Approximated Summarization of Data Provenance

Thesis submitted in partial fulfillment of graduate requirements for the degree “Master of Sciences” in Tel Aviv University School of Computer Science

By

Eleanor Ainy

Prepared under the supervision of Prof. Tova Milo and Dr. Daniel Deutch

July 2015
Abstract

Many modern applications involve collecting large amounts of data from multiple sources, and then aggregating and manipulating it in intricate ways. The complexity of such applications, combined with the size of the collected data, makes it difficult to understand how the resulting information was derived. *Data provenance* has proven helpful in this respect, however, maintaining and presenting the full and exact provenance information may be infeasible due to its size and complexity. We therefore introduce the notion of *approximated summarized provenance*, which provides a compact representation of the provenance at the possible cost of information loss. Based on this notion, we present a novel provenance summarization algorithm which, based on the semantics of the underlying data and the intended use of provenance, outputs a summary of the input provenance. Experiments measure the conciseness and accuracy of the resulting provenance summaries, and improvement in provenance usage time. In addition to the experiments that we have conducted, we have also developed a system called PROX. This system outputs a summarized representation of selected provenance, using the summarization algorithm, and further allows for approximate provisioning.
# Table of Contents

Acknowledgments iii

List of Figures 1

1: Introduction 3
   1.1 Contributions ........................................... 4
   1.2 Thesis organization .................................... 5

2: Model Description 6
   2.1 Workflows ................................................. 6
   2.2 Provenance Model ......................................... 7
   2.3 Valuations and Provisioning .............................. 9

3: Provenance Summarization 11
   3.1 Summarization Through Mappings ........................ 11
   3.2 Quantifying Summary Quality ............................ 12

4: Computing Provenance Summarizations 17
   4.1 Computing Summary Quality ............................... 17
   4.2 Finding a Summarization .................................. 18

5: Datasets and Use Cases 26
   5.1 Datasets ................................................... 26
   5.2 Use Cases .................................................. 27
6: Experiments

6.1 Algorithms Examined
6.2 Clustering Algorithm
6.3 Experimental Settings
6.4 wDist Experiment
6.5 TARGET-SIZE Experiment
6.6 TARGET-DIST Experiment
6.7 Varying Number of Algorithm Steps Experiment
6.8 Usage Time Experiment
6.9 Summarization and Candidate Computation Time Experiment
6.10 Other Datasets

7: The PROX System

7.1 System Architecture
7.2 PROX Web UI

8: Related Work

9: Conclusions

References
Acknowledgments

First, I would like to thank Prof. Tova Milo and Dr. Daniel Deutch, my supervisors, for their help, their endless support and knowledge. They have been extremely patient throughout this process and their encouragement as well as professional advice made it possible for me to complete this thesis and to learn a lot in the process.

Next, I would like to thank Susan B. Davidson from the Department of Computer and Information Science in the University of Pennsylvania and Pierre Bourhis from Lille University for their help with the research and their great input.

Finally, I would like to thank my family and boyfriend Or who believed in me, encouraged me and supported me during my studies.
List of Figures

2.1 Example Workflow ............................................. 6

6.1 Average Distance as a Function of wDist and TARGET-SIZE (MovieLens Dataset) ................................................. 36

6.2 Average Size as a Function of wDist and TARGET-DIST (MovieLens Dataset) ....................................................... 36

6.3 Average Size and Distance as a Function of wDist for Varying Number of Algorithm Steps (MovieLens Dataset) ................. 38

6.4 Usage Time (MovieLens Dataset) ....................................... 39

6.5 Candidate Computation and Summarization Times (MovieLens Dataset) 40

6.6 Average Distance as a Function of wDist and TARGET-SIZE (Wikipedia Dataset) .................................................... 41

6.7 Average Size as a Function of wDist and TARGET-DIST (Wikipedia Dataset) .......................................................... 41

6.8 Average Distance as a Function of wDist and TARGET-SIZE (DDP Dataset) ........................................................... 42

6.9 Average Size as a Function of wDist and TARGET-DIST (DDP Dataset) 42

7.1 System Architecture .................................................. 43

7.2 PROX Web UI - Selection View (Movies) ............................... 45

7.3 PROX Web UI - Selection View (Genres and Year) ................. 46

7.4 PROX Web UI - Summarization View ..................................... 47

7.5 PROX Web UI - Summary View (Groups) .............................. 48
7.6 PROX Web UI - Summary View (Group User Attributes) . . . . . . . 49
7.7 PROX Web UI - Summary View (Group Movie Attributes) . . . . . . 50
7.8 PROX Web UI - Summary View (Expression) . . . . . . . . . . . . . 51
7.9 PROX Web UI - Summary View - Evaluate Assignment (Annotations) 52
7.10 PROX Web UI - Summary View - Evaluate Assignment (Attributes) 53
I Introduction

Complex applications that collect, store and aggregate large-scale data, and interact with a large number of users, are found in a wide variety of domains. Notable examples are crowd-sourcing applications such as Wikipedia, social tagging systems for images, traffic information aggregators such as Waze, or hotel and movie ratings such as TripAdvisor and IMDb.

In the context of such applications, several questions arise related to how data was derived. As a user of the information, what is the basis for trusting it? How do contributions vary among crowd members based on characteristics such as age or gender? If some contribution seems wrong, how does the information change if we discard it? These questions are fundamentally important to better understand the application and its results.

At its core, the answer to these questions is based on the provenance of the collected data and resulting information, that is, who provided the data in what context and how the information was derived. However, provenance goes well beyond simply providing a log of the application execution. In particular, the algebraic model of provenance based on semirings of [21, 6] can be used to explain results by correlating input with output data, and tracking important details of the computational process that took place. As shown in [17], it can also be used to provision the result with respect to hypothetical scenarios, i.e. to observe changes to the result based on changes to the input without re-running the process. Detailed tracking of provenance is therefore an essential vehicle for the applications mentioned above.

As an example, consider a crowd-sourced application for movie reviews, where the number of movies, and number of reviews for each movie, may be very large. An
aggregated score for each movie is computed by combining the scores of multiple different users, possibly accounting for their previous reviews and for their preferences. These features, and the way in which they are used in the computation, should all be reflected in the provenance. In turn, this provenance may be presented to explain results such as the computed recommendations of movies, or to provision them, e.g. to determine how the average movie rating would change if we ignore ratings by some group of users.

The large amount of data and complexity of processing the data means that the resulting detailed provenance information can be overwhelming. Presenting it in full, as an explanation for a computation, may make it extremely hard for users to understand. In this thesis we therefore introduce the notion of approximated summarized provenance, which provides a compact representation of provenance at the possible cost of information loss. This compact representation will enable the user to see trends, for example that women aged 20-25 have tended to rate a particular movie more highly than men aged 20-25.

We next list the contributions of the thesis.

### 1.1 Contributions

- We start with a description of workflow provenance and the semiring provenance model. The described algebraic structures are the basis of our work. We also discuss the notion of truth valuations and provisioning.

- We present a novel algorithm that provides approximated summarization of provenance information for complex applications. The summarization is based in part on the semantics of the underlying data (such as gender, age or occupation of users), where annotations of “similar” data items are intuitively more amenable to be grouped together. More importantly, it is also geared towards the intended use of provenance (namely explanation and/or provisioning): we define a distance function between provenance expressions that is based on the
intended use, and optimizing this distance while still obtaining small expressions guides the summarization. We show that the problem of computing the exact distance is \#P-hard and present a sampling algorithm that approximates the distance of the summary from the original provenance expression and is used as a building block of the general summarization algorithm.

- We have developed PROX, a system for the management, presentation and use of data provenance for complex applications. Using PROX, participants can select provenance to summarize, view the summarization result and use it to gain insights on the application and its underlying data.

- We have conducted experiments with three datasets - MovieLens, Wikipedia and DDP (Data Dependent Process), in which we compared our algorithm to other approaches and showed that our approach gives better summarizations in terms of distance and size.

Our short paper was submitted and accepted to TAPP. Our full paper was also submitted.

1.2 Thesis organization

The rest of the thesis is organized as follows. Chapter 2 describes workflow provenance and the provenance model. Chapter 3 describes the notion of provenance summarization through mappings and the quality measurements for such summarization. In Chapter 4 we present a few propositions that, combined together, lead to our summarization algorithm. We end this chapter with an example of the full algorithm flow. Chapter 5 describes the datasets we used and also includes interesting use cases. We later describe our experimental results in Chapter 6. Our PROX system is discussed in Chapter 7. Related work is discussed in Chapter 8. Finally, conclusions are discussed in Chapter 9.
II Model Description

We now give an overview of the semiring provenance model of [21], and its extension to queries with aggregates in [7, 6]. This will serve as the basis for our work.

2.1 Workflows

We capture applications logic by a standard notion of workflows. One possible model for workflow [15, 6] consists of a specification and an associated set of executions. The specification can be thought of as an FSM (Finite State Machine), in which modules represent processing steps and edges indicate potential dataflow between the output port of one module to the input port of another module. In the model of [6], the workflow operates in the context of some global persistent state, i.e. some underlying database. Modules may be atomic, meaning that they are a query on the inputs to the module as well as the underlying database. Modules can also update the underlying database. A workflow execution (or “run”) is a repeated application of modules, which are ordered according to the workflow specification. ¹

![Figure 2.1: Example Workflow](image)

¹ This departs from the representation of executions and their provenance as multigraphs in [15]
Example 2.1.1. Consider a movie rating application, in which users rate movies and the ratings are aggregated using the application logic described by the workflow in Figure 2.1.

Certain information about users is known, such as gender and type (movie critic, director, audience, etc.), and stored in the Users table in the underlying database.

Reviews are collected by different reviewing modules, which crawl different reviewing platforms such as IMDb and newspaper web-sites. Each such module updates statistics in the Stats table in the underlying database, e.g. how many reviews the user has submitted (NumRate), what their max score is (MaxRate), etc. (alternatively, we could use sum or any other aggregation function). A reviewing module also consults Stats to output a sanitized review by implementing some logic. The sanitized reviews are then fed to an aggregator, which computes an aggregate movies scores.

There are many plausible logics for the reviewing modules; we exemplify one in which each module sanitizes the reviews by joining the users and statistics relations (depending on the module), keeping only reviews of users listed under the corresponding role (audience/critic) and who are “active”, i.e. who have submitted more than 2 reviews. The aggregator combines the reviews obtained from all modules to compute overall movie ratings (num, max).

2.2 Provenance Model

We next explain in general what the provenance model is and then use examples to illustrate the concepts described. We start by fixing a finite set $Ann$ of provenance annotations, corresponding to the basic units of data manipulated by the application, and which can be thought of as abstract variables identifying the data. Depending on the application, these annotations may correspond to different tuples in a database, to different users, to different questions, etc.

A correspondence between data manipulation and algebraic operations in the structure of a commutative semiring can then be defined. A commutative semiring is
a structure \((K, +_K, \cdot_K, 0_K, 1_K)\) where \((K, +_K, 0_K)\) and \((K, \cdot_K, 1_K)\) are commutative monoids. This means that the operations are associative and commutative with 0 and 1 standing for the neutral elements for addition and multiplication, respectively.

In addition, \(\cdot_K\) is distributive over \(+_K\), and \(a \cdot_K 0_K = 0_K a = 0_K\).

Given our set \(Ann\) of basic provenance annotations, the provenance semiring is the semiring of polynomials with natural coefficients, with indeterminates from the set \(Ann\). It is denoted \((N[Ann], +, \cdot, 0, 1)\), and was shown in [21] to capture provenance for positive relational queries:

- the \(+\) operation corresponds to the alternative use of data (as in union and projection)
- the \(\cdot\) operation corresponds to the joint use of data (as in join)
- 1 annotates data that is present
- 0 annotates data that is absent.

To capture aggregate queries, \(K\)-relations were further generalized by extending their data domain with aggregated values [7]. In this extended framework, relations have provenance also as part of their values, rather than just in the tuple annotations. Such a value is a formal sum \(\sum_i t_i \otimes v_i\), where \(v_i\) is the value of the aggregated attribute in the \(i^{th}\) tuple, while \(t_i\) is the provenance of that tuple. We can think of \(\otimes\) as an operation that pairs values (from a monoid \(M\)) with provenance annotations. Each such pair is called a tensor. The formal sum, presented by the \(\oplus\) operation is used to capture the aggregation function.

In [7, 17] the framework was also used to define provenance for nested aggregates and negation by introducing equation and inequality elements. Intuitively an equation such as \([(d_1 \cdot d_2) \otimes m > 3]\) is kept as an abstract token and can be used in conjunction with other semiring elements. Given concrete values for \(d_1, d_2\) and \(m\) one may test
the truth value of the equality and replace the equation by the truth value \(^2\). A precise algebraic treatment of aggregated values and the equivalence laws that govern them is based on semimodules and is described in [7]. We will focus, for simplicity, on the case where the values monoid \(M\) is that of real numbers with numbers addition and 0.

**Example 2.2.1.** The basic provenance annotation set \(Ann\) consists here of \(U_1, \ldots, S_1, \ldots\). The provenance-aware value stored as \(\text{MaxRate}\) in the aggregator’s output table, the \(\text{Movies}\) table, for the “MatchPoint” tuple would be:

\[
P = U_1 \cdot [S_1 \cdot U_1 \otimes 5 > 2] \otimes (3, 1) \oplus \\
U_2 \cdot [S_2 \cdot U_2 \otimes 3 > 2] \otimes (5, 1) \oplus \\
U_3 \cdot [S_3 \cdot U_3 \otimes 13 > 2] \otimes (3, 1) \oplus \ldots
\]

where \(U_i\) is a user identifier, \(S_i\) is the provenance of the user’s Stats tuple, and as aggregation we use a monoid of pairs to capture the aggregated rating (MAX rating with value 3 in the first tensor) and how many users contributed to this aggregated value (1 per tensor here but we will next show examples with other values). Intuitively, each numeric rating is associated with the provenance of the tuple obtained as the output of the reviewing module, namely the \(U_i\) annotation identifying the user. Each such sub-expression is multiplied by an inequality term serving as a conditional guard, indicating that the number of reviews recorded for the user is above the threshold of 2. Applying aggregation then results in coupling values (numeric reviews) with annotations to form the expression above.

### 2.3 Valuations and Provisioning

An important use of semiring provenance is for **provisioning**, i.e. examining changes to the application’s execution that are the result of some hypothetical modifications to the data (e.g. “How would the movie ratings change if we ignore some reviews sus-

\(^2\) The obtained semiring is denoted by \(K^M\) in [7]. For simplicity we will abuse notation here and just use \(K\)
pected as spam?”). This is formalized in [21] through the notion of truth valuations applied to annotations. Intuitively, specifying that $U_1$ is a spammer corresponds to mapping it to false (and that $U_1$ is reliable to mapping it to true), and recomputing the derived value w.r.t this valuation. Such valuation can again be extended in the standard way to a valuation $V : N[Ann] \rightarrow \{true, false\}$ using the following intuitive rules: (1) $0 \otimes m$ is interpreted as 0; (2) $1 \otimes m$ is interpreted as $m$; and (3) A comparison expression is interpreted as 1 if satisfied and as 0 otherwise.

Note that given a truth valuation for annotations, we obtain a real number for the expression by simply performing the substitution as defined above, and applying the basic semiring axioms $^3$.

**Example 2.3.1.** Consider the provenance expression $P$ of Example 2.2.1 and partial truth valuation that maps $S_1$ to 0 and $U_1$ to 1. Then $U_1 \cdot [S_1 \cdot U_1 \otimes 5 > 2] \otimes (3, 1)$ maps to $0 \otimes (3, 1) \equiv 0$: Although $U_1$ is mapped to 1, $S_1 \cdot U_1 \otimes 5$ is mapped to 0 and so the inequality does not hold, and the inequality expression is mapped to 0. In contrast, if $S_1$ is mapped to 1 then the condition would hold and we would have $(1 \cdot 1) \otimes (3, 1) \equiv 3$ (notice that $(3, 1) \equiv 3$ since we apply aggregation on a single user with score 3). Intuitively the second case corresponds to keeping the review, while the first one corresponds to discarding it.

$^3$Similarly, the provenance can capture scores of multiple movies and valuation then leads to a vector of values.
III Provenance Summarization

The provenance model described in the previous chapter provides full documentation of the transformations that took place. Since the resulting expression may be extremely long and complex, we would like to summarize the provenance expression, at the possible cost of information loss. We start by formalizing summarization through a notion of *mappings*, and then discuss how to quantify the quality of a summary.

3.1 Summarization Through Mappings

Let \( \text{Ann} \) be a domain of annotations (for the \( N[\text{Ann}] \) semiring) and \( \text{Ann'} \) be a domain of annotation summaries. Typically, we expect that \(| \text{Ann'} | < | \text{Ann} |\). We then define a mapping \( h : \text{Ann} \mapsto \text{Ann'} \) which maps each annotation to a corresponding summary. Abusing notation, this extends naturally to a homomorphism \( h : N[\text{Ann}] \mapsto N[\text{Ann}]' \), i.e. define \( h(a + b) = h(a) + h(b) \) and \( h(a \cdot b) = h(a) \cdot h(b) \).

This further extends to \( N[\text{Ann}'] \otimes M \) by the standard construction \( h(k \otimes m) = h(k) \otimes m \). Essentially, to apply \( h \) to a provenance expression \( p \), each occurrence of \( a \in \text{Ann} \) in \( p \) is replaced by \( h(a) \). The mapped expression, \( h(p) \), is a summary of the real provenance, in the sense that we lose track of some exact annotations and summarize the provenance using the abstract annotations in \( \text{Ann}' \).

**Example 3.1.1.** Consider the provenance-aware expression \( P \) obtained in Example 2.2.1. To simplify the example we focus on the reviews of users \( U_1, U_2, U_3 \) for the movie “Match Point”, and map all \( S_i \) annotations to 1 so we can discard the inequality terms. We thus obtain a simplified version of the provenance expression \( P \):

\[
\begin{align*}
U_1 & \cdot [1 \otimes 5 > 2] \otimes (3, 1) \oplus \\
U_2 & \cdot [1 \otimes 3 > 2] \otimes (5, 1) \oplus \\
U_3 & \cdot [1 \otimes 13 > 2] \otimes (3, 1)
\end{align*}
\]
which further simplifies to:
\[ P_s = U_1 \otimes (3, 1) \oplus U_2 \otimes (5, 1) \oplus U_3 \otimes (3, 1) \]

Next, we map user annotations to annotation summaries that intuitively reflect values of attributes of the corresponding users. Mapping \( U_1 \) and \( U_2 \) to an annotation summary called “Female” \(^1\), and applying congruences in the tensor structure, we obtain an expression that includes a maximum score of 5 collected from two female users:
\[ P'_s = Female \otimes (5, 2) \oplus U_3 \otimes (3, 1) \]

Another summary results from mapping annotations \( U_1 \) and \( U_3 \) to the annotation “Audience”:
\[ P''_s = Audience \otimes (3, 2) \oplus U_2 \otimes (5, 1) \]

In the example, we used two possible mappings \( h \) that combine reviews based on gender or role. In general there may be many possible mappings and the challenge is, given a provenance expression \( p \), to (a) define what a good mapping \( h \) is (correspondingly, what is a good summary \( h(p) \)), and (b) find such good \( h \).

### 3.2 Quantifying Summary Quality

Several, possibly competing, considerations need to be combined in quantifying the quality of a summary.

**Provenance size.** Since the goal of summarization is to reduce the provenance size (measured as the number of annotations), it is natural to use the size of the obtained expression (after simplifications) as a measure of its quality.

**Semantic Constraints.** The provenance expression obtained may be of little use if it is constructed by identifying multiple unrelated annotations. It is thus natural to impose constraints on which annotations may be grouped together. One simple example of such a constraint is to allow two annotations \( a, b \in Ann \) to be mapped to the same annotation in \( Ann' \) (or to 0 or 1) only if they annotate tuples in the

\(^1\) We later describe which mappings are possible and which are preferable to ours.
same input table, intuitively meaning that they belong to the same domain. Other constraints may be specified in the form of taxonomies, where available. Taxonomies give semantic relations between the underlying objects (users, movie, etc.), and are used to constrain homomorphisms by requiring that all annotations that are grouped together by mapping to the same annotation share a common ancestor.

Additional constraints involve restricting mappings based on the original input data, by requiring that annotations that are mapped together reference tuples that share values in some (or one of some) specified attributes. For example, we may specify that users that are grouped together must share a common attribute out of gender, age group, etc. This allows us to give a meaningful name to the new annotation for presentation purposes, based on the joint attribute.

Taxonomic information may also be useful for deciding between choices of mappings, and may be incorporated as part of the computation. For example, we may take into account the taxonomic distance between annotations and the annotation they are mapped to by using the MAX or SUM of these distances, and prefer mappings of annotations to a new annotation that is relatively close to them (e.g. mapping user annotations to annotation 'Guitarist' is preferable to mapping them to annotation 'Person').

**Distance.** Depending on the intended use of provenance, we may quantify the distance between the original and summary expressions. For that we again use the notion of valuations, and define distances with respect to a set $\mathcal{V}_{Ann}$ of valuations to the original annotations Ann.

**Example 3.2.1.** Consider a distance function designed to use provenance for provisioning in the presence of spammers. To simplify the example, we assume that there is a single spammer. In this case, the class of valuations considered consists of those assigning 0 to a single user annotation, and 1 to all others. A concrete aggregated value for each movie may then be computed by simply canceling every summand in which the mapped annotation is 0, and taking the aggregate values for the rest. (We use here the congruences $0 \otimes m \equiv 0$, $1 \otimes m \equiv m$, and the ability to embed the result
A central issue is how we transform a valuation in $V_{Ann}$, on the original annotations to one in $V_{Ann'}$, on the new annotation summaries. We propose that this will be given by a combiner function $\phi$ that sets a boolean value to $a' \in Ann'$ based on the truth values assigned to $a$ annotations that were mapped to it. For example, if $\phi$ is a disjunction of the truth values, then intuitively an annotation summary is cancelled only if all of the annotations it summarizes are cancelled. More formally, let $Ann, Ann'$ be two domains of annotations and let $h : Ann \mapsto Ann'$. Further let $\phi : Ann' \mapsto N[Ann]$ be a function such that for every $a' \in Ann'$, $\phi(a')$ is a polynomial only in elements of $\{a \in Ann \mid h(a) = a'\}$. In a sense, $\phi$ complements $h$, by specifying how the elements that are mapped to an annotation $a'$ should be combined. Now, any valuation $v_{Ann} \in V_{Ann}$ uniquely extends to a valuation $v_{Ann'} \in V_{Ann'}$ by defining $v_{Ann'}(a') = v_{Ann}(\phi(a'))$; note that the use of $v_{Ann}$ refers to its extension to the domain of $N[Ann]$. We use $v^{h,\phi}$ to denote the valuation obtained in such a way from a valuation $v$ and mappings $h, \phi$.

We next define the distance between a provenance expression $p$ and its summary $h(p)$ as an average over all truth valuations, of some property of $p$, $h(p)$, and the valuation. This property is based on yet another function we call $VAL\text{-FUNC}$, whose choice depends on the intended provenance use. We only require that it is a computable function fed as an additional input to the algorithm.

**Definition 3.2.2.** Let $p$ be a provenance expression over a set of annotations $Ann$, and let $p' = h(p)$, we define:

$$dist^{h,\phi}(p, p') = \frac{\sum_{v \in V_{Ann}} VAL\text{-FUNC}(v, v^{h,\phi}, p, p')}{|V_{Ann}|}$$

where $VAL\text{-FUNC}$ is some function measuring a property of the effect of the valuation over the two polynomials.
**VAL–FUNC functions**  We next give examples for natural choices of VAL–FUNC(v, v', p, p').

In all examples w(v) is some weighting over the valuation, e.g. the joint probability of the truth values it defines.

1. **Expected error**: \( w(v) \cdot |v(p) - v'(p')| \). Using the special case \( w(v) = 1 \) leads to comparing the overall error over all possible truth valuations out of the given set. This scenario makes sense when provenance is likely to be performed for multiple valuations, and it is all right to suffer some small error in each computation.

2. **Weighted fraction of disagreeing valuations**: 0 if \( v(p) = v'(p') \) and \( w(v) \) otherwise. Using the special case \( w(v) = 1 \) would be a reasonable choice if the user is to uniformly sample valuations and is interested in the probability of obtaining a correct/incorrect answer.

3. **Euclidean distance**: \( \text{euclidean-dist}(v(p), v(p')) \). This is well-defined when \( v(p) \) and \( v(p') \) are aggregation vectors rather than aggregation values (e.g. a vector of aggregated ratings of different movies of same genre).

**Example 3.2.3.** Observe that using \( |v(p) - v'(p')| \) as the VAL–FUNC, \( P''_s \) is at distance 0 from \( P_s \) w.r.t. valuations that map only a single user annotation to False. All these valuations yield the same value w.r.t. the two provenance expressions – if \( U_2 \) is mapped to True then the aggregated MAX value is 5 regardless of other truth values, and otherwise both \( U_1 \) and \( U_3 \) are mapped to True and so is Audience. In contrast, \( P'_s \) differs from \( P_s \) for the valuation that maps \( U_2 \) to False and the rest to True.

**Putting it all together**  In addition to obtaining a provenance summary with small distance, we of course wish to minimize the provenance expression size. The distance and size measurements are combined together to form a weighted average, where the weights are given as input parameters, that is used as a score given to
candidate mappings. We later describe how this score is used in our summarization algorithm and also how taxonomy distances are incorporated in the computation.

**Definition 3.2.4.** Let $p_0$ be an input provenance expression and let $p_{\text{cand}} = h(p_0)$. Also, let $w_{\text{Dist}}$ and $w_{\text{Size}}$ be user-defined weights ($w_{\text{Dist}} + w_{\text{Size}} = 1$), $r_{\text{Dist}}$ the approximated distance rank of $p_{\text{cand}}$ and $r_{\text{Size}}$ its size rank. We define a candidate mapping score as follows:

$$\text{CandidateScore}^{h,\phi}(p_0, p_{\text{cand}}) = w_{\text{Dist}} \times r_{\text{Dist}}^{h,\phi}(p_0, p_{\text{cand}}) + w_{\text{Size}} \times r_{\text{Size}}(p_{\text{cand}})$$

**Computational problems** Given a provenance expression $p_0$ with a set of annotations $Ann$, $\phi$ and VAL–FUNC functions, our goal is to explore the tradeoff between distance and size. This is studied in three flavors:

1. using input weights for size and distance, for obtaining a homomorphic expression $p'$, in each algorithm iteration, that minimizes the function CandidateScore.

2. minimizing the distance while obtaining a final homomorphic expression $p'$ of size less than some size bound TARGET–SIZE.

3. minimizing the size while obtaining a final homomorphic expression $p'$ of distance less than some distance bound TARGET–DIST. These three flavors are all studied using the summarization algorithm. To use the first flavor the user can choose weights for distance and size, TARGET–SIZE and TARGET–DIST according to her preferences. To use the second flavor, the user must set the distance weight to 1 (wDist=1) and TARGET–DIST to the maximal distance which is 1 (to cancel its effect). For the third flavor, the user must set the size weight to 1 and TARGET–SIZE to the minimal size which is 1. We study both a variant where the distance is computed with respect to all possible valuations (and then $V_{Ann}$ is not an explicitly given input) as well as a variant where a subset $V_{Ann}$ of valuations is given as input.
There are two main building blocks in a solution that summarizes a provenance expression. The first is, given a summary, compute its quality based on the measurements discussed above. The second is a search algorithm that explores multiple possible expressions, uses the first building block to compute the quality for each, and aims at finding the best ones. We next detail the two components.

4.1 Computing Summary Quality

Recall that a summary quality was defined through the notion of distance. Let \( \text{DIST-COMP} \) be the problem of computing the exact distance (w.r.t. all possible valuations) between two provenance expressions \( p \) and \( p' = h(p) \), given input \( p \), \( h \), \( \phi \) and \( \text{VAL-FUNC} \).

**Proposition 4.1.1.** \( \text{DIST-COMP} \) is \( \#P \)-hard in the size of the input provenance \( p \). This is true even if \( p \) includes no tensor elements.

**Proof.** The proof is by reduction from the \( \#P \)-DNF, the problem of counting the number of satisfying valuations to a DNF formula, which is known to be \( \#P \)-hard. Consider \( X' = \{ A \} \), and a mapping \( h \) that maps every annotation in \( X \) to this \( A \). Now given a formula \( f \), we claim that given \( \text{dist}(f, h(f)) \) one may compute the number of satisfying valuations to \( f \) via the following simple and efficient algorithm:

1. check if \( f \) and \( h(f) \) agree on the truth valuation sending all its annotations to \( true \),
2. if so then the number of unsatisfying valuations to \( f \) is exactly \( \text{dist}(f, h(f)) \), otherwise it is \( \text{dist}(f, h(f)) - 1 \).

On the other hand, approximating the distance is feasible.

---

**17**
Proposition 4.1.2. Given a provenance expression \( p \) and a homomorphism \( h \) on its annotations, and given \( 0 < \epsilon, 0 < \delta < 1 \), one can compute \( d' \) such that \( \text{Prob}\left( |d' - \text{dist}(p, h(p))| > \epsilon \right) < 1 - \delta \). The computation of \( d' \) may be performed in polynomial time with respect to \( |p|, \delta, \) and \( \frac{1}{\epsilon} \).

**Proof.** Absolute approximation may be obtained via a sampling algorithm. Each sample operates by (1) choosing a truth valuation for the annotations occurring in \( p \), (2) computing the value of \( p \) with respect to this valuation, (3) computing the corresponding valuation for annotations in \( h(p) \) based on \( h, \phi \), (4) increasing SuccCounter by the VAL-FUNC value computed using the values from (2) and (3), (5) increasing SampleCounter. Outputing \( \frac{\text{SuccCounter}}{\text{SampleCounter}} \) as an approximation of the distance. The convergence rate is then guaranteed by Chebyshev’s Inequality [9].

The proof is constructive in the sense that it involves a simple sampling-based approximation algorithm, that will be used as a building block in our summarization algorithm.

### 4.2 Finding a Summarization

Towards a summarization algorithm, we recall that the set of truth valuations \( V_{Ann} \) may be restricted, guided by the intended use (in the sequel we will assume that \( V_{Ann} \) is given as input). In this case, we observe that \( V_{Ann} \) may already dictate some simplifications that may be performed (see below).

**Proposition 4.2.1.** Given a provenance expression \( p \), finding a minimal \( p' \) such that \( \text{distance}(p, p') = 0 \) is in \( \text{PTIME} \) in \( p \) and in \( V_{Ann} \).

**Proof.** We first find the set of equivalence classes of annotations in \( Ann \). Let \( a_1 \) and \( a_2 \) be two annotations in \( Ann \). \( a_1 \) and \( a_2 \) are in the same equivalence class iff for each valuation \( \nu \) in \( V_{Ann} \): \( \nu(a_1) = \nu(a_2) \). To ensure that such equivalence classes exist, we assume that no valuation of \( V_{Ann} \) is contradictory. This can be checked in polynomial time. Let \( p' \) be a provenance expression obtained from \( p \) s.t. \( \text{distance}(p, p') = 0 \) and
$h$ be an homomorphism from $p$ to $p'$. Then for each annotation $a'$ in $p'$, $h^{-1}(a')$ is exactly the set of elements in the equivalence class of $a'$. It implies in particular that two annotations in $p'$ cannot belong to the same equivalence class. On the other hand, let $a_i$ be a representative element for the $i$th equivalence class (the order is arbitrary). Let $h$ be an homomorphism from the annotations of $p$ to $p'$ defined as follows: for each $i$ and for each element $a$ of the $i$th equivalence class, $h(a) = a_i$. Notice that the homomorphism is well defined because for each valuation $\nu$ in $V_{Ann}$, $\nu(a) = \nu(a_i)$. This shows that to find a minimal provenance formula we simply need to find the equivalence classes.

We next show that finding the set of equivalence classes is in PTIME. For each valuation $\nu_i$, we find the set of annotations assigned true, denoted by $T_{\nu_i}$, and the set of annotations assigned false, denoted by $F_{\nu_i}$. We compute the equivalence classes recursively. The initialization, done for the first valuation, $\nu_1$, returns $T_{\nu_1}$ and $F_{\nu_1}$. Let $S_1, ..., S_k$ be the equivalence classes for the first $i$ valuations. The equivalence classes for the $i + 1$ valuations are then obtained as follows: $\cup_{j \leq k} \{S_j \cap T_{\nu_{i+1}}\} \cup \{S_j \cap F_{\nu_{i+1}}\}$. We keep only the non empty sets. Notice that the number of equivalence classes is polynomial. Indeed, an element can belong to two equivalence classes iff there is a contradictory valuation in $V_{Ann}$. Therefore, finding the equivalence classes can be done in polynomial time.

**Equivalence Classes** The above proof is based on computing equivalence classes of annotations with respect to a set of valuations, with every two annotations being equivalent if they agree for *every* valuation in the set. The intuition is that there is no need to maintain these different annotations, since they in any case may not be differentiated. Replacing them by ("mapping them to") the same annotation will further allow simplifications of the expression based on the algebraic identities. This calls for a first step in the summarization algorithm, grouping together provenance annotations that are equivalent based on the above definition. Of course, this may still yield expressions of large size; we will thus perform a $A^*$-like search ([27]) of expressions, motivated by the next proposition.
Monotonicity  Let $p_0, p_1, ..., p_n$ be provenance expressions such that $p_i = h_i(p_{i-1})$ for some sequence of homomorphisms $h_i$. We define monotonicity of the distance and size functions, as follows: the distance function is increasing monotone iff for all $i > j$: $\text{distance}(p_0, p_i) \geq \text{distance}(p_0, p_j)$ and the size function is decreasing monotone iff for all $i > j$: $\text{size}(p_i) \leq \text{size}(p_j)$.

**Proposition 4.2.2.** All the VAL–FUNC functions previously mentioned yield increasing monotone distance and decreasing monotone size functions.

**Proof.** Let $p_0, p_1, ..., p_n$ be provenance expressions such that $p_i = h_i(p_{i-1})$ for some sequence of homomorphisms $h_i$. We must show that:

$\text{distance}(p_0, p_i) \geq \text{distance}(p_0, p_j)$ for all $i > j$. First, notice that if we prove that for $i = j + 1$, then the proof for the general case $i = k$ is pretty straightforward:

$\text{distance}(p_0, p_k) \geq \text{distance}(p_0, p_{k-1}) \geq \ldots \geq \text{distance}(p_0, p_k-(k-j-1)) \geq \text{distance}(p_0, p_k-j)$

So assume that $i = j + 1$. Also, assume $\phi = \lor$ (a similar proof exists for $\phi = \land$).

We first prove this proposition for VAL–FUNC($v, v', p, p'$) = $w(v) \cdot | v'(p') - v(p) |$.

According to Definition 3.2.2, to prove the distance inequality, it’s sufficient to prove the following inequality for all valuations $v$ defined on the annotations in $p_0$:

$| v'(p_i) - v(p_0) | \geq | v'(p_j) - v(p_0) |$

As a remainder, $v'$ is the valuation, defined on the new annotations, which is obtained from $v$, from the relevant homomorphism $h$ and the $\phi$ function. To obtain $p_i$ from $p_j$ ($p_i = h_i(p_j)$), annotations are mapped to a new annotation summary. There are two possible cases:

1. There exist annotations $a, a', b$ s.t. both $a, a'$ appear in $p_j$, but annotation $b$ replaces them in $p_i$. Since $p_i$ and $p_j$ are consecutive, all the other annotations in $p_j$ also appear in $p_i$.

2. There exist annotations $a, b$ s.t. $b$ is a new annotation (it does not appear in $p_0$ but it does appear in $p_j$) and $a$ is mapped to $b$ to obtain $p_i$ from $p_j$.  

20
To simplify the proof we only prove the proposition for the first case. The proof for
the second case is very similar. Let v be a valuation defined on the annotations in
\(p_0\). Consider the following cases:

a) Annotations \(a, a'\) are both assigned \(False\) by v: in this case, since \(\phi = \lor\), v’
also assigns \(False\) to b. So the annotation b “contributes” (the monoid values
associated with it) to the aggregated value of \(v'(p_i)\) the same contribution that
the original annotations “contribute” to the aggregated value of \(v'(p_j)\). Since the
provenance expressions “agree” on the other annotations, we can conclude that
in this case \(v'(p_i) = v'(p_j)\) and the weak inequality holds.

b) Annotations \(a, a'\) are both assigned \(True\) by v: v’ also assigns \(True\) to b, the
contribution of the annotations is the same so \(v'(p_i) = v'(p_j)\) is also true in this
case and the weak inequality holds.

c) Valuation v assigns \(True\) to a and \(False\) to a’: since \(\phi = \lor\), v’ also assigns
\(True\) to b. The contribution of a’ to the overall aggregated value of \(v'(p_j)\) is
canceled, whereas they both contribute to the value of \(v'(p_i)\) through the new
annotation b. If the aggregation function used is \(MAX\) or \(SUM\), that means that
\(v'(p_i) \geq v'(p_j)\) and for the same reason: \(v'(p_j) \geq v(p_0)\) and \(v'(p_i) \geq v(p_0)\) and
then the weak inequality holds. If the aggregation is \(MIN\), then \(v'(p_i) \leq v'(p_j)\),
\(v'(p_j) \leq v(p_0)\) and \(v'(p_i) \leq v(p_0)\) and then the inequality also holds.

d) Valuation v assigns \(False\) to a and \(True\) to a’: the same as c.

It remains to show that \(size(p_i) \leq size(p_j)\). We defined the provenance size as
the number of annotations the expression consists of (with repetitions). Since \(a, a'\)
are replaced by b, there are two possible forms for the new expression \(p_i\): either
\((b \cdot c_1 \cdot \ldots \cdot c_n) \otimes (AGG, 2) \oplus \ldots\) or \((b \cdot c_1 \cdot \ldots \cdot c_n) \otimes (v_1, 1) \oplus (b \cdot d_1 \cdot \ldots \cdot d_m) \otimes (v_2, 1)\ldots\nIn the second form the size remains the same and in the first form it decreases by one.
Either way, the size part of the proposition also holds. To conclude, the distance
function is indeed monotonically increasing and the size is indeed monotonically decreasing.

We next prove this proposition for: \( \text{VAL-FUNC}(v, v', p, p') = 0 \) if \( v(p) = v'(p') \) and \( = w(v) \) otherwise. It’s sufficient to prove the following inequality for all valuations \( v \) defined on the annotations in \( p_0 \): \( \text{VAL-FUNC}(v, v', p_0, p_i) \geq \text{VAL-FUNC}(v, v', p_0, p_j) \)

We prove this by contradiction. Assume the opposite is true, meaning:

\( \text{VAL-FUNC}(v, v', p_0, p_i) < \text{VAL-FUNC}(v, v', p_0, p_j) \)

Since the possible values for \( \text{VAL-FUNC} \) are 0 and \( w(v) \) this implies that:

\( \text{VAL-FUNC}(v, v', p_0, p_i) = 0 \) and \( \text{VAL-FUNC}(v, v', p_0, p_j) = w(v) \).

This further implies that: \( v(p_0) = v(p_i) \) and \( v(p_0) \neq v(p_j) \). We already proved that \( |v'(p_i) - v(p_0)| \geq |v'(p_j) - v(p_0)| \), but since \( |v'(p_i) - v(p_0)| = 0 \) and \( |v'(p_j) - v(p_0)| \neq 0 \) this leads to the following contradiction: \( 0 > |v'(p_j) - v(p_0)| \)

The proof for the size monotonicity is the same as above.

Finally, we prove this proposition for \( \text{VAL-FUNC}(v, v', p, p') = \text{euclidean-dist}(v'(p'), v(p)) \).

Similarly to the previous cases, it’s sufficient to prove the following inequality:

\( \text{euclidean-dist}(v'(p_i), v(p_0)) \geq \text{euclidean-dist}(v'(p_j), v(p_0)) \)

Assume \( v(p_0) = (a_1, a_2, ..., a_n) \), \( v(p_j) = (b_1, b_2, ..., b_n) \) and \( v(p_i) = (c_1, c_2, ..., c_n) \) where \( a_k, b_k, c_k \) are all aggregated values associated with object \( k \) (e.g. aggregated rating for movie \( k \)). We need to show that:

\[
\sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2 + \ldots + (b_n - a_n)^2} \geq \sqrt{(c_1 - a_1)^2 + (c_2 - a_2)^2 + \ldots + (c_n - a_n)^2}
\]

It’s sufficient to prove that for all \( k < n \): \( (b_k - a_k)^2 \geq (c_k - a_k)^2 \). Notice that we already proved that \( |b_k - a_k| \geq |c_k - a_k| \) (\( \text{VAL-FUNC} \) that returns an aggregated value for a single object rather than a vector of aggregated values for multiple objects), so the above inequality holds. The proof for the size monotonicity is the same as above.

\( \square \)

Naturally, not every choice of \( \text{VAL-FUNC} \) leads to monotonicity, e.g. a function that returns alternating constants, but, as the above proposition indicates, natural choices of functions do.
Input: A provenance expression \( p_0 \) with a set of annotations \( Ann \), \( \phi \) and \( \text{VAL-FUNC} \) functions, distance and size weights, size bound \( \text{TARGET-SIZE} \) and distance bound \( \text{TARGET-DIST} \)

Output: Summary provenance expression \( p_1 \)

\[
p' := \text{GroupEquivalent}(p_0, V_{Ann}) ;
\]

\[\text{while } \text{Size}(p') > \text{TARGET-SIZE} \text{ or } \text{ApproxDistance}(p_0, p', V_{Ann}) < \text{TARGET-DIST} \text{ do}
\]

\[\text{for every } h \in \text{CandidateHom}(p') \text{ do}
\]

\[h' : = h(p') ;
\]

\[\text{if } \text{CandidateScore}^{h, \phi}(p_0, p_{\text{cand}}) \text{ is minimal then}
\]

\[p'_{\text{prev}} := p' ;
\]

\[p' := p_{\text{cand}} ;
\]

\[\text{end}
\]

\[\text{end}
\]

\[\text{end}
\]

\[\text{if } \text{ApproxDistance}(p_0, p', V_{Ann}) \geq \text{TARGET-DIST} \text{ then}
\]

\[\text{return } p'_{\text{prev}} ;
\]

\[\text{end}
\]

\[\text{return } p' ;
\]

Algorithm 1: Provenance Summarization Algorithm

Provenance Summarization Algorithm The above proposition leads to Algorithm 1. Starting from the original set of annotations \( Ann \) and the given provenance expression \( p_0 \), the heuristic algorithm constructs the homomorphism \( h \) gradually, essentially by deciding on grouping of annotations. First, we obtain \( p' \) by grouping annotations that are equivalent w.r.t. the set of truth valuations (GroupEquivalent in line 1), as indicated by Proposition 4.2.1. Then, we iterate and in each step examine a set of possible single-step mappings (in CandidateHom) of two annotations to the same, new annotation name (line 3). For each such mapping we apply the obtained homomorphism to the current expression, computing \( h(p') \) (line 4) and approximating the distance between \( p_0 \) and \( p_{\text{cand}} = h(p') \). The \( p_{\text{cand}} \) with the smallest CandidateScore value is chosen (lines 5-8) and the process repeats until the stop condition is met. The stop condition for \( \text{TARGET-SIZE} \) (\( \text{TARGET-DIST} \)) is when the expression meets the size (resp. distance) bound (line 2).

If multiple candidates have minimal candidate scores, input taxonomies, if given as input, are used to break ties. For each such candidate, the taxonomy distances of the annotations from the new annotation they are mapped to are computed and
the MAX (or SUM) of these distances is computed. The candidate that minimizes this value is chosen. If no taxonomies are given as input, we arbitrarily choose a candidate with minimal score.

It is also possible to limit the number of algorithm iterations and by that control the running time of the algorithm. We call each algorithm iteration algorithm step. We will further discuss this “number of steps” input parameter in Chapter 6.

**Example 4.2.3.** Returning to our running example, assume now that the Movies table also includes a movie genre column. Further assume that the user would like to view scores of movies of certain genres and so the aggregator now aggregates multiple movies of the same user-specified genre. We next exemplify the algorithm flow, using the following provenance expression for the movies “Match Point” and “Blue Jasmine”: \( P_0 = P_{MP} \oplus_M P_{BJ} \) where \( P_{MP} = P_s \) is the provenance expression from Example 3.1.1 that consists of the three user reviews for the movie “Match Point”, \( P_{BJ} = U_2 \otimes (4,1) \) is the added review for the movie “Blue Jasmine” and \( \oplus_M \) is a formal sum for combining reviews of different movies (we will later see how this formal sum is used).

In each step, the algorithm examines the set of possible mappings of two annotations to the same new annotation. The mappings \( U_1, U_2 \rightarrow \text{Female} \) and \( U_1, U_3 \rightarrow \text{Audience} \) discussed in Example 3.1.1 are such possible single-step mappings that are examined by the algorithm. For simplicity, assume these are the only possible mappings. The algorithm computes the new provenance expressions that the candidate mappings yield:

\[
P'_0 = P'_{MP} \oplus_M P'_{BJ} = \text{Female} \otimes (5,2) \oplus U_3 \otimes (3,1) \oplus_M \text{Female} \otimes (4,1)
\]

\[
P''_0 = P''_{MP} \oplus_M P''_{BJ} = \text{Audience} \otimes (3,2) \oplus U_2 \otimes (5,1) \oplus_M U_2 \otimes (4,1)
\]

Also, a candidate score for each such candidate is computed. Assuming \( w_{\text{Dist}} = 1 \) and \( w_{\text{Size}} = 0 \), the candidate that is chosen in each step in the one that minimizes the distance from the original provenance \( P_0 \).

Assume we compute the distance w.r.t. the class of valuations that cancel a single annotation and the euclidean distance \( \text{VAL-FUNC} \). Keep in mind that evaluating a
valuation on this kind of provenance, that consists of reviews for different movies, results in a vector of aggregated ratings where each coordinate holds the aggregated rating of a certain movie. In this setting, $P_0''$ is at distance 0 from $P_0$ while $P_0'$ differs from $P_0$ for the valuation that cancels $U_2$. This is due to the fact that by canceling $U_2$ in $P_0$ we cancel the maximum rating for “Match Point” and the only rating for “Blue Jasmine”. Canceling $U_2$ in $P_0'$ does not have a similar effect since we use a disjunction of the truth values (of $U_1, U_2$) as our $\phi$ function, and so the new annotation “Female” is assigned the value true. Obviously, the euclidean distance between the aggregation vectors that are the result of evaluating this valuation on $P_0$ and $P_0'$ is greater than zero and so the overall distance over all considered valuations is greater than zero. This leads to the conclusion that the algorithm would choose $P_0''$ over $P_0'$ so the provenance for the next algorithm iteration is $P_1 = P_0''$ and so the algorithm continues.
V Datasets and Use Cases

We next describe the three provenance datasets that we used and show provenance examples for these datasets.

5.1 Datasets

1) MovieLens dataset, that includes ratings of different movies by users of MovieLens movie recommender that is based on collaborative filtering ([1]).

2) Wikipedia dataset - collected using the MediaWiki web API which is a Web service that provides convenient access to wiki features, data, and meta-data over HTTP. We also used YAGO Taxonomy ([4]) that contains rdfs:subClassOf facts derived from Wikipedia and WordNet. This taxonomy was used in our provenance summarization algorithm in order to improve the choices made by the algorithm when the input is a Wikipedia provenance expression. We used Wu-Palmer method for measuring semantic relatedness ([29]) in order to compute the distance between WordNet concepts in the taxonomy.

3) Data-Dependent Processes (DDP’s) dataset - we generated provenance expressions that represent data-dependent processes based on the structure described in [17].

Table 5.1 describes the provenance structure and the parameters used for each of these datasets. For example, all datasets share the same valuation classes - “Cancel Single Annotation” which we have demonstrated in the movies provenance examples throughout the thesis and “Cancel Single Attribute” which is the class of valuations that cancel all annotations that share the same attribute and assigns true to the rest (e.g. the valuation that cancels all Male users).
<table>
<thead>
<tr>
<th>Type</th>
<th>Structure</th>
<th>Mapping Constraints</th>
<th>Aggregation</th>
<th>Valuations Classes</th>
<th>φ Functions</th>
<th>VAL-FUNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movies</td>
<td>(UserID₁ · MovieTitle₁ · MovieYear₁) ⊗ (Rating₁₁, 1) ⊕ (UserID₂ · MovieTitle₂ · MovieYear₂) ⊗ (Rating₂₁, 1) ⊕ ...</td>
<td>Gender, Age Range, Occupation, Zip Code</td>
<td>MAX, SUM</td>
<td>Cancel Single Annotation, Cancel Single Attribute</td>
<td>Logical OR</td>
<td>Euclidean Distance</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>(Username₁ · PageTitle₁) ⊗ (EditType₁₁, 1) ⊕ (Username₂ · PageTitle₂) ⊗ (EditType₂₁, 1) ⊕ ...</td>
<td>For Users: isRegistered, Gender, Contribution Level. For Wikipedia Pages: Taxonomy Wordnet Concept</td>
<td>SUM</td>
<td>Same as above, but only valuations that are consistent with the taxonomy</td>
<td>Same as above</td>
<td>Same as above</td>
</tr>
<tr>
<td>DDP</td>
<td>(c₁, 1) · (0, [d₁ · d₂] ≠ 0) + (0, [d₂ · d₃] = 0) · (c₂, 1)</td>
<td>Mapping of DB (resp. cost) vars to new DB (cost) vars. Defined as part of the structure’s semirings.</td>
<td>Same as above</td>
<td>DB vars: Logical OR, Cost vars: MAX</td>
<td>Absolute Difference</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1: Description of Provenance and Summarization Parameters for Each Dataset

5.2 Use Cases

Example 5.2.1. A Wikipedia provenance expression represents different user edits of Wikipedia pages that belong to different categories. Each user edit can either be minor (0) or major (1). Consider the following provenance expression:

\[ P₀ = (\text{SalubriousToxin} \cdot \text{Adele}) \otimes (0, 1) \oplus \]
\[ (\text{Dubulge} \cdot \text{CelineDion}) \otimes (1, 1) \oplus \]
\[ (\text{Dr. Back-In-The-Street} \cdot \text{LoriBlack}) \otimes (1, 1) \oplus \]
\[ (\text{JaspertheFriendlyPunk} \cdot \text{AlecBaillie}) \otimes (1, 1). \]

This provenance includes 3 major (Dubulge, Dr. Back-In-The-Street and Jasper the Friendly Punk) and 1 minor (Salubrious Toxin) user edits of 4 Wikipedia pages - 2 pages whose title is a famous singer (Adele and Celine Dion) and 2 whose title is a famous guitarist (Lori Black and Alec Baillie).

To simplify the example, assume we only map user annotations to the same annotation if the associated users have a similar number of edits and then the new annota-
tion would describe their contribution level, e.g. “Top-Contributor” and “Reviewer”. Also assume we map Wikipedia pages to a new annotation only if the corresponding pages have the same parent WordNet concept in the taxonomy. Moreover, assume we use SUM aggregation and that we compute the distance w.r.t. $\phi = \lor$, euclidean distance $\text{VAL-FUNC}$ and the class of valuations that cancel a single annotation and are consistent with the taxonomy. A valuation is considered to be inconsistent if it assigns false to a Wikipedia category/WordNet concept $A$, but assigns true to a Wikipedia category/WordNet concept $B$ s.t. $B$ is a child of $A$ in the taxonomy. Assume the algorithm outputs the following summary:

$$P' = \left(\text{Top-Contributor} \cdot \text{<wordnet_guitarist>}\right) \otimes (2, 2) \oplus (\text{Reviewer} \cdot \text{<wordnet_singer>}) \otimes (1, 2).$$

According to the summary, two users that are top contributors edited Wikipedia pages of guitarists (one major edit each) and two simple reviewers edited Wikipedia pages of singers (one major edit and the other minor).

We next describe how the summary distance is computed before the algorithm chooses it as the optimal summary. To get the distance, the algorithm computes the $\text{VAL-FUNC}$ value for every examined valuation. Let's consider the valuation $v$ that cancels “Dubulge”. The evaluation result is a vector of aggregation results for each Wikipedia page/category:

$$v(p) = (\text{Adele}: 0, \text{Celine Dion}: 0, \text{Lori Black}: 1, \text{Alec Baillie}: 1).$$

Obtaining a new valuation $v'$ using the $\phi = \lor$, we get:

$$v'(p') = (\text{<wordnet_guitarist>}: 2, \text{<wordnet_singer>}: 1).$$

Notice that since we allow the mapping of both user and Wikipedia pages annotations, the evaluation results are vectors of different size. To compute the euclidean distance between the vectors, we first transform the original vector to a new vector of the same size of the vector that is the evaluation result on the new expression by using the aggregation function. In this case, we transform the vector $(\text{Adele}: 0, \text{Celine Dion}: 0, \text{Lori Black}: 1, \text{Alec Baillie}: 1)$ to:

$$(\text{<wordnet_guitarist>}: 2, \text{<wordnet_singer>}: 0).$$
Following this transformation it is possible to compute the euclidean distance between
the vectors as part of the distance computation.

By obtaining such a provenance summary it is easier to answer questions such as:
what are the most controversial or interesting topics, what are relatively popular
topics among top contributors, do top contributors make more major edits relative
to other users, etc (e.g. obtaining a summary similar to the above summary for
many Wikipedia users might lead us to the conclusion that top contributors prefer
to edit guitarist pages than singer pages). These are questions that are much harder
to answer using the original long provenance expression. We can also present such
summaries in a ui that makes it easier for the user to understand the summary and
get insights on the underlying data.

Example 5.2.2. A DDP (Data Dependent Process), described in [17], models an
application whose control flow is guided by a finite state machine, as well as by
the state of an underlying database. DDP provenance expressions are summaries
of executions where an execution is a multiplication of transitions. Each transition
is either based on a user’s choice or on a database query result. A user dependent
transition is of the form \( \langle c_k, 1 \rangle \) where \( c_k \) is the cost associated with the transition
(the user’s effort). A database dependent transition is of the form \( \langle 0, [d_i \cdot d_j] \neq 0 \rangle \) or
\( \langle 0, [d_i \cdot d_j] = 0 \rangle \). Consider the following DDP provenance example of two executions
(each consisting of two transitions): \( \langle c_1, 1 \rangle \cdot \langle 0, [d_1 \cdot d_2] \neq 0 \rangle + \langle 0, [d_2 \cdot d_3] = 0 \rangle \cdot \langle c_2, 1 \rangle \).

The aggregation function used is based on the semirings described in [17] - the
tropical semiring \((\mathbb{N}^\infty, \min, +, \infty, 0)\) and the semiring of polynomials with natural coef-
ficients \((\mathbb{N}[X], +, \cdot, 0, 1)\). The “attributes” that are used as constraints here are the
mappings of different database variables to new database variables and the mappings
of cost variables to new cost variables. If two database variables are mapped to a
single one, it means that either both tuples need to be present for the database query
to be satisfied or both should be missing. Similarly, if we know that user transitions
have more or less the same cost, it is possible to map the two cost variables to a new
cost variable. Also, assume we use the “Cancel Single Attribute” class of valuations.
The following is a possible summary for the above provenance, obtained by mapping $d_1,d_3$ to $D_1$ and $c_1,c_2$ to $C_1$:

$$\langle C_1,1 \rangle \cdot \langle 0, [D_1 \cdot d_2] \neq 0 \rangle + \langle 0, [d_2 \cdot D_1] \neq 0 \rangle \cdot \langle C_1,1 \rangle$$

which is equal to: $\langle C_1,1 \rangle \cdot \langle 0, [D_1 \cdot d_2] \neq 0 \rangle$.

This final summary represents a single execution that consists of two transitions - one user dependent transition and one database dependent transition.

Next we describe how the summary distance is computed before it is chosen. A valuation with respect to the DDP provenance expression, assigns boolean values to the database variables and 0 or 1 values to the cost variables. The valuation value, in this case, is either $\langle C,\text{True} \rangle$ if there exists an execution for which the evaluation of the relevant database query is true and the user’s effort is $C$, or $\langle C,\text{False} \rangle$ if no such execution exists.

Let’s examine the following valuation $v$ that cancels all the cost variables that have the same cost $C_1$:

$c_1 \rightarrow 0, c_2 \rightarrow 0, d_1 \rightarrow \text{True}, d_2 \rightarrow \text{True}, d_3 \rightarrow \text{True}$.

The value the valuation assigns to $p$, $v(p)$, is equal to $\langle 0,\text{True} \rangle$. Using MAX $\phi$ function on the cost variables and logical OR $\phi$ function on the DB variables, $C_1$ is assigned 0 and $D_1$ is assigned True. As a result, the evaluation result on the new expression is equal to: $\langle 0,\text{True} \rangle$.

Given the evaluation results on the original expression and on the new expression: $v(p) = \langle C_p,T_p \rangle$ and $v'(p') = \langle C'_p,T'_p \rangle$ respectively, the VAL-FUNC is the difference function which is defined as follows: if $T_p = T'_p = \text{True}$ then return $| C_p - C'_p |$. If $T_p = T'_p = \text{False}$ then return 0. If $T_p \neq T'_p$ then return the maximum possible difference in costs, which is the maximum cost per single transition (10) multiplied by the number of transitions per execution (5). Therefore, in this example, the difference between the evaluation results is equal to 0. This means that for this particular valuation, there is no error.

By summarizing this kind of provenance, analysts can test and explore the effect of hypothetical modifications to a DDP’s state machine and/or to the underlying
database (e.g. using the above summary, analysts can explore the effect of removing the database dependent transition). Exploring the effect of such modifications using the original provenance expression can be much more complicated.
VI Experiments

The main purpose of our experiments was to examine the effectiveness of our summarization algorithm, compared to other approaches, in terms of: (1) conciseness of the obtained provenance expression (measured by size), (2) accuracy of evaluations (measured by distance from original provenance), (3) faster provenance usage (“Usage Time” experiment) and (4) feasibility of summarization (“Summarization Time” experiment). The first two were examined as functions of the wDist weight (wDist experiment), and of the TARGET-DIST and TARGET-SIZE stop conditions (TARGET-DIST and TARGET-SIZE experiments). This covers the three computational problems that we have presented in section 3. Each experiment was conducted for the three datasets - MovieLens, Wikipedia and DDP. For each dataset, we generated multiple input provenance expressions, executed the experiments and averaged the results.

6.1 Algorithms Examined

In each experiment that we conducted, we executed the following algorithms for each dataset and compared different parameters of the result summary provenance: (1) Prov-Approx (Algorithm 1) - our provenance summarization algorithm. (2) Clustering - using only the MovieLens and Wikipedia datasets, since the DDP dataset is not suitable for the clustering experiment (it is not clear how to construct feature vectors to be used as input to the clustering algorithm). (3) Random - in which every pair of annotations was chosen randomly from the list of pairs that satisfy the mapping constraints. All three algorithms take into account the user-specified size and distance bounds (TARGET-SIZE and TARGET-DIST) and stop if and when they reach these bounds.
6.2 Clustering Algorithm

We used a library for hierarchical agglomerative clustering called HAC ([3]). Agglomerative Hierarchical Clustering (HAC), described in [13, 23], is a bottom-up algorithm that first treats each observation as a singleton cluster and then successively merges (or agglomerates) pairs of clusters. This library supports the following linkage criteria, i.e. a criteria that determines the distance between sets of observations as a function of the pairwise distances:

- “Single Linkage” - the distance between two clusters is calculated as the smallest distance between two objects in opposite clusters.
- “Average Linkage” - the distance between two clusters is calculated as the average of the distances between all pairs of objects in opposite clusters.
- “Centroid Linkage” - each cluster is represented by its centroid. The distance between two clusters is calculated as the distance between their centroids.
- “Complete Linkage” - the distance between two clusters is calculated as the largest distance between two objects in opposite clusters.
- “Median Linkage” - the distance between two clusters is the Euclidean distance between their weighted centroids.
- “Ward Linkage” - this method fuses those two clusters that result in the smallest increase in the total within-group error sum of squares. This quantity is defined as the sum of squared deviation of each object from the centroid of its own cluster.
- “Weighted Average” - average linkage where the sizes of the clusters are assumed to be equal. This method, similar to “Median”, weights small and large clusters equally.
All linkage criteria were examined in the experiments, but since they all yield similar results compared to our approach we present the “Single Linkage” results.

Not all provenance datasets are suitable for a Clustering algorithm, e.g. our DDP dataset. We next explain how the clustering approach was implemented for the MovieLens and Wikipedia datasets.

For a MovieLens provenance, each user, that rated k movies, was associated with a feature vector of the following form:

\[
\text{(UID, Gender, AgeRange, Occupation, ZipCode, (MovieTitle}_1 = \text{Rating}_1, ..., \text{MovieTitle}_k = \text{Rating}_k))
\]

\[
\text{(UID278, M, 45 − 49, tradesman/craftsman, 60482, (TheFury = 4.0, NearDark = 4.0))}
\]

In addition, we implemented a dissimilarity measure for computing the distance between each pair of feature vectors. We used Pearson Correlation Coefficient as a measure of similarity between the ratings vectors, that the feature vectors include as a single feature. Moreover, we added our mapping constraints to the clustering algorithm so that both our algorithm and the clustering algorithm would take into account the same constraints (we do not allow two clusters to merge if the users that belong to these clusters do not have at least one attribute in common).

Similarly, the clustering approach was also implemented for the Wikipedia dataset. Since we merge not only user annotations but also Wikipedia pages annotations, the clustering approach was implemented and executed for users and Wikipedia pages separately. As part of the users clustering implementation, each user, that edited k Wikipedia pages, was associated with a feature vector of the following form:

\[
\text{(UID, Gender, IsRegistered, ContributionLevel, (PageTitle}_1 = \text{NumMajorEdits}_1, ..., \text{PageTitle}_k = \text{NumMajorEdits}_k))
\]

\[
\text{(Dubulge, Male, RegisteredUser, TopContributor, (Adele = 20, CelineDion = 14, LoriBlack = 12, ...)})
\]

As part of the Wikipedia pages clustering implementation, each Wikipedia page, that was edited by k users, and is a descendant of m WordNet concepts in Yago
taxonomy, was associated with a feature vector of the following form:

\[(\text{PageTitle}, (\text{AncestorWordNetConcept}_1, ..., \text{AncestorWordNetConcept}_m), (\text{UID}_1 = \text{NumMajorEdits}_1, ..., \text{UID}_k = \text{NumMajorEdits}_k)), \text{e.g.} (\text{Adele}, (\text{wordnet\_singer}, \text{wordnet\_musician}, \text{wordnet\_performer}, \text{wordnet\_entertainer}, \text{wordnet\_person}, \text{wordnet\_causal\_agent}, \text{wordnet\_physical\_entity}), (\text{Dubulge} = 20, \text{Ebyabe} = 100, \text{Smalljim} = 112,...)).\]

We also implemented dissimilarity measures for computing the distance between pairs of user feature vectors and Wikipedia pages feature vectors and added our mapping constraints.

We next describe how we obtain the Clustering’s provenance summary in order to compare its quality to ours. Similarly to our summarization algorithm, each step of the Clustering algorithm, in which two clusters are merged, corresponds to a mapping of 2 annotations to an annotation summary (e.g. if two user clusters \(C_1\) and \(C_2\) are merged by the Clustering algorithm, this corresponds to mapping all the user annotations of users that belong to these clusters to a new annotation summary \(C\)). According to this mapping, we compute the provenance expression obtained by the Clustering approach and use it to check the stop conditions: TARGET-DIST, TARGET-SIZE, etc.

### 6.3 Experimental Settings

The experiments were conducted for different combinations of datasets, valuation classes and aggregation functions and all combinations have similar results. Specifically, two valuation classes were examined: (1) “Cancel Single Annotation” - each valuation in this class cancels a single annotation by assigning it false and assigning true to the rest. (2) “Cancel Single Attribute” - the class of valuations that cancel all annotations that share the same attribute and assigns true to the rest (e.g. the valuation that cancels all Male users). For space constraints, we present a set of representative results. It is important to note that the distance values we present, represent average error over all valuations, which we divide by the maximum pos-
sible error in order to normalize to $[0,1]$. Presenting the un-normalized distances results in the same graph trends.

We next describe our experimental results for the MovieLens dataset. Later we show some experimental results for the other two datasets.

![Diagram](a) ![Diagram](b)

**Figure 6.1:** Average Distance as a Function of wDist and TARGET-SIZE (MovieLens Dataset)

![Diagram](a) ![Diagram](b)

**Figure 6.2:** Average Size as a Function of wDist and TARGET-DIST (MovieLens Dataset)

### 6.4 wDist Experiment

The purpose of this experiment is to check the effect of the wDist and wSize weights on the summary provenance distance and size. For that purpose, we bounded the
maximum number of algorithm steps, the \textit{TARGET-DIST} was set to 1 (max distance) and the \textit{TARGET-SIZE} was also set to 1 (minimum size) so that they would have no effect as stop conditions. Figures 6.1a and 6.2a show the experiment’s results for the MovieLens dataset using “Cancel Single Attribute” valuation class, MAX aggregation and at most 20 steps. As expected, using our Prov-Approx approach, greater values of wDist yield smaller distance values and greater size values. The wDist has no effect on the Clustering and Random approaches (they do not take this parameter into account) so we averaged their results for different wDist values. As the wDist used increases our Prov-Approx approach yields smaller distance compared to the Clustering (starting from wDist = 0.1 as presented in the graph), as expected. Also, the Clustering approach yields greater size compared to our approach. The Random approach yields much greater distance and size values.

\section*{6.5 \textit{TARGET-SIZE} Experiment}

This experiment examines the problem variant in which the user aims to reach a certain \textit{TARGET-SIZE} value while keeping the result relatively “close” to the original provenance. For that purpose, we set the wDist and \textit{TARGET-DIST} values to 1. Figure 6.1b shows the results of the experiment for the MovieLens dataset. As expected, since the wDist was set to 1, our approach gave better distance values compared to the Clustering and Random approaches. The Random approach gave the worst results. For the Prov-Approx and the Clustering algorithms, as the \textit{TARGET-SIZE} increases, the size increases (the algorithms stop earlier) and as a result the distance is smaller. This is not always the case for the Random approach. Since this approach does not make the same choices for all the \textit{TARGET-SIZE} values but just randomly chooses a pair, there could be better distance choices made when smaller \textit{TARGET-SIZE} values are used (e.g. the distance that Random achieved using \textit{TARGET-SIZE} 130 is actually smaller than the one achieved using 140).
6.6 TARGET-DIST Experiment

This experiment examines another variant, in which the user wishes to bound the distance with the TARGET-DIST value while obtaining a small provenance expression. For that reason, wDist was set to 0 and TARGET-SIZE was set to 1 also. Figure 6.2b shows the results of the experiment for the MovieLens dataset. As expected, as the TARGET-DIST increases the size decreases until we reach a point from where we cannot decrease the size further. Moreover, as a result of using wDist = 0 the choices made were the ones that minimize the next expression’s size and that reflects in the results, where our approach reaches the smallest size values. Random gives the worst results. Also, Random does not make the same choices for all TARGET-DIST values, so it is also possible that it would make better choices for smaller TARGET-DIST values and that would yield better size, just like in TARGET-DIST 0.03.

Figure 6.3: Average Size and Distance as a Function of wDist for Varying Number of Algorithm Steps (MovieLens Dataset)

6.7 Varying Number of Algorithm Steps Experiment

This experiment shows the progress made by the algorithm. It examines the average distance and size as a function of wDist for different number of algorithm iterations. Figures 6.3a and 6.3b show the results. Using 20 and 30 maximum number of steps, as the wDist increases the distance decreases and the size increases, as expected.
As you recall, in each iteration we map two annotations to a new annotation, so we expect that the more steps we use the greater the result distances are and the smaller the result sizes are and indeed this is the case. Using 40 maximum number of steps, most of the runs ended before reaching the maximum number of steps. As a result, the runs ended in different steps of the algorithm. Some executed more steps than others. In this case the wDist does not affect much on the outcome. We can see that the difference in distance and size for different wDist values is not so great for this number of steps.

![Graph A](image1.png)  
**Figure 6.4a:** Usage Time Ratio (Using 20 Steps) 

![Graph B](image2.png)  
**Figure 6.4b:** Usage Time Ratio (Using 30 steps) 

6.8 Usage Time Experiment

This experiment examines the ratio between the average evaluation time of valuations on the summary and original provenance expressions, as a function of wDist. Figures 6.4a and 6.4b show the results for the MovieLens dataset using 20 and 30 maximum number of steps respectively. The **TARGET-DIST** was set to 1 and the **TARGET-SIZE** to 1 so that the only relevant stop condition would be the number of steps. The experiment was conducted as follows: we randomly chose 10 valuations and evaluated these valuations on the original expression and the summary for the three approaches. We then examined the ratio of evaluation time. As expected, using Prov-Approx, as the wDist increases, the result’s size is greater and the distance
is smaller. For that reason, the expression is closer to the original expression and so the ratio in evaluation time is greater. Also, using more algorithm steps, the ratio is smaller; using 30 steps the range is 0.3-0.5 (30% - 50% improvement in evaluation time) compared to 0.45-0.65, when using 20 steps. To conclude, the summary usage time is faster than the original provenance usage time. In addition, the Random and Clustering approaches are not affected by wDist, so we averaged the results for all the wDist values. As expected, our approach yields smaller ratio compared to the Random approach using smaller wDist values. As far as the Clustering approach, it yields much greater ratio than ours for all wDist values (less improvement in usage time).

![Graphs showing time per candidate and summarization time](image)

Figure 6.5: Candidate Computation and Summarization Times (MovieLens Dataset)

### 6.9 Summarization and Candidate Computation Time Experiment

This experiment examines the summarization time and also the average candidate computation time (distance and size computation for a candidate pair of annotations) as functions of provenance size. Figures 6.5a and 6.5b show the results for the MovieLens dataset, wDist weight set to 1 and 50 maximum number of steps. As expected, as the expression size decreases, the number of pairs to consider in each step decreases and so the number of distance calculations decreases and as a
result the summarization time decreases. Also, as the expression size decreases, the distance computation is faster, as expected.

![Graph](image)

Figure 6.6: Average Distance as a Function of wDist and TARGET-SIZE (Wikipedia Dataset)

![Graph](image)

Figure 6.7: Average Size as a Function of wDist and TARGET-DIST (Wikipedia Dataset)

### 6.10 Other Datasets

All the figures so far show results using the MovieLens dataset; we next describe results for the other two datasets (Wikipedia and DDP). Figures 6.6a, 6.7a, 6.8a and 6.9a show the results of the wDist experiment conducted on the Wikipedia and DDP datasets using 20 and 10 maximum number of steps respectively. Figures
6.6b, 6.7b, 6.8b and 6.9b show the results of the TARGET-SIZE and TARGET-DIST experiments for these datasets, using “Cancel Single Annotation” valuation class and sum aggregation for the Wikipedia dataset and “Cancel Single Attribute” for the DDP dataset. All results are similar to those obtained for the MovieLens dataset. Note that the DDP dataset is the only one that wasn’t compared to the Clustering approach since it’s unclear how to construct a Clustering competitor for this complex-structured data provenance.
To allow users to experience firsthand the advantages that summarization of data provenance holds, we have developed a system called PROX. PROX system uses the movies dataset that consists of ratings provided by users of MovieLens, a movies recommendation website which is based on collaborative filtering ([1]). We show that while full provenance is too big to be presented, PROX allows for a summarized representation of the provenance that provides a concise explanation of the recommendations, and further allows for approximate provisioning.

**Figure 7.1: System Architecture**

### 7.1 System Architecture

PROX server-side is implemented in Java using Spring to expose REST APIs to the client. The client-side is implemented in Angular JS. This web application is deployed to Apache Tomcat server on a Windows 7 machine. The system architecture is depicted in Figure 7.1. The server is comprised of three major services:

- A selection service that allows simple restriction of the provenance according
to user-defined selection criteria (such as provenance for a subset of movies, in
our examples)

- A summarization service that summarizes the selected provenance using the
  summarization algorithm (Algorithm 1) discussed in Chapter 4.

- An evaluator (provisioning) service that allows to use the summarized prove-
  nance for (approximated) exploration of hypothetical scenarios. As explained
  in Chapter 2, provisioning the result with respect to hypothetical scenarios
  allows to observe changes to the result based on changes to the input without
  re-running the process (e.g. it is possible to examine the scenario in which all
  Male users were not asked to rate a certain movie).

### 7.2 PROX Web UI

We developed a web UI that the user can interact with. It contains three views:

- The selection view that allows the user to choose movies whose provenance she
  would like to observe. The user can select movies according to their title and
  also search for movies she is interested in, as shown in Figure 7.2. If the user
  prefers to choose movies according to genre and year, she can instead specify
  the genres and year of release (Figure 7.3).

- The summarization view, shown in Figure 7.4, that allows the user to view
  the selected provenance that is used as input to the summarization algorithm
  and more importantly configure different parameters for the summarization
  algorithm such as aggregation function, valuations class, etc.

- The third view presents the summary expression in two subviews: expression
  view and groups view and allows the user to switch between them. The ex-
  pression view (Figure 7.8) shows the summary provenance in its polynomial
  form as exemplified throughout this thesis, with some related information, e.g.
its size. The groups view is presented in Figures 7.5, 7.6, 7.7. It shows the groups of users that the algorithm has chosen to map together, the annotations they were mapped to and information about each group. For instance, for the Male group in Figure 7.5, we can see the group size (16), the users that the algorithm mapped to Male, the movies they rated, their ratings and the aggregated rating (MAX here) of the group users (AGG:5). This last view also allows the user to apply valuations of her choice for provisioning, as presented in Figures 7.9 and 7.10.

---

**PROX Web UI - Selection View (Movies)**

First, select movies provenance by specifying the desired data components - either movie titles or movie genres and year.

 LEDs Choose movie titles

- **titan**
  - Chambermaid on the Titanic, The
  - Raise the Titanic
  - Remember the Titans
  - Titan A.F.
  - Titanic

 LEDs Choose movie genres and year

Get selected provenace!
First, select movies provenance by specifying the desired data components - either movie titles or movie genres and year.

- Choose movie titles
- Choose movie genres and year

Year: 1995

Figure 7.3: PROX Web UI - Selection View (Genres and Year)
Selected Provenance Expression


Provenance Size: 126

Next, to approximate this provenance expression choose input parameters to be used in our summarization algorithm:

Distance weight:

Size weight:

Distance bound:

Size bound: 1

Number of steps: 10

Aggregation: MAX

Valuation class: Cancel Single Annotation

VAL-FUNC: Euclidean Distance

Figure 7.4: PROX Web UI - Summarization View
To observe the algorithm in action, step use the left and right arrows.

To evaluate assignments, the next specify an assignment expression, you can evaluate button to get the evaluation result and execution time (in nanoseconds).

Specify an assignment expression:

- By choosing false
- By choosing false

Figure 7.5: PROX Web UI - Summary View (Groups)
To observe the algorithm’s results step by step use the left and right arrows.

To evaluate assignments on the resulting provenance expression, you can next specify an assignment and press the evaluate button to get the evaluation result along with the evaluation time (in nanoseconds).

Figure 7.6: PROX Web UI - Summary View (Group User Attributes)
Figure 7.7: PROX Web UI - Summary View (Group Movie Attributes)
Summary Provenance - Expression


Provenance Size: 75

To observe the algorithm's results step by step use the left and right arrows.

To evaluate assignments on the resulting provenance expression, you can next specify an assignment and press the evaluate button to get the evaluation result along with the evaluation time (in nanoseconds).

Specify an assignment:

◽ By choosing false annotations
◽ By choosing false attributes

Evaluate assignment!  Clear result

Figure 7.8: PROX Web UI - Summary View (Expression)
To observe the algorithm’s results step by step use the left and right arrows.

To evaluate assignments on the resulting provenance expression, you can next specify an assignment and press the evaluate button to get the evaluation result along with the evaluation time (in nanoseconds).

Specify an assignment:

❖ By choosing false annotations

- UID245
- Friday
- 1995
- UID1019
- Your False Value

❖ By choosing false attributes

Evaluate assignment!  Clear result

**Evaluation Result:**

<table>
<thead>
<tr>
<th>Movie Title</th>
<th>Aggregated Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Party Girl</td>
<td>5</td>
</tr>
<tr>
<td>Bye Bye, Love</td>
<td>5</td>
</tr>
<tr>
<td>Sleepover</td>
<td>1</td>
</tr>
<tr>
<td>Man of the House</td>
<td>4</td>
</tr>
<tr>
<td>Friday</td>
<td>5</td>
</tr>
</tbody>
</table>

Evaluation Time: 48118 nanoseconds
To observe the algorithm’s results step by step use the left and right arrows.

To evaluate assignments on the resulting provenance expression, you can next specify an assignment and press the evaluate button to get the evaluation result along with the evaluation time (in nanoseconds).

Specify an assignment:

- By choosing false annotations
- By choosing false attributes

Evaluation Result:

<table>
<thead>
<tr>
<th>Movie Title</th>
<th>Aggregated Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Party Girl</td>
<td>5</td>
</tr>
<tr>
<td>Bye Bye, Love</td>
<td>5</td>
</tr>
<tr>
<td>Sleepover</td>
<td>0</td>
</tr>
<tr>
<td>Man of the House</td>
<td>3</td>
</tr>
<tr>
<td>Friday</td>
<td>5</td>
</tr>
</tbody>
</table>

Evaluation Time: 1023004 nanoseconds
VIII Related Work

Provenance models have been extensively studied in multiple lines of research such as provenance for database transformations (see e.g. [18, 11, 8, 21, 10, 12]), for workflows (see e.g. [16, 14, 5, 22, 19, 28, 24]), for the web [2], for data mining applications [20], and many others, but typically full and exact provenance is presented (sometimes in an optimized form, e.g. factorized as in [25]). Provenance views have been proposed in context of workflows (see e.g. [16]), but the summarization obtained through these views is based on a notion of granularity levels, and is lossless rather than approximate.

A notion of approximate provenance was proposed in [26]. In particular, their focus was probabilistic databases, in which lineage is fundamental to processing probabilistic queries and understanding the data. Many state-of-the-art systems use a complete approach, e.g. Trio or Mystiq, in which the lineage for a tuple \( t \) is a Boolean formula which represents all derivations of \( t \). A consequence of ignoring some derivations is that their system may return an approximate query probability, instead of the true value. An application may be able to tolerate this difference, especially if the approximate answer can be obtained significantly faster. A second issue is that although a complete lineage approach explains all derivations of a tuple, it does not expose which facts are the most influential in that derivation. In large data sets, a derivation may become extremely large because it aggregates together a large number of individual facts. This makes determining which individual facts are influential an important and non-trivial task. To obtain approximate lineage, they compress the data by tracking only the most influential facts in the derivation. They propose two specific kinds of approximate lineage: (1) a conservative approximation called sufficient lineage that records the most important derivations for each tuple, and (2) polynomial lineage, which is more aggressive and can provide
higher compression ratios, and which is based on Fourier approximations of Boolean
expressions. Their notion of approximate lineage somewhat resembles ours, but is
limited to UCQs (and in particular allows no aggregates), geared towards proba-
bilistic computation, and does not account for semantic constraints.

Our notion of mapping to summarized annotations is also reminiscent of clustering.
In particular, our greedy algorithm resembles Agglomerative Hierarchical Clustering
([13, 23]) in some respects. Agglomerative Hierarchical Clustering (HAC) is a
bottom-up algorithm that first treats each observation as a singleton cluster and
then successively merges (or agglomerates) pairs of clusters until all clusters have
been merged into a single cluster that contains all observations. In order to decide
which clusters should be combined, a measure of dissimilarity between sets of obser-
vations is required. This is achieved by using a measure of distance between pairs
of observations, and a linkage criterion which specifies the dissimilarity of sets as a
function of the pairwise distances of observations in the sets. Examples of such link-
age criteria were given in Chapter 6. Similarly, our greedy algorithm successively
merges pairs of annotations until the stop condition is reached. There is however a
major difference between the two approaches. Our choices of annotations to merge
are based on the obtained summary provenance and its features such as distance
and size. As described in Chapter 6, in our experiments, in order to compare our
approach to HAC, we created a slightly modified HAC approach that takes into con-
sideration both our stop conditions and our semantic constraints (e.g. this modified
HAC approach does not merge user clusters in which users have nothing in com-
mon). To conclude, our approach is different since the function that we optimize is
one that depends on the provenance expression itself and its intended uses, which
leads to different design choices and to different results.
IX  Conclusions

We have studied summarization of provenance information. We have identified three desiderata for the assessment of candidate summaries: conciseness, semantic constraints satisfaction and small distance from original provenance. This has led us to the development of our summarization algorithm that finds an “optimal” summary according to these quality measurements. After comparing our approach to other approaches (Clustering and Random) by conducting different experiments using different provenance datasets, we conclude that our approach is indeed more fitting for the goal of finding quality summaries and allows the user to control the desired tradeoff between distance (that affects evaluation accuracy) and size (that affects presentation and usage time). As future work, we intend to explore a generalized version of the algorithm in which in each iteration we map k annotations to a new annotation rather than just 2. In this setting, we will also address the tradeoff between the number of annotations mapped to a new annotation in a single algorithm step and the overall number of steps till the stop condition is reached. Indeed, the more annotations mapped in a single step, the more work done by the algorithm in a single step and so less algorithm steps are required to reach the stop condition. An additional line of future work is to acheive further theoretical bounds on the algorithm’s performance and output quality.
References


Applications nowadays collect large amounts of information from various sources, and then aggregate and manipulate the information in various ways. The complexity of these applications, combined with the size of the information gathered, makes it difficult to understand the way the final information was received.

Data Provenance is useful and effective in answering such questions, but maintaining and presenting Provenance in full and in detail may be impractical due to the size and complexity of this information. Therefore, we propose the idea of Approximated Summarized Provenance, which provides a concise representation of information Provenance, despite the possible loss of some information.

Based on this idea, we propose a new algorithm for summarizing Provenance information, which relies on the semantics of the data and the way the user plans to use the data, returning a summary of the Provenance expression in the input.

Experiments evaluate the summarization accuracy and the improvement in the use of Provenance. In addition to the experiments we conducted, we developed a system named PROX. This system uses the summarization algorithm and presents a summary of the Provenance information selected by the user, also allowing Approximate Provisioning.
Data Provenance

Alinor Even

בנהנהות של פורפ', טובה מילוא וד"ר דניאל דויטש