented by \( \varphi : (X \rightarrow A) \), where both \( X \) and \( A \) are a set of attributes from \( R \). Such a dependency means any two tuples should have the same values for \( A \) if they have the same values for \( X \). Given database instance \( I \) and FD \( \varphi \), if no tuple pair violates \( \varphi \), we denote that \( D \models \varphi \). Usually, we use \( \Sigma \) to denote the set of FDs.

Given database \( D \) and FD set \( \Sigma \), if for every FD \( \varphi \in \Sigma \), \( D \models \varphi \), we say \( D \models \Sigma \).

**Semi-free attribute.** A semi-free attribute is an attribute restricted by at least a selection condition but projected out by the query. A free attribute is a semi-free attribute but not restricted in any selection condition.

**Proposition of semi-free attribute.** If a query is *monotone* without self-join, there is at most one tuple to be inserted into each table in the source database \( D \) by the side-effect free insertion, where the value of the projected attribute is the same as the insertion. This proposition guarantees that under data complexity, there are at most a constant number of possible insertion choices to be checked for a single insertion whether it has a view side-effect free solution.
Table 3.1. Notations for this chapter

<table>
<thead>
<tr>
<th>D and ∆D</th>
<th>source database and its update</th>
</tr>
</thead>
<tbody>
<tr>
<td>V and ∆V</td>
<td>view and update on view</td>
</tr>
<tr>
<td>Σ and Q</td>
<td>set of functional dependency and relational algebraic query</td>
</tr>
<tr>
<td>S, P, J and U</td>
<td>Selection, Projection, Join and Union</td>
</tr>
<tr>
<td>SPJ</td>
<td>Selection-Projection-Join</td>
</tr>
<tr>
<td>SPU</td>
<td>Selection-Projection-Union</td>
</tr>
</tbody>
</table>

Definition 3.2.1. *(FD-restricted view side-effect-free deletion propagation (FD-vsef-DP))* Given a source database $D$, a set of FDs $Σ$, a query $Q$, the materialized view $V$, and a set of tuple deletion $ΔV$, the FD-vsef-DP problem is to decide if there is a tuple set $ΔD$ such that

1. $D − ΔD \models Σ$, and
2. $Q(D − ΔD) = V − ΔV$.

Definition 3.2.2. *(FD-restricted view side-effect-free insertion propagation (FD-vsef-IP))* Given a source database $D$, a set of FDs $Σ$, a query $Q$, the materialized view $V$, and a set of inserted tuples $ΔV$, the FD-vsef-IP problem is to decide if there is a tuple set $ΔD$ such that

1. $D ∪ ΔD \models Σ$, and
2. $Q(D ∪ ΔD) = V ∪ ΔV$.

In this paper, we study the complexity of FD-vsef-IP in two cases single insertion when $|ΔV| = 1$, and group insertion when $|ΔV| > 1$. A query $Q$ is expressed in relational algebra including $S$ (selection), $P$ (projection), $J$ (join), and $U$ (union).

For the sake of clarity, we list notations again as follows.

3.3 Complexity results

In this section, we examine the impact of different combinations of the factors on both data complexity and combined complexity of the problems. *Data complexity* is the complexity expressed in terms of the size of the database only, while *combined complexity* is the complexity expressed in terms of both the size of the database and the query expression [104]. The complexity measure follows the work in [53] where the complexity results of the propagation problem are first established, and the studies in [36, 37, 105] which study the complexity of the view update problems. In general,
data complexity of a problem is always lower than its combined complexity, this is simple, because it incorporates both data size and expression size. In the following, we show specifically the results in domination positions including both data complexity and combined complexity of deletion cases and group/single insertion cases.

3.3.1 Deletion Propagation

In this section, we list the intractable cases on two aspects including both data and combined complexity. We remark that data complexity is the complexity expressed in terms of the size of the database only, while combined complexity is the complexity expressed in terms of both the size of the database and the query expression [104].

Data complexity aspects The first result can be stated as follows. The results are listed as the ascending order of complexity.

Theorem 7. FD-vsef-DP is NP-hard for P queries.

Proof Sketch: We construct a PTime reduction from 3SAT to this problem. Given a boolean variable set $X = \{x_1, \ldots, x_n\}$, the input of 3SAT problem is a formula $\phi = C_1 \land \ldots \land C_m$ where $C_i = \{l_1, l_2, l_3\}$ and $l_j$ is either $x_k$ or $\overline{x}_k$ for $k \in 1, 2, 3$. Reduction can be described as follows.

1. **Base instance.** Let $D$ contains only one relation $R$ including three attributes $(L, X, C)$. For each clause $C_i \in \phi$ and each literal $l_j \in C_i$ ($j \in \{1, 2, 3\}$), a tuple $t_{ij}$ is built and inserted into $R$ as follows. If $l_j$ is $x_k$, let $t_{ij} = (+, X_k, c_i)$. If $l_j$ is $\overline{x}_k$, let $t_{ij} = (-, X_k, c_i)$.
2. **FD set.** Let $\Sigma$ be $\{X \rightarrow L\}$.
3. **Witness query.** Let $Q$ be $\pi_C(R)$.
4. **Query result.** Let $Q(I) = \{(c_1), \ldots, (c_m)\}$.
5. **Feedback.** Let $\Delta$ be $\emptyset$. One can verify the $\phi$ is satisfied if and only if there is a valid repair of $I$.

Theorem 8. FD-vsef-DP is NP-hard for U queries.

Proof Sketch: We construct a PTime reduction from Monotone 3SAT problem. Similar with 3SAT problem, an instance of Monotone 3SAT problem is a formula $\phi = C_1 \land \cdots \land C_m$, where each clause $C_i$ includes only positive or negative literals. The reduction can be made as follows.

1. **Base relations.** First, suppose there are $n$ variables $x_1, \ldots, x_n$, then let $I$ contains $n$ relations $R_1, \ldots, R_n$. Each relation $R_i$ has attribute set $\{X, C\}$. Second, for each $C_j$, if $x_i \in C_j$, add $(+, C_j)$ to $R_i$, if $\overline{x}_i \in C_j$, add $(-, C_j)$ to $R_i$.
2. **FD set.** For each $R_i$, add rule $\emptyset \rightarrow X$ to the rule set $\Sigma$.
3. **Witness query.** Let $Q$ be $\pi_C(R)$.
4. **Query result.** Let $Q(I) = \{(c_1), \ldots, (c_m)\}$.
5. **Feedback.** Let $\Delta$ be $\emptyset$. One can verify the $\phi$ is satisfied if and only if there is a valid repair of $I$. 


query. Let \( Q \) be \( R_1 \cup \cdots \cup R_n \); (4) Query result. Initially, \( Q(I) \) includes \( m \) tuples where tuple \( t_i \) is \((+,C_j)\) if \( C_j \) contains positive; otherwise, it is \((-,-C_j)\); (5) Feedback. Let \( \Delta \) be \( \emptyset \). One can verify that \( \phi \) is satisfied if and only if there is a valid repair of \( I \).

**Theorem 9.** \( \text{FD-vsef-DP} \) is \( \text{NP} \) for RA queries.

**Combined complexity aspects** To analyse SPJ queries, we will use the term fact. Given a database instance \( I \) and an SPJ query \( Q \) in a form of \( \pi_A(\sigma_{\text{con}}(R_1 \times \cdots \times R_q)) \). A fact \( \mu \) of \( I \) is a tuple sequence \((t_1,t_2,\ldots,t_q) \in R_1 \times \cdots \times R_q\), where \( t_i \in R_i^I \) for each \( 1 \leq i \leq q \). If \((t_1,t_2,\ldots,t_q)\) satisfies the selection condition \( \text{con} \), then we denote it as \( Q(\mu) \in Q(I) \).

**Theorem 10.** \( \text{FD-vsef-DP} \) is \( \Sigma_2^P \text{-complete} \) for SPJ and SPJU queries.

**Proof.** We will prove the following two statements to show the correctness of the theorem. (i) We prove the upper bound of \( \text{FD-vsef-DP} \) for SPJU query is \( \Sigma_2^P \) by giving a \( \Sigma_2^P \) algorithm as follows. First, guess a sub-instance \( D \cup \Delta D \) of \( I \) satisfying \( \Sigma \). Then, determining whether \( Q(D \cup \Delta D) \cap \Delta = \emptyset \) and \( Q(I) \setminus \Delta \subseteq Q(D \cup \Delta D) \). The former question is in \( \text{coNP} \), because any SPJU query has a form of \( q_1 \cup \cdots \cup q_k \) where each \( q_i \) is a SPJ query, so that its complement can be solved by determining whether for there is a fact \( \mu \) of \( I_r \) such that \( Q(\mu) \in Q(D \cup \Delta D) \) and \( Q(\mu) \in \Delta \). The latter question is also in \( \text{coNP} \), because its complement can be solved by determining whether there is a fact \( \mu \) of \( I \) such that \( Q(\mu) \in Q(I) \setminus \Delta \) but \( \mu \) is not a fact of \( D \cup \Delta D \) (then it must be \( Q(\mu) \notin Q(I_r) \), because \( Q(\mu) \in Q(I) \)).

(ii) We prove the lower bound of \( \text{FD-vsef-DP} \) for SPJ query is \( \Sigma_2^P \text{-hard} \) by a reduction from QSAT_2 problem. An instance of QSAT_2 problem includes two variable sets \( X_1 = \{x_1,\ldots,x_{n'}\} \) and \( X_2 = \{x_{n'+1},\ldots,x_{n'+n''}\} \), and a 3-DNF boolean expression \( \phi \) with \( m \) clauses \( \{C_1,\ldots,C_m\} \), the task is to determine whether there is an assignment \( \tau \) for \( X_1 \) such that \( \phi \) is satisfied by all assignments for \( X_2 \). Let \( n = n' + n'' \), that is \(|X_1| + |X_2| = n\), we show the reduction as follows. (An example of the reduction for a QSAT_2 instance \( \phi = \exists x_1 x_2 \forall x_3 x_4 (x_1 \land x_2 \land x_3) \lor (x_1 \land \overline{x}_2 \land x_3) \lor (\overline{x}_1 \land x_3 \land x_4) \) is shown in Fig. 3.1.)

**Base instance \( I \).** We build \( D \) including \( n+m+3 \) relations \( \{S_i, i = 1,\ldots,n\} \cup \{R_1, \ldots, R_m\} \cup \{G_p, p = 1,2,3\} \), where \( S_i \) simulates \( x_i \), \( R_k \) simulates clause \( C_k \) and \( G_1,G_2,G_3 \) are three auxiliary relations. Concretely, (1) For each variable \( x_i \) (\( 1 \leq i \leq n \)), construct relation \( S_i = \{A_1,A_2\} \) and add
three tuples \((X, 1)\) and \((X, 0)\) and \((Y, B)\) to \(S_i\). (2) For each clause \(C_i\) \((1 \leq i < m)\), build a quintuple relation \(R_i\) \((A_1, A_2, A_3, A_4, A_5)\). We add 8 tuples into \(I[R_i]\). In the first 7 tuples, values of \(A_1, A_2, A_3\) refer to the 7 false value assignments of the 3 variables, values of \(A_4\) are always ‘−’, and the values of \(A_5\) are the ids of these 7 tuples. The last two tuples are auxiliary tuples \((Z, Z, Z, −, 8)\) and \((B, B, B, B, 9)\). (3) \(G_1\) includes two tuples \((X \cdots X), (Y \cdots Y)\); \(G_2\) includes three tuples \((0 \cdots 0), (1 \cdots 1), (B \cdots B)\); \(G_3\) includes eight tuples \((1 \cdots 1), (2 \cdots 2), \ldots, (9 \cdots 9)\).

**FD set \(\Sigma\).** For each relation \(S_i\), \(i \in [1, n']\), add FD: \(S_i.A_1 \rightarrow S_i.A_2\) into \(\Sigma\).

**Witness query \(Q\).** Construct the query \(Q\) as follows. We denote \(R_1 \times \cdots \times R_m, S_1 \times \cdots \times S_{n'}\) and \(S_{n'+1} \times \cdots \times S_n\) as \(R, S^1, S^2\). For each clause \(C_k \in \phi\), without loss of generality, it is assumed that \(C_k = x_{k1} \land \overline{x}_{k2} \land x_{k3}\), let the condition \(con_k\) be \((S_{k1}.A_2 = R_{k}.A_1) \land (S_{k2}.A_2 = R_{k}.A_2) \land (S_{k3}.A_2 = R_{k}.A_3)\). Let the condition \(con_{A_1}\) be \(R_1.A_4 = R_2.A_4 = \cdots = R_m.A_4\). Then, let the witness query \(Q = Q_0 \times Q_1 \times Q_2 \times Q_3\), where

\[
Q_0 = \pi_{R_1.A_4,\ldots,R_m.A_4}(\sigma_{con_1 \land \cdots \land con_m \land con_{A_1}}S^1 \times S^2 \times R),
\]

\[
Q_1 = \pi_{G_1.A_1,\ldots,G_1.A_{n'}}(\sigma_{S_1.A_1 = G_1.A_1 \land \cdots \land S_{n'.A_1 = G_1.A_{n'}}}(S^1 \times G_1)),
\]

\[
Q_2 = \pi_{G_2.A_1,\ldots,G_2.A_{n'}}(\sigma_{S_{n'+1}.A_2 = G_2.A_1 \land \cdots \land S_n.A_2 = G_2.A_{n'}}(S^2 \times G_2)),
\]

\[
Q_3 = \pi_{G_3.A_1,\ldots,G_3.A_{m}}(\sigma_{R_1.A_5 = G_3.A_1 \land \cdots \land R_{m}.A_5 = G_3.A_{m}}(R \times G_3)).
\]

**Query result \(Q(I)\).** Initially, let \(Q(I) = \{t, t'\} \times G_1 \times G_2 \times G_3\), where \(t = \underbrace{(\ldots, −)}_{m}, t' = \underbrace{B, \ldots, B}_m\).

**Feedback \(\Delta\).** Let \(\Delta = \{t\} \times G_1 \times G_2 \times G_3\).

Some key properties are introduced before the correctness proof.

**P1.** A solution \(I_r, Q(I_r) = Q(I) \setminus \Delta\) if and only if, \(Q_0(I_r) = \{t'\}, Q_1(I_r) = G_1, Q_2(I_r) = G_2\) and \(Q_3(I_r) = G_3\).

**P2.** \(Q_1(I_r) = G_1\) where \(I_r \models \Sigma\) if and only if, either \(S_i = \{(X, 0), (Y, B)\}\) or \(S_i = \{(X, 1), (Y, B)\}\) holds, for each \(S_i\) of \(I_r\) where \(1 \leq i \leq n'\). It simulates that each variable in \(X_1\) has one and only one assignments.

**P3.** \(Q_2(I_r) = G_2\) if and only if each \(S_i\) of \(I_r\) is the same as it is in \(I\) where \(n' + 1 \leq i \leq n\). It simulates that each variable in \(X_2\) can be assigned arbitrarily.

**P4.** \(Q_3(I_r) = G_3\) if and only if each \(R_i\) of \(I_r\) is the same as it is in \(I\) where \(1 \leq i \leq m\).

Next, we show the correctness of the reduction by following two statements.
⇒ If the answer of QSAT_2 instance \( \phi \) is ‘yes’, there must be a sub-instance \( I_r \) obtained from \( I \) by deleting some tuples such that \( I_r \models \Sigma \) and \( Q(I_r) = Q(I) \setminus \Delta \). Suppose \( \phi \) is satisfiable for all assignments of \( X_2 \) under the assignment \( \tau (X_1) \). Given \( \tau \), we will construct a repair \( I_r \) satisfying the conditions of FD-vsef-DP. First, delete \((Z, Z)\) from each \( S_i \) \((1 \leq i \leq n)\); Second, for each relation \( S_i \) satisfying \( 1 \leq i \leq n' \) of \( I_r \), let the corresponding variable of \( S_i \) be \( x_i \in X_1 \). If \( \tau(x_i) = 1 \), the tuple \((X, 0)\) will be deleted, otherwise, the tuple \((X, 1)\) will be deleted. Obviously, \( I_r \models \Sigma \) and \( Q_i(I_r) = G_i (1 \leq i \leq 3) \), since the RHS of \( P_2, P_3, P_4 \) are satisfied. One can verify that \( Q_0(I_r) = I' \) since that \( \phi \) is a tautology under \( \tau (X_1) \).

≤ One the other hand, if there is a sub-instance \( I_r \models \Sigma \) and \( Q(I_r) = Q(I) \setminus \Delta \), then we can construct an assignment \( \tau (X_1) \) such that \( \phi \) is true under any assignment of \( X_2 \). In \( I_r \), each \( R_k \) related with clause \( C_k \) is the same as it is in \( I \) where \( 1 \leq k \leq m \), each \( S_i \) related with variable in \( X_2 \) is the same as it is in \( I \) where \( n' + 1 \leq i \leq n \), and each \( S_i \) related with variable in \( X_1 \) excludes either \( \{(X, 0)\} \) or \( \{(X, 1)\} \), because of \( S_i \) should satisfy \( S_i.A_1 \rightarrow S_i.A_2 \), where \( 1 \leq i \leq n' \). Note that, in \( I_r \), we do not care about that whether \((Z, Z)\) is preserved in each \( S_i \). For each variable \( x_i \in X_1 \), let \( S_i \) be the corresponding relation in \( I_r \), then the assignment \( \tau \) can be built as follows,

\[
\tau(x_i) = \begin{cases} 
1, & \text{if } (X, 1) \text{ is in } S_i; \\
0, & \text{otherwise.}
\end{cases} \tag{3.3}
\]

\[
\tau(x_i) = \begin{cases} 
1, & \text{if } (X, 1) \text{ is in } S_i; \\
0, & \text{otherwise.}
\end{cases} \tag{3.4}
\]
To show that $\tau$ is a valid assignment for $\phi$, consider any tuple $\vec{t} \in S^1 \times S^2 \times R^1$ in $I_r$ such that $\vec{t}[S_i.A_2] = \tau(x)$ ($1 \leq i \leq n'$), $\vec{t}[S_i.A_2] \neq B$ ($n' + 1 \leq i \leq n$). It is not hard to verify that such $\vec{t}$ always violates at least one condition defined in $Q_0$ due to P1 where $Q_0(I_r) = \{t'\}$. That is, some condition $\text{con}_k$ such that $t$ does not satisfy $\text{con}_k$. Because $\vec{t}$ satisfies the conditions $t[R_k.A_4]=-'-$, then we have that $\vec{t}$ refer to the assignments on $\{A_1, A_2, A_3\}$ which does not appear in $R_k$, it means that clause $C_k$ is satisfied under such assignment. Because for each assignment on $X_2$, there is at least one clause is true, we have that $\phi$ is tautology under assignment $\tau$. \hfill \Box

**Theorem 11.** $FD$-$\text{vsef-DP}$ is PSPACE-complete for RA query.

**Proof.** $FD$-$\text{vsef-DP}$ is in PSPACE obviously, since the Relation algebraic Query Evaluation problem for RA is in PSPACE-complete [34]. We next prove that $FD$-$\text{vsef-DP}$ is PSPACE-hard by reduction from query evaluation problem. Given an instance of Query Evaluation problem $\langle I, q, t \rangle$ where $q$ is a relational algebraic query, it is to decide if tuple $t \in q(I)$. Without loss of generality, we assume that $I$ consists of $n$ relations, $R_1, \ldots, R_n$, each relation $R_i$ contains $c_i$ columns, and $t$ is a $d$-dimension tuple. Then, by means of the technique similar with the previous proof, an instance of $FD$-$\text{vsef-DP}$ can be built as follows.

**Base instance.** Let $I'$ contain relations $R'_1, \ldots, R'_n$ and two auxiliary relations $R'_x, R'_y$. Each $R'_i$ ($1 \leq i \leq n$) is obtained by adding an addition column $c_{i+1}$ to $R_i$ and filling the additional column using integer numbers in $[\Sigma_{1 \leq j \leq i-1}|R_j| + 1, \Sigma_{1 \leq j \leq i}|R_j|]$ to identify each tuple uniquely in $I'$. For convenience, let $N = |R_1| + \cdots + |R_n|$. Then let $R'_x = \{t\}$ and $R'_y = \{(t', -), (t'', +)\}$ where $t'$ and $t''$ are two different $d$-dimension tuples as long as they are different from $t$.

**Witness query $Q$.** We denote $q(\pi_I(I'))$ as $A_q$, construct the query $Q$ as follows,

$$Q_a(I') = \pi_{A_{c_1+1}}(R'_1) \cup \cdots \cup \pi_{A_{c_{n+1}}}(R'_n),$$

$$Q(I') = [Q_a(I') \times \{(-), (+)\} - \pi_{Q_a}(Q_a(I') \times ((R'_x - A_q) \cup R'_y))] \times R'_x.$$

Here, the operator $\pi_{Q_a}$ extracts all attributes in the scheme of $Q_a$, and the operator $\pi_I$ extracts all attributes in $I$.

**Query result $Q(I)$.** Initially, $Q(I) = \{(1), \ldots, (N)\} \times \{(-), (+)\} \times \{t\}$.

**Feedback $\Delta$.** Let $\Delta = \{(1,-), \ldots, (N, -)\} \times \{t\}$.

Obviously, the reduction described above can be finished in polynomial time. The correctness of the reduction can be obtained by observing that $Q(D \cup \Delta D) = Q(I) \setminus \Delta$ if and only if $A_1$ no
tuple in $I'$ disappears in $D \cup \Delta D$ since $Q(I) \setminus \Delta$ includes every unique identification number, and (2) $t \in q(I)$.

### 3.3.2 Insertion Propagation

We will show that FD-vsef-IP are NP-complete in most cases. This is perhaps not surprising for group insertion. Nevertheless, we show that these problems become PTime for a single insertion over some very restricted fragments. Specifically, we provide the complete picture of the complexity on these problems for views defined in various fragments of SPJU queries, identifying all those cases that are polynomial intractable. In summary, we obtain the following results in this paper.

<table>
<thead>
<tr>
<th>Query</th>
<th>Data Complexity</th>
<th>Combined Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single</td>
<td>Group</td>
</tr>
<tr>
<td>U</td>
<td>PTIME</td>
<td>NP-complete</td>
</tr>
<tr>
<td>SJ</td>
<td>PTIME</td>
<td>PTIME</td>
</tr>
<tr>
<td>SPJ</td>
<td>PTIME (no self-join)</td>
<td>NP-complete (finite dom., even without FD)</td>
</tr>
<tr>
<td>SJU</td>
<td>PTIME</td>
<td>NP-complete</td>
</tr>
</tbody>
</table>

Table 3.2. Results in this paper.

**Theorem 12.** FD-vsef-IP for $U$ query is NP-complete under group insertion on data complexity.

**Proof.** We construct a PTime reduction from the 3-colorability problem. The main idea is to use $\Delta V$ to encode a graph, and then to decide whether $\Delta V$ can be divided into Red, Green, and Blue tables under the FD-restriction. Specifically,

**Base instance** $D$. Let $D$ include three relations $Red$, $Green$ and $Blue$ with the same schema $\{A_1, B_1, \cdots, A_d, B_d\}$, where $d$ is the maximum degree of the input graph $G$. Initially, they are all empty.

**FD set** $\Sigma$. For $Red$, $Green$ and $Blue$, set $d$ FDs $A_i \rightarrow B_i$ for each one.

**Query** $Q$. Let query $Q := Red \cup Green \cup Blue$.

**View** $V$. Initially, $V$ is empty.

**Insertion** $\Delta V$. Let $\Delta V$ have $n$ tuples, and each tuple $t_i$ simulates a vertex $v_i$ in the input graph $G$ such that $\{B, \cdots, B\}$ initially. Then, for each edge $e(v_i, v_j)$ of $G$, find a $k$ such that $t_i[A_k, B_k]$ and
t_j[A_k, B_k] are both ‘(B, B)’. Note that, we can always find such a k(< d) since d is the maximum degree. Let (1) \( t_i[A_k] = t_j[A_k] = a_{ij} \); (2) \( t_i[B_k] = b_i \); (3) \( t_j[B_k] = b_j \).

One can easily verify that there is an FD-restricted side-effect-free insertion \( \Delta \) if and only if \( G \) can be colored by 3 colors.

**Theorem 13.** FD-vsef-IP for an SJ query is coNP-complete under group insertion on combined complexity.

**Proof.** For the upper bound, we should first briefly introduce some definition. Consider any SJ query \( Q \) with an equivalent standard form \( \sigma_c(R_1 \times \cdots \times R_m) \). Here, each \( R_i \) (\( 1 \leq i \leq m \)) is a table included in \( D \), and any two tables in \( R_1, \cdots, R_m \) may be the same table in the database, if \( Q \) includes self-join. For view insertion \( \Delta V \), we define its inverse on \( D \) with respect to \( Q \). For each \( i \), let \( \Delta V^{-1}_{R_i} \) be the projection on \( R_i \) of \( \Delta V \). In any query, if a self-joined table \( R_i \) has \( h \) occurrences, say \( R_{i_1}, \cdots, R_{i_h} \), let \( \Delta V^{-*}_{R_i} \) be the union \( \Delta V^{-1}_{R_{i_1}} \cup \cdots \cup \Delta V^{-1}_{R_{i_h}} \). Given any SJ query and database \( D \), for view insertion \( \Delta V \), we define its inverse as \( \Delta V^{-1} = \{ \Delta V^{-*}_{R_{i_1}}, \ldots, \Delta V^{-*}_{R_{i_m}} \} \).

Now, we present an NP algorithm to decide if there is no FD-restricted side-effect-free insertion as follows, (a) first compute \( \Delta V^{-1} \); (b) check whether \( D \cup \Delta V^{-1} \models \Sigma \), if no, then return yes; (c) guess a fact \( \mu \) of \( Q(D \cup \Delta V^{-1}) \), if \( Q(\mu) \notin V \cup \Delta V \), then return yes.

**Correctness.** We claim that \( \Delta V^{-1} \) is the minimum necessary insertion into \( D \) in order to produce the result \( \Delta V \).

For the lower bound, the insertion propagation problem for \( J \) query is coNP-hard [39].

The following result is surprising for SPJ queries. Contrary to its counterpart deletion case, insertion propagation is not always harder than deletion propagation.

**Theorem 14.** FD-vsef-IP for an SPJ query is NP-complete under group insertion with finite domain on data complexity, even without FD. It is PTime for an SPJ query without self-join under single insertion on data complexity.

**Proof.** The former part is actually a dual case of the singleton deletion propagation. To prove the former part, the main idea is to modify the reduction from monotone-3-sat of Theorem 2.1 in [53]. We build the source database \( D \) as the same, but initially there are only tuples with form like
$(a_i, x_i)$ of $R_1$ and $(x_i, c_j)$ of $R_2$. Therefore, the view only includes tuples like $(a_i, c_j)$. Then build the insertion $\Delta V$ including tuples like $(a, c_j)$ and $(a, c_j)$. Refer to [33], one can verify the correctness.

For the later part, we show an algorithm for this problem without FD. Without loss of generality, let a query be defined as a join on $k$ relations having no more than $n$ tuples.

A. For a possible insertion choice, we check all ways of value invention as follows,

(a) fix the projected attribute of the intended inserted tuples;

(b) find the scope values for each semi-free attribute;

(c) try all the possible values of each semi-free attribute and check whether it produces side-effect. Here we take all the absent values of a semi-free attribute as a variable $x$ different from all the active values of this attribute. Therefore, this step can be finished in $O((n + 1)^c)$ ($c$ is the number of selection conditions);

(d) take the side-effect free candidate choice of the above step, and check whether it has a consistent fill for the rest free attributes and all the variables, then check whether this step produces $\Delta$ after fixing the variable. If yes, we can claim that FD-vsef-IP has a solution.

B. If all the choices are not valid or consistent, we can decide that FD-vsef-IP has no view side-effect free insertion.

Then one can check the correctness of this algorithm.

Remark. We claim that in Step (d), no side-effect will be produced by fixing the variable, except the new tuple produced by all the newly inserted tuples, since there exists no value of other corresponding semi-free attribute which equals to it. Intuitively, the infinite domain permits the freedom of chase, this will make the decision easy, since the value invention is free. However, if the domain is finite, then this leads to the hardness of value invention, values must be invented carefully to avoid pruning of possible solution.

Compare with the above result, we next show an $SPJ$ query is even harder on combined complexity under finite domain unless $P=NP$. 
Theorem 15. FD-vsef-IP for an SPJ query is $\Sigma^P_2$-complete under group insertion with finite domain on combined complexity.

Proof. i. We first show a $\Sigma^P_2$ algorithm for SPJ queries as follows.

(a) guess $\Delta D$ base on $\Delta V$ by filling all the attributes projected out. When the domain is finite, then we can guess an invention, it shows the correctness.

(b) if $D \cup \Delta D \mid= \Sigma$, then decide whether $Q(D \cup \Delta D) = V \cup \Delta V$, i.e., there is no side-effect on view. As an SPJ query is monotone, we only need to decide whether $Q(D \cup \Delta D) \subseteq V \cup \Delta V$ and this decision problem is in $\text{coNP}$, because we can decide whether there is side-effect on view by guessing a fact $\mu$ of $D$ and checking whether $Q(\mu) \notin V \cup \Delta V$.

ii. We prove the lower bound of FD-vsef-IP for an SPJ query is $\Sigma^P_2$-hard by a reduction from the 3-CNF-SAT$_2$ problem. An instance of the 3-CNF-SAT$_2$ problem includes two variable sets $X_1 = \{x_1, \ldots, x_n\}$ and $X_2 = \{x_{n+1}, \ldots, x_{n'}\}$, and a 3-CNF boolean expression $\phi$ with $m$ clauses $\{C_1, \ldots, C_m\}$. The task is to determine whether there is an assignment $\tau$ for $X_1$ such that $\phi$ is satisfied by all the assignments for $X_2$. We show the reduction as follows. (An example of the reduction for a 3-CNF-SAT$_2$ instance $\phi = \exists x_1 x_2 \forall x_3 x_4 (x_1 + x_2 + x_3)(x_1 + \overline{x}_2 + x_3)(\overline{x}_1 + \overline{x}_3 + x_4)$ as shown in Fig.3.2)

Base instance $D$. Let $D$ have $n+m+1$ relations $S_1, \ldots, S_n$, $R_1, \ldots, R_m$ and $T$, where $S_i$ simulates existential variable $x_i$, $R_j$ simulates clause $C_j$ and the universal variables in it, and $T$ is an auxiliary relational table. Concretely,

(1) For each existential variable $x_i$ ($1 \leq i \leq n$), construct relation $S_i = \{A_1, A_2\}$, where $A_1$'s domain of $\text{dom}(A_1)$ is $\{B\}$ and $\text{dom}(A_2)$ is $\{0, 1\}$. Initially, each $S_i$ is empty.

(2) Let the auxiliary one be a unary relation $T(A)$, where $\text{dom}(A)$ is also $\{0, 1\}$. Initially, $T$ is also empty.

(3) For each clause $C_i$ ($1 \leq i < m$), build a quintuple relation $R_i(A_1, A_2, A_3, A_4, A_5)$. We add 8 tuples into table $R_i$, whose values of $A_1, A_2, A_3$ refer to the 8 value assignments of the 3 variables. The values of $A_4$ are the result of the corresponding assignments, i.e., seven ‘+’s or one
and the values of $A_5$ are the IDs of these tuples from ‘1’ to ‘8’.

**FD set** $\Sigma$. Set $fd: A_2 \rightarrow A_1$ for each relation $S_i$, and set $FD: A_1 A_2 A_3 \rightarrow A_5$ for each relation $R_i$.

**Query** $Q$. We first prepare all the join conditions used in a query as follows.

For each existential variable $x_i$ in QSAT instance (that is $i \in [1, n]$), build a join condition

$$con_\exists := S_1.A_1 = \cdots = S_n.A_1.$$  \hfill (3.5)

For the occurrences in clauses, we build a join condition with the conjunctive form such that

$$con_i := q_1 \land q_2 \land \cdots \land q_k,$$  \hfill (3.6)

where each $q_i$ is an equation $S_i.A_2 = R_j.A_p$, if $x_i$ occurs in the $p$-th position of clause $C_j$.

For each universal variable $x_i$ ($i \in [n + 1, n']$), we build a join condition $con_i$ also as a conjunctive form such that each $q_i$ is also an equation $R_i.A_p = R_{i'} A_{p'} = T.A$, if $x_i$ occurs in the $p$-th and $p'$-th positions of clauses $C_l$ and $C_{l'}$.

Let query $Q = Q_1 \times Q_2 \times Q_3$ such that,

$$Q_1 := \pi_{R_1.A_4, \ldots, R_m.A_4}(\sigma_{con_1 \land \cdots \land con_{n+n'}} S \times R \times T),$$  \hfill (3.7)

$$Q_2 := \pi_{S_1.A_1, \ldots, S_n.A_1}(\sigma_{con_\exists}(S))$$  \hfill (3.8)

$$Q_3 := \pi_A(T),$$  \hfill (3.9)

where $R$ is $R_1 \times \cdots \times R_m$, and $S$ is $S_1 \times \cdots \times S_n$.

**View** $V$. Initially, $V = \emptyset$ since $T$ is empty.
**Insertion** \( \Delta V \). Let \( \Delta V = t \times \{0, 1\} \), where \( t = (\underbrace{+}, \ldots, \underbrace{+}, \underbrace{B}, \ldots, \underbrace{B}) \).

\[
\begin{array}{ccc}
A_1A_2A_3A_4A_5 & A_1A_2A_3A_4A_5 & A_1A_2A_3A_4A_5 \\
0 0 1 + 1 & 0 0 0 + 1 & 0 0 0 + 1 \\
0 1 0 + 2 & 0 0 1 + 2 & 0 0 1 + 2 \\
0 1 1 + 3 & 0 1 1 + 3 & 0 1 0 + 3 \\
1 0 0 + 4 & 1 0 0 + 4 & 0 1 1 + 4 \\
1 0 1 + 5 & 1 0 1 + 5 & 1 0 0 + 5 \\
1 1 0 + 6 & 1 1 0 + 6 & 1 0 1 + 6 \\
1 1 1 + 7 & 1 1 1 + 7 & 1 1 1 + 7 \\
0 0 0 \ - 8 & 0 1 0 \ - 8 & 1 1 0 \ - 8 \\
\end{array}
\]

**Figure 3.2. Example for the reduction of theorem 15.**

One can verify the correctness of this reduction. Following are some key properties to prove the correctness.

**P1.** Find a view side-effect free solution \( \Delta D \). \( Q(D \cup \Delta D) = V \cup \Delta V \) if and only if \( Q_1(D \cup \Delta D) = \{+, \cdots, +\}, Q_2(D \cup \Delta D) = \{B, \cdots, B\} \) and \( Q_3(T) = \{0, 1\} \).

**P2.** \( Q_1(D \cup \Delta D) = \{+, \cdots, +\} \) where \( D \cup \Delta D \models \Sigma \) if and only if any tuple with “−” cannot be joined into the view when each \( S_i \) has to be filled with tuple “(B, 0)” or “(B, 1)” and \( T \) has to be filled with both tuples (0) and (1), because all the value combinations of \( A_1, A_2, \) and \( A_3 \) have been in each \( R_i \). It indicates that each universal variable can be assigned arbitrarily.

**P3.** \( Q_2(D \cup \Delta D) = \{B, \cdots, B\} \) if and only if each \( S_i \) has one of \\{(B, 0)\} and \( S_i = \{(B, 1)\} \) due to FD: \( S_i.A_1 \rightarrow S_i.A_2 \) and \( \text{dom}(A_1) = \{0, 1\} \). It indicates that each existential variable in \( X_1 \) has one and only one assignment.

**P4.** \( Q_3(D \cup \Delta D) = \{0, 1\} \) if and only if \( T \) has two unary tuples (0) and (1).

Next, we show the correctness of the reduction by following two statements.

⇒ If the answer of 3-CNF-SAT₂ instance \( \phi \) is ‘yes’, there must be an insertion \( \Delta D \) into \( D \) such that \( D \cup \Delta D \models \Sigma \) and \( Q(D \cup \Delta D) = V \cup \Delta V \). Suppose \( \phi \) is satisfiable for all the assignments of \( X_2 \) under
the assignment \( \tau(X_1) \). Given \( \tau \), we will construct the FD-restricted view side-effect-free insertion \( \Delta D \). For each relation \( S_i \) (1 \( \leq i \leq n \)), insert tuple \((B, 0)\), if \( \tau(x_i) = 1 \); Otherwise, insert tuple \((B, 1)\). At last, insert tuples \((0)\) and \((1)\) into \( T \). Obviously, \( D \cup \Delta D \models \Sigma \) and the RHS of \( P2, P3 \), and \( P4 \) are satisfied. One can verify that \( Q_1(D \cup \Delta D) = \{+, \cdots, +\} \) since \( \phi \) is a tautology under \( \tau(X_1) \).

\[ \Rightarrow \]

One the other hand, if there is an FD-restriction side-effect-free insertion, then we can define an assignment \( \tau(X_1) \) such that \( \phi \) is true under any assignment of \( X_2 \). In \( D \cup \Delta D \), (a) each \( R_j \) must stay unchanged due to \( "A_1 A_2 A_3 \rightarrow A_5" \); (b) Each \( S_i \) related to the existential variable in \( X_1 \) excludes either \( \{(B, 0)\} \) or \( \{(B, 1)\} \), because \( S_i \) should satisfy \( A_2 \rightarrow A_1 \); (c) \( T \) must have two tuples \((0)\) and \((1)\). For each existential variable \( x_i \in X_1 \), define assignment \( \tau \) as follows,

\[
\tau(x_i) := \begin{cases} 
1, & \text{if } (B, 1) \text{ is in } S_i; \\
0, & \text{otherwise}. 
\end{cases}
\]

To show \( \tau \) is a valid assignment for \( \phi \): (a) Consider any tuple \( \vec{t} \in S \times R \times T \) in \( D \cup \Delta D \) such that \( \vec{t}[S_i.A_2] = \tau(x) \) (1 \( \leq i \leq n \)) and there exists a "−" in \( \vec{t}[R_j.A_4] \) for some \( j \). It must violate some join condition "con_i" of universal variables. (b) Consider any other \( \vec{t} \in S \times R \times T \) in \( D \cup \Delta D \) such that \( \vec{t}[S_i.A_2] = \tau(x) \) (1 \( \leq i \leq n \)) but there is no "−" in \( \vec{t}[R_j.A_4] \) for any \( j \). It must be valid according to the join conditions, since all the assignments of universal variables are filled in \( \{A_1, A_2, A_3\} \) of each \( R_j \). Therefore, \( \tau \) must be valid.

It is not hard to verify that such \( \vec{t} \) always violates at least one condition defined in \( Q_0 \) due to \( P1 \) where \( Q_0(I_r) = \{t'\} \). That is, there is some condition con_k such that \( t \) does not satisfy con_k.

Because \( \vec{t} \) satisfies conditions \( t[R_k.A_4]="-" \), then \( \vec{t} \) refer to the assignments on \( \{A_1, A_2, A_3\} \), which does not appear in \( R_k \). It means clause \( C_k \) is satisfied under such an assignment. Because for each assignment on \( X_2 \), there is at least one true clause, then \( \phi \) is tautology under assignment \( \tau \).

Surprisingly, we show that on combined complexity, for an \( SJU \) query, \( FD-vsef-IP \) is \( \Sigma_2^p \)-complete, not \( coNP \)-complete as expected.

**Theorem 16.** \( FD-vsef-IP \) for an \( SJU \) query is \( \Sigma_2^p \)-complete under group insertion with finite domain on combined complexity.
Proof. i. We first show a $\Sigma^P_2$ algorithm for SJU queries as follows. Given a normal form of an SJU query $Q := Q_1 \cup \cdots \cup Q_k$,

(a) Divide $\Delta V$ into $k$ groups $\Delta V_1, \cdots, \Delta V_k$ (empty permitted);

(b) Compute $D \cup \Delta V_1^{-1} \cup \cdots \cup \Delta V_k^{-1}$ denoted as $D^*$;

(c) If $D^*$ satisfies the FD set, then use the NP-oracle to decide that ‘each SJ query $Q_i(D^*)$ has no side-effect on view $V \cup \Delta V$’ and ‘each tuple of $\Delta V$ is included in $Q(D^*)$’. If so, return yes.

Its correctness is guaranteed by the upper bound proof in Theorem 15.

ii. We prove the lower bound of $\text{FD-vsef-lp}$ for an SJU query is $\Sigma^P_2$-hard by a reduction from the 3-DNF-SAT_2 problem. Similar to 3-CNF-SAT_2, its instance also includes a set of existential variables $X_1 = \{x_1, ..., x_n\}$, a set of universal variables $X_2 = \{x_{n+1}, ..., x_{n'}\}$, and a 3-DNF boolean expression $\phi$ with $m$ clauses $\{C_1, \ldots, C_m\}$. The task is to decide whether there is an assignment $\tau$ for $X_1$ such that $\phi$ is satisfied by all the assignments for $X_2$. We show the reduction as follows. (An example reduction for a 3-DNF-SAT_2 instance $\phi = \exists x_1 x_2 \forall x_3 x_4 (x_1 \land x_2 \land x_3)(x_1 \land \overline{x}_2 \land x_3)(\overline{x}_1 \land \overline{x}_3 \land x_4)$ is shown in Fig. 3.3)

**Base instance $D$.** Let $D$ have $m + 4$ relations $S_+, S_-, R_1, \cdots, R_m$ and $G, T$, where $S_+$ and $S_-$ simulate existential variable $x_i$, $R_j$ simulates clause $C_j$ and the universal variables in it, and $G$ and $T$ are both auxiliary tables. Concretely,

1. For the existential variables in $X_1$, let unary relation $S_+ = \{A\}$ and $S_- = \{A\}$. Initially, both are empty.

2. Let $T$ be a unary relation $T\{A\}$. Initially, $T$ is empty. Let $G$ be a $4m$-ary relation $G\{A_1, \cdots, A_{4m}\}$. Initially, it has only one tuple $(B, \cdots, B)$.

3. For each clause $C_i$, let quaternary relation be $R_i\{A_1, A_2, A_3, A_4\}$. Initially, 7 tuples are filled into table $R_i$, and the values of $A_1, A_2, A_3$ refer to the 7 value assignments of the 3 variables making clause $C_i$ False. The values of $A_4$ are all ‘−’. At last, if there exists an existential variable, say $x_i$, in the $p$-th position clause $C_i$, then we substitute value ‘1’ of $A_p$ with $x_i$ and value ‘0’ with $\overline{x_i}$.

**FD set $\Sigma$.** Set FD: $\emptyset \rightarrow A$ for $S_+$ and $S_-$. 
Query $Q$. We first prepare all the join conditions used in a query as follows.

Existential variable: for the occurrences in clauses, we build a join condition in a conjunctive form such that

$$con_\exists := q_1 \land q_2 \land \cdots \land q_k,$$

where each $q$ is an equation $S_+.A = R_j.A_p$, if $x_i$ occurs in the $p$-th position of clause $C_j$.

Universal variable: same as the proof for an SPJ query, for each universal variable $x_i$ ($i \in [n+1,n']$), we build a join condition $con_i$ also in a conjunctive form such that each $q$ is also an equation $R_i.A_p = R'_{i}.A_{p'} = T.A$, if $x_i$ occurs in the $p$-th and $p'$-th positions of clauses $C_i$ and $C_{i'}$.

Let query $Q = Q_1 \cup Q_2$ such that,

$$Q_1 := \sigma_{con_\exists \land con_{n+1} \land \cdots \land con_{n+n'}}(S_+ \times R_1 \times \cdots \times R_m \times T),$$ (3.13)

$$Q_2 := (S_+ \cup S_-) \times G \times T$$ (3.14)

View $V$. Initially, $V = \emptyset$ since $T$ is empty.

Insertion $\Delta V$. Let $\Delta V = t \times \{x_1, \bar{x}_1, \cdots, x_n, \bar{x}_n\} \times \{0,1\}$, where $t = (B, \ldots, B)$. Note that $|\Delta V| = 4n$, i.e., it is $\text{poly}(n)$.

Figure 3.3. Example reduction of Theorem [16].
One can verify the correctness of this reduction. Following are some key properties to prove the correctness.

**P1.** Find a view side-effect free solution $\Delta D$. $Q(D \cup \Delta D) = V \cup \Delta V$ if and only if $Q_1(D \cup \Delta D) \subseteq \Delta V$ and $Q_2(D \cup \Delta D) = \Delta V$.

**P2.** $Q_2(D \cup \Delta D) = \Delta V$ if and only if $S_+ \cup S_- = \{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n\}$ and $T$ has two unary tuples (0) and (1).

**P3.** As required that $S_+ \cup S_- = \{x_i, \bar{x}_i | i \in [1, n]\}$, each pair ‘$x_i, \bar{x}_i$’ cannot be in the same table due to the FD-restriction. Additionally, $T$ has to be filled with both tuples (0) and (1). Therefore, $Q_1(D \cup \Delta D) \subseteq \Delta V$ where $D \cup \Delta D \models \Sigma$ if and only if any fact with $m$ “$-$”s violates the join condition for universal variables and clauses, because all the value combinations of $A_1, A_2, A_3$ have been in each $R_i$. It means that each universal variable can be assigned arbitrarily making clause False.

Next, we show the correctness of this reduction by following two statements.

$\Rightarrow$ If the answer of 3-DNF-SAT$_2$ instance $\phi$ is ‘yes’, there must be an insertion $\Delta D$ into $D$ such that $D \cup \Delta D \models \Sigma$ and $Q(D \cup \Delta D) = V \cup \Delta V$. Suppose $\phi$ is satisfiable for all the assignments of $X_2$ under the assignment $\tau (X_1)$. Given $\tau$, we will construct the FD-restricted side-effect-free insertion $\Delta D$. For each existential variable $x_i$, if $\tau(x_i) = 1$, then insert tuple $(x_i)$ into $S_+$ and $(\bar{x}_i)$ into $S_-$. Otherwise, insert tuple $(\bar{x}_i)$ into $S_+$ and $(x_i)$ into $S_-$. At last, insert tuples (0) and (1) into $T$. Obviously, $D \cup \Delta D \models \Sigma$ and the RHS of P2 and P3 are satisfied. One can verify that there is no result with $m$ ‘$-$’s since $\phi$ is a tautology under $\tau (X_1)$.

$\Leftarrow$ One the other hand, if there is an FD-restriction side-effect-free insertion, then we can define an assignment $\tau (X_1)$ such that $\phi$ is True under any assignment of $X_2$. In $D \cup \Delta D$, both $S_+$ and $S_-$ related to the existential variable excludes either $(x_i)$ or $(\bar{x}_i)$, because of $S_+$ and should satisfy $\emptyset \rightarrow A$, and $T$ must have two tuples (0) and (1).
For each existential variable $x_i \in X_1$, define assignment $\tau$ as follows,

$$\tau(x_i) := \begin{cases} 
1, & \text{if } (x_i) \in S_+; \\
0, & \text{otherwise}. 
\end{cases} \quad (3.15)$$

One can verify that $\tau$ is a valid assignment for $\phi$ according to the statement of $P3$.

In fact, when data is small and query is simple, it is likely to understand and interpret query results for human, but things are different once the volume of data grows and query gets complex, it is going to be no more easy to understand, explain and also impossible to debug manually or by crowd. Previous work explained the specified query result in several different ways, such as causality and responsibility \cite{65,67,79}, which aims to measure the robustness a specified output with respect to given tuple and rank them based on such measurement to tracing errors, intuitively, the more robust an output, the less impact of given tuple. Also some work \cite{74,77} explained it as predicates of a query and so on.

Differently, view propagation is to find a tuple set, such that the deletion (insertion) of it will result in an exactly deletion (insertion) of the intended tuples in the query result without producing any side effect.

### 3.4 Conclusion

In this paper, we study the computational complexity of view propagation problem for conjunctive queries, and figure out its complexity hierarchy. From the results, we know that when the data gets big in volume, automatic propagation analysis is solvable in polynomial time, unfortunately, once conjunctive query gets complex simultaneously, it is intractable. Our work also make several contributions to previous known results for the problems of side-effect free deletion propagation and also could be applied in an update forbidden case of view propagation.
Chapter 4

BOUNDED VIEW PROPAGATION

Traditionally, view propagation seeks a minimum deletion or insertion on source data. They believe such measurement could show the robustness and explain the impact well. However, in this traditional way, the candidate update on source database is picked aimlessly in advance, so that updated database may be very distant from the original one no matter if it is the maximum one. Additionally, this definition will suffer very high computational complexity.

4.1 Introduction

The candidate update on source database could not always be picked randomly, often can be limited as some parts of the database in advance. This is the case in many scenarios, for example, when some tables involved in the conjunctive query are canonical dictionaries, however, if the minimum deletion on source database is located at this table, it is not reasonable, since this table can never be the cause of unintended results. This is also called ‘update forbidden’ in view propagation analysis. Another thing is, if the minimum deletion is very distance from the original source data, usually it is also not reasonable to explain the results, but we know that for conjunctive query, the view propagation has high computational complexity, typically $\text{NP-hard}$ even on data complexity, therefore, this method is too unwise.

Further more, one can specify the scope of candidate update and check the necessary and sufficient reason that result in the view deletion exactly. Consider the social network application, e.g., Facebook [106], allow us to build a friend relation, such a relation is several links to the persons who are our friends. Look at the following query.

A query $Q$ is to find all persons that user $u_0$ is able to got in touch with by at most two men-in-middle. In fact, this is a path query. The relation needed for answering $Q$ is only a table $\text{friends}(\text{uid}, \text{fid})$ for friends relation where $\text{uid}$ is the id of user, and $\text{fid}$ is the id of one’s friend, a tuple $(u, f)$ in it represents that $f$ is a friend of $u$, and $u$ is also a friend of $f$. Given these, $Q$ can be
Figure 4.1. Example data and the result of query $Q$:

\[ Q := \pi_{f_1.uid, f_3.fid} (\sigma_{f_1.uid = u_0 \land f_1.fid = f_2.uid \land f_2.fid = f_3.uid} (f_1 \times f_2 \times f_3)) \]

where the join condition is given as

\[ \text{cond}_f := f_1.fid = f_2.uid \land f_2.fid = f_3.uid \]

Fig. 4.1 shows an example tables and query result.

Since each user has fixed related tuples, such as Facebook may require each person has at most 5000 links out, for sake of simple, suppose the limitation of number of links is 4, then consider following tasks,
a) One wants to check whether $u_1$, as a friend of $u_0$, is the critical person who is sufficient and necessary to contact $u_3$ for $u_0$, i.e., deletion of some links of $u_1$ will exactly kick off $(u_0, u_3)$ from the results of query $Q$ without any side effect. Due to the limitation of number of links, we know that $u_1$ has at most 4 friends, then the potential is exactly the tuple set $\{t_1, t_3, t_4, t_5\}$, then we can check all possible subsets of it, at last, even if delete all links of $u_1$, $(u_0, u_3)$ is still in the result of query $Q$. Therefore, there is no solution contained in $\{t_1, t_3, t_4, t_5\}$.

b) One wants to check whether $u_2$, as another friend of $u_0$, is the critical person who is sufficient and necessary to contact $u_3$ for $u_0$. Due to the limitation of number of links, we know that $u_2$ has also at most 4 friends, then the potential is exactly the tuple set $\{t_4, t_5, t_6, t_7, t_8\}$, then we can check all possible subsets of it, then we find that if delete tuple $(u_2, u_3)$ from friend, $(u_0, u_3)$ is kicked off the result of query $Q$. Therefore, there exists solution contained in $\{t_4, t_5, t_6, t_7, t_8\}$, i.e., $u_2$ is the critical one verified by deletion propagation analysis.

Contributions. We formally study the bounded view propagation including deletion and insertion cases, a weak version of view propagation where the candidate update on the source database is limited as subset of ‘potential’ $C$ which is a fixed small tuple set of $D$. This definition avoids the unreasonable update or explanation, and also leads to a lower computational complexity. In this paper, we study the computational complexity of this problem for conjunctive queries, and also make several contributions to previous known results for the problems of side-effect free deletion propagation.

Concretely, we formally define bounded view propagation problem, it decreases the computational complexity no matter the structure of given conjunctive query.

We show the fixed potential is actually a dichotomy for both deletion and insertion propagation, and figure out the results on combined complexity which is somehow neglected previously. A detailed complexity analysis (of the combined complexity) is given for conjunctive queries, specifically, for deletion propagation, the data complexity drops from NP-hard to PTIME, we show the combined complexity drops to $\text{NP}(k)$-complete which is in Boolean Hierarchy contained in $\Delta^p_2$; For insertion propagation, our complexity analysis show that combined complexity also drops to $\text{NP}(k)$-complete from $\Sigma^p_2$-complete. These results also figure out the combined complexity class where the general
deletion and insertion propagation problem for conjunctive query are actually in $\Sigma^P_2$-complete instead of only NP-hard.

Finally, based on the results of computational complexity, for view propagation, we map out a complete picture of the computational complexity hierarchy for conjunctive queries on both data and combined aspects, it implies the transition between data and combined complexity of this problem. Moreover, this bounded version is also an update forbidden case of view propagation, our results can be also applied in such case.

Figure 4.2. The data complexity of bounded, general, $fd$-restricted deletion propagation

4.2 definition, preparation and overview of results

Let’s begin with some necessary definition. A schema is a finite sequence $\mathbf{R} = \langle R_1, \ldots, R_m \rangle$ of distinct relation symbols, where each $R_i$ has an arity $r_i > 0$ and includes several attributes, denoted by $R_i = \{A_1, \ldots, A_{r_i}\}$. Each attribute $A_i$ has a corresponding set $\text{dom}(A_i)$ which is the domain of values appearing in $A_i$. An database instance $D$ (over $\mathbf{R}$) is a sequence $\langle R_1^D, \ldots, R_m^D \rangle$, such that each $R_i^D$ is a finite set of tuples $\{t_1, \ldots, t_N\}$, each tuple $t_k$ belongs to the set $\text{dom}(A_1) \times \cdots \times \text{dom}(A_{r_i})$. We use $R.A_i$ to indicate the attribute $A_i$ of relation $R$, and also we denote $R^D$ as $R$ without loss of clarity.
Definition 4.2.1. (bounded deletion propagation, b-DP) Given a source database $D$, potential $C \subseteq D$ which is tuple set of size less than some constant $b > 0$, a query $Q$, its view $V = Q(D)$ and a set of tuples $\Delta V$, it is to determine if there is a tuple set $\Delta D$ such that

1. $\Delta D \subseteq C$,
2. $Q(D - \Delta D) = V - \Delta V$.

If $C$ is not specified definitely but also bounded by a constant $b$, then we call it constant bounded deletion propagation, constant b-DP. Obviously, b-DP is a special case of constant b-DP. If $C$ is not bounded and equals to source database $D$, then it is the general deletion propagation problem.

In our definition of bounded insertion propagation problem, only input a constant as bound.

Definition 4.2.2. (bounded insertion propagation, b-IP) Given a source database $D$, a constant $b > 0$, a query $Q$, its view $V = Q(D)$ and a set of tuples $\Delta V$, it is to determine if there is a tuple set $\Delta D$ such that

1. $|\Delta D| \leq b$,
2. $Q(D + \Delta D) = V + \Delta V$.

The conjunctive query $Q$ is written by operations in relational algebra including selection, projection, join.

We examine the impact of different combinations of these factors on both data and combined complexity of these problems. Data complexity is the complexity expressed in terms of the size of the database only, while combined complexity is the complexity expressed in terms of both the size of the database and the query expression [104].

4.2.1 Preparation

Boolean Hierarchy BH. The boolean hierarchy is the hierarchy of boolean combinations (intersection, union and complementation) of NP sets. Equivalently, the boolean hierarchy can be described as the class of boolean circuits over NP predicates. It is defined as follows:

- NP$(0)$ is PTIME.
- NP$(1)$ is NP.
- NP(2k) is the class of languages which are the intersection of a language in NP(2k − 1) and a language in coNP.

- NP(2k + 1) is the class of languages which are the union of a language in NP(2k) and a language in NP.

- Boolean Hierarchy is the union of all NP(k).

Here, for example,

\[ \text{NP}(2) = \{ L_1 \cap L_2 | L_1, L_2 \in \text{NP} \} \]

\[ \text{NP}(3) = \{ (L_1 \cap L_2) \cup L_3 | L_1, L_2, L_3 \in \text{NP} \} \]

\[ \text{NP}(4) = \{ ((L_1 \cap L_2) \cup L_3) \cap L_4 | L_1, L_2, L_3, L_4 \in \text{NP} \} \]

\[ \ldots \]

**SAT-UNSAT Problem**: Given two Boolean expressions \( \phi, \phi' \), both in conjunctive normal form with three literals per clause, it is to determine if \( \phi \) is satisfiable and \( \phi' \) is not. This problem is \( \text{NP}(2) \)-complete.

Given a Boolean expression \( \phi \), we call \( \phi \in \text{SAT} \) if it is satisfiable; and \( \phi \in \text{UNSAT} \) if it is unsatisfiable, i.e., \( \neg \phi \) is a tautology.

**SAT(k) Problem** [107]. Given \( k \) Boolean expressions \( \phi_1, \ldots, \phi_k \), define predicate \( f \) such that

\[
p(\phi_k) := \begin{cases} 
\phi_k \in \text{SAT}, & \text{if } k \text{ is odd;} \\
\phi_k \in \text{UNSAT}, & \text{if } k \text{ is even.}
\end{cases}
\]

Then SAT(k) problem is to determine if the formula \( f(p(\phi_1), \ldots, p(\phi_k)) \) is true, where the formula is defined based on predicates \( f \) and \( k \) Boolean expressions \( \phi_1, \ldots, \phi_k \) as follow,

\[
f(p(\phi_1), \ldots, p(\phi_k)) :=
\begin{cases} 
p(\phi_1), & \text{if } k = 1; \\
f(p(\phi_1), \ldots, p(\phi_{k-1})) \land p(\phi_k), & \text{if } k \text{ is even}; \\
f(p(\phi_1), \ldots, p(\phi_{k-1})) \lor p(\phi_k), & \text{if } k \text{ is odd.}
\end{cases}
\]
It is shown that SAT(k) problem is $\text{NP}(k)$-complete \cite{107}. Especially, if the given boolean expressions $\phi_1, \ldots, \phi_k$ are all in conjunctive normal form with three literals per clause, then it is call CNF(k) problem.

To simplify the proof in this paper, we here give the following lemma.

**Lemma 3.** CNF(k) problem is $\text{NP}(k)$-complete.

*Proof.* (1) Upper bound. CNF(k) is a special case of SAT(k), therefore CNF(k) is in $\text{NP}(k)$. Specifically, to decide if $f_k$ is true can be answered by a query to NP-oracle, and no more than $k$ queries is required.

(2) Lower bound. Given an instance of SAT(k), say boolean expressions $\phi_1, \cdots, \phi_k$, by the reduction used in the proof of NP-completeness of 3SAT problem. Specifically, for each $k$, $\phi_k$ is reduced to another boolean expression $\phi'_k$ in conjunctive normal form with three literals per clause, such that $\phi_k$ is satisfiable iff $\phi'_k$ is satisfiable. Then, $p(\phi_k)$ is true iff $p(\phi'_k)$ is true. Then, the input formula $f(p(\phi_1), \cdots, p(\phi_k))$ is true iff $f(p(\phi'_1), \cdots, p(\phi'_k))$ is true. Therefore, CNF(k) is $\text{NP}(k)$-hard. $\square$

Obviously, SAT-UNSAT problem is a special case of CNF(k), where $k = 2$, i.e., $(\phi_1 \in \text{SAT}) \land (\phi_2 \in \text{UNSAT})$ where $\phi_1, \phi_2$ are CNF.

**3-DNF-SAT$_2$ Problem.** An instance of 3-DNF-SAT$_2$ problem includes a set of existential variables $X_1 = \{x_1, \ldots, x_{n'}\}$ and a set of universal variables $X_2 = \{x_{n'+1}, \ldots, x_{n''}\}$, and a 3-DNF boolean expression $\phi$ with $m$ clauses $\{C_1, \ldots, C_m\}$, it is to determine if there is an assignment $\tau$ for $X_1$ such that $\phi$ is satisfied by all assignments for $X_2$.

**3-CNF-SAT$_2$ Problem.** Similar to 3-DNF-SAT$_2$ Problem, given existential and universal variables, 3-CNF boolean expression $\phi$, it is also to determine if there is an assignment $\tau$ for existential variables such that $\phi$ is satisfied by all assignments for universal ones.

It is shown that 3-DNF-SAT$_2$ and 3-CNF-SAT$_2$ problems are both $\Sigma^P_2$-complete. \cite{108}

**4.2.2 Overview of results**

The complexity measure follows the work \cite{53} where the complexity results of propagation problem were first established and the studies \cite{105}, \cite{36}, \cite{37} where the complexity of view update problems was studied.
First, we figure out the precise class of the bounded view propagation problem for conjunctive
query is exactly in Boolean Hierarchy and for each $k$, it has instance which is $\text{NP}(k)$-complete.

Then, we also find the precise class of the general deletion and insertion propagation on combine
complexity, show that both deletion and insertion cases are $\Sigma^P_2$-complete improving results before.

At last, summarize all results in the past and this paper, we map out the hierarchy of the
transition from data complexity to combined complexity [4.2]. Our results also imply that if join
operation is hard to process (possibly due to big data or complex query), then view propagation is
hard to implementation no matter the potential is small.

4.3 Bounded Deletion Propagation

**Theorem 17.** For any $k > 0$, $b$-DP for conjunctive query is $\text{NP}(2k)$-hard on combined complexity.

**Proof.** We first prove it is $\text{NP}(2)$-hard by a reduction from SAT-UNSAT problem. Then inductively
prove it is $\text{NP}(2k)$-hard for general $k$ by reduction from SAT(k) problem.

To show $b$-DP is $\text{NP}(2)$-hard. The task is to simulate the two 3-CNF expressions $\phi, \phi'$, and
their intersection.

i **Simulate $\phi \in \text{SAT.**** Let $\phi$ has $m$ clauses $\{C_1, \ldots, C_m\}$ with variables $X = \{x_1, \ldots, x_n\}$.

Source database $D_{\phi}$. Let $D_{\phi}$ has $m$ relations $R_1, \ldots, R_m$, where $R_i$ simulates clause $C_i$
and variables in it. Concretely, for each clause $C_i$ ($i \in [m]$), initiate a quintuple relation
$R_i (A_1, A_2, A_3, A_4, A_5)$.

(a) Add 7 tuples into table $R_i$, corresponding to 7 different 0-1 combinations assigned to
variables in $C_i$, which make clause $C_i$ satisfied. Let the values of $A_4$ of all such 7 tuples
are `+' which is the value of $C_i$ under corresponding assignments. Let the values of $A_5$
of all such 7 tuples are `x' which is the value of $C_i$ under corresponding assignments.

(b) Add one more tuple $(\Delta, \Delta, \Delta, -, c)$ into each $R_i$ respectively.

(c) Add one more tuple $(\Delta, \Delta, \Delta, -, x)$ into the first table $R_1$. 
Intuitively, the combination simulates all possible truth assignments of variables, signs ‘+’ and ‘-’ identify whether the intermediate join paths consist of variables, signs 'x' and 'c' generates ambiguity.

**Query $Q_\phi$.** Query is defined as follow.

(a) To identify join path of variables, define a join condition

$$cond_{sign} := R_1.A_4 = \cdots = R_m.A_4$$

so that the intermediate join paths are divided into two types, ‘+’ and ‘-’.

(b) To guarantee the consistency of assignment for variables, specify join conditions here. Without loss of generality, suppose the variable $x_i$ ($i \in [n]$) occurs in the $k_i$ clauses, define join condition with conjunctive form such that

$$cond_i := \bigwedge_{j=1}^{k_i} q_j$$

such that each $q$ is also an equation $R_l.A_p = R_{l'}.A_{p'}$, if $x_i$ occurs in the $p$-th and $p'$-th positions of clauses $C_l$ and $C_{l'}$ where $p, p' \in \{1, 2, 3\}$ and $l, l' \in \{1, \cdots, m\}$. At last, let the variable assignment join condition be

$$cond_{var} := \bigwedge_{i=1}^{n} cond_i$$

(c) Let $R$ is $R_1 \times \cdots \times R_m$, then we define the query $Q$ such that,

$$Q := \pi_{R_1.A_5}(\sigma_{\neg cond_{sign} \land cond_{var}}R)$$

**View $V_\phi$.** Initially, one can verify that $V_\phi = Q_\phi(D_\phi)$ has two tuples $\{(x), (c)\}$.

**View deletion $\Delta V_\phi$.** Let $\Delta V_\phi = \{(c)\}$. 
**Simulating \( \phi' \in \text{UNSAT} \).** It is similar with SAT case. Let \( \phi' \) has \( m' \) clauses \( \{C'_1, \ldots, C'_m\} \) with variables \( X' = \{x'_1, \ldots, x'_n\} \).

**Source database \( D_{\phi'} \).** Let \( D_{\phi'} \) has \( m' \) relations \( R'_i \) simulating each clauses. Differently, each \( R'_i \) is a quadruple relation \( R'_i(A_1, A_2, A_3, A_4) \).

(a) Add 7 tuples into each table \( R'_i \), whose values of \( A_1, A_2, A_3 \) also refer to the 7 truth assignments. Let values of \( A_4 \) be ‘+’ representing result of the corresponding assignments.

(b) Add one more tuple \((\Delta, \Delta, \Delta, +)\) into each table \( R'_i \) respectively.

(c) Add one more tuple \((\Delta, \Delta, \Delta, -)\) into table \( R'_1 \).

Intuitively, sign ‘+’ generates ambiguity, sign ‘−’ helps to guarantee the existence of intermediate join paths from the SAT part.

**Query \( Q_{\phi'} \).** Also define join conditions as follows,
(a) To identify join path of variables, build a join condition

\[ \text{con}'_{\text{sign}} := R'_1.A_4 = \cdots = R'_m.A_4 \]

so that the intermediate join paths are also divided into two types `+' and `'.'

(b) Using similar join conditions defined in SAT part to guarantee the consistency of assignment for variables. Without loss of generality, suppose each variable \( x'_i \) occurs in the \( k'_i \) clauses, define join condition with conjunctive form such that

\[ \text{cond}'_i := \bigwedge_{j=1}^{k'_i} q'_j \]

where each \( q' \) is the same as SAT part. At last, let the variable assignment join condition be

\[ \text{cond}'_{\text{var}} := \bigwedge_{i=1}^{n} \text{cond}'_i \]

(c) Let \( R' \) is \( R'_1 \times \cdots \times R'_m \), then we define the query \( Q_{\varphi'} \) such that

\[ Q_{\varphi'} := \pi R'_1.A_4(\sigma_{\text{cond}'_{\text{sign}} \wedge \text{cond}'_{\text{var}}} R') \]

**View** \( V_{\varphi'} \). Initially, one can verify that \( V \) has also two tuples \{(+), (-)\}.

**View deletion** \( \Delta V_{\varphi'} \). Let \( \Delta V_{\varphi'} = \{(+), (-)\} \).

An example reduction is shown in Fig. 4.3 for an instance be \( \varphi = (x_1 + x_2 + x_3)(x_2 + \overline{x}_3 + x_4)(\overline{x}_4 + \overline{x}_5 + x_6) \) and \( \varphi' = (x'_1 + x'_2 + x'_3)(x'_3 + x'_4 + \overline{x}'_5)(x'_4 + \overline{x}'_6 + \overline{x}'_7) \).

iii **Simulate the `conjunction` of** \( \varphi \in \text{SAT} \) **and** \( \varphi' \in \text{UNSAT} \).

**Source database** \( D \). Combine \( D_\varphi \) and \( D_{\varphi'} \), i.e., including \( m+m' \) relations \{\( R_1, \cdots, R_m, R'_1, \cdots, R'_m \)\}. 
Query \( Q_{\phi,\phi'} \). Combine queries \( Q_{\phi} \) and \( Q_{\phi'} \) as follow,

\[
Q := Q_{\phi} \times Q_{\phi'}
\]

View \( V_{\phi,\phi'} \). According to \( Q_{\phi,\phi'}(D) \), we have \( V_{\phi,\phi'} \) has four tuples

\[
\{(x,+), (x,-), (c,+), (c,-)\}
\]

View deletion \( \Delta V_{\phi,\phi'} \). Let \( \Delta V_{\phi,\phi'} \) be

\[
\{(x,+), (c,+), (c,-)\}
\]

Potential \( C \). The potential is the set of tuple \((\Delta, \Delta, \Delta, -, c)\) in \( R_m \) and the tuple \((\Delta, \Delta, \Delta, +)\) in \( R_1' \).

NP(2)-hard is guaranteed by correctness of this reduction shown as following statements.

**For SAT part:** \( \Rightarrow \) If \( \phi \) is satisfiable, i.e. \( \phi \) could be satisfied by some assignment \( \tau \), then deletion permitted tuple \((\Delta, \Delta, \Delta, -, c)\) can be safely deleted from \( R_m \), such that

\[
Q_{\phi}(D_{\phi} - \{(\Delta, \Delta, \Delta, -, c)\}) = V_{\phi} - \Delta V_{\phi}.
\]

If \( \phi \) has only one clause, where \( m = 1 \), it must be satisfiable. The first seven tuples certainly preserves result ‘(x)’. The assignment \( \tau \) makes every clause true, so that at least a join path of type ‘+’ will be preserved. Therefore, the result ‘(x)’ can never be deleted at the same time.

\( \Leftarrow \) One the other hand, if there is a side-effect-free deletion from permitted tuple \((\Delta, \Delta, \Delta, -, c)\) of \( R_m \), then \( \phi \) must be satisfiable. As shown by the case (a) above, if \( \phi \) has only one clause, it is satisfiable no matter what \( \tau \), at the same time, \((\Delta, \Delta, \Delta, -, x)\) of \( R_1 \) ensure the result. If \( \phi \) has two or more clauses, deletion of \((\Delta, \Delta, \Delta, -, c)\) of \( R_m \) prunes away two join paths of type ‘-’ starting from tuple \((\Delta, \Delta, \Delta, -, x)\) and \((\Delta, \Delta, \Delta, -, c)\) in \( R_1 \), by contradiction, if the input \( \phi \)
is unsatisfiable, then there will no join path of type ‘+’, then there will no result ‘x’, results in a side effect. Therefore, \( \phi \) must be satisfiable.

**For UNSAT part:** \( \Rightarrow \) If the UNSAT part \( \phi' \) is unsatisfiable, i.e. \( \phi' \) could not be satisfied by any assignment, then deletion permitted tuple \((\Delta, \Delta, \Delta, +)\) can be safely deleted from \( R'_1 \), such that \( Q_{\phi'}(D_{\phi'} - \{(\Delta, \Delta, \Delta, +)\}) = \{(-)\} \). If \( \phi' \) is unsatisfiable, then there is no join path with \( m \) ‘+’s, i.e., seven tuples in the front of each \( R'_i \) can not produce any result. Once \((\Delta, \Delta, \Delta, +)\) is deleted, only ‘-’ is obtained.

\( \Leftarrow \) If there is a side-effect-free deletion from permitted tuple \((\Delta, \Delta, \Delta, +)\) of \( R'_1 \), then \( \phi' \) must be unsatisfiable. Since there is no join path of type ‘+’, if \( \phi' \) is satisfiable, then there must be at least assignment \( \tau \) making all clauses true, so that there is at least a join path producing result ‘+’, contradicts, therefore, \( \phi' \) is unsatisfiable.

**For intersection:** \( \Rightarrow \) If \( \phi \) is satisfiable and \( \phi' \) is unsatisfiable, i.e., \( \phi \) could be satisfied by some assignment \( \tau \), and all clauses of \( \phi' \) can not be made true by any assignment, then \((\Delta, \Delta, \Delta, -, c)\) of \( R_m \) can be safely deleted from \( R_m \), and \((\Delta, \Delta, \Delta, +)\) can be safely deleted from \( R'_1 \) at the same time, so that

\[
Q_{\phi}(D_{\phi} - \{(\Delta, \Delta, \Delta, -, c)\}) = \{(x)\}
\]

\[
Q_{\phi'}(D_{\phi'} - \{(\Delta, \Delta, \Delta, +)\}) = \{(-)\}
\]

Then the result \( Q_{\phi,\phi'} \) is exactly \( \{(x,-)\} \).

\( \Leftarrow \) As proved in SAT and UNSAT part, if there is a side-effect free deletion from permitted tuple \((\Delta, \Delta, \Delta, -, c)\) of \( R_m \), and \((\Delta, \Delta, \Delta, +)\) can be safely deleted from \( R'_1 \) at the same time, then \( \phi \) must be satisfiable and \( \phi' \) is unsatisfiable. Since if not, at least one more result in \( V_{\phi,\phi'} \) except \( \{(x,-)\} \) could be produced, contradicts.

iv **Simulate the ‘disjunction’ of** \((\phi \in \text{SAT} \land \phi' \in \text{UNSAT})\) **and** \( \phi'' \in \text{SAT} \). Now, to mimic \( \text{CNF}(3) \). First, we use the same reduction to build an instance for the third boolean expression \( \phi'' \).

However, several modifications should be put on the tables and queries as follows,
For $\phi$. Extend table $R_1$ with one more attribute $A_6$, then set the value of $A_6$ as ‘$a$’ of tuples where $A_5$ is ‘x’, and set it as ‘$b$’ of tuples where $A_5$ is ‘c’.

For $\phi'$. Extend table $R_1'$ one more attribute $A_5$, then set the value of $A_5$ as ‘$b$’ of tuples where $A_4$ is ‘+’, and set it as ‘$a$’ of tuples where $A_4$ is ‘-’.

$\phi''$. Extend table $R_1''$ with two more attributes $A_6, A_7$, then set the value of $A_6, A_7$ as ‘$a,a$’ of tuples where $A_5 = x$, and substitute the last tuple $(\Delta, \Delta, c)$ with two new tuples,

$$(\Delta, \Delta, c, a, b)$$

$$(\Delta, \Delta, c, b, a)$$

For $Q_3$. First rewrite $Q_\phi$ by adding attribute $R_1.A_6$ into projection operation, rewrite $Q_{\phi'}$ by adding attribute $R_1'.A_5$ into projection operation, and rewrite $Q_{\phi''}$ by adding attributes $R_1''.A_6, R_1''.A_7$ into projection operation. At last, define

$$cond_{\vee} := (R_1.A_6 = R_1''.A_6) \land (R_1'.A_5 = R_1''.A_7)$$

then build the final query

$$Q_3 := \pi_{R_1.A_5, R_1'.A_4, R_1''.A_5}(\sigma_{cond_{\vee}}(Q_\phi \times Q_{\phi'} \times Q_{\phi''}))$$

For $V$ and $\Delta V$. One can verify that the initial view has three tuples

$$V = \{(x, -, x), (x, +, c), (c, -, c)\}$$

View deletion $\Delta V$ is the last three tuples while preserving the first tuple.

Intuitively, view deletion is defined as the results which represents “the two parts of disjunction are both false, i.e., ($\phi \notin \text{SAT} \lor \phi' \notin \text{UNSAT}$) and $\phi'' \notin \text{SAT}$”. The values ‘$a$’ and ‘$b$’ is exactly to encode the join paths only presents if the two parts are both false, concretely, $(a, b)$ and $(b, a)$ identify the result $(x, +, c)$ and $(c, -, c)$ while $(a, a)$ uniquely identifies the $(x, -, x)$ which is preserved.
Note that, the reduction of disjunction does not increase the size of $V$. All steps are polynomially doable.

\textbf{NP(3)-hard} is guaranteed by correctness of this reduction shown as following statements.

⇒ If at least one of two parts ($\phi \in \text{SAT} \land \phi' \in \text{UNSAT}$) and $\phi'' \in \text{SAT}$ is true, then the deletion permitted tuple in the corresponding table can be safely deleted, while others remain unchanged. The deletion strategy is to delete tuples only in one part, and the others remain unchanged no matter if they are both true. Therefore, result $(x,-,x)$ is always preserved. Meanwhile, at least one of $\{(x,+),(c,-)\}$ of $V_{\phi,\phi'}$ and $\{(c)\}$ of $V_{\phi''}$ could be deleted so as to prune away $(x,+,c)$ and $(c,-,c)$.

⇐ If there is a source deletion, such that the renewed query result is $((x,-,x))$, then there must be an assignment making at least one of both parts true. Consider that, to delete $(x,+,c)$ and $(c,-,c)$, then there must be one of following two cases: (a). source deletion of part $(\phi,\phi')$ prunes away both ‘(x,+)' and ‘(c,-)' simultaneously; (b). source deletion of part $\phi''$ prunes away ‘(c)’. However, the preservation of $(x,-,x)$ guarantees ($\phi \in \text{SAT} \land \phi' \in \text{UNSAT}$) or $\phi'' \in \text{SAT}$. Therefore, when $|C| \leq 4$, the problem is \textbf{NP(3)-hard}.

v **Simulation for general** $k$. For general $k$, alternatively using the reduction of ‘conjunction’ and ‘disjunction’ to mimic CNF($k$). The correctness of such alternative reduction can be proved inductively. We omit the proof here. Intuitively, the size of initial view, i.e., query result, will not increase when mimic the disjunctions, and at most increase doubly, therefore for even $k > 2$, the size of view $|V| \leq 2k$, and for odd $k$, it is less than $2(k-1)$. Moreover, for each reduction of UNSAT, only 1 tuple is permitted to delete, for SAT, no more than 2, so our reduction is overall polynomially doable. At last, to build query $Q_k$ for disjunction when $k$ is odd, the table newly built should be extended with at most $k$ attributes to identify the join paths.

Therefore, if there are $k + 4$ deletion permitted tuples, then it is at least \textbf{NP(k)-hard}.

Next, we show the unbounded version is really harder than bounded case.
4.4 Bounded Insertion Propagation

In additional, we will show that the bounded condition is also a dichotomy for insertion propagation, if there is attribute with finite domain in relation.

**Theorem 18.** $b$-IP for conjunction query is NP-(k)-complete with finite domain on combined complexity.

*Proof.* The proof is based on the reduction using in the proof of theorem 17, but it is more complicated than that, therefore, we here only give the idea, proof sketch.

To simulate CNF(k) problem, we start from building the source database $D$, query $Q$, view $V$ and view deletion $\Delta V$ as deletion case. Then make a modification on $D$ and $Q$ so that change the initial view $V$ become empty, then it is only to insert $\Delta V' = V - \Delta V$, i.e., insert the final result of deletion case.

The idea of modification is to add constant number of tables as ‘on-off’ switches to control whether the query result corresponding the true instances of CNF(k) is going to be shown in the view, while using the finite domain limitation and unexpected result with insertion on switch tables to guarantee that the insertion on source database could be only placed on the switches and the first table corresponding to each SAT instance, all the other table is no benefit to insert any new tuple. At first, let all switches be off, i.e., let these tables are empty, then all join paths are pruned away no matter the CNF(k) instance is true or false, so that the query result is empty. Then, let the view insertion guarantee switch-on, that is, the right tuple enabling join paths should be inserted. And once insertion occurs, input instance of CNF(k) is true iff there is a right insertion $\Delta D$ such that $Q(D + \Delta D) = V + \Delta V$.

Some techniques should be explained here,

(a) to control the result of ‘conjunction’ in CNF(k), one switch table is used to join all the tuples, so that once switch is on, query result is right the answer of type like $(x, -)$ iff two parts of the conjunction are both true.

(b) to control the result of ‘disjunction’ in CNF(k) is a little difficult, two switch tables are add to join two part of the disjunction respectively, so that once the two switches are on, query result is right the answer of type like ‘$(x, -, x)$’ iff at least one of the two parts of the disjunction is true.

The tricky technique here is to design join condition like in the proof of theorem 17, so that when
both parts are false, no possible insertion can be placed to remedy both parts without generating redundancy.

Finally, at most $k$ switch tables are built, meanwhile at most $2k$ tuples are permitted to insertion, so that the reduction is still polynomially doable.

\[\square\]

4.5 Conclusion

The computational complexity of view propagation problem for conjunctive queries has been studied in this chapter. The comprehensive complexity hierarchy has been figured out. The results tell that when the data gets big in volume, automatic propagation analysis is doable in polynomial time, unfortunately, once the conjunctive query gets complex simultaneously, it is intractable even for bounded cases. This work also makes several contributions to previous known results for the problems of side-effect free deletion propagation and also could be applied in an update forbidden case of view propagation.
Data analysis has become a growing concern for smart database systems and applications. Conventional database systems usually do not provide automatic or semi-automatic data analytical tools. It is hard to conduct data analysis or debugging manually which relies on explanations of query results. This is especially the case for large-scale and complex systems. However, due to the rapid growth of data volume, it is turning more difficult to understand, interpret, and debug through investigating data manually [77,109]. To assist users and administrators to better understand their data and resolve problems effectively and efficiently, resilience of query results has been recognized as a fundamental need to derive explanations for query results and surprising observations so as to help data understanding [67,76].

5.1 Introduction

Several methods have been proposed to address this problem by building data provenance [53, 110,112] to perform analysis on causality [113,114], responsibility [66], update propagations [53,65] and so on. Nevertheless, there is no sufficient theoretical analysis of applying these methods to improve the database debugging process. In fact, theoretical study, especially on the computational complexity, is necessary to provide a guidance on how to apply the methods.

To provide a complexity study for data provenance based methods, an important problem is regarding resilience decision. Given a query \( q \) which is initially true, the resilience of database instance \( d \) essentially refers to a set of tuples (facts), denoted as \( res \), such that the deletion of \( res \) makes \( q \) false, i.e., \( q(d - res) \) is empty. The resilience decision problem can be defined as follows.

**Definition 5.1.1 (RES decision problem [64])**. Given a database \( d \), a fixed integer \( k_r > 0 \), and a boolean query \( q \) where \( q(d) \) is already known and initially true, it is to decide if there is a set of facts \( res \subseteq d \) such that \( |res| < k_r \) and \( q(d - res) \) is false.
The resilience decision problem is a fundamental problem in the theoretical study of database operations such as query result explaining, error tracing and so on. In these applications, the common fundamental work is regarded as a question answering task, called *why-provenance*. It studies why some specified partial result $T$ of a given query $q$ is obtained for database instance $d$. There are several ways to define such “why”. Based on a specific definition of “why”, by a search performed on $d$, we are able to trace the corresponding witnesses which are claimed to be the reason of generating $T$.

**Example 1** (Efficient system migration guarantee reliability [64]). Suppose a database instance is recording the information of server users, service requests, and log of access. The example database instance is shown in the figure 5.1.

<table>
<thead>
<tr>
<th>users</th>
<th>uid</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>1</td>
<td>Alice</td>
</tr>
<tr>
<td>$t_2$</td>
<td>2</td>
<td>Bob</td>
</tr>
<tr>
<td>$t_3$</td>
<td>3</td>
<td>Charlie</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>requests</th>
<th>type</th>
<th>details</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_4$</td>
<td>IMAP</td>
<td>Email-In</td>
</tr>
<tr>
<td>$t_5$</td>
<td>SMTP</td>
<td>Email-Out</td>
</tr>
<tr>
<td>$t_6$</td>
<td>DB</td>
<td>Database</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>access_log</th>
<th>uid</th>
<th>type</th>
<th>server</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_7$</td>
<td>1</td>
<td>IMAP</td>
<td>yoda</td>
</tr>
<tr>
<td>$t_8$</td>
<td>2</td>
<td>DB</td>
<td>yoda</td>
</tr>
<tr>
<td>$t_9$</td>
<td>1</td>
<td>SMTP</td>
<td>yoda</td>
</tr>
<tr>
<td>$t_{10}$</td>
<td>1</td>
<td>DB</td>
<td>yoda</td>
</tr>
<tr>
<td>$t_{11}$</td>
<td>3</td>
<td>IMAP</td>
<td>loki</td>
</tr>
<tr>
<td>$t_{12}$</td>
<td>3</td>
<td>DB</td>
<td>yoda</td>
</tr>
<tr>
<td>$t_{13}$</td>
<td>2</td>
<td>SMTP</td>
<td>loki</td>
</tr>
<tr>
<td>$t_{14}$</td>
<td>1</td>
<td>DB</td>
<td>homer</td>
</tr>
</tbody>
</table>

Figure 5.1. Example database instance

Suppose an old server called yoda needs to retire from a department. The department needs to figure out the relationship between this server and others, so that the migration to other servers could be done more efficiently. Obviously, the migration is efficient if the number of service transfers from yoda to other servers is minimized.

In terms of database, the administrator would like to understand why the following database query $q$ evaluates to be true in advance:

$$q : - \text{users}(u, n), \text{access}_\log(u, t, \text{‘yoda’}), \text{requests}(t, d)$$

Observe that email service for Alice and database service are the reasons making query $q$ true. Due to such two factors, to perform efficient migration with reliability guarantee, email service for Alice should be moved to another email server, and the databases on yoda need to be migrated to a different server.
\[ q(n, t, s, d) \]

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>server</th>
<th>detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1 IMAP</td>
<td>yoda</td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td>2 DB</td>
<td>yoda</td>
<td></td>
</tr>
<tr>
<td>Alice</td>
<td>1 SMTP</td>
<td>yoda</td>
<td></td>
</tr>
<tr>
<td>Alice</td>
<td>1 DB</td>
<td>yoda</td>
<td></td>
</tr>
<tr>
<td>Charlie</td>
<td>3 DB</td>
<td>yoda</td>
<td></td>
</tr>
</tbody>
</table>

As aforementioned, many potential changes to the data could correct the query result of \( q \), however, a natural explanation should be simple and concise, and more likely to be an efficient solution to a potential task or the base of the right solution. Therefore, for the underlying context, the minimum number of tuples (deleting tuples \( t_1 \) and \( t_6 \)) whose deletion provides meaningful hints. Such tuples are not explicit recommendations for a particular action (e.g., it would not actually remove a user), but they give indications for possible actions (e.g., update a user’s settings).

Causal responsibility [?] and deletion propagation [53] are two problems similar with that investigated in this paper, which studies how a subset of the source data impacts a query result. Given a query-tuple-database instance \((q, t, d)\), causal responsibility [113] is to find a minimum tuple deletion \( \Gamma \) in the source data other than \( t \) such that in \( d - \Gamma \), query \( q \) is true when \( t \) emerges, but it would become false once \( t \) is deleted. In the above example, tuple \( t_1 \) is the target \( \Gamma \) with respect to \( t_6 \). On the other hand, deletion propagation [53] investigates two ways to define view propagation [74], which are source side effect free (S-sef) and view side effect free (V-sef). Suppose \( d \) is a database instance, \( q \) is a relational query on \( d \), and \( q(d) \) is the materialized view. If there exists tuple \( t \subseteq q(d) \) which is specified as the intended testing result, then S-sef is to find a set of facts \( r \subseteq d \) small enough such that \( q(d - r) \subseteq q(d) - \{t\} \), while V-sef is to find some set of fact \( r \) such that \( q(d - r) = q(d) - \{t\} \). Intuitively, S-sef intends to prevent unnecessary damage towards source data, while V-sef aims to prevent unnecessary damage on materialized views. However, resilience decision problem investigated in this paper is different.

Here we show an example of employing resilience to illustrate the possible solution and hardness.

**Example 2.** Suppose a company maintained a file management database containing two tables as follows. The first table "department" is recording the affiliation of the employees. The second table
"authorization" is recording files which group owns the authorization of the use of them. An testing materialized view is defined as a conjunction: “show the files that a user is authorized to access”.

<table>
<thead>
<tr>
<th>department</th>
<th>authorization</th>
<th>q(u, f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>did</td>
<td>uid</td>
<td>did</td>
</tr>
<tr>
<td>dept_1</td>
<td>usr_1</td>
<td>dept_1</td>
</tr>
<tr>
<td>dept_2</td>
<td>usr_2</td>
<td>dept_2</td>
</tr>
<tr>
<td>dept_2</td>
<td>usr_3</td>
<td>dept_2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Suppose query

\[ q_1 := \text{department}(d, 'usr_1'), \text{authorization}(d, f) \]

is to check whether deleting only one fact form either \text{department} or \text{authorization} is good enough to enforce that the result of \( q_1 \) is false. Consider a suspicion testing on the result of \( q_1 \). There are two candidate facts that can be deleted in \( d \) which are shown below:

- (dept_1, usr_1) from \text{department}, and
- (dept_1, file_1) from \text{authorization}

We can easily verify that each of them is a ‘res’ for the given query \( q(usr_1, x) \).

Then consider another test to suspicious tuple on the result of query

\[ q_2 := \text{department}(d, 'usr_2'), \text{authorization}(d, f). \]

We need to check if it becomes false after the deletion of a single fact from either \text{department} or \text{authorization}. We have three alternative candidate facts shown below,

- (dept_2, usr_2) from \text{department},
- (dept_2, file_2) from \text{authorization}, and
- (dept_2, file_3) from \text{authorization}

We can see that only the first fact is the expected solution of resilience. It is because the absence of the first fact makes \( q_2 \) to become false. However, neither the second fact nor the third fact can be a
solution. Thus, resilience (dept$_2$, usr$_2$) is regarded as a suspicious error or candidate explanation for the result of $q_2$.

In [64], the authors showed a polynomial reduction from RES to S-sef. The work indicates that the study of the lower bound of RES is necessary because it determines the lower bound of both of the two problems. Therefore, we are motivated to investigate the resilience decision problem to provide a comprehensive analysis of its parameterized complexity.

5.2 related works

<table>
<thead>
<tr>
<th>Table 5.1. Polynomial tractable cases of S-sef</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>PTIME</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 5.2. Hard cases of S-sef</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>NP-complete</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>co-W[1]-complete</td>
</tr>
<tr>
<td>co-W[SAT]-hard</td>
</tr>
<tr>
<td>co-W[t]-hard</td>
</tr>
<tr>
<td>co-W[P]-hard</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 5.3. Polynomial tractable cases of V-sef</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>PTIME</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>FPT</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Table 5.4. Hard cases of V-sef

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Citations</th>
<th>Query Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP-complete</td>
<td>Buneman et al. 2002 [53]</td>
<td>selection-free conjunctive queries</td>
</tr>
<tr>
<td></td>
<td>Cong et al. 2012 [51]</td>
<td>non-key-preserving conjunctive queries</td>
</tr>
<tr>
<td></td>
<td>Kimelfeld et al. 2012 [55]</td>
<td>non-head-domination conjunctive queries</td>
</tr>
<tr>
<td></td>
<td>Kimelfeld et al. 2013 [56]</td>
<td>non level-k head-domination conjunctive queries</td>
</tr>
<tr>
<td>NP(k)-complete</td>
<td>Miao et al. 2017 [?      ]</td>
<td>conjunctive queries for bounded source deletions</td>
</tr>
<tr>
<td>Σ²_p-complete</td>
<td>Miao et al. 2016 [103]</td>
<td>conjunctive queries under general settings</td>
</tr>
</tbody>
</table>

Some efforts have been spent on studying the classical computational complexity of S-sef and V-sef. The previous results regarding S-sef are presented in Table 5.1 and 5.2. The previous results regarding V-sef are summarized in Table 5.3 and 5.4. The complexity of relational query languages was first investigated by Chandra and Merlin [115] about 40 years ago. Then, it became one of the most primary concerns in the theoretical study of the database field. Unfortunately, for query evaluation, the available complexity results of query languages are shown in a general way and seem too rough. A few years later, two new metrics are proposed in [116] to measure the complexity of query evaluation. These two new metrics are data complexity and combined complexity. By considering these two metrics, the complexity of query evaluation can be investigated in a more reasonable way. Intuitively, data complexity helps with identifying the scenarios where the query size is small with respect to the database instance involving a reasonable number of join operations. Therefore, it is represented as a function of the size of the database instance and the considered query is regarded as a fixed one. Differently, combined complexity focuses on more general scenarios where both the considered query and database instance are not fixed. Thus, the considered query and database instance are both regarded as variables of the complexity function. The combined complexity for a specific class of queries involving join operations is usually inevitably exponential higher than the data complexity. Because of this reason, data complexity is more meaningful for practical query evaluations, thus it draws a lot of attentions.

Thanks to such two metrics, more complexity results of the view side effect free (V-sef) problem are obtained [51][56]. For data complexity, Kimelfeld et al. [54] introduced a dichotomy called ‘head-domination’. It is shown that for conjunctive queries without self-join, V-sef is polynomial tractable by the unidimensional algorithm for the queries with the head-domination property, and no PTAS even without the head-domination property. The functional dependency restricted version is a natural extension of V-sef. Things change too much with the presence of functional dependency. Thus, the work in [55] introduced another extended dichotomy, called ‘fd-head domination’. Besides single deletion, they also identified a
trichotomy, called ‘level k-domination’, for multi-tuple deletion in \[56\]. It is claimed that finding a k-factored constant-optimal solution is polynomial tractable for a query with the level k-domination property, but it is NP-hard to derive a constant optimal solution. For combined complexity, there exist many results for different fragments of select-project-join-union queries \[51, 52, 103\]. The authors in \[51, 52\] presented the intractable cases and illustrated a key-preserving condition to recognize a polynomial tractable case. The authors in \[103\] investigated the functional dependency restricted V-sef problem. Similarly, several polynomial tractable and intractable cases are presented for V-sef.

Moreover, there are some results on the source side effect free (S-sef) problem \[51, 53, 64\]. The work in \[64\] is very related to this paper. Freire et al. identified the ‘triad’ property for dual hypergraph representation of conjunctive queries. Based on this work, we can know that the resilience decision problem is polynomial decidable if the dual hypergraph of the given query \(q\) excludes the triad structure. Otherwise, the problem is NP-complete. Similarly, if a set of functional dependencies are defined in advance, then the ‘triad’ structure should be changed to ‘fd-induced triad’ which is more general. Then, a tight dichotomy follows.

The view update problem is another related problem. It has been widely investigated for many years. The example works include \[35, 37, 40, 41, 58\]. The view update problem studies how to perform an update on the source data in order to remove ambiguity or achieve the expected update to a specified view while guaranteeing the semantic correctness. Most of the previous works mainly focus on recognizing the conditions under which uniqueness of update can be guaranteed if the update is carried out. It is worth mentioning that an update is not always ideally unique in practice. Therefore, such works are only effective for the restricted scenarios. A more practical study is to identify an update to database instance \(d\) which can enable the specified update to \(q(d)\) while minimizing the side-effect, e.g., unintended update. This is essentially the same as view propagation.

However, the two complexity metrics may not seem completely reasonable. As aforementioned, since the given queries and database instances are processed equally, the practical relationship between query size and database is totally overlooked. Thus, the combined complexity becomes too restrictive. On the other hand, the size of database instance \(d\) is typically much larger than that of query \(q\) \((q \sim o(d))\). Because of this reason, data complexity is always more meaningful and practical for query evaluation. Therefore, more efforts are spent on investigating data complexity. However, as mentioned in \[\ldots\], “polynomial time in the context of data complexity means time \(O(d^q)\), and in fact the known algorithms that place the above-mentioned languages in PTIME have precisely such a running time. Besides this, in the case of fix-point logic, this is known to be inherently unavoidable. Even if \(q < n\), it is not reasonable to consider \(q\) fixed,
because even for small values of $q$, a running time of $n^q$ hardly qualifies as tractable, especially in view of the fact that $n$ is typically huge”.

In this paper, we revisit the resilience decision problem to enrich the results in between. We study the parameterized complexity of this problem. Note that in this case, a tractable class requires a time within the data size to the power of a fixed constant but independent of the query size. This is the typical paradigm of the parameterized complexity theory.

5.3 Preparation

In the context of classical computational complexity, a language $L$ is regarded as a subset of $\Sigma^*$, which includes all the possible strings over alphabet $\Sigma$. Since the problem concerned by the parameterized complexity theory [?] usually has at least two inputs, a language is then defined as $L \subseteq \Sigma^* \times \Sigma^*$, which is different from the definition for classical computational complexity. Such a language is called a parameterized language. A parameterized language $L$ with two inputs is often represented as pair $(x, k)$, where $x$ could be a graph, an algebraic structure, or something else, and parameter $k$ is usually a natural number ($k \in \mathbb{N}$). Then, a considered language is defined as $L \subseteq \Sigma^* \times \mathbb{N}$. For a specific $k$, $L_k = \{(x, k) \mid (x, k) \in L\}$ always denotes the $k$-th slice of $L$. In this paper, the input of the resilience decision problem is defined as the following pair

$$(\langle q, d \rangle, \langle k_r, k \rangle),$$

where $k_r$ is the size of resilience. Here, we investigate two cases. The first case is parameterized by the size of query $q$. The other case is parameterized by the number of variables in query $q$. Then, there are two choices for parameter $k$, with one for each case.

During last two decades, several well-established hierarchies are proposed for the parameterized complexity theory, and some well-known classes are identified.

**Fixed parameter tractable.** The parameterized complexity theory studies a language slice-wisely. A problem is tractable with respect to the piece if for every non-negative integer $k$, the membership of $L_k$ can be decided in a time polynomial to the size of input $x$ but could be a power of $k$ or even higher depending on $k$. Formally, let $c > 0$ be a constant and $f : \mathbb{N} \to \mathbb{N}$ be a computable function. Then a parameterized problem $P$ is fixed-parameter tractable if there exists an algorithm $A$, such that for any input $(x, k)$, $A$ outputs yes in $f(k) \times |x|^c$ time if and only if $(x, k)$ is yes. Class FPT is the set of all the problems which are fixed parameter tractable.
**W-Hierarchy.** For the problems which are probably not fixed parameter tractable, W-hierarchy is introduced in the parameterized complexity theory. Considering the classical complexity theory, W-Hierarchy is analogue to Polynomial-Hierarchy in which problems are classified under the parameterized perspective [?]. Many complete problems can be identified in W-Hierarchy. Specifically, FPT is at the bottom level W[0]. For each \( i > 0 \), W[i] contains the problems whose every instance \((x, k)\) can be reduced to combinatorial circuit that has weft at most \( i \) in a fixed parameter tractable time (fpt-reduction). Such series of complexity classes W[i] are jointly in W-Hierarchy. Obviously, for all \( j \geq i \), W[j] contains W[i], which means the problems in W[j] are likely harder and further from having a fixed parameter tractable algorithm than those in W[i]. Especially, W-Hierarchy has another two classes W[P] and W[SAT].

Now, we briefly introduce several useful concepts for the database theory.

**Database and its instance.** A database schema \( S \) is a vector of distinct relations \( \langle R_1, \ldots, R_m \rangle \). Any relation \( R \) has an arity of \( r \) if it possess \( r \) attributes \( A_1, \ldots, A_r \). \( \text{dom}(A_j) \) denotes the corresponding domain of attribute \( A_j \). Let \( \text{dom}(A_1) \times \cdots \times \text{dom}(A_r) \) be the domain of relation \( R \), denoted as \( \text{dom}(R) \). Any element of \( \text{dom}(R) \) is said to be a fact or a tuple. A database instance is defined as \( \{d; R_1, \ldots, R_m\} \), representing a schema \( \{R_1, \ldots, R_m\} \) over some subset \( d \subseteq \text{dom}(R_1) \times \cdots \times \text{dom}(R_m) \). In this paper, once the schema is fixed, we simply use “\( d \)” to denote a database instance.

**Boolean database queries.** Boolean query \( q \) is a function over database schema \( S \) which is compatible with \( q \). It maps the set of database instances over \( S \) to \{true, false\}. Intuitively, false represents an empty result of the its corresponding non-boolean query. In this paper, the expressive capability of queries is limited to the first-order logic. Inside this scope, conjunctive queries are most widely studied. By the datalog-style notation, a conjunctive query is written as

\[
q := R_1(\bar{x}_1), R_2(\bar{x}_2), \ldots, R_k(\bar{x}_k)
\]

where each \( R_i \) is an atom, each atom has an \( r_i \)-ary vector \( \bar{x}_i \), and each \( x \in \bar{x}_i \) could be a constant or a variable. Let \( q(d) \) be the query result which is either true or false. \( q(d) \) is true if there exist facts \( \{t_1, t_2, \ldots, t_k\} \) in \( d \) that can be mapped to \( \bar{x}_1, \bar{x}_2, \ldots, \bar{x}_k \) consistently. Otherwise, \( q(d) \) is false. For example, query \( q_1 \) in Example 2

\[
q_1 := \text{department}(d, \text{‘usr.1’}), \text{authorization}(y, z)
\]
is a conjunctive query including two atoms (or relations) where the first atom has a variable \(d\) and a specific constant ‘usr 1’, while the second atom has two variables. From the perspective of relational algebra, a boolean conjunctive query can be written as a paradigm with the combination of selection and join operations equivalently. Based on a conjunctive query, a positive query can be written as a disjunction of some conjunctive queries, i.e., \(q : = q_1 \lor q_2 \lor \cdots \lor q_k\) where each \(q_i\) is a conjunctive query. From the relational algebra perspective, it is also equivalent to the query class written by union, selection and join operations. In a classical semantic of disjunction, \(q(d)\) is true if at least some \(q_i\) is true. Otherwise, \(q(d)\) is false. Furthermore, a first-order query is a positive query with negations. It is also equivalent to an arbitrary first-order formula by only using the predicates \(R_1, \ldots, R_m\). Therefore, they can be listed as conjunctive queries, positive queries and first-order queries in an increasing order of expressive capability. In this paper, queries can be defined in terms of one of these three forms.

5.4 Parameterized complexity results for different query classes

The parameterized complexities of different cases is examined in this section, by considering the form of the query \(q\), the number of variables contained in the query \(q\) and the size of the query \(q\) (i.e., the number of atoms appearing in it). Note that the size of any query \(q\) is no more than the number of variables in it, because each non-trivial query should contain at least one variable. Thus, the complexity of the case parameterized by the number of variables should be no higher than the case parameterized by the size of query for the same query class.

5.4.1 Hard Cases

The hard cases shown here are all related with the complement of the classes in W-Hierarchy. The complement of a parameterized problem is the parameterized problem resulting from reversing the yes and no answers. For any \(i \in \mathbb{N}\), a parameterized problem is in co-W[\(i\)] if its complement is in W[\(i\)]. If any co-W[\(i\)]-complete problem is fixed-parameter tractable, then co-W[\(i\)] = FPT = co-FPT = W[\(i\)] follows. However, it may cause the failure of the Exponential Time Hypothesis \(\text{[?]}.\) Therefore, to show that a problem may not be fixed-parameter tractable, co-W[\(i\)]-completeness is a strong theoretical evidence.

**Theorem 19.** RES(\(d, q, k_r, k\)) is co-W[1]-complete if \(q\) is a conjunctive query and \(k\) is either the size of \(q\) or the number of variables in \(q\).

**Proof.** To show the upper bound is co-W[1], a polynomial reduction could be derived from RES (\(d, q, k_r, k\)) for conjunctive queries to the weighted satisfiability problem for 2-CNF formulas. The complementary deci-
sion problem of RES \((d, q, k_r, k)\) is to find out whether no set of facts, denoted as \(res\), from \(d\) can be found such that \(q(d - res)\) is \(false\) and \(|res| < k_r\). Without loss of generality, suppose \(q := p_1(\bar{x}_1), \ldots, p_k(\bar{x}_k)\).

It is to ask whether an assignment of the variables in \(q\) could be found such that each atom in \(q\) could be mapped to at least \(k_r + 1\) different facts from database instance \(d\).

**Lemma 4.** The complement of RES has a solution for constant \(k_r\) if and only if \(k_r + 1\) disjoint join paths could be found from the database instance \(d\) as the witnesses of query \(q\).

A fact \(t\) is said to be **consistent** with an atom \(p\) only if (1) the \(i\)-th entry of fact \(t\) is the constant \(c\) whenever the \(i\)-th entry of atom \(p\) is some constant \(c\); and (2) the \(i\)-th and \(j\)-th entries of fact \(t\) are the same constant whenever the \(i\)-th entry of atom \(p\) is a variable, say \(x\), and \(j\)-th entry of atom \(p\) is also the variable \(x\).

We here introduce \(k_r + 1\) indicators, \(I_{i,t}^{(j)}\) where \(0 \leq j \leq k_r\) for each atom \(p_i\) in the given query \(q\) and each fact \(t\) consistent with atom \(p_i\).

\[
I_{i,t}^{(j)} := \begin{cases} 
1, & \text{if } p_i \text{ is mapped to fact } t \text{ in the } j\text{-th join path;} \\
0, & \text{otherwise.}
\end{cases}
\]

A 2-CNF can be defined as follows by using only these indicators introduced above.

**Fact:** a 2-CNF formula \(\phi_t\) can be built for each fact \(t\):

\[
\bigwedge_{0 \leq j_1 \neq j_2 \leq k_r, 1 \leq i \leq k} \left( \neg I_{i,t}^{(j_1)} \lor \neg I_{i,t}^{(j_2)} \right)
\]

**Atom:** a 2-CNF formula \(\phi_i\) can be built for each atom \(p_i\) and two different facts \(t\) and \(t'\) consistent with it:

\[
\bigwedge_{0 \leq j \leq k_r} \left( \neg I_{i,t}^{(j)} \lor \neg I_{i,t'}^{(j)} \right).
\]

**Conjunction of atoms:** for any two atoms \(p_i\) and \(p_{i'}\) and any two facts \(t\) and \(t'\), if \(p_i\) and \(p_{i'}\) have some common variable \(v\), and \(t\), \(t'\) is consistent with \(p_i\) and \(p_{i'}\) respectively, but facts \(t\) and \(t'\) have different constants in the position of \(v\), (i.e., \(t\) and \(t'\) are not able to occur in the same join path simultaneously), then a 2-CNF expression \(\phi_{(i,t),(i',t')}\) can be built as

\[
\bigwedge_{0 \leq j \leq k_r} \left( \neg I_{i,t}^{(j)} \lor \neg I_{i',t'}^{(j)} \right).
\]
Combine these 2-CNF formulas introduced, the complete 2-CNF formula $F$ can be built as follows

$$
\bigwedge_{1 \leq i \neq i' \leq k, t \neq t' \in d} \phi_{(i,t),(i',t')} \bigwedge_{1 \leq i \leq k, t \in d} \phi_i \bigwedge_{t \in d} \phi_t
$$

Let the parameter of 2-CNF weighted satisfiability problem be $(k_r + 1) \cdot k$, where $k$ is the parameter of RES, i.e., the size of query $q$.

**Example 3.** Consider an instance of $\text{RES}(d, q, 1, 2)$, where query $q$ contains $k = 2$ atoms

$$
q \leftarrow r(x_1, x_2, '1'), s(x_2, '2', x_3)
$$

and a database instance containing four facts

$$
d \leftarrow r_1(a, a, '1'), r_2(a, b, '1'), s_1(a, '2', b), s_2(b, '2', b).
$$

Note that $k_r = 1$ and parameter $k = 2$, thus we have variables

$$I_r^{(0)}_{r_1}, I_r^{(1)}_{r_1}, I_r^{(0)}_{r_2}, I_r^{(1)}_{r_2},
$$

$$I_s^{(0)}_{s_1}, I_s^{(1)}_{s_1}, I_s^{(0)}_{s_2}, I_s^{(1)}_{s_2}.
$$

The 2-CNF instance could be shown as follows.

(a) Build the expression for ‘facts’:

$$\phi_{r_1} \land \phi_{r_2} \land \phi_{s_1} \land \phi_{s_2},$$

where

$$\phi_{r_1} = \neg I_{r_1}^{(0)} \lor \neg I_{r_1}^{(1)},$$

$$\phi_{r_2} = \neg I_{r_2}^{(0)} \lor \neg I_{r_2}^{(1)},$$

$$\phi_{s_1} = \neg I_{s_1}^{(0)} \lor \neg I_{s_1}^{(1)},$$

$$\phi_{s_2} = \neg I_{s_2}^{(0)} \lor \neg I_{s_2}^{(1)}.$$

The expression for ‘facts’ is true if one of the variables for each fact is true, that is, each fact should be mapped to at least one atom in some join path.
(b) Build the expression for ‘atoms’: $E_r \land E_s$, where

$$
\phi_r = (-I_{r,r_1}^{(0)} \lor -I_{r,r_2}^{(0)}) \land (-I_{r,r_1}^{(1)} \lor -I_{r,r_2}^{(1)}),
$$

$$
\phi_s = (-I_{s,s_1}^{(0)} \lor -I_{s,s_2}^{(0)}) \land (-I_{s,s_1}^{(1)} \lor -I_{s,s_2}^{(1)}).
$$

The goal of the expression for ‘atoms’ is to ensure that it is true if each atom is mapped to one and at most one fact in each join path.

(c) Build the expression for ‘conjunction of atoms’:

$$
\phi_{(r,r_1),(s,s_2)} \land \phi_{(r,r_2),(s,s_1)},
$$

where

$$
\phi_{(r,r_1),(s,s_2)} = (-I_{r,r_1}^{(0)} \lor -I_{s,s_2}^{(0)}) \land (-I_{r,r_1}^{(1)} \lor -I_{s,s_2}^{(1)}),
$$

$$
\phi_{(r,r_2),(s,s_1)} = (-I_{r,r_2}^{(0)} \lor -I_{s,s_1}^{(0)}) \land (-I_{r,r_2}^{(1)} \lor -I_{s,s_1}^{(1)}).
$$

The goal of the expression for ‘join’ is to ensure that each join path is valid if it is true. One can verify the formula is true if and only if there exists a solution for instance $RES(d,q,1,2)$.

This reduction can be conducted in time

$$
O \left( (k_r + 1)^2 k \cdot n + (k_r + 1)k \cdot n^2 + k^2(k_r + 1) \cdot n^2 \right)
$$

Note that any assignment of the variables in given query $q$ should guarantee that $I_{i,t}^{(j)}$ is true if atom $p_i$ in $q$ is mapped to the fact $t$. Exactly $(k_r + 1)k$ variables and all the clauses are made true by the assignment. On the other hand, $\phi_i$ ensures that if an assignment makes all clauses true with $(k_r + 1)k$ true variables, then for each of the $k$ atoms, the assignment should contain exactly a set of variables $I_{i,t}^{(j)}$ which are made true. Clause $\phi_{(i,t),(i',t')}$ ensures that an assignment of the variables of $q$ does map each atom $p_i$ to a fact $t$. And $\phi_t$ guarantees the existence of at least $k + 1$ disjoint joining paths. Then, one can easily verify that the 2-CNF formula built in the reduction has a satisfying assignment with $k$ true variables if and only if RES problem has no solution for the constant $k_r$, i.e., $k_r$ assignments of the variables in $q$ can be found such that all the atoms are mapped to the facts from the database instance $d$, and these assignments are indeed disjoint with each other.
To show the lower bound is co-W[1]-hard, a reduction can be easily built from the clique problem which is W[1]-complete to the complement of RES \((d, q, k_r, k^2)\).

- Given clique problem instance \((G, k)\), build a database containing one binary relation \(G(\cdot, \cdot)\), i.e., \(G(x_i, x_j)\) if \((v_i, v_j) \in E\).
- Add one more distinct dummy fact \((x, x)\).
- Let \(k_r\) be 1.

Then a query with a bounded size depending on \(k\) can be defined as

\[
q := \bigwedge_{1 \leq i < j \leq k} G(x_i, x_j).
\]

In fact, the size of such query is bounded by \(O(k^2)\).

It is easy to check that the query result \(q(d)\) is initially true, and any set of arbitrary \(k_r\) facts makes the query still true if and only if \(G\) has a clique of size \(k\) because the dummy element \((x, x)\) should be deleted so that \(q\) is guaranteed to be false.

\textbf{Theorem 20.} RES\((d, q, k_r, k)\) is co-W[1]-complete if query \(q\) is a positive query and \(k\) is the size of query \(q\).

\textbf{Proof.} We first show the upper bound is co-W[1]. Any positive query \(q\) can be rewritten as a disjunction of some conjunctive queries. Recall that the number of conjunctive queries introduced into the transformed expression is at most exponential to the size of \(q\). Such rewriting thus can be conducted in a fixed parameter tractable time. However, it is non-trivial to build the new dummy element and the corresponding parameter \(k\), so that a detailed reduction is given here. Note that there is no solution of size \(k\) in \(d\) for query \(q\) if and only if for each conjunctive query \(q_i\) in the transformed disjunction of \(q\), no set of fact of size \(k\) whose removal causing failure of the query \(q_i\). Therefore, an FPT reduction can be built from each instance \((d, q_i, k_r, k)\) to the corresponding clique instance \((G_i, k_i)\), such that no solution of size \(k\) in \(d\) for query \(q_i\) if and only if \(G_i\) has a clique of size \(k_i\).

To clarify the correctness, all the different \(k_i\)'s should be uniformed. Let \(k_c\) be the parameter of clique problem, then we have \(k_c = \max_i \{k_i\}\). In each \(G_i\), create \(k_c - k_i\) new vertices adjacent to each other, and add enough edges such that all the other vertices in \(G_i\) are also adjacent to these new added vertices. It is easily to check that RES has a solution of size \(k\) for a positive query if and only if \(G\) (disjoint set of all the \(G_i\)'s) has a clique of size \(k_c\).
As for the lower bound, because conjunctive query is a special case of positive query, the case of conjunctive queries can be thus simply reduced to the case of positive queries, then it is immediately co-W[1]-hard.

However, resilience decision problem becomes much harder when the input query turns to a positive query.

**Theorem 21.** \( \text{RES}(d,q,k_r,k) \) is co-W[SAT]-hard if query \( q \) is a positive query and \( k \) is the number of variables in \( q \).

**Proof.** To show the lower bound is co-W[SAT]-hard, a reduction can be built from the \( k \)-Weighted SAT problem to RES. Formally, given a boolean formula \( F \) (without restriction on depth) and an integer \( k \geq 0 \), the \( k \)-weighted SAT problem is to ask whether there exists a satisfying assignment to \( F \) with a Hamming weight of exactly \( k \).

Dummy element together with the technique in [?] are used to construct this reduction.

- First construct a database consisting of three relations:

  \[ Pos(x,y), \ Neg(x,y), \ Dummy(x). \]

- Then the database instance \( d \) is obtained by filling \( Pos(X,Y) \) up with facts \((1,1),\ldots,(n,n)\), filling \( Neg(x,y) \) up with facts \((i,j)\) for each pair \(1 \leq i \neq j \leq n\), and filling \( Dummy(x) \) up with single fact \((a)\), for each variable \(x_1,\ldots,x_n\) appearing in the formula \( F \).

- The query \( q \) is constructed as follows. The first part

  \[ P := (\exists z_1,\ldots,z_k) \bigwedge_{1 \leq i \neq j \leq k} \text{Neg}(z_i,z_j). \]

  is first introduced as a simulation of the \( k \) Hamming-weighted solution. Then, for each literal \( l_i \) of variable \( x_i \) in the formula \( F \), we define a clause

  \[ \tau(l_i) := \begin{cases} \bigwedge_{1 \leq j \leq k} \text{Neg}(i,z_j) & l_i \text{ is negative literal;} \\ \bigvee_{1 \leq j \leq k} \text{Pos}(i,z_j) & l_i \text{ is positive literal.} \end{cases} \]
For simplicity, let $F_\tau$ be the clause transformed by $\tau$. Combine these clauses to make up the final query

$$q := (P \land F_\tau) \lor \text{Dummy}(a).$$

- At last, let $k' = 1$.

It is easy to check that (a) the number of variables is bounded by $k$, (b) the dummy elements enforce the boolean query $q$ to be true initially, and (c) unary fact $(a)$ must be deleted in order to guarantee $q$ is false, therefore, deletion of any $k'$ facts makes the query still true if and only if $F$ has no satisfying assignment with Hamming weight of exactly $k$.

**Theorem 22.** Let query $q$ be a first-order query, $\text{RES}(d, q, k_r, k)$ is co-W[P]-hard if $k$ is the number of variables in $q$, while co-W[t]-hard if $k$ is the size of $q$.

The proof is basically established by reductions from “weighted circuit satisfiability” and “depth-$t$ weighted circuit satisfiability” separately, which are the typical problems in W[P]-hard and W[t]-hard [?].

### 5.4.2 Triangle query and refined characteristic

Next, we show the upper bound of an important case of this problem is fixed-parametric tractable, and provide a kernelization algorithm. At first, a list of definitions should be introduced.

As argued in [64], dual hypergraph $H(q)$ of a query $q$ is defined as follows. Let $q := a_1, \ldots, a_m$ be a self-join-free conjunctive query, and $\text{var}(q)$ be the set of variables in $q$. Its dual hypergraph $H(q)$ has vertex set $V = \{a_1, \ldots, a_m\}$. Each variable $x_i \in \text{var}(q)$ determines the hyper-edge consisting of all those atoms in which $x_i$ occurs: $e_i = \{a_j | x_i \in \text{var}(a_j)\}$.

**Triad Query.** Based on the hypergraph of queries, an important class of conjunctive queries, called triad query, could be defined immediately as follows,

**Definition 5.4.1** (Triad query [64]). A triad is a set of three atoms in a query, $T = \{S_1, S_2, S_3\}$ such that for every pair of $i, j$, there is a path in its hypergraph $H(q)$ from $S_i$ to $S_j$ that uses no variable occurring in the other atom of $T$.

**Definition 5.4.2.** Triangle query $q_\triangle$ is a special conjunctive query of the form as follows,

$$q_\triangle := R(x, y), S(y, z), T(z, x).$$
For the RES decision problem, a dichotomy on the complexity is identified by Freire et al. \cite{64}, which shows that the problem is NP-complete if the input $q$ is a triad query, otherwise, it is polynomial tractable.

Intuitively, a triad is a triple of vertices with robust connectivity. Meanwhile, we observe that every triad case could be easily reduced from the triangle case $RES(q_\triangle)$ polynomially. Moreover, as shown by Freire et al. \cite{64}, triad makes the $RES$ problem hard.

\textbf{Definition 5.4.3 (Cell, Witness).} Given an $l$-ary fact $t = (u_1, \ldots, u_l) \in d$, for each $0 < i < l + 1$, value $u_i$ is a cell of fact $t$, and it could be denoted as $t[u_i]$. Given a query with $m$ atoms, say $q := a_1(x_1), \ldots, a_1(x_m)$, a witness is a set of facts $w = \{t_1, t_2, \ldots, t_m\}$, where for each $1 \leq i \leq m$, atom $a_i(x_i)$ can be mapped to fact $t_i$. Given a tuple set $w = \{t_1, t_2, \ldots, t_m\}$, if $q(w) = w'$, then $w$ is called the witness of query result $w'$.

Given a database instance $d$, let $q := a_1(x_1, x_2), \ldots, a_m(x_m, x_1)$ be a cyclic conjunctive query without self join, i.e., $a_i \neq a_j$ for each $i \neq j$. Let $w'$ be an arbitrary query result and let $w = \{t_1, t_2, \ldots, t_m\}$ be a witness of $w'$ such that $q(w) = w'$, for each $t \in w$. We also refer each cell $t[x_i]$ as a vertex, and refer all the cells of fact $t$ as a directed edge. Such a graph is called a witness graph. Actually, a witness graph can be regarded as a directed instantiation of the dual hypergraph of query $q$.

We now give the definition of data planarity, a property of database instances with respect to queries.

\textbf{Definition 5.4.4 (Data planarity).} Let $d$ be a database instance and $q$ be a query. $d$ is said to be of planarity with respect to $q$ if its corresponding witness graph is planar.

\textbf{Example 4.} An example database instance is shown as follows.

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th></th>
<th>y</th>
<th>z</th>
<th></th>
<th>z</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>r₁</td>
<td>1</td>
<td>2</td>
<td>s₁</td>
<td>2</td>
<td>3</td>
<td>t₁</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>r₂</td>
<td>1</td>
<td>4</td>
<td>s₂</td>
<td>4</td>
<td>3</td>
<td>t₃</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>r₃</td>
<td>2</td>
<td>5</td>
<td>s₃</td>
<td>5</td>
<td>3</td>
<td>t₄</td>
<td>6</td>
</tr>
<tr>
<td>T</td>
<td>r₄</td>
<td>5</td>
<td>2</td>
<td>s₄</td>
<td>2</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>r₅</td>
<td>2</td>
<td>4</td>
<td>s₅</td>
<td>4</td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In this database instance, suppose a triangle query and its result are shown as follows,

$q_\triangle := R(x, y), S(y, z), T(z, x)$

According to the definitions, for $i = 1, 3, 4$, $w = \{r_i, s_i, t_i\}$ is a witness of $z_i$, and $w = \{r_2, s_2, t_1\}$ is also a witness. Intuitively, its witness graph is shown in Fig. 5.2.

Obviously, the witness graph is planar. Therefore, we know that this database instance $d$ is of planarity with respect to triangle query $q_\triangle$. 
Refined characteristic. It has been already known that RES is NP-complete even if the input is self-join-free queries with triad, but it is polynomial tractable if without triad. This is because triangle query $q_\triangle$ can be reduced to these queries while preserving hardness.

One may suppose that if the input data is somehow simple with respect to the triangle query, it may be possible to solve RES in polynomial time. However, we will show that it is impossible at least when the data is as simple as be of planarity.

Theorem 23. Let $q_\triangle$ be an self-join-free conjunctive query and input database is of planarity, then RES($q_\triangle$) is still NP-complete.

Proof. To strengthen the hardness result for query with triad, construct a Karp-reduction from the NP-complete vertex cover problem to RES($q_\triangle$). Input an arbitrary graph $G = (V,E)$ with an natural number $k$, the output is expected as yes, if and only if there exists a vertex set $C \subseteq V$ of size $k$ such that each edge of $E$ is incident to at least one vertex of $C$. Vertex cover is still NP-complete even if the input graph is planar, and cubic, i.e., very vertex has a degree of three. Based on this, our reduction can also ensure that the database built is also of planarity.

Concretely, given an instance $(G,k)$ of vertex cover, where $G = (V,E)$ is a planar graph, we construct an instance $(d_G,k_G)$ of RES($q_\triangle$) as follows, for each $u$ of $V$, suppose its neighbors are $x,y,z$, say $N(u) = \{x,y,z\}$. Build a gadget of planarity $g(u)$ which can be shown as follows,
Figure 5.3. The first possible gadget for the instance of $g(u)$

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

Figure 5.4. The second possible gadget for the instance of $g(u)$

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 5.5. The third possible gadget for the instance of $g(u)$

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>
Figure 5.6. The fourth possible gadget for the instance of $g(u)$

To illustrate the intuition of the instance, we provide the visualization of $g(u)$ as follows,

Figure 5.7. Four possible visualizations of gadget $g(u)$

And we also should show the visualization of connecting two gadgets of individual vertex $u$ and $v$ as following,

Figure 5.8. Connecting the gadgets of two adjacent vertices

**Observation 1.** each directed edge is a tuple in database instance $d$, and each directed triangle is a tuple of $q_{\Delta}(d)$. 
The idea is similar to [?], the triangle is then attached to the vertex gadgets \( g(u) \) and \( g(v) \) as follows. For triangle \( \Delta_{uv} \) and vertex gadget \( g(u) \), add the edges \( v(u), u(v) \) by inserting a tuple \( v(u), u(v) \) into the next corresponding atom, where \((1,6)\) is a docking edge that has not been used before. Vertex gadget \( g(v) \) is attached analogously. By taking no account of direction consistency, since \( G \) is cubic, the three docking edges that each vertex gadget provides is sufficient and each docking edge will be used. Planarity is ensured since its underlying graph is planar (consider removing the direction), by using the docking edges of \( u \) according to the relative order of neighbors of \( u \) given by an embedding of \( G \) into database. Planarity of all gadgets are guaranteed, thus yielding the holistic planarity.

The intuition of this construction is the following. Each edge \((u,v)\) of the original graph \( G \) must have at least one of its endpoints in the vertex cover. Correspondingly, for each triangle \( \Delta_{uv} \) at least one edge must be deleted. Consider the graph \( \kappa(u) \) induced by the vertex set

\[
\{1,2,3,4,5,6,7\} \cup \{ux,u(x),uy,u(y),uz,u(z)\}
\]

the minimum number of edge deletions to make \( \kappa(u) \) triangle-free is six.

However, if one of the outer edges \((ux,u(x)), (uy,u(y)), (uz,u(z))\) is deleted, it is possible to delete the other two outer edges while only deleting seven edges. Note that this is the minimum number of edge deletions, equals to the size of minimum contingency set, to make \( \kappa(u) \) triangle-free, say query result is empty, under the constraint of “having to use one of the outer edges”. If we do so, all triangles in edge gadgets for edges incident to \( u \) are destroyed. Conversely, if there is a solution for the constructed instance of RES \( (q_{\Delta}) \), there always is an optimal solution for RES \( (q_{\Delta}) \) which does not contain the third edge \((u(y),y(u))\) and consequently activates \( \kappa(u) \) or \( \kappa(y) \), making the deletion of all the outer edges of one of these two graphs possible. There are at most \( k \) vertex gadgets corresponding to members of the vertex cover, then we set

\[
k_G := 7k + 6(|V| − k) = 6|V| + k
\]

then, it follows immediately that RES\((q_{\Delta})\) is NP-complete.

However, we can not ignore the consistency of direction of edges in each common triangle. We denote the docking edges \((1,2)\) and \((2,1)\) as A-edge, \((1,6)\) and \((6,1)\) as B-edge, \((7,6)\) and \((6,7)\) as C-edge one by one.

A local view of connecting two vertices can be shown as following,
Observation 2. inside every gadget $g$, the direction of B-edge is the same as that of A-edge, and for two adjacent gadgets $g$ and $g'$ connected by no C-edges, the directions of A-edge and B-edge of $g$ are different from that of $g'$.

This property results in a problem that the consistency of directed edges after connecting all these gadgets in a casual manner while keeping planarity.

Consider such an odd circle as following,

Obviously, according to the observation mentioned above, there is a conflict of the directions of A3-edge and B3-edge. It is infeasible to implement this in $\text{RES}(q_\Delta)$ if do not change the manner of connection. However, a good news is that, an observation is that the C-edge of each gadget could be used to tolerate any direction, so that each C-edge provide a chance to solve a conflict. We can connect these gadgets consistently by a careful arrangement of the neighbor of every C-edge. To do this, we distribute the C-edges to resolve all the direction conflict greedily. That is, for each vertex $u \in V$ and at least two adjacent edges $(u, x_1), (u, x_2)$ of $u$ are unlabeled, if there is an adjacent vertex, say $x_i$, whose C-edge is unused yet, label $(u, x_i)$ by C-edge connection, repeat until the edge labeled graph $G_d$ is obtained.

To reset the graph to a planar one, we could reset the layout of the gadgets corresponding to all vertices. The basic idea is based on an observation that, $\{4, 6, 7\}$ of every gadget $g$ can be set as either
clockwise or anticlockwise direction, therefore, the gadget of the vertex adjacent to \( \{4, 6, 7\} \)-induced part can be set up in any manner without resulting in a conflict.

The correctness of the reset follows since the input is a cubic graph, and note that any vertex has at least two neighbors have the same direction of A-edge. Here the claim accomplishes this proof.

\[\square\]

5.5 Kernelization for triangle queries

Then we provide a kernelization algorithm, and show that on the parametric aspect, the \( \text{RES}(q_\Delta) \) problem is fixed parameter tractable when the input data is of planarity, so that if the input constant \( k_r \) is not large, this problem could be solved efficiently.

We know that \( \text{RES}(q_\Delta) \) is still \( \text{NP-complete} \) even if \( d \) is of planarity with respect to \( q_\Delta \). But, fortunately, we claim it is fixed parameter tractable. We next provide a kernelization.

Our kernelization algorithm for any instance \( (d, k_r, q_\Delta) \) with the planarity property could produce a kernel consisting of 11\( k_r \) facts. This kernelization for instance of planarity is based on the idea for kernelizing triangle edge deletion [117]. We show the reductions in the following.

**Reduction 1.** Remove all the facts that are not included in any witness of any query result \( t \in q_\Delta(d) \).

For example, \( r_5 \) and \( s_5 \) would be removed, since they are not included in any witness of any of the results in the above table.

We apply this simple data reduction, and it follows immediately. Then the second reduction rule is very powerful.

**Reduction 2.** If the witness of some query result in \( q_\Delta(d) \) contains only one fact \( t \) which is contained in the witness of another different query result, delete \( t \) and reduce \( k_r \) by 1.

For example, witness \( w = \{r_2(1,4), s_2(4,3), t_2(3,1)\} \) contains only one fact \( t_1(3,1) \) which belongs to another witness \( w' = \{r_2(1,2), s_2(2,3), t_2(3,1)\} \) of result \( (1,2,3) \). Therefore, remove fact \( r_2(1,3) \) and reduce \( k_r \) by 1. Note that due to the different directions, witness \( w = \{r_4(5,2), s_4(2,6), t_4(6,5)\} \) contains only one fact \( r_4(5,2) \) which does not belong to another witness \( w' = \{r_3(2,5), s_3(5,3), t_3(3,2)\} \) of result \( (2,5,3) \), and so does \( r_3(2,5) \).

**Reduction 3.** If the witness of some query result does not contain a fact which is in the witness of another query result, then delete an arbitrary fact of this witness and reduce \( k_r \) by 1.

For example, witness \( w = \{r_4(5,2), s_4(2,6), t_4(6,5)\} \) contains no fact which belongs to any other witness. Therefore, remove all the facts of witness \( w \) and reduce \( k_r \) by 1. Similarly, witness \( w' = \)
\{r_3(2,5), s_3(5,3), t_3(3,2)\} contains no fact belonging to another witness. Therefore, all the facts of witness \(w'\) can be removed and reduce \(k_r\) by 1.

The correctness of this reduction can be easily verified. One fact of the three facts \(r, s, t\) in the witness of any query result has to be removed. Then this reduction always chooses a fact \(r\), which covers all the witnesses covered by \(s\) or \(t\).

Let any two facts \((u,x),(u,y)\) \(\in d\) be a dock of cell \(v\) if and only if \(\{(u,v,x),(u,v,y)\} \subseteq q_\Delta(d)\) or \(\{(x,u,v),(y,u,v)\} \subseteq q_\Delta(d)\) or \(\{(v,x,u),(v,y,u)\} \subseteq q_\Delta(d)\). The cell contained in both edges of a dock is called the dock cell of \(v\).

For example, two facts \((1,4),(1,2)\) is a dock of cell \(3\), because we have two results, i.e., \(\{(1,4,3),(1,2,3)\} \subseteq q_\Delta(d)\). Intuitively, in general, the docking edges \((u,x),(u,y)\) and cell \(v\) induced subgraph is a claw which can be shown as follows,

![Claw Diagram]

We have the following property identifying a structure which holds in the database instance of planarity: let \(d\) be a database instance and \(S\) be the solution of \(\text{RES}(q_\Delta)\). Let cell set \(M\) include all the cells included in \(S\), and for each cell \(c \in M\), let \(|S[u]|\) be the number of facts in \(S\) including \(u\). Then, let \(|S[u]| \geq 2\). If \(d\) is of planarity, then cell set \(C\) excludes \(M\), say \(C \setminus M\) should contain fewer than \(2|S[u]| - 1\) cells with a dock \(B \subseteq S\) with \(u\) as the dock cell.

If draw any cell as a vertex, and any tuple as an edge, then the worst case of Reduction 3 can be visualized as the following figure.

![Reduction 3 Visualized]

Figure 5.11. At most 4 vertices could be added
For simplicity, we first suppose each fact $t$ only appears once in the database. Thus, there are at most $2|S[u]| - 1$ neighbors of $u$, and they are contained in $C \setminus M$ and have a dock $B \subseteq S$ with $u$ as the base vertex. As already shown, every vertex $c \in C \setminus M$ has a dock $B \subseteq S$, and this implies that the corresponding dock cell is in $M$. It follows that

$$|C \setminus M| \leq 2|S[u]| - 2 \leq 4|S| - 2|M|.$$  

We use the fact that the sum over all $|S[u]|$ counts every edge in $S$ exactly twice. Cell set $C$ is partitioned into $M$ and $C \setminus M$, hence

$$|C| \leq |M| + |C \setminus M| \leq 4|S| - |M|.$$  

Due to the planarity of a database instance, we have the Euler’s formula implying that

$$|S| \leq 3|M| - 6.$$  

Then we have

$$|M| \geq |S| + 2.$$  

We can obtain the upper bound

$$|C| \leq 4|S| - |S| - 2 \leq 11|S| - 2 \leq 11k_r.$$  

Therefore, for the general case, we know that each fact $t$ appears at most three times in the database, and distributes in the three tables. Therefore, the final results of the kernel size should be

$$|d'| \leq (3|C|)^2 \leq 121k_r^2.$$  

and the following theorem follows.

**Theorem 24.** For a triangle query, the problem $\text{RES}(q_\Delta)$ admits a problem kernel with the number of facts less than $121k_r^2$, which can be computed in $O(k_r n \sqrt{n})$. 
5.6 Conclusion

In this paper, we examine the complexity of the resilience decision problem under the traditional and parameterized perspectives. We find that the decision problem is NP-complete for a triangle query but fixed parameter tractable when the data has the planarity property. Our result implies that if the solution of RES is bounded by \( k \), then the data could be pre-processed (kernelized) as a much simpler case which has only \( O(k^2) \) facts.
Chapter 6

CONCLUSION

This dissertation studies the key problem of computational complexity and algorithm for data evaluation and repairing. This dissertation first studies the computational complexity of and algorithms for the database inconsistency evaluation. We define and use the minimum tuple deletion to evaluate the database inconsistency. For such minimum tuple deletion problem, we study the relationship between the size of rule set and its computational complexity. We show that the minimum tuple deletion problem is still NP-complete, if given two conditional functional dependencies and three attributes involved in them; And it is NP-hard to approximate the minimum tuple deletion problem within $\frac{17}{16}$ if given three conditional functional dependencies and four attributes involved in them. We design a near optimal approximated algorithm for computing the minimum tuple deletion, the ratio is $2 - \frac{1}{2^r}$, where $r$ is the number of conditional functional dependencies in the given rule set $\Sigma$. Under the unique gaming conjecture, this ratio is near optimal, its hard to improve it with a constant independent of $n$.

To guide the data repairing, this dissertation also investigates the data repairing method by using query feedbacks, formally studies two decision problems, functional dependency restricted deletion and insertion propagation problem, corresponding to the feedbacks of deletion and insertion. A comprehensive analysis on both combined and data complexity of the cases is provided by considering different relational operators and feedback types. We have identified the intractable and tractable cases to picture the complexity hierarchy of these problems, and provided the efficient algorithm on these tractable cases.

We also examine the complexity of the resilience decision problem by means of parameterized complexity. We find that a triangle query is fixed parameter tractable if the data is of planarity property with the triangle query. Our result implies that if the solution of RES is bounded by $k$, then the data could be pre-processed (kernelized) as a much simpler graph which has only $O(k^2)$ facts.

From the results, we know that when data volume becomes big, automatic propagation analysis is solvable in polynomial time. Unfortunately, when a conjunctive query becomes complex simultaneously, it is intractable even for the bounded case. Our work also makes contributions to the previous results for the problems of side-effect free deletion propagation, and could be applied in the update forbidden case of view propagation.
REFERENCES

2016, no. 1, p. 18, 2016.

sensor networks,” in The 34th Annual IEEE International Conference on Computer Communications
(INFOCOM 2015), April 2015, pp. 531–539.

in sensor networks,” in The 10th International Conference on Wireless Algorithms, Systems, and


1–17. [Online]. Available: http://dx.doi.org/10.1007/11965893_1

seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems,
http://doi.acm.org/10.1145/1376916.1376940


http://dx.doi.org/10.1002/int.v22:7


